BOB CARPENTER

PROBABILITY AND STATIS-TICS

Preface

This book provides a short, accessible introduction to *applied* probabilistic modeling and statistical inference.[1]

[1] Application areas include the physical sciences, biological sciences, social sciences, engineering, health and medicine, business and finance, education, government, and sport and leisure—anywhere there is data from which we want to draw conclusions or make decisions.

Goal

After reading this book, you should have a solid grasp of how to draw sound inferences from noisy measurements and how to make reasoned decisions in the face of uncertainty.

Approach

This book differs from other books covering similar material in that it

- is short, top-down, and example-driven;
- assumes only introductory-level mathematics and computation;[2]
- develops probability theory and statistical inference through simulation;
- takes an unapologetically Bayesian approach to modeling and inference;
- is fully reproducible with open-source code, figures, and text.

[2] Specifically, this book assumes some familiarity with algebra, integral calculus, and exposure to basic programming concepts.

Licensing

Acknowledgements

Thanks to Mitzi Morris for proofreading.

What is Probability?

The fundamental nature of probability as used in applied statistical inference was described succintly by John Stuart Mill as part of his larger program of characterizing inductive reasoning,[3]

> We must remember that the probability of an event is not a quality of the event itself, but a mere name for the degree of ground which we, or some one else, have for expecting it. ... Every event is in itself certain, not probable; if we knew all, we should either know positively that it will happen, or positively that it will not. But its probability to us means the degree of expectation of its occurrence, which we are warranted in entertaining by our present evidence.

[3] Mill, John Stuart. 1882. *A System of Logic: Raciocinative and Inductive*. Eighth edition. Harper & Brothers, Publishers, New York. Part III, Chapter 18.

Mill is saying that our estimate of the probability of an event fundamentally depends on how much we know. For example, if you ask me now to forecast the weather on a random day in Edinburgh next year, I'll forecast 52% because it rains on average 191 days per year in Edinburgh.[4] If I know the day is in December, I'll make an estimate of 65% conditioned on knowing that it rains on average 20 days in December. Continuing to add information, if it's December 7 and I have up-to-date radar, then my estimate of the chance of rain on December 8 or December 9 might be anything, depending on the current meteorological conditions.[5]

[4] World Weather and Climate (2018) Edinburgh rainfall.

[5] The current forecast from weather.com as of 11:30 pm December 7, 2018 is a 60% chance of rain on December 8 and a 10% chance of rain on December 9.

Putting Mill's view in more modern terms, probability is a relative measure of uncertainty conditioned on available information. In other words, probability involves statements about an agent's or collection of agents' knowledge of the world, not about the world directly.[6] This allows us to believe the world is deterministic, while still reasoning probabilistically based on available evidence. Apparently, this was Pierre-Simon Laplace's position, as he wrote,[7]

> We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes.

[6] In philosophical terms, the nature of probability is *epistemic* (based on knowledge) rather than *ontological* (based on metaphysics) or *deontic* (based on belief).

[7] Pierre-Simon Laplace. 1814. *A Philosophical Essay on Probabilities*. English translation of the 6th edition, Truscott, F.W. and Emory, F.L. 1951. Dover Publications. page 4.

It also provides us the wiggle room to take sides with Albert Einstein, who wrote,[8]

> The theory [quantum mechanics] says a lot, but does not really bring us any closer to the secret of the "old one." I, at any rate, am convinced that He does not throw dice.

Probability theory merely provides a mathematically and logically consistent approach to quantifying uncertainty and performing inductive inference.

[8] Albert Einstein. 1926. Personal letter to Max Born. December 4.

Random Variables and Event Probabilities

Random variables

Let Y be the result of a fair coin flip. Not a general coin flip, but a specific instance of flipping a specific coin at a specific time. Defined this way, Y is what's known as a *random variable*, meaning a variable that takes on different values with different probabilities.[9]

Probabilities are scaled between 0% and 100% as in natural language. If a coin flip is fair, there is a 50% chance the coin lands face up ("heads") and a 50% chance it lands face down ("tails"). For concreteness and ease of analysis, random variables will be restricted to numerical values. For the specific coin flip in question, the random variable Y will take on the value 1 if the coin lands heads and the value 0 if it lands tails.

[9] Random variables are conventionally written using upper-case letters to distinguish them from ordinary mathematical variables which are bound to single values and conventionally written using lower-case letters.

Events and probability

An outcome such as the coin landing heads is called an *event* in probability theory. For our purposes, events will be defined as conditions on random variables. For example, $Y = 1$ denotes the event in which our coin flip lands heads. The functional $\Pr[\,\cdot\,]$ defines the probability of an event. For example, for our fair coin toss, the probability of the event of the coin landing heads is written as

$$\Pr[Y = 1] = 0.5.$$

In order for the flip to be fair, we must have $\Pr[Y = 0] = 0.5$, too. The two events $Y = 1$ and $Y = 0$ are mutually exclusive in the sense that both of them cannot occur at the same time. In probabilistic notation,

$$\Pr[Y = 1 \text{ and } Y = 0] = 0.$$

The events $Y = 1$ and $Y = 0$ are also exhaustive, in the sense that at least one of them must occur. In probabilistic notation,

$$\Pr[Y = 1 \text{ or } Y = 0] = 1.$$

In these cases, events are conjoined (with "and") and disjoined (with "or"). These operations apply in general to events, as does negation. As an example of negation,

$$\Pr[Y \neq 1] = 0.5.$$

Sample spaces and possible worlds

Even though the coin flip will have a specific outcome in the real world, we consider alternative ways the world could have been. Thus even if the coin lands heads ($Y = 1$), we entertain the possibility that it could've landed tails ($Y = 0$). Such counterfactual reasoning is the key to understanding probability theory and applied statistical inference.

An alternative way the world could be, that is, a *possible world*, will determine the value of every random variable. The collection of all such possible worlds is called the *sample space*.[10] The sample space may be conceptualized as an urn containing a ball for each possible way the world can be. On each ball is written the value of every random variable.[11]

Now consider the event $Y = 0$, in which our coin flip lands tails. In some worlds, the event occurs (i.e., 0 is the value recorded for Y) and in others it doesn't. An event picks out the subset of worlds in which it occurs.[12]

[10] The sample space conventionally written as Ω, the capitalized form of the last letter in the Greek alphabet.

[11] Formally, a random variable X can be represented as a function from the sample space to a real value, i.e., $X : \Omega \to \mathbb{R}$. For each possible world $\omega \in \Omega$, the variable X takes on a specific value $X(\omega) \in \mathbb{R}$.

[12] Formally, an event is defined by a subset of the sample space, $E \subseteq \Omega$.

Simulating random variables

We are now going to turn our attention to computation, and in particular, simulation, with which we will use to estimate event probabilities.

The primitive unit of simulation is a function that acts like a random number generator. But we only have computers to work with and they are deterministic. At best, we can created so-called *pseudo-random number generators*. Pseudorandom number generators, if they are well coded, produce deterministic streams of output that appear to be random.[13]

For the time being, we will assume we have a primitive pseudorandom number generator `uniform_01_rng()`, which behaves roughly like it has a 50% chance of returning 1 and a 50% chance of returning 0.[14]

Suppose we want to simulate our random variable Y. We can do so by calling `uniform_01_rng` and noting the answer.

[13] There is a large literature on pseudo-random number generators and tests for measurable differences from truly random streams.

[14] The name arises because random variables in which every possible outcome is equally likely are said to be *uniform*.

A simple program to generate a realization of a random coin flip, assign it to an integer variable y, and print the result could be coded as follows.[15]

```
int y = uniform_01_rng()
print 'y = ' y
```

The variable y is declared to be an integer and assigned to the result of calling the `uniform_01_rng()` function.[16] The print statement outputs the quoted string `y =` followed by the value of the variable y. Executing the program might produce the following output.

```
y = 1
```

If we run it a nine more times, it might print

```
y = 1
y = 1
y = 1
y = 1
y = 0
y = 0
y = 1
y = 1
y = 0
```

When we say it might print these things, we mean the results will depend on the state of the pseudorandom number generator.

Seeding a simulation

Simulations can be made exactly reproducible by setting what is known as the *seed* of a pseudorandom number generator. This seed establishes the deterministic sequence of results that the pseudorandom number generator produces. For instance, contrast the program

```
seed_rng(1234)
for (n in 1:10) print uniform_01_rng()
for (n in 1:10) print uniform_01_rng()
```

which produces the output

```
0 1 1 1 1 1 0 0 1 1
1 1 0 1 0 1 0 0 0 0
```

with the program

[15] Computer programs are presented using a consistent pseudocode, which provides a sketch of a program that should be precise enough to be coded in a concrete programming language. R implementations of the pseudocode generate the results and are available in the source code repository for this book.

[16] The use of a lower-case y was not accidental. The variable y represents an integer, which is the type of a realization of a random Y representing the outcome of a coin flip. In code, variables are written in typewriter font (e.g., y), whereas in text they are written in italics like other mathematical variables (e.g., y).

```
seed_rng(1234)
for (n in 1:10) print uniform_01_rng()
seed_rng(1234)
for (n in 1:10) print uniform_01_rng()
```

which produces

```
 0 1 1 1 1 1 0 0 1 1
 0 1 1 1 1 1 0 0 1 1
```

Resetting the seed in the second case causes exactly the same ten pseudorandom numbers to be generated a second time. Every well-written pseudorandom number generator and piece of simulation code should allow the seed to be set manually to ensure reproducibility of results.[17]

[17] Replicability of results with different seeds is a desirable, but stricter condition.

Using simulation to estimate event probabilities

We know that $\Pr[Y = 1]$ is 0.5 because it represents the flip of a fair coin. Simulation based methods allow us to estimate event probabilities straightforwardly if we can generate random realizations of the random variables involved in the event definitions.

For example, we know we can generate multiple simulations of flipping the same coin. That is, we're not simulating the result of flipping the same coin ten different times, but simulating ten different realizations of exactly the same random variable, which represents a single coin flip.

The fundamental method of computing event probabilities will not change as we move through this book. We simply simulate a sequence of values and return the proportion in which the event occurs as our estimate.

For example, let's simulate 10 values of Y again and record the proportion of the simulated values that are 1. That is, we count the number of time the event occurs in that the simulated value $y^{(m)}$ is equal to 1.

```
occur = 0
for (m in 1:M)
  y[m] = uniform_01_rng()
  occur = occur + (y[m] == 1)
estimate = occur / M
print 'estimated Pr[Y = 1] = ' estimate
```

The equality operator is written as `==`, as in the condition `y[m] == 1` to distinguish it from the assignment statement `y[m] = 1`, which

sets the value of `y[m]` to 1. The condition expression `y[m] == 1` evaluates to 1 if the condition is true and 0 otherwise.

If we let `uniform_01_rng(M)` be the result of generating `M` pseudo-random coin flip results, the program can be shortened to

```
y <- uniform_01_rng(M)
occur = sum(y == 1)
estimate = occur / M
```

A condition such as `y == 1` on a sequence returns a sequence of the same length with value 1 in positions where the condition is true. For instance, if

```
y = (2, 1, 4, 2, 2, 1)
```

then

```
y == 2
```

evaluates to

```
(1, 0, 0, 1, 1, 0).
```

Thus `sum(y == 1)` is the number of positions in the sequence `y` which have the value 1. Running the program provides the following estimate based on ten simulation draws.

```
1 1 0 1 0 1 0 0 0 0
estimated Pr[Y = 1] =  0.4
```

Let's try that a few more times.

```
0 0 0 0 0 1 1 1 1 0  estimated Pr[Y = 1] =  0.4
0 0 0 1 0 1 0 0 1 1  estimated Pr[Y = 1] =  0.4
1 1 0 1 0 1 1 0 0 1  estimated Pr[Y = 1] =  0.6
0 0 1 1 0 1 0 1 0 1  estimated Pr[Y = 1] =  0.5
1 0 0 0 0 1 0 1 0 1  estimated Pr[Y = 1] =  0.4
0 1 0 1 0 1 0 0 0 1  estimated Pr[Y = 1] =  0.4
1 0 0 1 0 1 0 0 0 1  estimated Pr[Y = 1] =  0.4
0 1 0 0 0 1 0 0 0 1  estimated Pr[Y = 1] =  0.3
0 1 0 0 0 0 0 0 0 0  estimated Pr[Y = 1] =  0.1
1 0 1 0 0 1 1 0 0 1  estimated Pr[Y = 1] =  0.5
```

The estimates are close, but not very exact. What if we use 100 simulations?

```
1 1 1 1 0 0 0 1 0 0 1 1 0 0 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 1 0 1 1
estimated Pr[Y = 1] =  0.59
```

That's closer than most of the estimates based on ten simulation draws. Let's try that a few more times without bothering to print all 100 simulated values,

```
estimated Pr[Y = 1] =  0.48
estimated Pr[Y = 1] =  0.51
estimated Pr[Y = 1] =  0.62
estimated Pr[Y = 1] =  0.54
estimated Pr[Y = 1] =  0.5
estimated Pr[Y = 1] =  0.52
estimated Pr[Y = 1] =  0.49
estimated Pr[Y = 1] =  0.52
estimated Pr[Y = 1] =  0.42
estimated Pr[Y = 1] =  0.45
```

What happens if we let $M = 10,000$ simulations?

```
estimated Pr[Y = 1] =  0.5
estimated Pr[Y = 1] =  0.5
estimated Pr[Y = 1] =  0.51
estimated Pr[Y = 1] =  0.49
estimated Pr[Y = 1] =  0.5
estimated Pr[Y = 1] =  0.5
estimated Pr[Y = 1] =  0.51
estimated Pr[Y = 1] =  0.5
estimated Pr[Y = 1] =  0.5
estimated Pr[Y = 1] =  0.49
```

Now the estimates are very close to the true probability being estimated (i.e., 0.5, because the flip is fair). This raises the questions of how many simulation draws we need in order to be confident our estimates are close to the values being estimated.

Law of large numbers

Visualization in the form of simple plots goes a long way toward understanding concepts in statistics and probability. A traditional way to plot what happens as the number of simulation draws M increases is to keep a running tally of the estimate as each draw is made and plot the estimated event probability $\Pr[Y = 1]$ for each $m \in 1 : M$.[18]

To calculate such a running tally of the estimate at each online, we can do this:

```
occur = 0
for (m in 1:M)
```

[18] See, for example, the quite wonderful little book, Bulmer, M.G., 1965. *Principles of Statistics*. Oliver and Boyd, Edinburgh.

```
y[m] = uniform_01_rng(M)
occur = occur + (y[m] == 1)
estimate[m] = occur / m
```

Recall that the expression `(y[m] == 1)` evaluates to 1 if the condition holds and 0 otherwise. The result of running the program is that `estimate[m]` will hold the estimate $\Pr[Y = 1]$ after m simulations. We can then plot the estimates as a function of the number of draws using a line plot to display the trend.

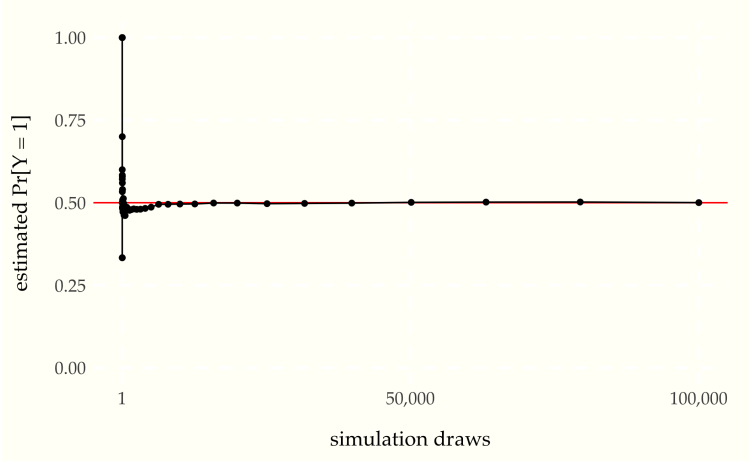


Figure 1: Monte Carlo estimate of probability that a coin lands head as a function of the number of simulation draws. The line at 0.5 marks the true probability being estimated. Plotted on a linear scale, it is clear how quickly the estimates converge to roughly the right value.

The linear scale of the previous plot obscures the behavior of the estimates. Consider instead a plot with the x-axis on the logarithmic scale.

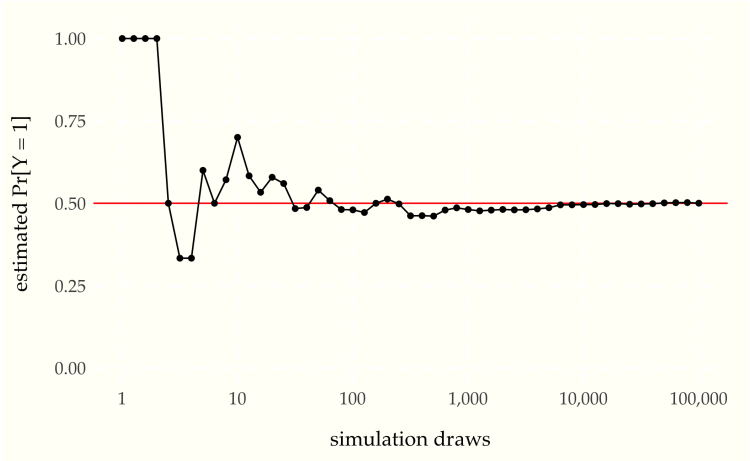


Figure 2: Monte Carlo estimate of probability that a coin lands head as a function of the number of simulation draws. The line at 0.5 marks the true probability being estimated. The log-scaled x-axis makes the early rate of convergence more evident.

With a log-scaled x-axis, the values between 1 and 10 are plotted with the same width as the values between 10 000 and 100 000; both take up 20% of the width of the plot. On the linear scale, the values

between 1 and 10 take up only $\frac{10}{100\,000}$, or 0.01%$, of the plot, wehreas values between 10 000 and 100 000 take up 90% of the plot.

Plotting the progression of multiple simulations demonstrates the trend in errors.

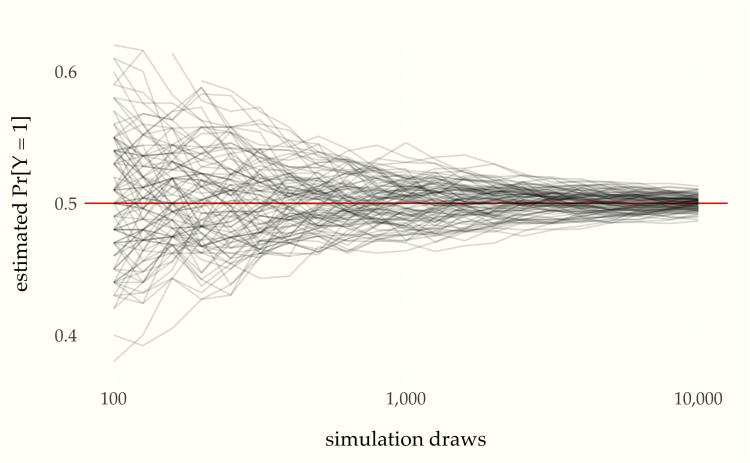


Figure 3: Each of the one hundred grey lines represents the ratio of heads observed in a sequence of coin flips, the size of which indicated on the x-axis. The line at 0.5 indicates the probability a coin lands heads in a fair coin toss. The convergence of the ratio of heads to 0.5 in all of the sequences is clearly visible.

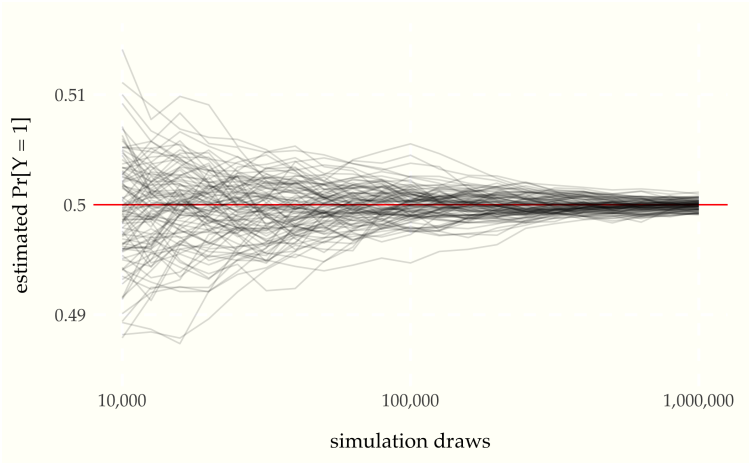


Figure 4: Continuing where the previous plot left off, each of the one hundred grey lines represents the ratio of heads observed in a sequence of coin flips. The values on the x axis is one hundred times larger than in the previous plot, and the scale of the y-axis is one tenth as large. The trend in error reduction appears the same at the larger scale.

The *law of large numbers*[19] says roughly that as the number of simulated values grows, the average will converge to the expected value. In this case, our estimate of $\Pr[Y = 1]$ can be seen to converge to the true value of 0.5 as the number of simulations M increases. Because the quantities involved are probabilistic, the exact specification is a little more subtle than the ϵ-δ proofs in calculus.

[19] Which technically comes in a strong and weak form.

Simulation notation

We will use parenthesized superscripts to pick out the elements of a sequence of simulations. For example,

$$y^{(1)}, y^{(2)}, \ldots, y^{(M)}$$

will be used for M simulations of a single random variable Y.[20] It's important to keep in mind that this is M simulations of a single random variable, not a single simulation of M different random variables.

[20] Each $y^{(m)}$ is a possible realization of Y, which is why they are written using lowercase.

Before we get going, we'll need to introduce *indicator function* notation. For example, we write

$$\mathrm{I}[y^{(m)} = 1] = \begin{cases} 1 & \text{if } y^{(m)} = 1 \\ 0 & \text{otherwise} \end{cases}$$

The indicator function maps a condition, such as $y^{(m)} = 1$ into the value 1 if the condition is true and 0 if it is false.[21]

[21] Square bracket notation is used for functions when the argument is itself a function. For example, we write $\Pr[Y > 0]$ because Y is a random variable, which is modeled as a function. We also write $\mathrm{I}[x^2 + y^2 = 1]$ because the standard bound variables x and y are functions from contexts defining variable values.

Now we can write out the formula for our estimate of $\Pr[Y = 1]$ after M draws,

$$\Pr[Y = 1] \quad \approx \quad \frac{\mathrm{I}[y^{(1)} = 1] \; + \; \mathrm{I}[y^{(2)} = 1] \; + \; \cdots \; + \; \mathrm{I}[y^{(M)} = 1]}{M}$$

That is, our estimate is the proportion of the simulated values which take on the value 1. It quickly becomes tedious to write out sequences, so we will use standard summation notation, where we write

$$\mathrm{I}\Big[y^{(1)} = 1] + \mathrm{I}[y^{(2)} = 1] + \cdots + \mathrm{I}[y^{(M)} = 1\Big] \; = \; \sum_{m=1}^{M} \mathrm{I}[y^{(m)} = 1]$$

Thus we can write our simulation-based estimate of the probability that a fair coin flip lands heads as[22]

$$\Pr[Y = 1] \approx \frac{1}{M} \sum_{m=1}^{M} \mathrm{I}[y^{(m)} = 1]$$

The form $\frac{1}{M} \sum_{m=1}^{M}$ will recur repeatedly in simulation — it just says to average over values indexed by $m \in 1 : M$.[23]

[22] In general, the way to estimate an event probability $\phi(Y)$ where ϕ defines some condition, given simulations $y^{(1)}, \ldots, y^{(M)}$ of Y, is as

$$\Pr[\phi(Y)] = \frac{1}{M} \sum_{m=1}^{M} \mathrm{I}[\phi(y^{(m)})].$$

[23] We are finally in a position to state the *strong law of large numbers* as the event probability of a limit,

$$\Pr\left[\lim_{M \to \infty} \frac{1}{M} \sum_{m=1}^{M} \mathrm{I}\Big[y^{(m)} = 1\Big] \; = \; 0.5\right],$$

where each $y^{(m)}$ is a separate fair coin toss.

Central limit theorem

The law of large numbers tells us that with more simulations, our estimates become more and more accurate. But they do not tell us how quickly we can expect that convergence to proceed. The *central limit theorem* provides the convergence rate.

First, we have to be careful about what we're defining. First, we define the *error* for an estimate as the difference from the true value,

$$\left(\frac{1}{M}\sum_{m=1} \mathrm{I}[y^{(m)} = 1]\right) - 0.5$$

The *absolute error* is just the absolute value[24] of this,

$$\left|\left(\frac{1}{M}\sum_{m=1} \mathrm{I}[y^{(m)} = 1]\right) - 0.5\right|$$

[24] In general, the absolute value function applied to a real number x is written as $|x|$ and defined to be x if x is non-negative and $-x$ if x is negative.

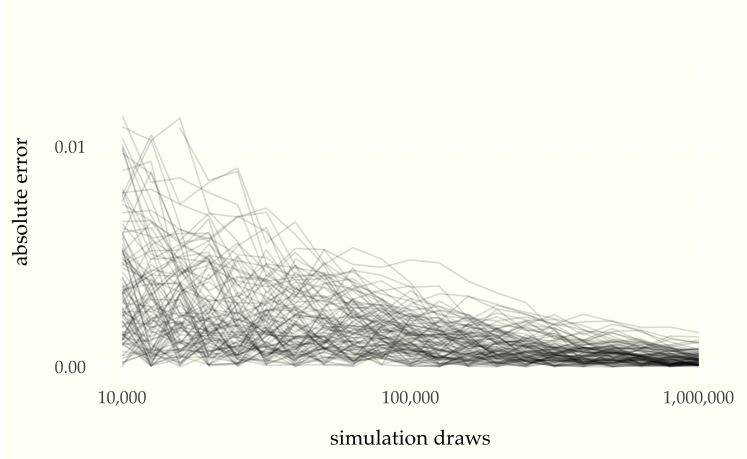


Figure 5: The absolute error of the simulation-based probability estimate as a function of the number of simulation draws. One hundred sequences of one million flips are shown.

Plotting both the number of simulations and the absolute error on the log scale reveals the rate at which the error decreases with more draws.

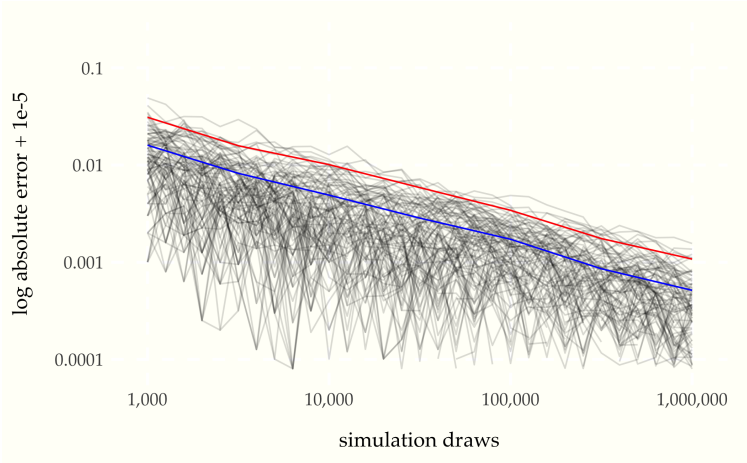


Figure 6: Absolute error versus number of simulation draws for 100 simulated sequences of $M = 1\,000\,000$ draws. The blue line is at the 68 percent quantile and the red line at the 95 percent quantile of these draws. The relationship between the log number of draws and log error is revealed to be linear.

We can read two points (x_1, y_1) and (x_2, y_2) off of the graph for $x_1 = 10\,000$ and $x_2 = 100\,000$ as

```
x[1], y[1] =   10000, 0.00490
x[2], y[2] =  100000, 0.00172
```

which gives us the following values on the log scale

```
log x[1], log y[1] =  9.21, -5.32
log x[2], log y[2] = 11.51, -6.36
```

Using the log scale values, the estimated slope of the reduction in quantile bounds is

```
estimated slope
(log y[2] - log y[1])
 / (log x[2] - log x[1])  =  -0.45
```

If we let ϵ_M be the value of one of the quantile lines at M simulation draws, the linear relationship plotted in the figures have the form[25]

$$\log \epsilon_M = -\frac{1}{2} \log M + \text{const.}$$

When writing "const" in a mathematical expression, the presumption is that it refers to a constant that does not depend on the free variables of interest (here, M, the number of simulation draws). Ignoring constants lets us focus on the order of the dependency. The red line and blue line have the same slope, but different constants.

Seeing how this plays out on the linear scale requires exponentiating both sides of the equation and reducing,

$$\begin{aligned}
\exp(\log \epsilon_M) &= \exp(-\tfrac{1}{2} \log M + \text{const}) \\
\epsilon_M &= \exp(-\tfrac{1}{2} \log M) \times \exp(\text{const}) \\
\epsilon_M &= \exp(\log M)^{-\frac{1}{2}} \times \exp(\text{const}) \\
\epsilon_M &= \exp(\text{const}) \times M^{-\frac{1}{2}}
\end{aligned}$$

Dropping the constant, this relationship between the expected absolute error ϵ_M after M simulation draws may be succinctly summarized using proportionality notation,[26]

$$\epsilon_M \propto \frac{1}{\sqrt{M}}.$$

This is a fundamental result in statistics derived from the *central limit theorem*. The central limit theorem governs the accuracy of almost all simulation-based estimates. We will return to a proper formulation when we have the scaffolding in place to deal with the pesky constant term.

In practice, what does this mean? It means that if we want to get an extra decimal place of accuracy in our estimates, we need one hundred (100) times as many draws. For example, the plot shows error rates bounded by roughly 0.01 with 10 000 draws, yielding

[25] A line through the points (x_1, y_1) and (x_2, y_2) has

$$\text{slope} = \frac{y_2 - y_1}{x_2 - x_1}.$$

[26] In general, we write

$$f(x) \propto g(x)$$

if there is a positive constant c that does not depend on x such that

$$f(x) = c \times g(x).$$

For example,

$$3x^2 \propto 9x^2,$$

with $c = \frac{1}{3}$.

estimates that are very likely to be within $(0.49, 0.51)$. To reduce that likely error bound to 0.001, that is, ensuring estimates are very likely in $(0.0499, 0.501)$, requires 100 times as many draws (i.e., a whopping 1 000 000 draws).[27]

Usually, one hundred draws will provide a good enough estimate of most quantities of interest in applied statistics. The variability of the estimate based on a single draw depends on the variability of the quantity being estimated. One hundred draws reduces the expected estimation bound to one tenth of the variability of a single draw. Reducing that variability to one hundredth of the variability of a single draw would require ten thousand draws. In most applications, the extra estimation accuracy is not worth the extra computation.

[27] For some perspective, 10 000 is the number of at bats in an entire twenty-year career for a baseball player, the number of field goal attempts in an entire career of most basketball players, and the size of a very large disease study or meta-analysis in epidemiology.

Multiple Random Variables and Probability Functions

Multiple random variables

Random variables do not exist in isolation. We started with a single random variable Y representing the result of a single, specific coin flip. Suppose we fairly flip the coin three times? Then we can have random variables Y_1, Y_2, Y_3 representing the results of each of the flips. We can assume each flip is independent in that it doesn't depend on the result of other flips. Each of these variables Y_n for $n \in 1:3$ has $\Pr[Y_n = 1] = 0.5$ and $\Pr[Y_n = 0] = 0.5$.

We can combine multiple random variables using arithmetic operations. We have already seen comparison operators in writing the event $Y = 1$. If $Y_1, \ldots, Y_{10}$ are random variables representing ten coin flips, then we can define their sum as

$$Z = Y_1 + Y_2 + Y_3$$

We can simulate values of Z by simulating values of Y_1, Y_2, Y_3 and adding them.

```
y1 = uniform_01_rng()
y2 = uniform_01_rng()
y3 = uniform_01_rng()
z = y1 + y2 + y3
print 'z = ' z
```

It is easier and less error prone to collapse similar values into arrays and operate on the arrays collectively, or with loops if necessary.

```
for (n in 1:3)
  y[n] = uniform_01_rng()
z = sum(y)
print 'z = ' z
```

Running this program a few times we get

```
 z = 2
```

```
z = 3
z = 1
z = 3
z = 1
```

We can use simulation to evaluate the probability of an outcome that combines multiple random variables. For example, to evaluate $\Pr[Z = 2]$, we run the simulation many times and count the proportion of results that are two.[28]

[28] The sum is calculated using notation `sum(y[m, ])`, which is defined to be `sum(y[m, ]) = y[m, 1] + ... + y[m, N]`, where `N` is the number of entries in row `m` of the variable `y`.

```
for (m in 1:M)
  for (n in 1:3)
    y[m, n] = uniform_01_rng()
  z[m] = sum(y[m, ])
Pr_is_two = sum(z == 2) / M
```

As in our other probability estimates, we simulate the variable of interest Z a total of M times, yielding $z^{(1)}, \ldots, z^{(m)}$. Here, that requires simulating $y_1^{(m)}, y_2^{(m)}, y_3^{(m)}$ and adding them for each $z^{(m)}$. We then just count the number of times `Z` is simulated to be equal to `2` and divide by the number of simulations.

Letting $M = 100\,000$, and running five times, we get

```
Pr[Z == 2] = 0.375
Pr[Z == 2] = 0.374
Pr[Z == 2] = 0.374
Pr[Z == 2] = 0.373
Pr[Z == 2] = 0.378
```

Nailing down that final digit is going to require one hundred times as many iterations (i.e, $M = 10\,000\,000$ iterations). Let's see what that looks like.

```
Pr[Z == 2] = 0.375
Pr[Z == 2] = 0.375
Pr[Z == 2] = 0.375
Pr[Z == 2] = 0.375
Pr[Z == 2] = 0.375
```

We can do the same for the other numbers, to get a complete picture of taking the probability of each number of heads in separate coin flips.

```
Pr[Z == 0] = 0.125
Pr[Z == 1] = 0.375
Pr[Z == 2] = 0.375
Pr[Z == 3] = 0.125
```

What if we flip four coins instead of three?

```
Pr[Z == 0] = 0.062
Pr[Z == 1] = 0.250
Pr[Z == 2] = 0.375
Pr[Z == 3] = 0.250
Pr[Z == 4] = 0.062
```

Discrete random variables

So far, we have only considered random numbers that take a finite number of integer values. A random variable that only takes values in the integers, i.e., values in

$$\mathbb{Z} = \ldots -2, -1, 0, 1, 2, \ldots$$

is said to be a *discrete random variable.*[29]

[29] In general, any countable set of numerical values could be used as values of a discrete random variable. A set of values is *countable* if each of its members can be assigned a unique counting number in $\mathbb{N} = 0, 1, 2, \ldots$. The integers $\mathbb{Z}$ can be mapped to natural numbers $\mathbb{N}$ by interleaving,

$\mathbb{Z}$		$\mathbb{N}$
0	$\mapsto$	0
-1	$\mapsto$	1
1	$\mapsto$	2
-2	$\mapsto$	3
2	$\mapsto$	4
	$\vdots$	

Probability mass functions

It's going to be convenient to have a function that maps each possible outcome in a variable to its probability. In general, this will be possible if and only if the variable is discrete, as defined in the previous section.

For example, if we reconsider $Z = Y_1 + \cdots Y_4$, the number of heads in four separate coin flips, we can define a function[30]

$p_Z(0)$	=	1/16	TTTT
$p_Z(1)$	=	4/16	HTTT, THTT, TTHT, TTTH
$p_Z(2)$	=	6/16	HHTT, HTHT, HTTH, THHT, THTH, TTTH
$p_Z(3)$	=	4/16	HHHT, HHTH, HTHH, THHH
$p_Z(4)$	=	1/16	HHHH

There are sixteen possible outcomes of flipping four coins. Because the flips are separate and fair, each possible outcome is equally likely. The sequences corresponding to each count of heads (i.e., value of Z) are recorded in the rightmost columns. The probabilities are derived by dividing the number of ways a value for Z can arise by the number of possible outcomes.

This function p_Z was constructed to map a value u for Z to the event probability that $Z = u$,[31]

$$p_Z(u) = \Pr[Z = u].$$

A function defined as above is said to be the *probability mass function* of the random variable Z. Every discrete random variable has a unique probability mass function.

[30] We implicitly assume that functions return zero for arguments not listed.

[31] Conventionally, this is written as

$$p_Z(z) = \Pr[Z = z],$$

but that can be confusing with upper case Z denoting a random variable and lower case z denoting an ordinary variable.

Probablity mass functions represent probabilities of a discrete set of outcomes. The sum of all such probabilities must be one because at least one of the outcomes must occur.[32]

With large numbers of counts based on simulation, we can more readily apprehend what is going on with a plot. Discrete simulations are typically plotted using bar plots, where the outcomes are arrayed on the x axis with a vertical bar over each one whose height is proportional to the frequency of that outcome.

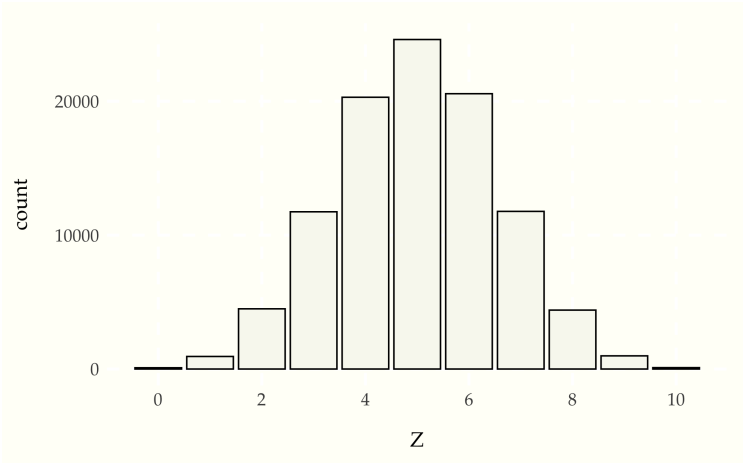


Figure 7: Plot of $M = 100\,000$ simulations of the probability mass function of a random variable defined as the number of heads in ten specific coin flips.

The actual frequencies are not relevant, only the relative sizes. A simple probability estimate from simulation provides a probability for each outcome proportional to its height.[33]

This plot can easily be repeated to see what happens as the number of bins grows.

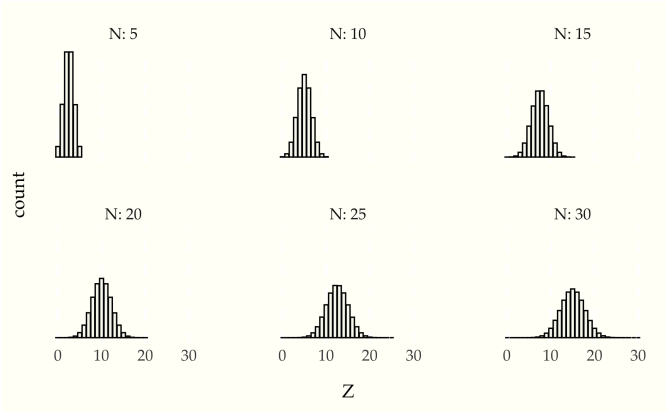


Figure 8: Plot of $M = 1\,000\,000$ simulations of a variable Z representing the number of heads in N coin flips. Each plot represents a different N. Because the bars are the same width and the x axes are scaled to the same range in all plots, the total length of all bars laid end to end is the same in each plot; similarly, the total area of the bars in each plot is the same.

[32] More formally, if Y is a discrete random variable, then

$$\sum_{u} p_Y(u) = 1,$$

where the summation variable u ranges over all possible values of Y. We are going to start writing this with the standard overloading of lower and upper case Y as

$$\sum_{y} p_Y(y) = 1.$$

[33] And proportional to its area because the bars are of equal width.

Dice

Let Y be a random variable representing a fair throw of a six-sided die. We can describe this variable easily through it's probability mass function, which is uniform (i.e., assigns each possible outcome the same probability).

$$p_Y(y) = \begin{cases} \frac{1}{6} & \text{if } y \in 1:6 \\ 0 & \text{otherwise} \end{cases}$$

Games like *Monopoly* use a pair of six-sided dice and consider the sum of the results. That is, Y_1 and Y_2 are fair six-sided die rolls and $Z = Y_1 + Y_2$ is the result. Games like *Dungeons & Dragons* use a trio of six-sided dice and consider the sum of the results. In that scenario, Y_1, Y_2, Y_3 are the results of fair six-sided die rolls and $Z = Y_1 + Y_2 + Y_3$. Dungeons and Dragons also uses four six-sided die of which the best 3 are summed to produce a result. Let's simulate some of these approaches and see what the results look like based on $M = 100\,000$ simulations.[34]

[34] The simulations are identical to before, only using `1:6` in the range of uniform variables.

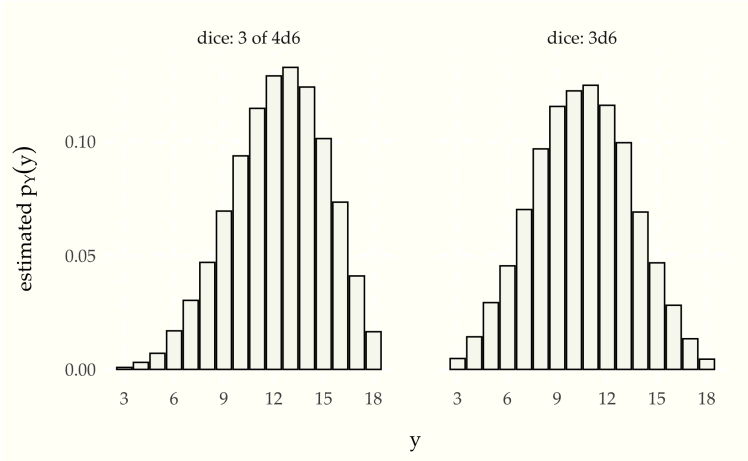


Figure 9: Estimated $p_Y(y)$ for case of Y being the sum of three six-sided dice (3d6) or the sum of the highest three of four six-sided dice (3 of 4d6).

Dungeons and Dragons also uses 20-sided dice.[35] The fifth edition of the game introduced the notion of *advantage*, where two 20-sided dice are rolled and the higher result retained, as well as *disadvantage*, which retains the lower result of the two dice. Here's a simulation using $M = 100\,000$. The counts are converted to estimated probabilities on the vertical axis in the usual way by dividing by M.

The most likely roll is a 20 when taking the best of two rolls and the most likely roll is 1 when taking the worst of two rolls.[36] The min and max plots are mirror images of each other as is to be expected by the consecutive nature of the numbers and the min/max operations.

[35] In physical games, an icosahedral die is used. The icosahedron is a regular polyhedron with 20 equilateral triangular faces, giving it the largest number of faces among the five Platonic solids.

[36] The chance for a 20 when taking the best of two 20-sided die rolls is $1 - \left(\frac{19}{20}\right)^2 \approx 0.098$; the chance of rolling 1 is $\left(\frac{1}{20}\right)^2 = 0.0025$. The probabilities are reversed when taking the worst of two 20-sided die rolls.

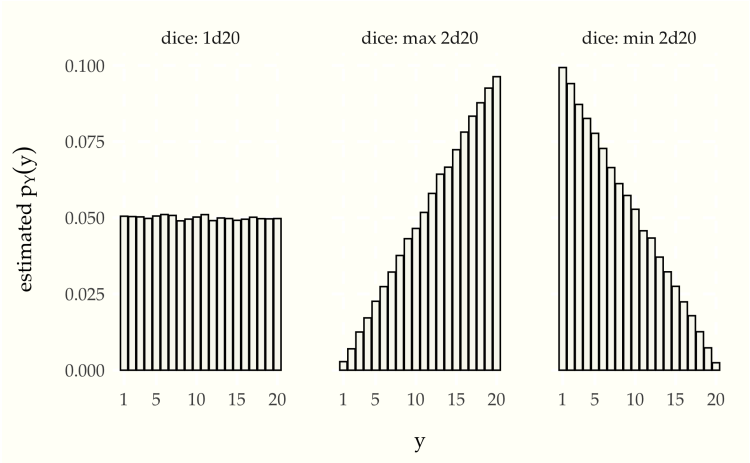


Figure 10: Estimated $p_Y(y)$ for case of Y being a single twenty-sided die (d20), the higher two twenty-sided die rolls (max 2d20), and the lower of two 20-sided die rolls (min 2d20).

Spinners and the Bernoulli distribution

Some games, such as *All Star Baseball*, come with spinners rather than dice. The beauty of spinners is that they can be divided into two areas, one with a 27% chance of occurring in a fair spin and one with a 73% chance of occurring.[37] Now suppose we have a random variable Y representing the result of a fair spin. Its probability mass function is

[37] 27% is roughly the chance of a hit in an at bat.

$$p_Y(y) \ = \ \begin{cases} 0.27 & \text{if } \ y = 1 \\ 0.73 & \text{if } \ y = 0 \end{cases}$$

To simplify our notation, we are going to start defining useful functions that can be used as probability mass functions. Our first example is the so-called Bernoulli[38] distribution. We define it as a function with somewhat peculiar notation,

[38] Named after Jacob Bernoulli (1654–1705), one of several prominent mathematicians in the family.

$$\text{Bernoulli}(y \mid \theta) \ = \ \begin{cases} \theta & \text{if } \ y = 1 \\ 1 - \theta & \text{if } \ y = 0 \end{cases}$$

The vertical bar (|) separates the *variate* argument y, which we think of as an outcome, from the *parameter* argument $\theta \in [0, 1]$, which determines the probability of the outcome. In this case, the variate y is discrete, and can take on only the values zero and one, so we write $y \in 0 : 1$. The parameter θ, on the other hand, is continuous and can take on any value between zero and one (inclusive of endpoints), so we write $y \in [0, 1]$.[39]

[39] Interval notation $[0, 1]$ is used for the set of values x such that $0 \leq x \leq 1$. Parentheses are used for exclusive endpoints, so that $(0, 1)$ is taken to be the set of x such that $0 < x < 1$.

This notation allows us to simplify our baseball example. Going back to our example random variable Y which had a 27% chance of being 1 and a 73% chance of being 0, we can write

$$p_Y(y) = \text{Bernoulli}(y \mid 0.27).$$

To simplify notation even further, we will say that a random variable U has a Bernoulli distribution and write

$$U \sim \text{Bernoulli}(\theta)$$

to indicate that the probability mass function of U is

$$p_U(u) = \text{Bernoulli}(u \mid \theta).$$

Cumulative distribution functions

In Dungeons & Dragons, the players are often concerned with probabilities of rolling higher than a given number (or equivalently, rolling lower than a given number. For example, they may need to roll a 15 to sneak by an orc. Such probabilities are conventionally given in the form of cumulative distribution functions. If Y is a random variable, its *cumulative distribution function* F_Y is defined by

$$F_Y(y) = \Pr[Y \leq y].$$

The event probability on the right is calculated the same way as always in a simulation, by counting the number of simulated values in which the condition holds and dividing by the number of simulations.

We can plot the cumulative distribution function for the straight twenty-sided die roll and the rolls with advantage (best of two rolls) or disadvantage (worst of two rolls). Here's the result using the same simulations as in the last plot, with $M = 100\,000$.

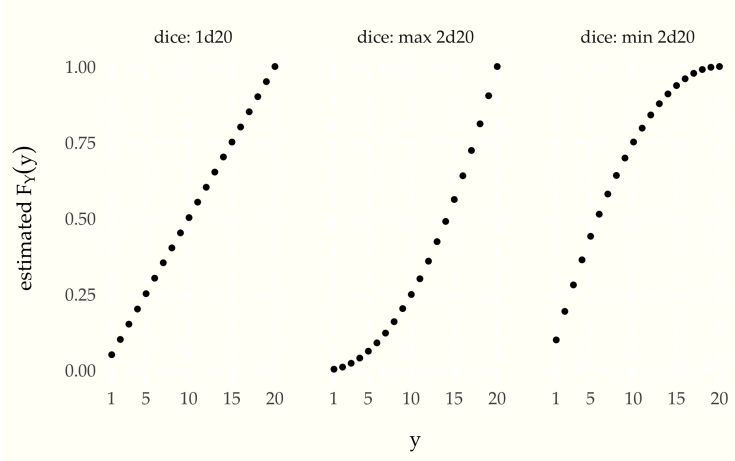


Figure 11: Cumulative distribution function for three variables corresponding to rolling a single 20-sided die, or rolling two 20-sided dice and taking the best or worst result.

** TODO(carpenter): get the right CDF plot and fix aspect ratio. **

The plot is rendered as a point plot, though this isn't quite sensible for discrete distributions—the intermediate values are not real. It's easy to see that there's a 50% chance of rolling 5 or lower in a single die throw; with the best of 2 it's more like a 20% chance and with the worst of 2, it's more like a 75% chance.

Usually in Dungeons & Dragons, players care about rolling more than a given number, not less, or they'd have to be subtracting all the time. This is where the complementary cumulative distribution function comes in. For a random variable Y, the *complementary cumulative distribution function* is

$$F_Y^{\complement}(y) \;=\; 1 - F_Y(y) \;=\; \Pr[Y > y].$$

It's easier to see with a plot how it relates to the usual cumulative distribution function.

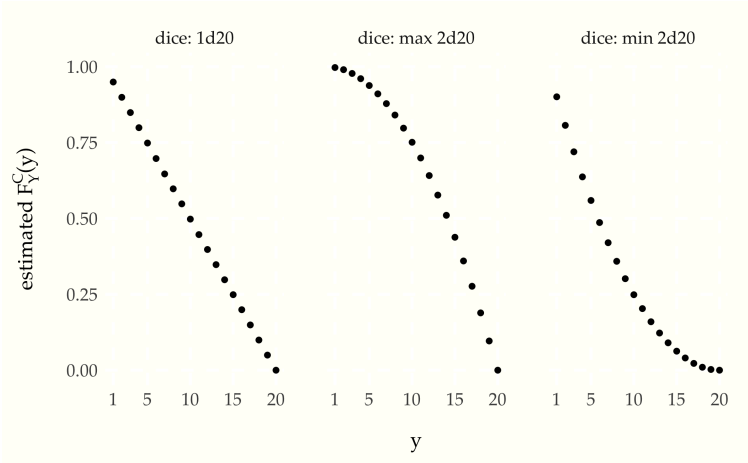


Figure 12: Complementary cumulative distributions for a single 20-sided die, the best of two dice, and the worst of two dice.

Infinite discrete random variables

Consider an experiment in which a coin is tossed until a heads appears Let the random variable U be the number of tosses that came up tails before the first head comes up. The legal sequences are H (0 tails), TH (1 tails), TTH (2 tails), and so on. There is no upper limit to how many tails may appear before the first heads.

Here's some code to create M simulations of the variable U.[40]

```
for (m in 1:M)
  u[m] = 0
  while (uniform_01_rng() == 0)
    u[m] += 1
```

This looks dangerous! The body of a while-loop (here `u[m] + 1`) is executed iteratively as long as the condition is true.[41] Shouldn't

[40] This code uses a while loop, which repeats as long as its condition evaluates to true (i.e., 1). Here, the condition compares the output of the random number generator directly rather than assigning to an intermediate value. We have also introduced the increment operator +=, which adds the value of the right hand side to the variable on the left hand side.

[41] If we write `while (1 + 1 == 2)` we produce what is known as an *infinite loop*, i.e., one that never terminates.

we be worried that the random number generator will just continue to throw tails (i.e., 0) so that the program never terminates?[42] In this case, no, because the odds are vanishingly small that U gets large. For example,

$$\begin{aligned}\Pr[U < 10] &= p_U(0) + p_U(1) + \cdots p_U(9) \\ &= \tfrac{1}{2} + \cdots + \tfrac{1}{1024} \\ &\approx 0.999.\end{aligned}$$

Going further, $\Pr[U < 20] \approx 0.999\,999$, and so on. So there's not much chance of running very long at all, much less forever.

With the concern of non-termination out of the way, let's see what we get with $M = 50$ simulations of U.

```
1  0  0  0  0  2  0  0  0  1
1  9  0  0  0  4  1  2  0  0
0  1  1  0  2  2  0  1  1  1
0  4  1  1  1  1  1  3  0  2
1  3  1  3  3  1  8  1  2  0
```

It's very hard to discern a pattern here. There are a lot of zero values, but also some large values. For cases like these, we can use a bar plot to plot the values. This time, we're going to use $M = 10\,000$ to get a better picture of the pattern.

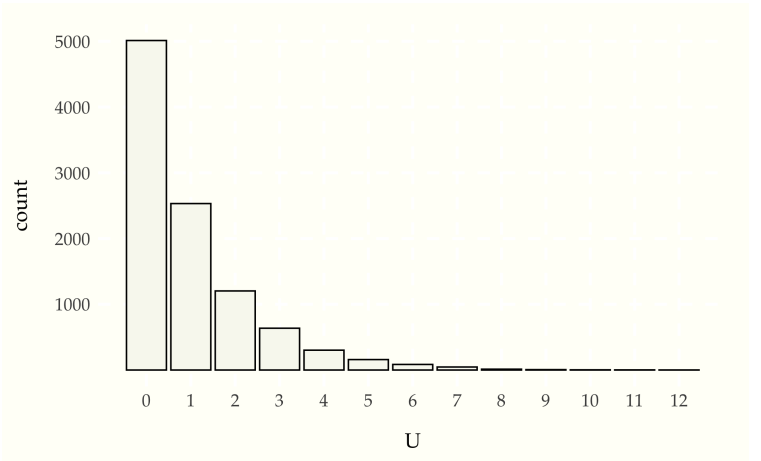


Figure 13: Frequency of outcomes in 10 000 simulation draws of U, the number of tails seen before a head in a coin-tossing experiment.

The x-axis represents the value of U and the y-axis the number of times that value arose in the simulation.[43] Each additional throw of tails appears to cut the probability of occurrence in half result seems to have about half the probability of the previous one. This is what we should expect because each coin toss brings a 50% probability of a tails result. This exponential decay[44] in the counts with the number

[42] The answer is "yes," in general, because programmers are error prone.

[43] Despite U having infinitely many possible values, it will only take on finitely many of them in a finite sample.

[44] Exponential decay means each additional outcome is only a fraction as likely as the previous one.

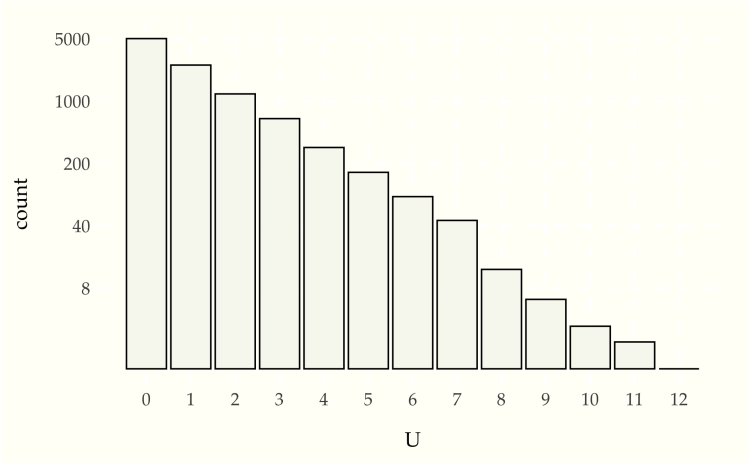


Figure 14: Frequency of outcomes in 10 000 simulation draws of U, the number of tails seen before a head in a coin-tossing experiment, this time with the outcome count on the log scale to illustrate the exponentially decreasing probabilities of each successive number of tails.

of tails thrown is more obvious when plotted on the log scale.

There is a 50% probability that the first toss is heads, yielding a sequence of zero tails, and $U = 0$. Each successive number of tails is half as likely as the previous, because another tail will have to be thrown, which has a 50% probability.[45]

Thus the overall probability mass function for U is[46]

$$\begin{array}{rcll} p_U(0) &=& \frac{1}{2} & = \frac{1}{2} \\ p_U(1) &=& \frac{1}{2} \times \frac{1}{2} & = \frac{1}{4} \\ p_U(2) &=& \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} & = \frac{1}{8} \\ & \vdots & & \\ p_U(u) &=& \underbrace{\frac{1}{2} \times \cdots \times \frac{1}{2}}_{u+1 \text{ times}} & = \left(\frac{1}{2}\right)^{u+1} \\ & \vdots & & \end{array}$$

Even though there are infinitely many possible realizations of the random variable U, simulation may still be used to compute event probabilities, such as $\Pr[U \leq 3]$, by

```
for (m in 1:M)
  u[m] = sim_u()
  leq3[m] = (u[m] <= 3)

print 'est Pr[U <= 3] = ', sum(leq3) / M
```

Let's see what we get with $M = 100 000,

```
 est Pr[U <= 3] = 0.937
```

Writing out the analytic answer involves an infinte sum,

[45] In symbols,

$$\Pr[U = n+1] \;=\; \frac{1}{2}\Pr[U = n].$$

[46] An elementary result of calculus is that

$$\sum_{n=0}^{\infty} \frac{1}{2^{n+1}} \;=\; \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots \;=\; 1.$$

$$\Pr[U \leq 3] \;=\; \sum_{u=0}^{\infty} p_U(u)\mathrm{I}[u \leq 3].$$

We can recognize that all of the terms where $u > 3$ are zero, so that this reduces to

$$\begin{aligned}\Pr[U \leq 3] &= p_U(0) + p_U(1) + p_U(2) + p_U(3) \\ &= \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \tfrac{1}{16} \\ &\approx 0.938.\end{aligned}$$

Simulation is not that clever. It just blindly simulates values of u, many of which turn out to be larger than three. In the sequence of simulated values, many were larger than three—the histogram summarized a much larger simulation, none of the values of which were that large.

Given that we only ever run a finite number of iterations and thus only ever see a finite number of values, how does simulation get the right answer?[47] Isn't there some kind of bias from only visiting smallish values? The answer is "no" precisely because the values that are not simulated are so rare. Their total probability mass, when added together is small, so they cannot have much influence on the simulated answer. For simulations to get the right answer,[48] they need only visit the typical values seen in simulation, not the extreme values.[49]

[47] Spoiler alert! The technical answer, like so much in statistics, is that the central limit theorem kicks in.

[48] Given some qualifications!

[49] The sparsity problem grows exponentially worse in higher dimensions and uncountably worse with continuous random variables.

The Chevalier de Méré's challenge

Antoine Gombaud, the Chevalier de Méré,[50] challenged Blaise Pascal to explain how it was possible that thre probability of throwing at least one six in four throws of a single six-sided die is slightly greater than $\frac{1}{2}$, whereas the probability of throwing two sixes in 24 throws of a pair of six-sided die was slightly less than $\frac{1}{2}$.[51] We can evaluate these claims by simulation directly.[52]

To represent the problem in probabilistic notation, we introduce a random variable $Y_{1,k} \in 1:6$ and $Y_{2,k} \in 1:6$ for each of the two dice in each of the $k \in 1:24$ throws. Define the outcome of the game as the random variable

$$Z \;=\; \begin{cases} 1 & \text{if there is a } k \text{ such that } Y_{1,k} = Y_{2,k} = 6, \text{ and} \\ 0 & \text{otherwise} \end{cases}$$

That is, $Z = 1$ if there is at least one double-six in the 24 throws. The Chevalier de Méré was inquiring about the the value of the event

[50] Self appointed!

[51] Smith, D. A., 1929. *A Source Book on Mathematics*. McGraw-Hill, New York; cited in Bulmer, 1967, p. 26.

[52] Working out the example analytically, there is a $\frac{35}{36}$ chance of *not* throwing double six with two six-sided dice, and so $\left(\frac{35}{36}\right)^{24}$ is the probability of *not* throwing at least one double six in 24 throws, and so

$$1 - \left(\frac{35}{36}\right)^{24} \approx 0.491$$

is the probability of throwing at least one double-six in 24 fair throws of a pair of six-sided dice.

probability $\Pr[Z = 1]$, i.e., the chance of winning by throwing at least one double six in 24 fair throws of a pair of dice.

We will introduce variables for the full range of simulated random variable values and simulation indexes in the following program.[53]

[53] The simulation indexes use parentheses rather than the traditional brackets and come first so that, e.g., the simulated value `y(m)` will consist of a 2×24 collection of values, matching the size of Y.

```
for (m in 1:M)
  for (k in 1:24)
    y(m)[1, k] = uniform_rng(1:6)
    y(m)[2, k] = uniform_rng(1:6)
    z(m)[k] = y(m)[1, k] + y(m)[2, k]
  success(m) = (sum(z(m) == 12) > 0)
print 'Pr[double-six in 24 throws] = ' sum(success) / M
```

Let's run that for $M = 100\,000$ simulations a few times and see what the estimated event probabilities look like.

```
 Pr[double-six in 24 throws] = 0.491
 Pr[double-six in 24 throws] = 0.490
 Pr[double-six in 24 throws] = 0.491
 Pr[double-six in 24 throws] = 0.491
 Pr[double-six in 24 throws] = 0.492
```

This shows the result to be around 0.49. The Chevalier de Méré should not bet that he'll roll at least one pair of sixes in 24 throws! To nail down the last digit, we could use 10 000 000 simulations rather than 100 000. As shown in the previous note, calculating the result analytically yields 0.491 to three decimal places, which is in agreement with the simulation-based estimates.

The Chevalier de Méré was reputedly perplexed by the difference between the chance of at least one double-six in 24 throws of two dice versus the chance of at least one six in 4 throws of a single die.[54]

[54] The probability of at least one six in four die rolls works out to

$$1 - \left(\frac{5}{6}\right)^4 \approx 0.518.$$

As noted above, the probability of at least one double six in 24 die rolls is ≈ 0.491.

Sampling without replacement

Drawing from a deck of cards is typically done without replacement. Once a card is drawn, it may not be drawn again. A traditional deck of playing cards consists of 52 cards, each marked with a value

$$2, 3, \ldots, 10, \mathrm{J}, \mathrm{Q}, \mathrm{K}, \mathrm{A}$$

and a suit from

$$\clubsuit, \diamondsuit, \heartsuit, \spadesuit$$

The lettered values are called jack, queen, king, and ace, and the suits are called clubs, diamonds, hearts, and spades. Traditionally

the diamonds and hearts are colored red and the clubs and spades colored black (here they are indicated by unfilled vs. filled shading).

A hand of cards consists of some number of cards drawn from a deck. When cards are drawn from a deck, they are not replaced. This is called sampling without replacement. Thus a hand of cards can contain at most 52 cards, because after 52 cards are drawn, there are none left.[55] Drawing without replacement also affects probabilities of particular hands.

[55] In some games, multiple decks are often used.

For example, drawing two cards from a fresh deck, the chance of getting two aces is not $\left(\frac{4}{52}\right)^2$, but rather $\left(\frac{4}{52} \times \frac{3}{51}\right) \approx 0.0045$. The chance of drawing an ace on the first draw is $\frac{4}{52}$ because there are 4 aces among the 52 cards and each card is assumed to be equally likely to be drawn from any deck. But after the first ace is drawn, there are only 51 cards remaining, and among those, only 3 aces. So the chance of the second card being an ace is only $\frac{3}{51}$.

We can verify that with a quick simulation.

```
total = 0
for (m in 1:M)
  y <- draw_cards(2)
  if (is_ace(y[1]) && is_ace(y[2]))
    total += 1
print 'Pr[draw 2 aces] = ' total / M
```

Let's run that with $M = 10\,000$, a few

```
 Pr[draw 2 aces] = 0.0049
 Pr[draw 2 aces] = 0.0049
 Pr[draw 2 aces] = 0.0058
 Pr[draw 2 aces] = 0.0044
 Pr[draw 2 aces] = 0.0039
 Pr[draw 2 aces] = 0.0051
 Pr[draw 2 aces] = 0.0039
 Pr[draw 2 aces] = 0.0044
```

Curiously, we are now not getting a single digit of accuracy, even with 10 000 draws. What happened?

A fundamental problem with accuracy of simulation-based estimates is that rare events are hard to estimate with random draws. If the event of drawing two aces only has a 0.45% chance (roughly 1 in 200) of occurring, we need a lot of simulation events to see it often enough to get a good estimate of even that first digit. With 10 000 draws, the number of two-ace draws we expect to see is about 50 if they occur at roughly a 1 in 200 hands rate. We know from prior experience that estimating a number with only 50 draws is not going

to be very accurate. So what we need to do is increase the number of draws. Let's run that again with $M = 1\,000\,000$ draws.

```
Pr[draw 2 aces] = 0.0046
Pr[draw 2 aces] = 0.0045
Pr[draw 2 aces] = 0.0044
Pr[draw 2 aces] = 0.0045
```

Now with an expected 5 000 occurrences of a two-ace hand, we have a much better handle on the relative accuracy, having nailed down at least the first digit and gotten close with the second digit.

Error versus relative error

Suppose we have an estimate $\hat{y}$ for a quantity y. One natural way to measure the accuracy of the estimate is to consider its *error*,

$$\text{err} = \hat{y} - y.$$

If the estimate is too high, the error will be positive, and if the estimate is too low, the error will be negative. The problem with this standard notion of error arises when the estimand y is very small or very large.

Now consider an estimand of $y = 0.01$ and an estimate of $\hat{y} = 0.015$. The error is just $y - \hat{y} = 0.005$\$, which looks small. But compared to the magnitude of y, which is only 0.01, the error is relatively large.

The *relative error* of an estimate $\hat{y}$ of a quantity y can be defined relative to the scale of the estimand as

$$\text{rel\_err} = \frac{\hat{y} - y}{|y|}.$$

This delivers results that are scaled in units of of y. The relative error for our estimate $\hat{y} = 0.015$ for an estimand $y = 0.01$ has relative error of

$$\frac{0.015 - 0.01}{|0.01|} = 0.5.$$

That's a 50% relative error, which now looks quite large compared to the 0.005 error.[56]

If the sign of error doesn't matter, errors are often reported as absolute values, i.e., as *absolute error* and *absolute relative error*.

The central limit theorem provides guarantees only about the error; to calculate its implications for relative error, the true value of the estimand must be known.

[56] An estimate of $\hat{y} = 0.015$ has an error of -0.005 and a relative error of -0.5, or 50% too low.

The Earl of Yarborough's wager

If an event has a probability of θ, the *odds* of it happening are given by the function

$$\text{odds}(\theta) = \frac{\theta}{1-\theta}.$$

For example, if there is a 25% chance of an event happening, the odds of it happening are $\frac{0.25}{1-0.25} = \frac{1}{3}$. In other words, it's three times as probable that the event does not occur than that it occurs. Odds are written as $1:3$ and pronounced "one to three" rather than being written as $\frac{1}{3}$ and pronounced "one in three".[57]

The Earl of Yarborough[58] reputedly laid a thousand to one odds against drawing a 13-card whist hand that contained no card higher than a 9.[59] There is a total of 32 cards that are 9 or lower (and hence 20 cards 10 or higher). We can use the same argument to calculate the Earl's odds of drawing his eponymous hand at

$$\begin{aligned}\Pr\left[\text{draw a Yarborough}\right] &= \tfrac{32}{52} \times \tfrac{31}{51} \times \cdots \times \tfrac{20}{40} \\ &= \textstyle\prod_{n=0}^{12} \frac{32-n}{52-n} \\ &\approx 0.00055\end{aligned}$$

The n in the second line represents the number of cards drawn previously. The true odds are roughly one in 2000, or rounded to the nearest integer

$$1:1817 \approx \frac{0.00055}{1-0.00055}$$

Rounded to the nearest integer, the odds are 1817:1 against, so the Earl should expect to profit from his bet.[60]

[57] When reporting odds, it is common to report the odds as "three to one *against*" for an event with a 25% probability.

[58] Many early developments in probability were bankrolled by gambling aristocrats.

[59] Bulmer, 1967, p. 26.

[60] Modulo the fact that one rare event might ruin him—these market-making schemes require a large bankroll and many repetitions to avoid ruin.

Binomial and repeated binary trials

Suppose we have a sequence of random variables, $V_1, \ldots, V_N$, each with a Bernoulli distribution $V_n \sim \text{Bernoulli}(\theta)$. That is, each V_n takes on the value 1 with a probabilty of θ.

We can think of $V_1, \ldots, V_N$ as N repeated binary trials, each with a θ chance of success.[61] That is, each V_n is a completely independent trial and each trial has a θ chance of success. By independent, we mean that the success of Y_n does not depend on $Y_{n'}$ if $n \neq n'$.

What can we say about the number of successes in N trials? Let

$$\begin{aligned}Y &= V_1 + \cdots + V_N \\ &= \textstyle\sum_{n=1}^{N} V_n,\end{aligned}$$

[61] The term "success" is the conventional name for the result 1 in an abstract binary trial, with result 0 being "failure".

and the question reduces to what we can say about the random variable Y. Repeated binary trials come up so often that the distribution of Y has a name, the *binomial distribution*. Pascal figured out that for any number of trials $N \geq 0$, chance of success $\theta \in [0, 1]$, the probability of a total number of successes $y \in 0 : N$ is

$$p_Y(y) = \text{Binomial}(y \mid N, \theta),$$

where[62]

$$\text{Binomial}(y \mid N, \theta) \ = \ \binom{N}{y} \times \theta^y \times (1 - \theta)^{N-y}.$$

[62] The value $\binom{N}{y}$ is called the *binomial coefficient* due to its use here, and defined by

$$\binom{N}{y} = \frac{N!}{(N-y)! \times y!}.$$

The value of the *factorial* $m!$ for $m > 0$ is

$$m! = m \times (m-1) \times (m-2) \times \cdots 1.$$

The recursive definition has base case $0! = 1$ and inductive case $(n+1)! = n \times n!$. The postfix factorial operator binds more tightly than multiplication, so this resolves as $n \times (n!)$.

Variance of the Bernoulli and binomial

If $Y \sim \text{Bernoulli}(\theta)$, then

$$\begin{aligned} \mathbb{E}[Y] &= \textstyle\sum_{y \in 0:1} p_Y(y) \times y \\ &= \text{Bernoulli}(0 \mid \theta) \times 0 + \text{Bernoulli}(1 \mid \theta) \times 1 \\ &= (1 - \theta) \times 0 + \theta \times 1 \\ &= \theta. \end{aligned}$$

Plugging this into the formula for variance yields

$$\begin{aligned} \text{var}[Y] &= \mathbb{E}\left[(Y - \mathbb{E}[Y])^2\right] \\ &= \textstyle\sum_{y \in 0:1} p_Y(y) \times (y - \theta)^2 \\ &= (1 - \theta) \times (0 - \theta)^2 + \theta \times (1 - \theta)^2 \\ &= (1 - \theta) \times \theta^2 + \theta \times (1 - 2 \times \theta + \theta^2) \\ &= \theta^2 - \theta^3 + \theta - 2 \times \theta^2 + \theta^3 \\ &= \theta - \theta^2 \\ &= \theta \times (1 - \theta). \end{aligned}$$

The multinomial distribution

The binomial distribution is for repeated binary trials—the multinomial distribution extends the same idea to repeated categorical trials. Just as a multiple coin tosses can be represented by a binomial distribution, multiple die rolls can be represented by multinomial distributions.

Counts and the Poisson distribution

Consider two binomial random variables,

$$Y \sim \text{binomial}(N, \theta),$$

and

$$Z \sim \text{binomial}(2 \times N, \frac{1}{2}\theta).$$

The variable Z has a maximum value that is twice as large of that of Y, yet it has the same expectation,

$$\mathbb{E}[Y] = \mathbb{E}[Z] = N \times \theta.$$

Poisson is the limit of the binomial in the sense that

$$\text{Poisson}(y \mid \lambda) \;=\; \lim_{N \to \infty} \text{Binomial}(y \mid N, \frac{1}{N}\lambda)$$

We can plot this out in a series of binomials: binomial(N, lambda / N) for N in ceil(lambda), *= sqrt(10) ...

Expectations and Variance

Almost all quantities of interest in statistics, ranging from parameter estimates to event probabilities and forecasts can be expressed as expectations of random variables. Expectation ties directly to simulation because expectations are computed as averages of samples of those random variables.

Variance is a measure of the variation of a random variable. It's also defined as an expectation. Curiously, it is on the quadratic scale—it's defined in terms of squared differences from expected values. This chapter will show why this is the natural measure of variation and how it relates to expectations.

The expectation of a random variable

Suppose we have a discrete random variable Y. Its *expectation* is defined as its average value, which is a weighted average of its values, where the weights are the probabilities.

$$\mathbb{E}[Y] \ = \ \sum_{y \in Y} y \times p_Y(y).$$

The notation $y \in Y$ is meant to indicate the summation is over all possible values y that the random variable Y may take on.

For example, if Y is the result of a fair coin flip, then

$$\mathbb{E}\,[Y] \ = \ 0 \times \frac{1}{2} + 1 \times \frac{1}{2} \ = \ \frac{1}{2}.$$

Now suppose Y is the result of a fair six-sided die roll. The expected value of Y is

$$\mathbb{E}\,[Y] \ = \ 1 \times \frac{1}{6} + 2 \times \frac{1}{6} + \cdots + 6 \times \frac{1}{6} \ = \ \frac{21}{6} \ = \ 3.5.$$

Now suppose U is the result of a fair twenty-sided die roll. Using the same formula as above, the expectation works out to $\frac{210}{20} = 10.5$.[63]

[63] In general, the expectation of a die roll is half the number of faces plus 0.5, because

$$\sum_{n=1}^{N} n \times \frac{1}{N} \ = \ \frac{1}{N} \sum_{n=1}^{N} n \ = \ \frac{1}{N} \frac{N \times (N+1)}{2} \ = \frac{N+1}{2}.$$

Expectations as simulation averages

We have been computing expectations all along for event probabilities. Expectations are the natural calculation for simulations, because

$$\mathbb{E}\,[Y] \;=\; \lim_{M\to\infty} \frac{1}{M} \sum_{m=1}^{M} y^{(m)},$$

where the $y^{(m)}$ are individual simulations of Y.

For any finite M, we get an estimate of the expectation defined as

$$\mathbb{E}\,[Y] \;=\; \frac{1}{M} \sum_{m=1}^{M} y^{(m)}.$$

Event probabilities as expectations of indicators

The event probability estimates from sampling can be viewed as calculating the expectation of the value of an indicator function. For example,[64]

$$\begin{aligned} \text{Prob}[Y = 1] \;&=\; \mathbb{E}\,[\mathrm{I}\,[Y = 1]] \\ &\approx\; \frac{1}{M} \sum_{m=1}^{M} \mathrm{I}\Big[y^{(m)} = 1\Big]. \end{aligned}$$

[64] The expectation is implicitly the expectation of a new random variable defined as $Z = \mathrm{I}[Y = 1]$.

The indicator function converts an event to a numerical 1 or 0, then the expectation does the averaging. The second line provides an estimate of the expectation based on a finite number of simulations of the random variable $y^{(m)}$ used to define the event.

The variance of a random variable

Variance is a measure of how much a random variable can be expected to vary around its expected value. For example, consider a random variable Y representing a fair throw of a six-sided die. The expected value is $\mathbb{E}[Y] = 3.5$, but the result may vary between 1 and 6.

Variance measures the expected square difference between a random variable and its expected value.

$$\text{var}[Y] \;=\; \mathbb{E}\Big[(Y - \mathbb{E}[Y])^2\Big].$$

The nested expectation, $\mathbb{E}[Y]$, is just the expectation of Y, and as such it's a constant. The expectation operator $\mathbb{E}[\,\cdot\,]$ captures its random variables. Thus we could have written this as

$$\mathbb{E}\Big[(Y - c)^2\Big],$$

where the constant $c = \mathbb{E}[Y]$.

In our example of the six-sided die, the expectation is the average roll, $\mathrm{E}[Y] = 3.5$. As is often the case with averages of discrete random variables, 3.5 is not itself a possible value of the variable. That is, it's not possible to roll a 3.5 on a six-sided die.

We can continue working out the expectation notation for our example, expanding

$$\begin{aligned}
\mathbb{E}\left[(Y-3.5)^2\right] &= \textstyle\sum_{y \in Y} p_Y(y) \times (y-3.5)^2 \\
&= \textstyle\sum_{y \in 1:6} \frac{1}{6} \times (y-3.5)^2 \\
&= \tfrac{1}{6}\left((1-3.5)^2 + (2-3.5)^2 + (3-3.5)^2 + (4-3.5)^2 + (5-3.5)^2 + (6-3.5)^2\right) \\
&= \tfrac{1}{6}\left(6.25 + 2.25 + 0.25 + 0.25 + 2.25 + 6.25\right) \\
&\approx 2.92.
\end{aligned}$$

We can, of course, use simulation to calculate variances.

```
sum_sq_diffs = 0
for (m in 1:M) {
  y = uniform_rng(1:6)
  sum_sq_diffs += (y - 3.5)^2
print 'estimated var[Y] = ' sum_sq_diffs / M
```

And running that for $M = 1\,000\,000$ iterations gives us

```
 estimated var[Y] = 2.91
```

We see that the extreme values have an outsized influence on the sum of squared errors. That's because we're dealing with squared error, which reduces small errors (e.g., $0.5^2 = 0.25$) and magnifies large errors (e.g., $5^2 = 25$). We can see a plot of the squared differences from the mean that wold be involved in calculating the variance of a 20-sided die.

Why squared error?

Why are we calculating average squared difference from the expectation rather than, say, average absolute difference? The answer has to do with the nature of averages. It turns out the average is the x that minimizes the sum of squared differences,

$$\mathrm{sqe}(x, y) = \sum_{n \in 1:N} (y_n - x)^2,$$

or equivalently, minimizes the average (or mean) squared difference,

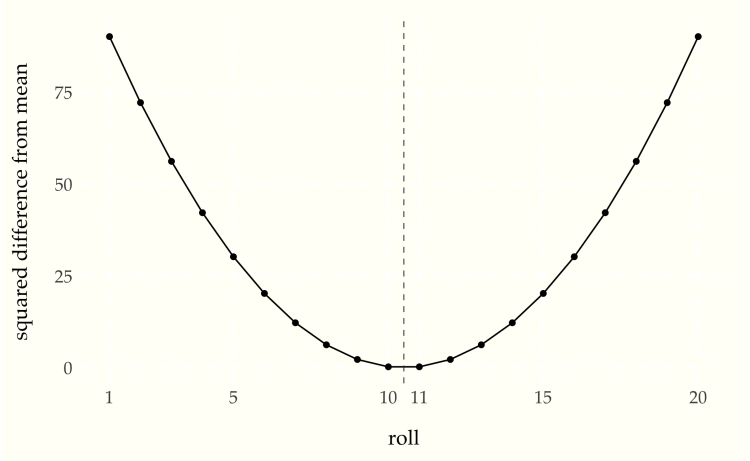


Figure 15: Squared differences from the mean of 10.5 for a 20-sided die roll. The mean is indicated with a dashed vertical line.

$$\text{msqe}(x, y) = \frac{1}{N} \sum_{n \in 1:N} (y_n - x)^2.$$

That is,

$$\text{avg}(y) \ = \ \frac{1}{N} \sum_{n=1}^{N} y_n \ = \ \text{argmin}_x \ \text{msqe}(x, y)$$

The notation $\text{argmin}_x \, f(x)$ is meant to return the x at which $f(x)$ takes on its minimum value. For example, $\text{argmin}_u (u - 1)^2 + 7 = 1$, whereas the minimum value is given by $\min_u (u - 1)^2 + 7 = 7$.

Averages themselves are important because expectations are defined in terms of averages, and event probabilities are defined in terms of expectations. Furthermore, the central limit theorem governs the convergence of averages as estimates of quantities of interest in terms of squared error.

Standard deviations for scale correction

The difficulty in managing variance is that it has different units than the variable it is measuring the variation of. For example, if Y reprents the fill of a bottle with units in milliliters, then $\text{var}[Y]$ has units of squared milliliters. Squared milliliters are not natural. As we saw even with the calculation for a single six-sided die, values of 1 and 6 had 6.25 squared difference from the expected value of 3.5, whereas 3 and 4 had only 0.25 squared difference from the expected value. Small values (below one) get smaller ($0.5^2 = 0.25$) and large values are amplified ($2.5^2 = 6.25$).

Instead of dealing with variance, with its quadratic scale and mismatched units, it is common to convert back at the end to ordinary

units by taking a square root. The *standard deviation* of a random variable is defined as the square root of its variance,

$$\mathrm{sd}[Y] = \sqrt{\mathrm{var}\,[Y]}.$$

While it is not known why Karl Pearson coined the term "standard deviation",[65] the key consideration here is that $\mathrm{sd}[Y]$ has the same units as the variable Y itself and is thus much easier to interpret.

[65] Pearson, K., 1893. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London, Series A*. Nov 16: 329–333, which on page 330 coins the term stating only, "standard deviations—a term used in the memoir for what corresponds in frequency curves to the error of mean square."

Joint, Marginal, and Conditional Probabilities

Diagnostic accuracy

What if I told you that you just tested positive on a diagnostic test with 95% accuracy, but that there's only a 16% chance that you have the disease in question? This running example will explain how this can happen.[66]

Suppose the random variable $Z \in 0 : 1$ represents whether a particular subject has a specific disease.[67] Now suppose there is a diagnostic test for whether a subject has the disease. Further suppose the random variabel $Y \in 0 : 1$ represents the result of applying the test to the same subject.[68]

Now suppose the test is 95% accurate in the sense that

- if the test is administered to a subject with the disease (i.e., $Z = 1$), there's a 95% chance the test result will be positive (i.e., $Y = 1$), and
- if the test is administered to a subject without the disease (i.e., $Z = 0$), there's a 95% chance the test result will be negative (i.e., $Y = 0$).

We're going to pause the example while we introduce the probability notation required to talk about it more precisely.

[66] The example diagnostic accuracies and disease prevalences we will use for simulation are not unrealistic—this kind of thing happens all the time in practice, which is why there are often batteries of follow-up tests after preliminary positive test results.

[67] $Z = 1$ if the subject has the disease and $Z = 0$ if not.

[68] $Y = 1$ if the test result is positive and $Y = 0$ if it's negative.

Conditional probability

Conditional probability notation expresses the diagnostic test's accuracy for people have the disease ($Z = 1$) as

$$\Pr[Y = 1 \mid Z = 1] = 0.95$$

and for people who don't have the disease ($Z = 0$) as

$$\Pr[Y = 0 \mid Z = 0] = 0.95.$$

We read $\Pr[\text{A} \mid \text{B}]$ as the *conditional probability* of event A given that we know event B occurred. Knowing that event B occurs often, but

not always, gives us information about the probability of A occurring.

The conditional probability function $\Pr[\,\cdot \mid \mathrm{B}]$ behaves just like the unconditional probability function $\Pr[\,\cdot\,]$. That is, it satisfies all of the laws of probability we have introduced for event probabilities. The difference is in the semantics—conditional probabilities are restricted to selecting a way the world can be that is consistent with the event B occurring.[69]

For example, the sum of exclusive and exhaustive probabilities must be one, and thus the diagnostic error probabilities are one minus the correct diagnosis probabilities,[70]

$$\Pr[Y = 0 \mid Z = 1] = 0.05$$

and

$$\Pr[Y = 1 \mid Z = 0] = 0.05.$$

[69] Formally, an event B is modeled as a set of ways the world can be, i.e., a subset $\mathrm{B} \subseteq \Omega$ of the sample space Ω. The conditional probability function $\Pr[\,\cdot \mid \mathrm{B}]$ can be interpreted as an ordinary probability distribution with sample space B instead of the original Ω.

[70] These error rates are often called the false positive rate and false negative rate, the positive and negative in this case being the test result and the false coming from not matching the true disease status.

Joint probability

Joint probability notation expresses the probability of multiple events happening. For instance, $\Pr[Y = 1, Z = 1]$ is the probability of both events, $Y = 1$ and $Z = 1$, occurring.

In general, if A and B are events,[71] we write $\Pr[\mathrm{A}, \mathrm{B}]$ for the event of both A and B occurring. Because joint probability is defined in terms of conjunction ("and" in English), the definition is symmetric,

$$\Pr[\mathrm{A}, \mathrm{B}] = \Pr[\mathrm{B}, \mathrm{A}].$$

[71] We use capital roman letters for event variables to distinguish them from random variables for which we use capital italic letters.

In the context of a joint probability $\Pr[\mathrm{A}, \mathrm{B}]$, the single event $\Pr[\mathrm{A}]$ is called a *marginal probability*.[72]

[72] Because of symmetry, $\Pr[\mathrm{B}]$ is also a marginal probability.

Joint probability is defined relative to conditional and marginal probabilities by

$$\Pr[\mathrm{A}, \mathrm{B}] \;=\; \Pr[\mathrm{A} \mid \mathrm{B}] \times \Pr[\mathrm{B}].$$

In words, the probability of events A and B occurring is the same as the probability of event B occurring times the probabilty of A occurring given that B occurs.

The relation between joint and conditional probability involves simple multiplication, which may be rearranged by dividing both sides by $\Pr[\mathrm{B}]$ to yield

$$\Pr[\mathrm{A} \mid \mathrm{B}] \;=\; \frac{\Pr[\mathrm{A}, \mathrm{B}]}{\Pr[\mathrm{B}]}.$$

Theoretically, it is simplest to take joint probability as the primitive so that this becomes the definition of conditional probability. In

practice, all that matters is the relation between conditional and joint probability.

Dividing by zero: not-a-number or infinity

Conditional probabilities are not well defined in the situation where $\Pr[B] = 0$. In such a situation, $\Pr[\mathrm{A}, \mathrm{B}] = 0$, because B has probability zero.

In practice, if we try to evaluate such a condition using standard computer floating-point arithmetic, we wind up dividing zero by zero.

```
Pr_A_and_B = 0.0
Pr_B = 0.0
Pr_A_given_B = Pr_A_and_B / Pr_B
print 'Pr[A | B] = ' Pr_A_given_B
```

Running that, we get

```
 Pr[A | B] = NaN
```

The value `NaN` indicates what is known as the *not-a-number* value.[73] This is a special floating-point value that is different from all other values and used to indicate a domain error on an operation. Other examples that will return not-a-number values include `log(-1)`.

[73] There are technically two types of not-a-number values in the IEEE 754 standard, of which we will only consider the non-signaling version.

If we instead divide a positive or negative number by zero,

```
print '1.0 / 0.0 = ' 1.0 / 0.0
print '-3.2 / 0.0 = ' -3.2 / 0.0
```

we get

```
 1.0 / 0.0 = Inf
 -3.2 / 0.0 = -Inf
```

These values denote positive infinity (∞) and negative infinity ($-\infty$), respectively. Like not-a-number, these are special reserved floating-point values with the expected interactions with other numbers.[74]

[74] For example, adding a finite number and an infinity yields the infinity and subtracting two infinities produces a not-a-number value. For details, see 754-2008—IEEE standard for floating-point arithmetic.

Simulating the diagnostic example

What we don't know so far in the running example is what the probability is of a subject having the disease or not. This probability, known as the *prevalence* of the disease, is often the main quantity of interest in an epidemiological study.

Let's suppose in our case that 1% of the population has the disease in question. That is, we assume

$$\Pr[Z = 1] = 0.01$$

Now we have enough information to run a simulation of all of the joint probabilities. We just follow the marginal and conditional probabilities. First, we generate Z, whether or not the subject has the disease, then we generate Y, the result of the test, conditional on the disease status Z.

```
for (m in 1:M)
  z[m] = bernoulli_rng(0.01)
  if (z[m] == 1)
    y[m] = bernoulli_rng(0.95)
  else
    y[m] = bernoulli_rng(0.05)
print 'estimated Pr[Y = 1] = ' sum(y) / M
print 'estimated Pr[Z = 1] = ' sum(z) / M
```

The program computes the marginals for Y and Z directly. This is straightforward because both Y and Z are simulated in every iteration (as `y[m]` and `z[m]` in the code). Marginalization using simulation requires no work whatsoever.[75]

[75] Marginalization can be tedious, impractical, or impossible to carry out analytically.

Let's run that with $M = 100\,000$ and see what we get.

```
estimated Pr[Y = 1] = 0.060
estimated Pr[Z = 1] = 0.010
```

We know that the marginal $\Pr[Z = 1]$ is 0.01, so the estimate is close to the true value for Z; we'll see below that it's also close to the true value of $\Pr[Y = 1]$.

We can also use the simulated values to estimate conditional probabilities. To estimate, we just follow the formula for the conditional distribution,

$$\Pr[A \mid B] = \frac{\Pr[A, B]}{\Pr[B]}.$$

Specifically, we count the number of draws in which both A and B occur, then divide by the number of draws in which the event B occurs.

```
for (m in 1:M)
  z[m] = bernoulli_rng(0.01)
  if (z[m] == 1)
    y[m] = bernoulli_rng(0.95)
  else
```

```
    y[m] = bernoulli_rng(0.05)
  y1z1[m] = (y[m] == 1 & z[m] == 1)
  y1z0[m] = (y[m] == 1 & z[m] == 0)

print 'estimated Pr[Y = 1 | Z = 1] = ' sum(y1z1) / sum(z == 1)
print 'estimated Pr[Y = 1 | Z = 0] = ' sum(y1z0) / sum(z == 0)
```

We set the indicator variable `y1z1[m]` to 1 if $Y = 1$ and $Z = 1$ in the m-th simulation; `y1z0` behaves similarly. The operator `&` is used for conjuncton (logical and) in the usual way.[76]

Recall that `z == 0` is an array where entry m is 1 if the condition holds, here $z^{(m)} = 0$.

The resulting etsimates with $M = 100\,000$ draws are pretty close to the true values,

```
estimated Pr[Y = 1 | Z = 1] = 0.952
estimated Pr[Y = 1 | Z = 0] = 0.051
```

The true values were stipulated as part of the example to be $\Pr[Y = 1 \mid Z = 1] = 0.95$ and $\Pr[Y = 1 \mid Z = 0] = 0.05$. Not surprisingly, knowing the value of Z (the disease status) provides quite a bit of information about the value of Y (the diagnostic test result).

Now let's do the same thing the other way around,and look at the probability of having the disease based on the test result, i.e., $\Pr[Z = 1 \mid Y = 1]$ and $\Pr[Z = 1 \mid Y = 0]$.[77]

```
estimated Pr[Z = 1 | Y = 1] = 0.159
estimated Pr[Z = 1 | Y = 0] = 0.001
```

Did we make a mistake in coding up our simulation? We estimated $\Pr[Z = 1 \mid Y = 1]$ at around 16%, which says that if the subject has a positive test result ($Y = 1$), there is only a 16% chance they have the disease ($Z = 1$). How can that be when the test is 95% accurate and simulated as such?

The answer hinges on the prevalence of the disease. We assumed only 1% of the population suffered from the disease, so that 99% of the people being tested were disease free. Among the disease free ($Z = 0$) that are tested, 5% of them have false positives ($Y = 1$). That's a lot of patients, around 5% times 99%, which is nearly 5% of the population. On the other hand, among the 1% of the population with the disease, almost all of them test positive. This means roughly five times as many disease-free subjects test positive for the disease as disease-carrying subjects test positive.

[76] The logical and operation is often written as `&&` in programming languages to distinguish it from bitwise and, which is conventionally written `&`.

[77] The program is just like the last one.

Analyzing the diagnostic example

The same thing can be done with algebra as we did in the previous section with simulations.[7][8] Now we can evaluate joint probabilities, e.g.,

$$\begin{aligned} \Pr[Y = 1, Z = 1] &= \Pr[Y = 1 \mid Z = 1] \times \Pr[Z = 1] \\ &= 0.95 \times 0.01 \\ &= 0.0095. \end{aligned}$$

[7][8] Computationally, precomputed algebra is a big win over simulations in terms of both compute time and accuracy. It may not be a win when the derivations get tricky and human time is taken into consideration.

Similarly, we can work out the probability the remaining probabilities in the same way, for example, the probability of a subject having the disase and getting a negative test result,

$$\begin{aligned} \Pr[Y = 0, Z = 1] &= \Pr[Y = 0 \mid Z = 1] \times \Pr[Z = 1] \\ &= 0.05 \times 0.01 \\ &= 0.0005. \end{aligned}$$

Doing the same thing for the disease-free patients completes a four-by-four table of probabilities,

Probability	$Y = 1$	$Y = 0$
$Z = 1$	0.0095	0.0005
$Z = 0$	0.0450	0.8500

For example, the top-left entry records the fact that $\Pr[Y = 1, Z = 1] = 0.0095$. The next entry to the right indicates that $\Pr[Y = 0, Z = 1] = 0.0005$.

The marginal probabilities (e.g., $\Pr[Y = 1]$) can be computed by summing the probabilities of all alternatives that lead to $Y = 1$, here $Z = 1$ and $Z = 0$

$$\Pr[Y = 1] = \Pr[Y = 1, Z = 1] + \Pr[Y = 1, Z = 0]$$

We can extend our two-by-two table by writing the sums in what would've been the margins of the original table above.

Probability	$Y = 1$	$Y = 0$	$Y = 1$ or $Y = 0$
$Z = 1$	0.0095	0.0005	*0.0100*
$Z = 0$	0.0450	0.8500	*0.9900*
$Z = 1$ or $Z = 0$	*0.0545*	*0.8505*	**1.0000**

Here's the same table with the symbolic values.

Probability	$Y = 1$	$Y = 0$	$Y = 1$ or $Y = 0$
$Z = 1$	$\Pr[Y = 1, Z = 1]$	$\Pr[Y = 0, Z = 1]$	$\Pr[Z = 1]$
$Z = 0$	$\Pr[Y = 1, Z = 0]$	$\Pr[Y = 0, Z = 0]$	$\Pr[Z = 0]$
$Z = 1$ or $Z = 0$	$\Pr[Y = 1]$	$\Pr[Y = 0]$	$\Pr[\,]$

For example, that $\Pr[Z = 1] = 0.01$ can be read off the top of the right margin column—it is the sum of the two table entries in the top row, $\Pr[Y = 1, Z = 1]$ and $\Pr[Y = 0, Z = 1]$.

In the same way, $\Pr[Y = 0] = 0.8505$ can be read off the right of the bottom margin row, being the sum of the entries in the right column, $\Pr[Y = 0, Z = 1]$ and $\Pr[Y = 0, Z = 0]$.

The extra headings define the table so that each entry is the probability of the event on the top row and on the left column. This is why it makes sense to record the grand sum of 1.00 in the bottom right of the table.

Joint and conditional distribution notation

Recall that the probability mass function $p_Y(y)$ for a discrete random variable Y is defined by

$$p_Y(y) = \Pr[Y = y].$$

As before, capital Y is the random variable, y is a potential value for Y, and $Y = y$ is the event that the random variable Y takes on value y.

The *joint probability mass function* for two discrete random variables Y and Z is defined by the joint probability,

$$p_{Y,Z}(y, z) = \Pr[Y = y, Z = z].$$

The notation follows the previous notation with Y, Z indicating that the first argument is the value of Y and the second that of Z.

Similarly, the conditional probablity mass function is defined by

$$p_{Y|Z}(y \mid z) = \Pr[Y = y \mid Z = z].$$

It can equivalently be defined as

$$p_{Y|Z}(y \mid z) \;=\; \frac{p_{Y,Z}(y, z)}{p_Z(z)}.$$

The notation again follows that of the conditional probability function through which the conditional probability mass function is defined.

Bayes's Rule

Bayes's rule relates the conditional probability $\Pr[\mathrm{A} \mid \mathrm{B}]$ for events A and B to the inverse conditional probability $\Pr[\mathrm{A} \mid \mathrm{B}]$ and the marginal probability $\Pr[\mathrm{B}]$. The rule requires a partition of the event

A into events $\mathrm{A}_1, \ldots, \mathrm{A}_K$, which are mutually exclusive and exhaust A. That is,

$$\Pr[\mathrm{A}_k, \mathrm{A}_{k'}] = 0 \text{ if } k \neq k',$$

and

$$\begin{aligned} \Pr[\mathrm{A}] &= \Pr[\mathrm{A}_1] + \cdots + \Pr[\mathrm{A}_K]. \\ &= \textstyle\sum_{k \in 1:K} \Pr[\mathrm{A}_k]. \end{aligned}$$

The basic rule of probability used to derive each line is noted to the right.

$$\begin{aligned} \Pr[\mathrm{A} \mid \mathrm{B}] &= \frac{\Pr[\mathrm{A}, \mathrm{B}]}{\Pr[\mathrm{B}]} && \text{[conditional definition]} \\ &= \frac{\Pr[\mathrm{B} \mid \mathrm{A}] \times \Pr[\mathrm{A}]}{\Pr[\mathrm{B}]} && \text{[joint definition]} \\ &= \frac{\Pr[\mathrm{B} \mid \mathrm{A}] \times \Pr[\mathrm{A}]}{\sum_{k \in 1:K} \Pr[\mathrm{B}, \mathrm{A}_k]} && \text{[marginalization]} \\ &= \frac{\Pr[\mathrm{B} \mid \mathrm{A}] \times \Pr[\mathrm{A}]}{\sum_{k \in 1:K} \Pr[\mathrm{B} \mid \mathrm{A}_k] \times \Pr[\mathrm{A}_k]}. && \text{[joint definition]} \end{aligned}$$

Letting A be the event $Y = y$, B be the event $Z = z$, and A_k be the event $Y = k$, Bayes's rule can be instantiated to

$$\Pr[Y = y \mid Z = z] = \frac{\Pr[Z = z \mid Y = y] \times \Pr[Y = y]}{\sum_{y' \in 1:K} \Pr[Z = z \mid Y = y'] \times \Pr[Y = y']}.$$

This allows us to express Bayes's rule in terms of probabilty mass functions as

$$p_{Y|Z}(y \mid z) = \frac{p_{Z|Y}(z \mid y) \times p_Y(y)}{\sum_{y' \in 1:K} p_{Z|Y}(z \mid y') \times p_Y(y')}.$$

Bayes's rule can be extended to infinite partitions of the event B, or in the probability mass function case, a variable Y taking on infinitely many possible values.

Fermat and the problem of points

Blaise Pascal and Pierre de Fermat studied the problem of how to divide the pot[79] of a game of chance that was interrupted before it was finished. As a simple example, Pascal and Fermat considered a game in which each turn a fair coin was flipped, and the first player would score a win a point if the result was heads and the second

[79] The *pot* is the total amount bet by both players.

player if the result was tails. The first player to score ten points wins the game.

Now suppose a game is interrupted after 15 flips, at a point where the first player has 8 points and the second player only 7. What is the probability of the first player winning the match were it to continue?

We can put that into probability notation by letting Y_n be the number of points for the first player after n turns.

$Y_{n,1}$ be the number of heads for player 1 after n flips, $Y_{n,2}$ be the same for player 2. Let Z be a binary random variable taking value 1 if the first player wins and 0 if the other player wins. Fermat managed to evaluate a formula like Fermat evaluated $\Pr[Z = 1 \mid Y_{n,1} = 8, Y_{n,2} = 7]$ by enumerating the possible game continuations and adding up the probabilities of the ones in which the first player wins.

We can solve Fermat and Pascal's problem by simulation. As usual, our estimate is just the proportion of the simulations in which the first player wins. The value of `pts` must be given as input—that is the starting point for simulating the completion of the game, assuming neither player yet has ten points.[80]

[80] For illustrative purposes only! In robust code, validation should produce diagnostic error messages for invalid inputs.

```
for (m in 1:M)
  while (pts[1] < 10 & pts[2] < 10)
    toss = uniform_01_rng()
    if (toss == 1) pts[1] += 1
    else pts[2] += 1
    if (pts[1] == 10) win[m] = 1
    else if (pts[2] == 10) win[m] = 0
print 'est. Pr[player 1 wins] = ' sum(win) / M
```

If the while-loop terminates because one player has ten points, then `wins[m]` must have been set in the previous value of the loop.[81]

[81] In general, programs should be double-checked (ideally by a third party) to make sure *invariants* like this one (i.e., `win[m]` is always set) actually hold. Test code goes a long way to ensuring robustness.

Let's run that a few times with $M = 100\,000$, starting with the `pts` set to `(8, 7)`, to simulate Fermat and Pascal's problem.

```
est. Pr[player 1 wins] = 0.688
est. Pr[player 1 wins] = 0.687
est. Pr[player 1 wins] = 0.688
est. Pr[player 1 wins] = 0.688
est. Pr[player 1 wins] = 0.690
```

This is very much in line with the result Fermat derived by brute force, namely $\frac{11}{16} \approx 0.688$.[82]

[82] There are at most four more turns required, which have a total of $2^4 = 16$ possible outcomes, HHHH, HHHT, HHTH, ..., TTTH, TTTT, of which 11 produce wins for the first player.

Independence of random variables

Informally, we say that a pair of random variables is independent if knowing about one variable does not provide any information about

the other. If X and Y are the variables in question, this property can be stated directly in terms of their probability mass functions as

$$p_X(x) = p_{X|Y}(x \mid y).$$

In practice, we use an equivalent definition. Random variables X and Y are said to be *independent* if

$$p_{X,Y}(x, y) = p_X(x) \times p_Y(y).$$

for all x and y.[83]

By way of example, we have been assuming that a fair dice throw involves the throw of two independent and fair dice. That is, if Y_1 is the first die and Y_2 is the second die, then Y_1 is independent of Y_2.

In the diagnostic testing example, the disease state Z and the test result Y are not independent.[84]. This can easily be verified because $p_{Y|Z}(y \mid z) \neq p_Y(y)$.

[83] This is equivalent to requiring the events $X \leq x$ and $Y \leq y$ to be independent for every x and y. Events A and B are said to be *independent* if $\Pr[\text{A}, \text{B}] = \Pr[\text{A}] \times \Pr[\text{B}]$.

[84] That would be a very poor test, indeed!

Independence of multiple random variables

It would be nice to be able to say that a set of random $Y_1, \ldots, Y_N$ was independent if each of its pairs of random variables was independent. We'd settle for being able to say that the joint probability factors into the product of marginals,

$$p_{Y_1,\ldots,Y_N}(y_1, \ldots, y_N) \ = \ p_{Y_1}(y1) \times \cdots \times p_{Y_N}(y_N).$$

But neither of these is enough.[85] For a set of random variables to be *independent*, the probability of each of its subsets must factor into the product of its marginals.[86]

[85] Analysis in general and probability theory in particular defeat simple definitions with nefarious edge cases.

[86] More precisely, $Y_1, \ldots, Y_N$ are *independent* if for every $M \leq N$ and permutation π of $1 : N$ (i.e., a bijection between $1 : N$ and itself), we have

$$p_{Y_{\pi(1)},\ldots,Y_{\pi(M)}}(u_1, \ldots, u_M) \ = \ p_{Y_{\pi(1)}}(u_1) \times \cdots \times p_{Y_{\pi(M)}}(u_M)$$

for all $u_1, \ldots, u_M$.

Conditional independence

Often, a pair of variables are not independent only because they both depend on a third variable. The random variables Y_1 and Y_2 are said to be *conditionally independent* given the variable Z if they are independent after conditioning,

$$p_{Y_1,Y_2|Z}(y_1, y_2 \mid z) \ = \ p_{Y_1|Z}(y_1 \mid z) \times p_{Y_2|Z}(y_2 \mid z).$$

Conditional expectations

The expectation $\mathbb{E}[Y]$ of a random variable Y is its average value (weighted by density or mass, depending on whether it is continuous or discrete). The conditional expectation $\mathbb{E}[Y \mid A]$ given some event A is defined to be the average value of Y conditioned on the event A,

$$\mathbb{E}[Y \mid A] \;=\; \int_Y y \times p_{Y|A}(y \mid A)\,\mathrm{d}y,$$

where $p_{Y|A}$ is the density of Y conditioned on event A occurring. This conditional density $p_{Y|A}$ is defined just like the ordinary density p_Y only with the conditional cumulative distribution function $F_{Y|A}$ instead of the standard cumulative distribution function F_Y,

$$p_{Y|A}(y \mid A) \;=\; \frac{\mathrm{d}}{\mathrm{d}y} F_{Y|A}(y \mid A).$$

The conditional cumulative distribution function $F_{Y|A}$ is, in turn, defined by the conditioning on the event probability,

$$F_{Y|A}(y \mid A) \;=\; \Pr[Y < y \mid A].$$

This also works to condition on zero probability events, such as $\Theta = \theta$, by taking the usual definition of conditional density,

$$\mathbb{E}[Y \mid \Theta = \theta] \;=\; \int_Y y \times p_{Y|\Theta}(y \mid \theta)\,\mathrm{d}y.$$

When using discrete variables, integrals are replaced with sums.

Independent and identically distributed variables

If the variables $Y_1, \ldots, Y_N$ are not only independent, but also have the same probability mass functions (i.e., $p_{Y_n} = p_{Y_m}$ for all $m, n \in 1 : N$), we say that they are *independent and identically distributed*, or "iid" for short. Many of our statistical models, such as linear regression, will make the assumption that observations are conditionally independent and identically distributed.

Finite-State Markov Chains

Random processes

A finite sequence of random variables is said to be a random vector. An infinite sequence

$$Y = Y_1, Y_2, \ldots$$

of random variables is said to be a *random process*.[87] A trivial example is a sequence of independent Bernoulli trials in which each Y_t is drawn independently according to $Y_t \sim \text{bernoulli}(\theta)$. A sequence of independent Bernoulli trials is called a *Bernoulli process.*

In this chapter, we will restrict attention to processes Y whose elements take on values $Y_t \in 0 : N$ or $Y_t \in 1 : N$ for some fixed N.[88] The Bernoulli process is finite in this sense because each Y_t takes on boolean values, so that $Y_t \in 0 : 1$.

[87] We consider only discrete random processes where the set of indexes is the counting numbers $1, 2, 3, \ldots$. Nevertheless, the set of indexes is infinite, so much of the approach to finite vectors has to be reworked.

[88] The choice of starting at 0 or 1 is a convention that varies by distribution. For example, Bernoulli and binomial variates may take on value zero, but categorical values take on values in $1 : N$.

Finite-State Markov chains

A random process Y is said to be a *Markov chain* if each element is generated conditioned on only the previous element, so that

$$p_{Y_{t+1}|Y_1,\ldots,Y_t}(y_{t+1} \mid y_1, \ldots, y_t) \;=\; p_{Y_{t+1}|Y_t}(y_{t+1} \mid y_t)$$

holds for all $y_1, \ldots, y_{t+1}$. In this chapter, we only consider Markov chains in which the Y_t are finite random variables taking on values $Y_t \in 0 : N$ or $Y_t \in 1 : N$, the range depending on the type of variable.[89]

The Bernoulli process discussed in the previous section is a trivial example of a finite Markov chain. Each value is generated independently, so that for all $y_1, \ldots, y_{t+1}$, we have

$$\begin{aligned} p_{Y_{t+1}|Y_1,\ldots,Y_t}(y_{t+1} \mid y_1, \ldots, y_t) &= p_{Y_{t+1}|Y_t}(y_{t+1} \mid y_t) \\ &= \text{bernoulli}(y_{t+1} \mid \theta). \end{aligned}$$

[89] We generalize in two later chapters, first to Markov chains taking on countably infinite values and then to ones with continous values.

Fish in the stream

Suppose a person is ice fishing for perch and pike, and notes that if they catch a perch, it is 95% likely that the next fish they catch is a perch, whereas if they catch a pike, it is 20% likely the next fish they catch is a pike.[90] We'll treat the sequence of fish types as a random process $Y = Y_1, Y_2, \ldots$ with values

$$Y_t \ = \ \begin{cases} 1 & \text{if fish } t \text{ is a pike, and} \\ 2 & \text{if fish } t \text{ is a perch.} \end{cases}$$

[90] This is a thinly reskinned version of the classic exercise involving cars and trucks from Ross, S.M., 2014. *Introduction to Probability Models.* Tenth edition. Academic Press. Exercise 30, page 279.

The sequence Y forms a Markov chain with transition probabilities

$$\begin{aligned} \Pr[Y_{t+1} = 1 \mid Y_t = 1] &= 0.20 \\ \Pr[Y_{t+1} = 1 \mid Y_t = 2] &= 0.05 \end{aligned}$$

The easiest way to visual a Markov chain with only a few states is as a state transition diagram. In the case of the pike and perch, the transition diagram is as follows.

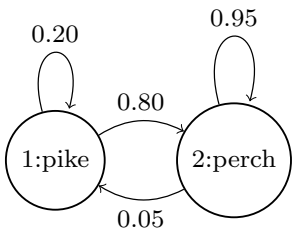


Figure 16: State diagram for finite Markov chain generating sequences of fishes. The last fish observed determines the current state and the arrows indicate transition probabilities to the next fish observed.

Like all such transition graphs, the probabilities on the edges going out of a node must sum to one.

Let's simulate some fishing. The approach is to generate the type of each fish in the sequence, then report the overall proportion of pike.[91] We will start with a random fish drawn according to bernoulli(1/2).

[91] With some sleight of hand here for compatiblity with Bernoulli variates and to facilitate computing proportions, we have recoded perch as having value 0 rather than 2.

```
y[1] = bernoulli_rng(0.5)
for (t in 2:T)
  y[t] = bernoulli_rng(y[t - 1] = 1 ? 0.2 : 0.05)
print 'simulated proportion of pike = ' sum(y) / M
```

Now let's assume the fish are really running, and run a few simulated chains until $T = 10\,000$.

```
 simulated proportion of pike =   NA
 simulated proportion of pike =   NA
 simulated proportion of pike =   NA
 simulated proportion of pike =   NA
 simulated proportion of pike =   NA
```

The proportion of pike is roughly 0.06.

Ehrenfest's Urns

Suppose we have two urns, with a total of N balls distributed between them. At each time step, a ball is chosen uniformly at random from among the balls in both urns and moved to the other urn.[92]

[92] This model was originally introduced as an example of entropy and equilibrium in P. Ehrenfest and T. Ehrenfest. 1906. Über eine Aufgabe aus der Wahrscheinlichkeitsrechnung, die mit der kinetischen Deutung der Entropievermehrung zusammenhängt. *Mathematisch-Naturwissenschaftliche Blätter* No. 11 and 12.

The process defines a Markov chain Y where transitions are governed by

$$p_{Y_{t+1}|Y_t}(y_{t+1} \mid y_t) = \begin{cases} \frac{y_t}{N} & \text{if } y_{t+1} = y_t - 1, \text{ and} \\ 1 - \frac{y_t}{N} & \text{if } y_{t+1} = y_t + 1. \end{cases}$$

The transition probabilities make sure that the value of Y_t remains between 0 and N. For example,

$$\Pr[Y_{t+1} = 1 \mid Y_t = 0] = 1$$

because $1 - \frac{y_t}{N} = 1$. Similarly, if $Y_t = N$, then $Y_{t+1} = N - 1$.

What happens to the distibution of Y_t long term? It's easy to compute by simulation of a single long chain:[93]

[93] We've used a function borrowed from R here called `table`, defined by

$$\text{table}(y, A, B)[n] = \sum_{t=1}^{T} \text{I}[y_t = n]$$

for $n \in A : B$. For example, if

$$y = (0, 1, 2, 1, 1, 3, 2, 2, 1),$$

then

$$\text{table}(y, 0, 4) = (1, 4, 3, 1, 0),$$

because there is one 0, four 1s, three 2s, a single 3, and no 4s among the values of y.

```
y[1] = floor(N / 2)
for (t in 2:T)
  z[t] = bernoulli_rng(y[t - 1] / N)
  y[t] = y[t - 1] + (z[t] ? -1 : +1)
p_Y_t_hat = table(y, 0, N) / T
```

Let's run that with $N = 10$ and $T = 100\,000$ and display the results as a bar plot.

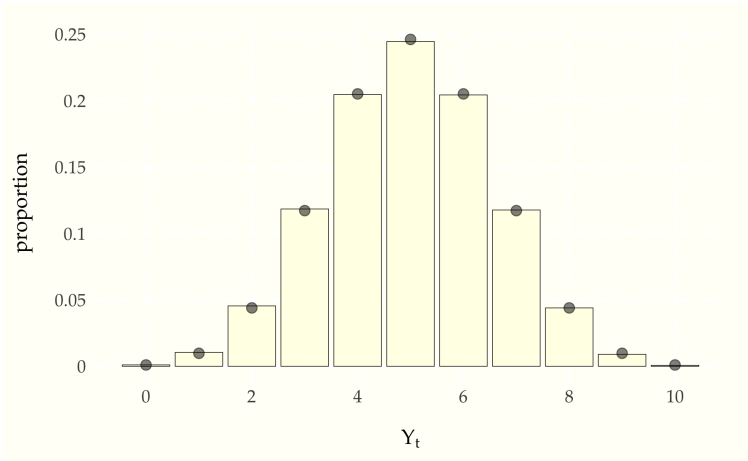


Figure 17: Long-term distribution of number of balls in the first urn of the Ehrenfest model in which N balls are distributed between two urns, then at each time step, a ball is chosen uniformly at random move to the other urn. The simulation is based on total of $T = 100\,000$ steps with $N = 10$ balls, starting with 5 balls in the first urn. The points on the top of the bars are positioned at the mass defined by the binomial distribution, $\text{binomial}(Y_t \mid 10, 0.5)$.

The distribution of Y_t values is the binomial distribution, as shown by the agreement between the points (the binomial probability mass function) and the bars (the empirical proportion Y_t spent in each state).[94]

[94] In the Markov chain Monte Carlo chapter later in the book, we will see how to construct a Markov chain whose long-term frequency distribution matches any given target distribution.

Page Rank and the random surfer

Pagerank,[95] the innovation behind the original Google search engine ranking system, can be modeled in terms of a random web surfer whose behavior determines a Markov chain. The web is modeled as a set of pages, each of which has a set of outgoing links to other pages. When viewing a particular page, our random surfer chooses the next page to visit by

[95] Page, L., Brin, S., Motwani, R. and Winograd, T., 1999. The PageRank citation ranking: Bringing order to the web. Stanford InfoLab Technical Report. Section 2.5 Random Surfer Model.

- if the current page has outgoing links, then with probability λ, choose the next page uniformly at random among the outgoing links,
- otherwise (with probability $1 - \lambda$), choose the next page to visit uniformly at random among all web pages.

Translating this into the language of random variables, let $Y = Y_1, Y_2, \ldots$ be the sequence of web pages visited. Our goal now is to define the transition function probabilistically so that we may simulate the random surfer. Let $L_i \subseteq 1 : N$ be the set of outgoing links from page i; each page may have any number of outgoing links from o to N.

The process Y is most easily described in terms of an auxiliary process $Z = Z_1, Z_2, \ldots$ where Z_t represents

the decision whether to jump to a link from the current page. We define Z by setting $Z_t = 0$ if the page Y_t has no outgoing links, and otherwise setting

$$Z_t \sim \text{bernoulli}(\lambda).$$

If $Z_t = 1$, we can generate Y_{t+1} uniformly from the links L_{Y_t} from page Y_t,

$$Y_{t+1} \sim \text{uniform}\left(L_{Y_t}\right).$$

If $Z_t = 0$, we simply choose a web page uniformly at random from among all N pages,

$$Y_{t+1} \sim \text{uniform}(1 : N).$$

This sequence is easy to simulate with `L[n]` denoting the outgoing links from page `n`. We start from a page `y[1]` chosen uniformly at

random among all the pages. Then we just simulate subsequent pages according to the process described above.

```
y[1] <- uniform_rng(1:N)
for (t in 2:T)
  last_page = y[t - 1]
  out_links = L[last_page]
  z[t] <- empty(out_links) ? 0 : bernoulli_rng(lambda)
  y[t] <- uniform(z[t] ? out_links : (1:N))
```

Suppose we have the following graph.

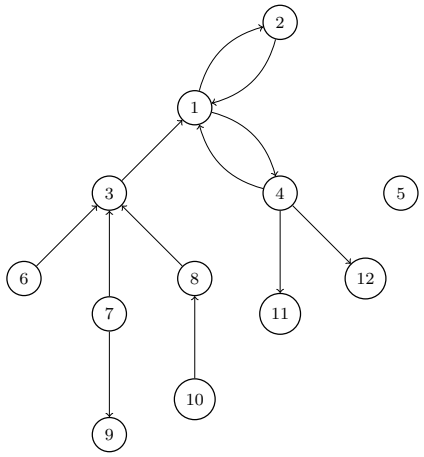


Figure 18: A simplified web. Each node represents a web page and each edge is a directed link from one page to another web page.

We can simulate $T = 100\,000$ page visits using the algorithm shown above and display the proportion of time spent on each page.

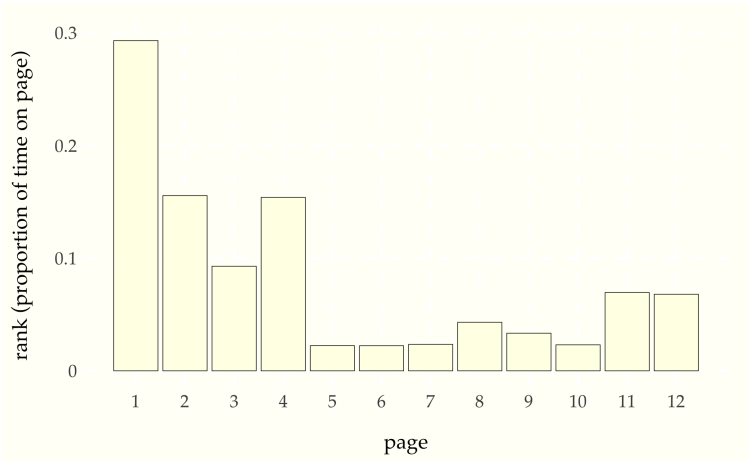


Figure 19: Proportion of time spent on each page by a random surfer taking $T = 100\,000$ page views starting from a random page with a web structured as in the previous diagram.

Page 1 is the most central hub. Pages 5, 6, 7, and 10 have no links coming into them and can only be visited by random chance, so all should have the same chance of being visited by the random surfer.

Pages 11 and 12 are symmetric, and indeed have the same probability. There is a slight difference between the views of page 9 and 10 in that it possible ot get to 9 from 7, but 10 is only visited by chance.

Stationary Distributions and Markov Chains

Stationary Markov chains

A random process $Y = Y_1, Y_2, \ldots$ is said to be *stationary* if the marginal probability of a sequence of elements does not depend on where it starts in the chain. In symbols, a discrete-time random process Y is stationary if for any $t \geq 1$ and any sequence sequence $u_1, \ldots, u_N \in \mathbb{R}^N$ of size N, we have

$$p_{Y_1,\ldots,Y_N}(u_1, \ldots, u_N) = p_{Y_t,\ldots,Y_{t+N}}(u_1, \ldots, u_N)$$

None of the chains we will construct for practical applications will be stationary in this sense, because we would need to know the appropriate initial distribution $p_{Y_1}(y_1)$. For example, consider the fishes example in which we know the trandition probabilities, but not the stationary distribution. If we run long enough, the proportion of pike stabilizes

Stationary distributions

Although we will not, in practice, have Markov chains that are stationary from $t = 1$, we will use Markov chains that have stationary distributions in the limit as $t \to \infty$. For a Markov chain to be stationary, there must be some q such that

$$p_{Y_t}(u) = q(u)$$

for all t, starting from $t = 0$. Instead, we will have an equilibrium distribution q that the chain approaches in the limit as t grows. In symbols,

$$\lim_{t\to\infty} p_{Y_t}(u) = q(u).$$

Very confusingly, this equilibrium distribution q is also called a *stationary distribution* in the Markov chain literature, so we will stick to that nomenclature. We never truly arrive at $q(u)$ for a finite

t because of the bias introduced by the initial distribution $p_{Y_1}(u) \neq q(u)$. Nevertheless, as with our earlier simulation-based estimates, we can get arbitrarily close after suitably many iterations.[96]

[96] The last section of this chapter illustrates rates of convergence to the stationary distribution, but the general discussion is in the later chapter on continuous-state Markov chains.

Reconsider the example of a process $Y = Y_1, Y_2, \ldots$ of fishes, where 1 represents a pike and 0 a perch. We assumed the Markov process Y was governed by

$$\begin{aligned} \Pr[Y_{t+1} = 1 \mid Y_t = 1] &= 0.20 \\ \Pr[Y_{t+1} = 1 \mid Y_t = 0] &= 0.05 \end{aligned}$$

Rewriting as a probability mass function,

$$p_{Y_{t+1}|Y_t}(j \mid i) = \theta_{i,j},$$

where $\theta_{i,j}$ is the probabiltiy of a transtion to state j given that the process is in state i. For the pike and perch example, θ is fully defined by

$$\begin{aligned} \theta_{1,1} &= 0.20 \\ \theta_{1,2} &= 0.80 \\ \hline \theta_{2,1} &= 0.05 \\ \theta_{2,2} &= 0.95. \end{aligned}$$

These numbers are normally displayed in the form of a *transition matrix*, which records the transitions out of each state as a row, with the column indicating the target state,

$$\theta = \begin{bmatrix} 0.20 & 0.80 \\ 0.05 & 0.95 \end{bmatrix}.$$

The first row of this transition matrix is $(0.20, 0.80)$ and the second row is $(0.05, 0.95)$. Rows of transition matrices will always have non-negative entries and sum to one, because they are the parameters to categorical distributions.[97]

[97] Vectors with non-negative values that sum to one are known as *unit simplexes* and matrices in which every row is a unit simplex is said to be a *stochastic matrix*. Transition matrices for finite-state Markov chains are always stochastic matrices.

Now let's take a really long run of the chain with $T = 1\,000\,000$ fish to get a precise estimate of the long-run proportion of pike.

```
initial state = 0.000000; simulated proportion of pike = 0.059
initial state = 1.000000; simulated proportion of pike = 0.059
```

The initial state doesn't seem to matter. That's because the rate of 5.9% pike is the stationary distribution. More formally, let $\pi = (0.059, 1 - 0.059)$ and note that[98]

$$\pi_i = \sum_{j=1}^{2} \pi_j \times \theta_{j,i}.$$

[98] In matrix notation, if π is considered a row vector, then

$$\pi = \theta\, \pi.$$

If π satisfies this formula, then it is said to be the *stationary distribution* for θ.

If a Markov chain has a stationary distribution π and the initial distribution of Y_1 is also π, then it is stationary.

Reducible chains

The Markov chains we will use for sampling from target distributions will be well behaved by construction. There are, however, things that can go wrong with Markov chains that prevent them from having stationary distributions. The first of these is reducibility. A chain is reducible if it can get stuck in a state from which other states are not guaranteed to be revisited with probability one.

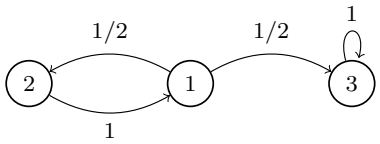


Figure 20: State diagram for a reducible finite Markov chain. The chain will eventually get stuck in state 3 and never exit to visit states 1 or 2 again.

If we start the chain in state 1, it will eventually transition to state 3 and get stuck there.[99] It's not necessary to get stuck in a single state. The same problem arises if state 3 has transitions out, as long as they can't eventually get back to state 1.

[99] State 3 is what is known as a *sink state.*

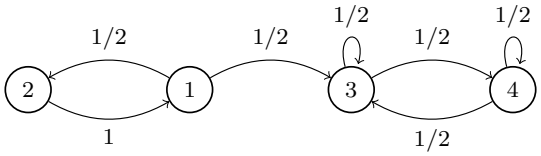


Figure 21: State diagram for another reducible finite Markov chain. The chain will eventually get stuck in state 3 and 4 and never exit to visit states 1 or 2 again.

In this example, the chain will eventually fall into a state where it can only visit states 3 and 4.

Periodic chains

A Markov chain can be constructed to cycle through states in a regular (probabilistic) pattern. For example, consider the following Markov chain transitions.

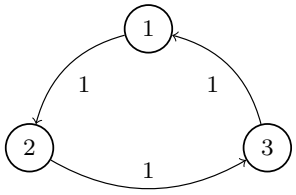


Figure 22: State diagram for finite Markov chain generating periodic state sequences $\ldots, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, \ldots$.

Regular cycles like this defeat the existence of a stationary distribution. If $Y_1 = 2$, the entire chain is deterministically defined to be

$$Y = 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, \ldots.$$

Clearly $p_{Y_t} \neq p_{Y_{t+1}}$, as each concentrates all of its probability mass on a different value.

On the other hand, this chain is what is known as wide-state stationary in that using long-running frequency estimates are stable. the expected value is $\frac{1+2+3}{3} = 2$ and the standard deviatiation is $\sqrt{\frac{1^2+0^2+1^2}{3}} \approx 0.47$. More formally, the wide-state expectation is calculated as

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} Y_t \to 2.$$

The definition of periodicity is more subtle than just deterministic chains. For example, the following transition graph is also periodic.

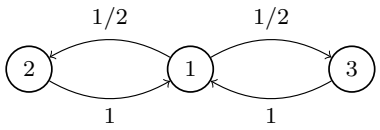


Figure 23: State diagram for finite Markov chain generating periodic state sequences alternatiing between state 1 and either state 2 or state 3.

Rather than a deterministic cycle, it cycles between the state 1 and the pair of states 2 and 3. A simulation might look like

$$y^{(1)} = 1, 2, 1, 2, 1, 2, 1, 3, 1, 3, 1, 2, 1, 3, 1, 2, \ldots$$

Every other value is a 1, no matter whether the chain starts in state 1, 2, or 3. Such behavior means there's no stationary distribution. But there is a wide-sense stable probability estimate for the states, namely 50% of the time spent in state 1, and 25% of the time spent in each of states 2 and 3.

Convergence of finite-state chains

In applied statistics, we proceed by simulation, running chains long enough that they provide stable long-term frequency estimates. These stable long-term frequency estimates are of the stationary distribution categorical(π). All of the Markov chains we construct to sample from target distributions of interest (e.g., Bayesian posterior predictive distributions) will be well-behaved in that these long-term frequency estimates will be stable, in theory.[100]

[100] In practice, we will have to be very careful with diagnostics to avoid poor behavior due to floating-point arithmetic combined with approximate numerical algorithms.

In practice, none of the Markov chains we employ in calculations will be stationary for the simple technical reason that we don't know the stationary distribution ahead of time and thus cannot draw Y_1 from it.[101] What we need to know is conditions under which a Markov chain will "forget" its initial state after many steps and converge to the stationary distribution.

All of the Markov chains we will employ for applied statistics applications will be well behaved in the sense that when run long enough, the distribution of each element in the chain will approach the stationary distribution. Roughly, when t is large enough, the marginal distribution p_{Y_t} stabilizes to the stationary distribution. The well-behavedness conditions required for this to hold may be stated as follows

Fundamental Convergence Theorem. If a Markov chain $Y = Y_1, Y_2, \ldots$ is (a) irreducible, (b) aperiodic, and (c) has a stationary distribution categorical(π), then

$$\lim_{t \to \infty} P_{Y_t}(u) \to \text{categorical}(u \mid \pi).$$

What this means in practice is that we can use a single simulation,

$$y^{(1)} = y_1^{(1)}, y_2^{(1)}, \ldots, y_T^{(1)}$$

to estimate the parameters for the stationary distribution. More specifically, if we define π by

$$\widehat{\pi}_i = \frac{1}{T} \sum_{t=1}^{T} \mathrm{I}[y_t^{(1)} = i]$$

then we can estimate the stationary distribution as categorical$(\widehat{\pi})$.

As a coherence check, we often run a total of M simulations of the first T values of the Markov chain Y.

$$\begin{aligned} y^{(1)} &= y_1^{(1)}, y_2^{(1)}, \ldots, y_T^{(1)} \\ y^{(2)} &= y_1^{(2)}, y_2^{(2)}, \ldots, y_T^{(2)} \\ &\vdots \\ y^{(M)} &= y_1^{(M)}, y_2^{(M)}, \ldots, y_T^{(M)} \end{aligned}$$

We should get the same estimate from using $y^{(m)}$ from a single simulation m as we get from using all of the simulated chains $y^{(1)}, \ldots, y^{(M)}$.[102]

How fast is convergence?

The fundamental theorem tells us that if a Markov chain $Y = Y_1, Y_2, \ldots$ is ergodic (aperiodic and irreducible) and has a stationary distribu-

[101] In the finite case, we actually can calculate it either through simulation or as the eigenvector of the transition matrix with eigenvalue one (which is guaranteed to exist). An eigenvector of a matrix is a row vector π such that

$$c \times \pi = \theta \, \pi,$$

where c is the eigenvalue. This is why Google's PageRank algorithm is known to computational statisticians as the "billion dollar eigenvector." One way to calculate the relevant eigenvector of a stochastic matrix is by raising it to a power, starting from any non-degenerate initial simplex vector λ,

$$\lim_{n \to \infty} \lambda \, \theta^n = \pi.$$

Each

$$\theta^n = \underbrace{\theta \times \theta \times \cdots \times \theta}_{n \text{ times}}$$

is a transition matrix corresponding to taking n steps in the original transition matrix θ.

[102] We'd expect lower error from using all of the chains as we have a larger sample with which to estimate.

tion, then the distribution of Y_t converges to the stationary distribution in rhw limit. But it doesn't tell us how fast.

As with everything else, we'll go at this by simulation to establish intuitions. In particular, we'll consider three chains that have bernoulli(0.5) as their stationary distribution (a fair coin toss).

First, we will consider a Markov chain producing independent Bernoulli draws.

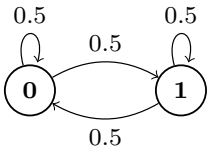


Figure 24: State diagram for finite Markov chain generating independent draws.

Whether it is currently in state 0 or state 1, there is a 50% chance the next state is 0 and a 50% chance it is 1. Thus each element of the process is generated independently and is identically distributed,

$$Y_t \sim \text{bernoulli}(0.5).$$

Therefore, the stationary distribution must also be $\pi = (0.5, 0.5)$, because

$$\begin{aligned} \pi_1 &= \pi_1 \times \theta_{1,1} + \pi_2 \times \theta_{2,1} \\ 0.5 &= 0.5 \times 0.5 + 0.5 \times 0.5 \end{aligned}$$

and

$$\begin{aligned} \pi_2 &= \pi_1 \times \theta_{1,2} + \pi_2 \times \theta_{2,2} \\ 0.5 &= 0.5 \times 0.5 + 0.5 \times 0.5. \end{aligned}$$

We can simulate 100 values and print the first 99 to see what the chain looks like.

```
1 1 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1
1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 0 0 1 1 1 1
0 1 0 1 0 1 0 1 0 1 0 1 1 1 0 0 1 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 1
```

An initial segment of a Markov chain $Y = Y_1, Y_2, \ldots, Y_T$ can be visualized as a traceplot, a line plot of the value at each iteration.

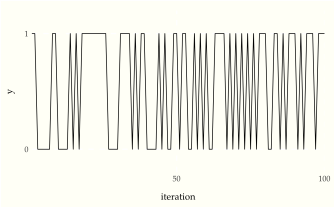


Figure 25: Traceplot of chain producing independent draws, simulated for 100 time steps. The horizontal axis is time (t) and the vertical axis the value of e iteration number and the value is the value (Y_t).

The flat segments are runs of the same value. This Markov chain occassionally has runs of the same value, but otherwise mixes quite well between the values.

So how fast do estimates of the stationary distribution based on an initial segment $Y_1, \ldots, Y_T$ converge to $\frac{1}{2}$? Because each Y_t is independent and identically distributed, the central limit theorem tells us that the rate of convergence, as measured by standard deviation of the distribution of estimates, goes down as $\frac{1}{\sqrt{T}}$ with an initial segment $Y_1, \ldots, Y_T$ of the chain goes down in error as $\sqrt{T}$

Now consider a Markov chain which is still symmetric in the states, but with a tendency to stay in the same state.

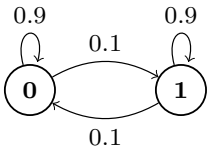


Figure 26: State diagram for correlated draws.

It has the same stationary distribution of 0.5. Letting $\theta = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$ be the transition matrix and $\pi = (0.5, 0.5)$ be the probabilities of the stationary distribution we see that the general formula is satisfied by this Markov chain,

$$\begin{aligned} \pi_1 &= \pi_1 \times \theta_{1,1} + \pi_2 \times \theta_{2,1} \\ 0.5 &= 0.5 \times 0.9 + 0.5 \times 0.1 \end{aligned}$$

The same relation holds for π_2,

$$\begin{aligned} \pi_2 &= \pi_1 \times \theta_{1,2} + \pi_2 \times \theta_{2,2} \\ 0.5 &= 0.5 \times 0.1 + 0.5 \times 0.9 \end{aligned}$$

We can simulate from the chain and print the first 99 values, and then print the traceplot.

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1
1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
```

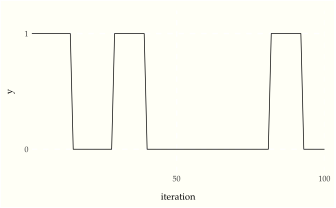


Figure 27: Traceplot for chain with correlated draws.

As expected, there are now long runs of the same value being produced. This leads to much poorer mixing and a longer time for estimates based on the draws to converge.

Finally, we consider the opposite case of a symmetric chain that favors moving to a new state each time step.

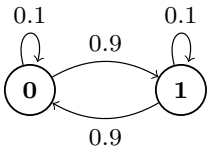


Figure 28: State diagram for anticorrelated draws.

Sampling, printing, and plotting the values produces

```
1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1
0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0
```

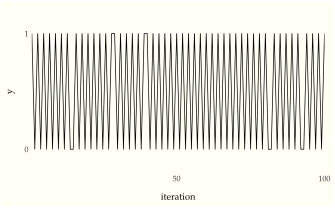


Figure 29: Traceplot of chain with anticorrelated draws.

The draws form a dramatic sawtooth pattern as as they alternate between zero and one.

Now let's see how quickly estimates based on long-run averages from the chain converge in in a side-by-side comparison. A single chain is enough to illustrate the dramatic differences.

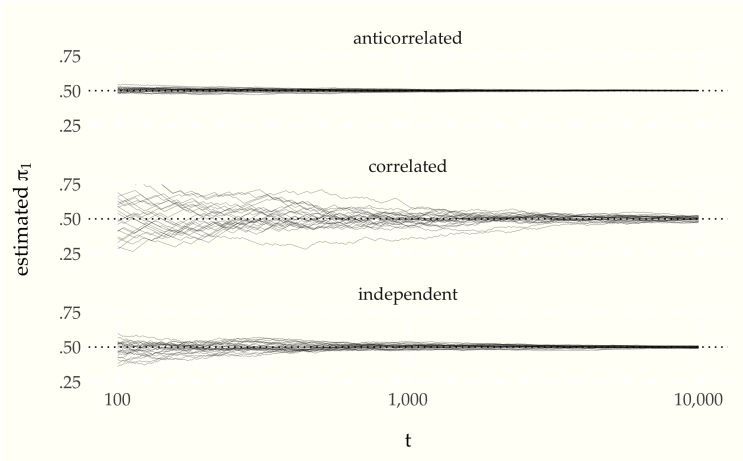


Figure 30: Estimate of the stationary probability π_1 of state 1 as a function of t under three conditions, correlated, independent, and anticorrelated transitions. For each condition, 25 simulations of a chain of size $T = 10\,000$ are generated and overplotted.

Reversibility

These simple Markov chains wind up being reversible, in that the probability of being in a state i and then transitioning to state j is the same as that of being in state j and transitioning to state i. In symbols, a discrete-valued Markov chain $Y = Y_1, Y_2, \ldots$ is *reversible* with respect to π if

$$\pi_i \times \theta_{i,j} \;=\; \pi_j \times \theta_{j,i}.$$

Reversibility is sufficient for establishing the existence of a stationary distribution.[103] Markov chains can have stationary distributions without being reversible.[104] But all of the Markov chains we consider for practical applications will turn out to be reversible.

[103] If a discrete Markov chain is reversible with respect to π, then π is also the unique stationary distribution of the Markov chain.

[104] The reducible chains with we saw earlier are examples with stationary distributions that are not reversible.

Infinite Discrete Markov Chains

All of the Markov chains we have until now have had a finite number of states. In this chapter, we consider Markov chains with a countably infinite number of states. That is, they are still discrete, but can take on arbitrary integer values.

Drunkard's walk

The so-called *drunkard's walk* is a non-trivial Markov chain which starts with value 0 and moves randomly right one step on the number line with probability θ and left one step with probability $1 - \theta$.

The initial value is required to be zero,

$$p_{Y_1}(y_1) \ = \ 1 \ \text{ if } \ y_1 = 0.$$

Subsequent values are generating with probability θ of adding one and probability $1 - \theta$ of subtracting one,

$$p_{Y_{t+1}|Y_t}(y_{t+1} \mid y_t) \ = \ \begin{cases} \theta & \text{if } \ y_{t+1} = y_t + 1, \text{and} \\ 1 - \theta & \text{if } \ y_{t+1} = y_t - 1. \end{cases}$$

Another way to formulate the drunkard's walk is by setting $Y_1 = 0$ and setting subsequent values to

$$Y_{t+1} = Y_t + 2 \times Z_t - 1.$$

where $Z_t \ \sim \ \text{bernoulli}(\theta)$. Formulated this way, the drunkard's walk Y is a transform of the Bernoulli process Z. We can simulate drunkard's walks for $\theta = 0.5$ and $\theta = 0.6$ and see the trend over time.

```
y[1] = 0
for (m in 2:M)
  z[m] = bernoulli_rng(theta)
  y[m] = y[m - 1] + (z[m] ? 1 : -1)
```

We'll simulate from both processes for $M = 1000$ steps and plot.

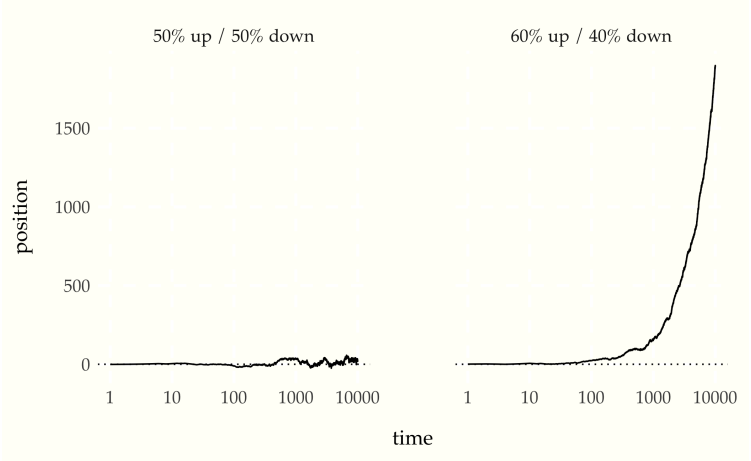


Figure 31: Drunkard's walks of 10,000 steps with equal chance of going left or right (blue) versus a sixty percent chance of going left (red). The dotted line is drawn at the starting point. As time progresses, the biased random walk drifts further and further from its starting point.

For the balanced drunkard, the expected drift per step is zero as there is equal chance of going in either direction. After 10,000 steps, the expected position of the balanced drunkard remains the origin.[105] For the unbalanced drunkard, the expected drift per step is $0.6 \times 1 + 0.4 \times -1 = 0.2$. Thus after 10,000 steps, the drunkard's expected position is $0.2 \times 10\,000 = 2\,000$.

[105] Contrary to common language usage, the expected position being the origin after 10 000 steps does not imply that we should expect the drunkard to be at the origin. It is in fact very unlikely that the drunkard is at the origin after 10 000 steps, as it requires exactly 5 000 upward steps, the probability of which is $\text{binomial}(5\,000 \mid 10\,000, 0.5) = 0.008$.

Gambler's Ruin

Another classic problem which may be understood in the context of an infinite discrete Markov chain is the gambler's ruin. Suppose a gambler sits down to bet with a pile of N chips and is playing a game which costs one chip to play and returns one chip with a probability of θ.[106] The gambler is not allowed to go into debt, so if the gambler's fortune ever sinks to zero, it remains that way in perpetuity. The results of the bets at times $t = 1, 2, \ldots$ can be modeled as an independent and identically distributed random process $Z = Z_1, Z_2, \ldots$ with

[106] The original formulation of the problem, involving two gamblers playing each other with finite stakes, was analyzed in Christiaan Huygens. 1657. *Van Rekeningh in Spelen van Geluck.* Here we assume one player is the bank with an unlimited stake.

$$Z_t \sim \text{bernoulli}(\theta).$$

As usual, a successful bet is represented by $Z_t = 1$ and an unsuccessful one by $Z_t = 0$. The gambler's fortune can now be defined recursively as a time series $Y = Y_1, Y_2, \ldots$ in which the initial value is given by

$$Y_1 = N$$

with subsequent values defined recursively by

$$Y_{n+1} = \begin{cases} 0 & \text{if } Y_n = 0, \text{ and} \\ Y_n + Z_n & \text{if } Y_n > 0. \end{cases}$$

Broken down into the language of Markov chains, we have an initial distribution concentrating all of its mass at the single point N, with mass function

$$p_{Y_1}(N) = 1.$$

Each subsequent variable's probability mass function is given by

$$p_{Y_{t+1}|Y_t}(y_{t+1} \mid y_t) = \begin{cases} \theta & \text{if } y_{t+1} = y_t + 1 \\ 1 - \theta & \text{if } y_{t+1} = y_t - 1. \end{cases}$$

These mass functions are all identical in that $p_{Y_{t+n+1}|Y_{t+n}} = p_{Y_{t+1}|Y_t}$. In other words, Y is a time-homogeneous Markov chain.

We are interested in two questions pertaining to the gambler. First, what is their expected fortune at each time t? Second, what is the probability that they have fortune zero at time t.[107] Both of these calculations have simple simulation-based estimates.

[107] A gambler whose fortune goes to zero is said to be *ruined.*

Let's start with expected fortune and look out $T = 100$ steps. Suppose the chance of success on any given bet is θ and their initial fortune is N. The simulation of the gambler's fortune is just a straightforward coding of the time series.

```
y[1] = N
for (t in 2:T)
  z[t] = bernoulli_rng(theta)
  y[t] = y[t - 1] + (z[t] ? 1 : -1)
```

Now if we simulate that entire process M times, we can calculate the expected fortune as an average at each time $t \in 1 : T$.

```
for (m in 1:M)
  y(m)[t] = N
  for (t in 2:T)
    z(m)[t] = bernoulli_rng(theta)
    y(m)[t] = y(m)[t - 1] + (z[t] ? 1 : -1)
for (t in 1:T)
  expected_fortune[t] = mean(y(1:M)[t])
```

Let's run $M = 10\,000$ simulations for $T = 50$ starting with a stake of $N = 5$ with several values of θ and plot the expected fortunes.

Next, we'll tackle the problem of estimating the probability that a gambler has been run out of money at time t. In symbols, we are going to use simulations $y^{(1)}, \ldots, y^{(M)}$ of the gambler's time series,

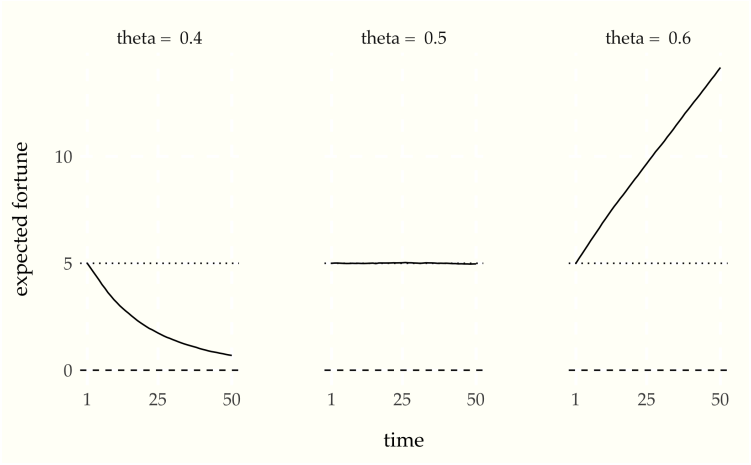


Figure 32: Expected returns for gambler starting with stake N and having a θ chance at each time point of increasing their fortune by 1 and a $1 - \theta$ chance of reducing their fortune by 1. The horizontal dotted line is at the initial fortune and the dashed line is at zero.

$$\begin{aligned} \Pr[Y_t = 0] &= \mathbb{E}\left[\mathrm{I}\left[Y_t = 0\right]\right], \\ &\approx \frac{1}{M} \sum_{m=1}^{M} \mathrm{I}\left[y_t^{(m)} = 0\right]. \end{aligned}$$

This last term can be directly calculated by adding the indicator variables to the calculations before.

```
for (m in 1:M)
  y(m)[t] = N
  for (t in 2:T)
    z(m)[t] = bernoulli_rng(theta)
    y(m)[t] = y(m)[t - 1] + (z[t] ? 1 : -1)
    ruined(m)[t] = (y(m)[t] == 0)
for (t in 1:T)
  estimated_pr_ruin[t] = mean(ruined(1:M)[t])
```

So let's run that and plot the probability of ruin for the same three choices of θ, using $M = 5\,000$ simulations. But this time, we'll run for $T = 200$ time steps.

Even in a fair game, after 50 bets, there's nearly a 50% chance that a gambler starting with a stake of 5 is ruined; this probabiltiy goes up to nearly 75% after 200 bets.

Queueing

Suppose we have a checkout line at a store (that is open 24 hours a day, 7 days a week) and a single clerk. The store has a queue, where customers line up for service. The queue begins empty. Each hour a random number of customers arrive and a random number of

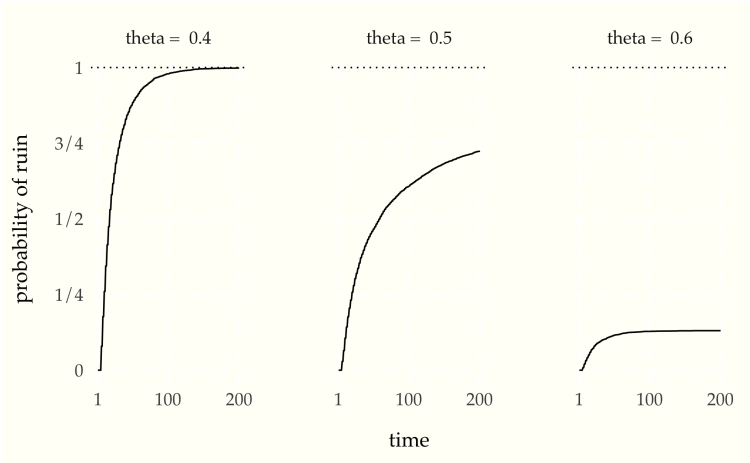


Figure 33: Probability of running out of money for a gambler starting with stake N and having a θ chance at each time point of increasing their fortune by 1 and a $1 - \theta$ chance of reducing their fortune by 1. The horizontal dotted line is at 100 percent.

customers are served. Unserved customers remain in the queue until they are served.

To make this concrete, suppose we let $U_t \in 0, 1, \ldots$ be the number of customers that arrive during hour t and that it has a binomial distribution,

$$U_t \sim \text{binomial}(1000, 0.005).$$

Just to provide some idea of what this looks like, here are 20 simulated values,

```
 7  0  3  4  4  3  3  7  2  4  6  7  6  4  5  7  2  4  3  6
```

We can think of this as 1000 potential customers, each of which has a half percent chance of deciding to go to the store any hour. If we repeat, the mean number of arrivals is 5 and the standard deviation is 2.2.

Let's suppose that a cleark can serve up to V_t customers per hour, determined by the clerk's rate ϕ,

$$V_t \sim \text{binomial}(1000, \phi).$$

If $\phi < 0.005$, there is likely to be trouble. The clerk won't be able to keep up on average.

The simulation code just follows the definitions.

```
queue[1] = 0
for (t in 2:T)
  arrive[t] = binomial_rng(1000, 0.005)
  serve[t] = binomial_rng(1000, phi)
  queue[t] = max(0, queue[t - 1] + arrive[t] - serve[t])
```

The `max(0, ...)` is to make sure the queue never gets negative. If the number served is greater than the total number of arrivals and customers in the queue, the queue starts empty the next time step.

Let's try different values of ϕ, the average server rate, and plot two weeks of service.[108]

[108] 24hours/day $\times$ 14 days $=$ 336hours

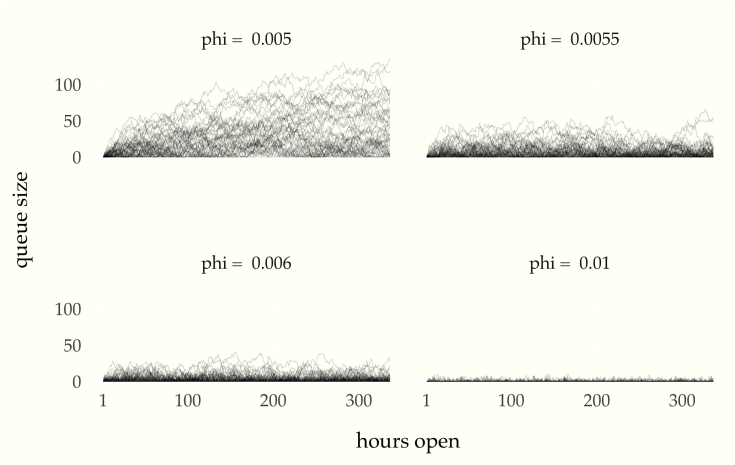


Figure 34: Multiple simulations of queue size versus time for a queue with binomial$(1000, 0.005)$ customers arriving per hour (an average of 5), and a maximum of binomial$(1000, \phi)$ customers served per hour, plotted for various ϕ (as indicated in the row labels).

As can be seen in the plot, the queue not growing out of control is very sensitive to the average service rate per hour. At an average rate of five cusotmers served per hour (matching the average customer intake), the queue quickly grows out of control. With as few as five and a half customers served per hour, on average, it becomes stable long term; with seven customers served per hour, things settle down considerably. When the queue goes up to 50 people, as it does with $\phi = 0.0055$, wait times are over ten hours. Because of the cumulative nature of queues, a high server capacity is required to deal with spikes in customer arrival.

Continuous Random Variables

So far, we have only considered discrete random variables, i.e., variables taking only integer values. But what if we want to use random variables to represent lengths or volumes or distances or masses or concentrations or any of the other continuous properties in the physical world? We will need to generalize our approach so far.

Continuous random variables take on real values. The set of real numbers is *uncountable* in the sense of being strictly larger than the set of integers.[109]

[109] Georg Cantor developed the technique of diagonalization to show it was impossible to have a one-to-one map from the reals to the integers, thus proving the set of reals is uncountable.

The mathematics of probability is the same for real values. Even more importantly from a practical standpoint, the way we calculate event probabilities and estimates remains the same with continuous quantities. The notion of a probability mass function, on the other hand, must be replaced by its continuous equivalent, the probability density function.

Spinners and uniform continuous variables

Suppose Θ is a random variable representing the angle at which a fair spin of a spinner lands. We will use degrees and thus suppose the value of Θ is between 0 and 360[110]

[110] The end points are the same, representing a complete rotation of 360 degrees; they are labeled as such in the plot.

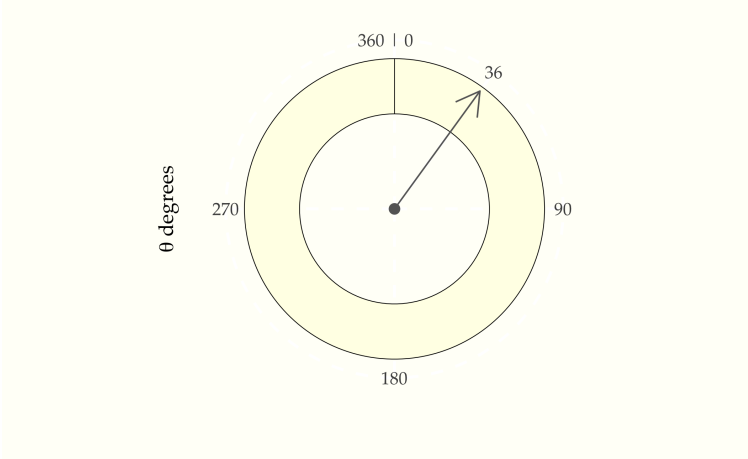


Figure 35: A spinner resting at 36 degrees, or ten percent of the way around the circle. A fair spin might land anywhere between 0 and 360 degrees.

What does fair mean for continuous probabilities? At the least, we should be able to say that the probality of landing in any band is the same no matter where the band lies on the circle. That is, landing between 0 and 36 degrees should be the same as landing between 200 and 236 degrees. Also, because 36 degrees is one tenth of the way around the circle, the chance of landing in any 36 degree band has to be 10%.[111] We can express that in probability notation as

$$\Pr[0 \leq \Theta \leq 36] \ = \ \frac{1}{10}$$

and

$$\Pr[200 \leq \Theta \leq 236] \ = \ \frac{1}{10}.$$

We are not talking about the probability of Θ taking on a particular value, but rather of it falling in a particular interval.[112] For continuous random variables, outcomes do not have probability mass. Instead, probability mass is assigned continuously based on the probability of a variable falling in a region.

[111] A circle can be divided into ten bands of 36 degrees to create exhaustive and exclusive intervals, the event probabilities of landing in which must be the same by fairness and must total one because they exhaust all possible outcomes.

[112] In general, the probability of a fair spinner Θ falling in interval is the fraction of the circle represented by the interval, i.e.,

$$\Pr[\theta_1 \leq \Theta \leq \theta_2] = \frac{\theta_2 - \theta_1}{360}.$$

for $0 \leq \theta_1 \leq \theta_2 \leq 360$.

The paradox of vanishing point probabilities

In our first example, we took a fair spinner to land at exactly 36 degrees; it could've been 36.0376531 degrees or even an irrational number such as $0.3333\cdots$.[113] What's the probability the spinner landed on exactly 36 degrees? Paradoxically, the answer is zero.

[113] When I learned decimal representations, we wrote $0.\bar{3}$ for the decimal representation of $\frac{1}{3}$.

$$\Pr[\Theta = 36] = 0.$$

Why must this be? If the probability of any specific outcome was greater than zero, every other possible value would have to have the same probability to satisfy fairness. But then if we summed them all up, the total would be greater than one, which is not possible. Something has to give, and that something is the idea of particular point outcomes having probability mass in a continuous distribution. The paradox arises because some number must come up, but every number has zero probability.

Simulating uniform values

We will assume that our programming language comes equipped with a function `uniform_rng(L, H)` that generates numbers uniformly in the interval $[L, H]$.

For instance, the following program simulates from the uniform interval.

```
for (m in 1:M)
  y[m] = uniform_rng(0, 1)
print 'y = ' y
```

Let's simulate $M = 10$ draws and look at the result,

```
 0.1137 0.6223 0.6093 0.6234 0.8609 0.6403 0.0095 0.2326 0.6661 0.5143
```

These are only printed to a few decimal places. As usual, it's hard to get a sense for the sequence of values as raw numbers. The most popular way to summarize one-dimensional data is with a *histogram*, as shown in the following plot.

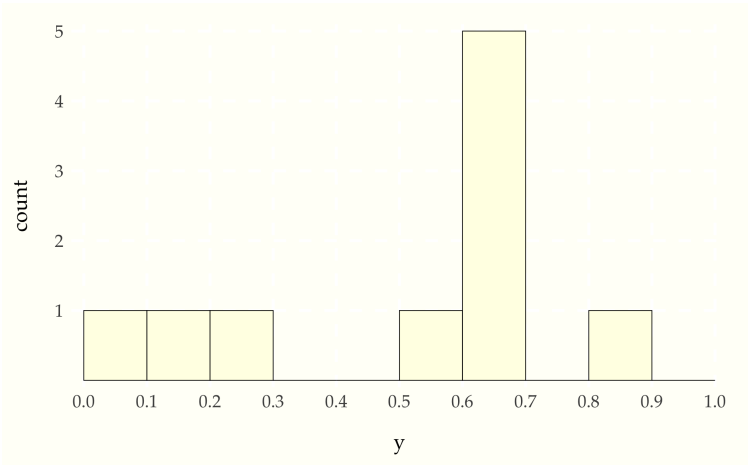


Figure 36: Histogram of 10 draws from a uniform$(0,1)$ distribution.

The range of values from 0 to 1 is broken up into ten equally spaced bins, 0 to 0.1, 0.1 to 0.2, up to 0.9 to 1.0. Each bin is then drawn as a rectangle with a height proportional to the number of values that fell into the bin.

Even though the distribution draws uniformly in the interval, with only ten draws, the probability of having one draw in each bin is small,[114] whereas the probability of having many values in a single bin is relatively high.[115] As usual, we turn to repetition and sizing to see what's going on.

```
List of 1
 $ aspect.ratio: num 0.5
 - attr(*, "class")= chr [1:2] "theme" "gg"
 - attr(*, "complete")= logi FALSE
 - attr(*, "validate")= logi TRUE
```

Calculating π via simulation

Now that we have a continuous random number generator, there are all kinds of values we can compute. Here, we show how to cal-

[114] The first draw can be in any bin, the second in any of 9 bins, the third in any of 8 bins, and so on, yielding a probability for each bin containing a single draw of

$$\prod_{n=1}^{10} \frac{n}{10} \approx 0.00036.$$

[115] For example, the probability of having a bin with exactly five draws involves a choice of the distinguished bin, a choice of which of the five draws go in the distinguished bin, then the probabilities of the distinguished bins and other bins,

$$\binom{10}{1} \times \binom{10}{5} \times \left(\frac{1}{10}\right)^5 \times \left(\frac{9}{10}\right)^5 \approx 0.015.$$

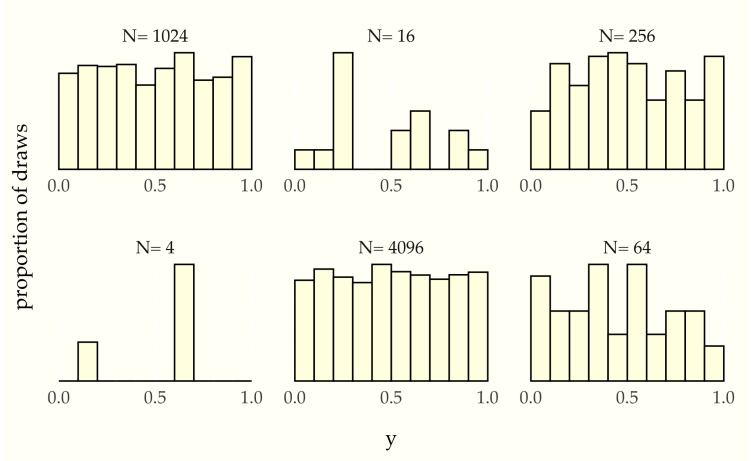


Figure 37: Histograms for uniform(0, 1) samples of increasing sizes. The proportion of draws falling into each bin becomes more uniform as the sample size increases. With each sample plotted to the same height, the vertical count axis varies in scale among the plots.

culuate the first few digits of π. We'll carry this out by formulating an event probability over some continuous random variables whose probability is a fraction of π.

We start with the basic fact of algebra that that π is the area of a unit radius circle.[116] We then assume there are two independent, uniform random variables X and Y,

$$X, Y \sim \text{uniform}(-1, 1).$$

Simulations $x^{(m)}, y^{(m)}$ of these variables pick out a point on the plane within a square bounded by -1 and 1 in both dimensions. Here is a plot of the square in which the draws will fall. Also shown is a circle of unit radius inscribed within that square. Draws may or may not fall within the circle.[117]

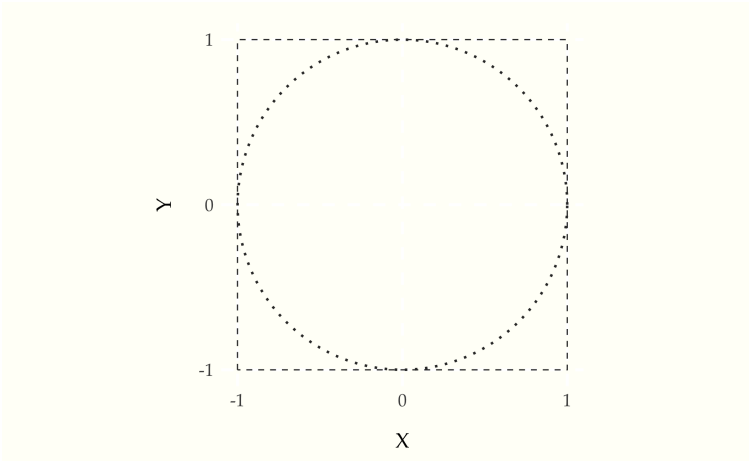


Figure 38: A unit circle (dotted line), centered at the origin is inscribed in a square (dashed lines) with axes running from -1 to 1.

A point (x, y) will fall within the unit circle if[118]

[116] The area of a circle of radius r is $\pi \times r^2$, so when $r = 1$, the area is π.

[117] Technically, the bearer of area is a disc and the line around its border a circle. Mathematicians are picky because, topologically speaking, a disc has two dimensions whereas a circle has but one.

[118] A point falls on a circle of radius r if $x^2 + y^2 = r^2$.

$$x^2 + y^2 \leq 1.$$

Let's see what this looks like with $M = 250$ draws. The resulting plot is known as a *scatterplot*—it places values at their (x, y) coordinates, resulting in them being "scattered."

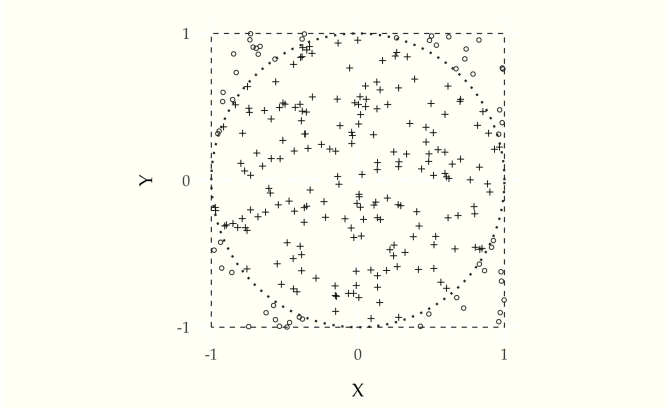


Figure 39: $M = 250$ simulated draws of $(x^{(m)}, y^{(m)})$ from a uniform$(-1, 1)$ distribution. Points within the circle are plotted using $+$ and those outside it with $\circ$.

For random variables $X, Y \sim \text{uniform}(-1, 1)$, the event of their falling within the unit circle is $X^2 + Y^2 \leq 1$. Because X and Y are drawn uniformly from within the square, the probability of their being within the circle is proportional to the circle's area. The circle's area is π, whereas the overall area of the square is 4. So the probability of a random draw within the square being within the circle is

$$\Pr[X^2 + Y^2 \leq 1] = \frac{\pi}{4}.$$

We know how to estimate event probabilities using simulation. The code here is straightforward.

```
for (m in 1:M)
  x[m] = uniform_rng(-1, 1)
  y[m] = uniform_rng(-1, 1)
  inside[m] = (x[m]^2 + y[m]^2 <= 1)
print 'Pr[in circle] = ' sum(inside) / M
print 'esimtated pi = ' 4 * sum(inside) / M
```

We recover the simulation-based estimate of π by multiplying the event probability of $X^2 + Y^2 \leq 1$ by four.

Let's run this with $M = 1\,000\,000$ and see what we get,

```
Pr[in circle] = 0.784
estimated pi = 3.138
```

We actually knew the answer ahead of time here, $\pi \approx 3.14159$. The simulation-based estimate is on the right track.[119] At least the first couple of digits are correct.

If you want to do this all the old-fashioned way with calculus, note that the top half of the circle is given by the function $y = \sqrt{1 - x^2}$. Integrating this from -1 to 1 and doubling it thus produces the required value,

$$\pi = 2 \times \int_{-1}^{1} \sqrt{1 - x^2} \, \mathrm{d}x.$$

Simulation-based methods are largely used in practice to solve nasty integrals without analytic solutions.

[119] But not going to win any π digit-generating contests, either. Remember, we need one hundred *times* as many draws for each subsequent digit of precision using i.i.d. simulation.

The curse of dimensionality

In most introductory examples, including the ones in this book, intuitions are developed based on one or two dimensions—essentially, we use what can be visualized. It turns out that intuitions based on a few dimensions are not only useless, but harmful, when extended to higher dimensions. This section attempts to explain why, and along the way, introduce the key notion for sampling of the typical set.

Working by example, let's start with the base case of a single random variable X_1 with a uniform distribution over the line in the interval $(0, 1)$,

$$X_1 \sim \text{uniform}(0, 1).$$

The length of our unit line is one (hence the name), so the probability of falling in an interval is proportional to the interval's length.[120]

Now extend this to two dimensions, letting the pair of random variables (X_1, X_2) be uniformly distributed in the unit square, $(0, 1) \times (0, 1)$,

$$X_1, X_2 \sim \text{uniform}(0, 1).$$

The area of a $1x1$ square is one, so that the probability of falling in a region within that square is proportional to the area of the region.[121]

Going one dimension further, let (X_1, X_2, X_3) be random variables uniformly distributed in a unit cube, $(0, 1)^3 = (0, 1) \times (0, 1) \times (0, 1)$,

$$X_1, X_2, X_3 \sim \text{uniform}(0, 1).$$

The unit cube has unit volume, $1 \times 1 \times 1 = 1$. Thus the probability of falling in a region within that cube is proportional to the volume of the region.[122]

[120] This also explains why a point has probability zero—it has length zero.

[121] Thus a point or a line has zero probability in a square.

[122] Thus a plane has zero probability in a cube. This generalizes in the obvious way, so this will be the last note.

Now we can just go all the way, and let $X = (X_1, \ldots, X_N)$ be an N-dimensional random variable generated uniformly in an N-dimensional unit hypercube, $(0,1)^N$,

$$X_n \sim \text{uniform}(0,1) \text{ for } n \in 1:N.$$

As before, the hypervolume is $1 \times 1 \times \cdots \times 1 = 1$, so the probability of lying in a region of the hypercube is proportional to the hypervolume of the region.[123]

We saw that in two dimensions, the probability of a random draw from a unit square lying in the inscribed circle was approximately 0.74. As dimensionality increases, this number goes to zero quickly. In other words, most of the probability mass (as measured by hypervolume) in high dimensions lies in the corners. Let's see just how much by simulation.

[123] The fact that we go from line to square to cube and then run out of words should be a tipoff that we're barely acquainted with high dimensions. To be fair, we have "tesseract" for a four-dimensional hypercube, but that's a technical term that might have lain buried in introductions to geometry if not for the Marvel Universe, and even there, it was known colloquially as "the cube."

```
euclidean_length = x.sqrt(sum(x^2))

for (log2_N in 1:max_log2_N)
  N <- 2^log2_N
  for (m in 1:M)
    for (n in 1:N)
      x(m)[n] = uniform_rng(-0.5, 0.5)
    d(m) = euclidean_length(x(m))
  mean_d[log2N] = mean(d)
  min_d[log2N] = min(d)
  max_d[log2N] = max(d)
```

We take `x^2` to operate elementwise on the members of `x`, e.g., $(1,2,3)^2 = (1,4,9)$. The square root, sum, and other operaitons should be self explanatory. The Euclidean length function is being defined with *lambda abstraction*, where the `x.` indicates that the argument to the function is `x` and the result is the result of plugging the value for `x` into the body, `sqrt(sum(x^2))`.

Let's plot what we get out to 1 000 dimensions or so.

While it may seem intuitively from thinking in two dimensions that draws should be uniformly dispersed and thus appear near the origin, they in fact do two things that are quite surprising based on our low-dimensional intuitions,

- draws get further and further away from the origin, on average, as dimension increases, and
- draws accumulate in a thin shell of distance from the origin, with no draws being anywhere near the origin in high dimensions.

The first fact is obvious when you consider the definition of distance is the square root of the squared dimension values, which we

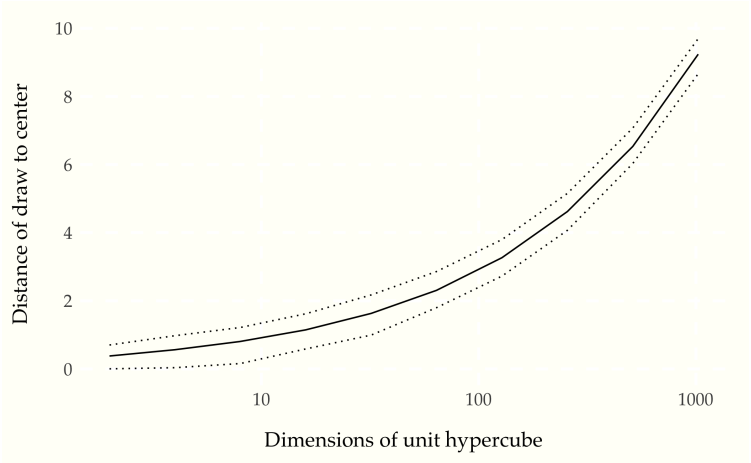


Figure 40: Plot of the average distance (solid line) of a uniform draw from a hypercube to the center of the hypercube as a function of the number of dimensions. The minimum and maximum distance (dotted lines) are shown based on $M = 10\,000$ simulations.

will write as

$$||x|| = \sqrt{x_1^2 + x_2^2 + \cdots + x_N^2}.$$

In this form, it is clear that as N increases, so does $||x||$ as we simply keep adding squared terms x_n^2 with each additional dimension.

Although the draws accumulate in a thin shell of distance from the origin, it would be wrong to conclude that they are anywhere near each other. They are further from each other, on average, than they are from the origin. For example, let's simulate just for 100 dimensions,

```
for (m in 1:M)
  for (n in 1:N)
    x[n] = uniform_rng(-0.5, 0.5)
    y[n] = uniform_rng(-0.5, 0.5)
  d(m) = euclidean_length(x - y)
print 'min = ' min(d) ';  mean = ' mean(d) '; max = ' max(d)
```

Let's run that for $M = 10\,000$ and $N = 100$ and see what we get.

```
 min = 3.0;  mean = 4.1;  max = 5.0
```

We see that the average distances between randomly generated points is even greater than the average distance to the origin.

Continuous Distributions and Densities

Continuous cumulative distribution functions

Suppose $\Theta \sim \text{uniform}(0, 360)$ is the result of spinning a fair spinner. The cumulative distribtution function is defined exactly as for discrete random variables,[124]

$$F_\Theta(\theta) = \Pr[\Theta \leq \theta].$$

That is, it's the probability the random variable is less than or equal to θ. In this case, because the spinner is assumed to be fair, the cumulative distribution function is

$$F_\Theta(\theta) = \frac{\theta}{360}.$$

This is a linear function of θ, i.e., $\frac{1}{360} \times \theta$, as is reflected in the following plot.

[124] Note that we have moved from Roman to Greek letters, but have kept to our capitalization convention for random variables—Θ is the capitalized form of θ.

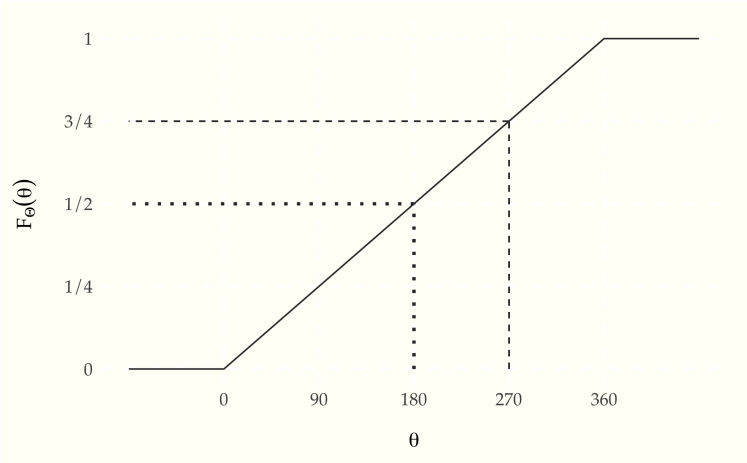


Figure 41: Cumulative distribution function for the angle θ (in degrees) resulting from a fair spin of a spinner. The dotted line shows the value at 180 degrees, which is a probability of one half and the dashed line at 270 degrees, which is a probability of three quartersxs.

We can verify this result using simulation. To estimate cumulative distribution functions, we take M simulated values $\theta^{(m)}$ and then sort them in ascending order.

```
for (m in 1:M)
```

```
  theta[m] <- uniform_rng(0, 360)
theta_ascending <- sort(theta)
prob <- (1:M) / M
```

The expression `(1:M)` denotes the sequence $1, 2, \ldots, M$, so that `(1:M) / M` denotes $\frac{1}{M}, \frac{2}{M}, \ldots, \frac{M}{M}$. The trick is to put the sorted random variable the x-axis and the probability values on the y axis. Here's a run with $M = 1\,000$ simulated values.

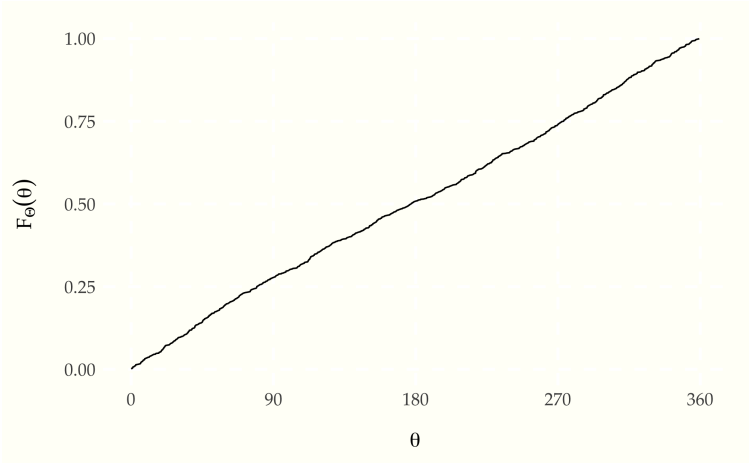


Figure 42: Plot of the cumulative distribution function of a random variable Θ representing the result of a fair spin of a spinner from 0 to 360 degrees. As expected, it is a simple linear function because the underlying variable Θ has a uniform distribution.

Even with $M = 1\,000$, this is pretty much indistinguishable from the one plotted analytically.

As with discrete parameters, the cumulative distribution function may be used to calculate interval probabilities, e.g.,[125]

$$\begin{aligned} \Pr[180 < \Theta \leq 270] &= \Pr[\Theta \leq 270] \ - \ \Pr[\Theta \leq 180] \\ &= F_\Theta(270) - F_\Theta(180) \\ &= \tfrac{3}{4} - \tfrac{1}{2} \\ &= \tfrac{1}{4}. \end{aligned}$$

[125] With continuous variables, the interval probabilities are open below ($180 < \Theta$) and closed above ($\Theta \leq 270$), due to the definition of the cumulative distribution function as a closed upper bound ($F_\Theta(\theta) = \Pr[\Theta \leq \theta]$).

The log odds transform

Now that we have seen how to generate uniform random numbers from 0 to 360, it is time to consider generating standard uniform variates from 0 to 1. Suppose Θ is a random variable with a standard uniform distribution, i.e., $\Theta \sim \text{uniform}(0, 1)$. Because probabilities are scaled from zero to one, we can think of Θ as denoting a random probability.

Given a probability value $\theta \in (0, 1)$, we can define its *log odds* by

$$\text{logit}(\theta) = \log \frac{\theta}{1 - \theta}.$$

This is just the natural logarithm of the odds, $\frac{\theta}{1-\theta}$. Now let

$$\Phi = \text{logit}(\Theta)$$

be the random variable representing the log odds. We say that Φ is a transform of Θ, because its value is determined by the value of Θ.

Simulating transformed variables is straightforward.

```
for (m in 1:M)
  theta[m] = uniform_rng(0, 1)
  alpha[m] = logit(theta[m])
print 'alpha = ' alpha[1:10] ' ... '
```

We can run this and see the first ten values,

```
 -2.05 0.50 0.44 0.50 1.82 0.58 -4.65 -1.19 0.69 0.06
  ...
```

To understand the distribution of values of Φ, let's look at histograms. First, we have the uniform draws of Θ,

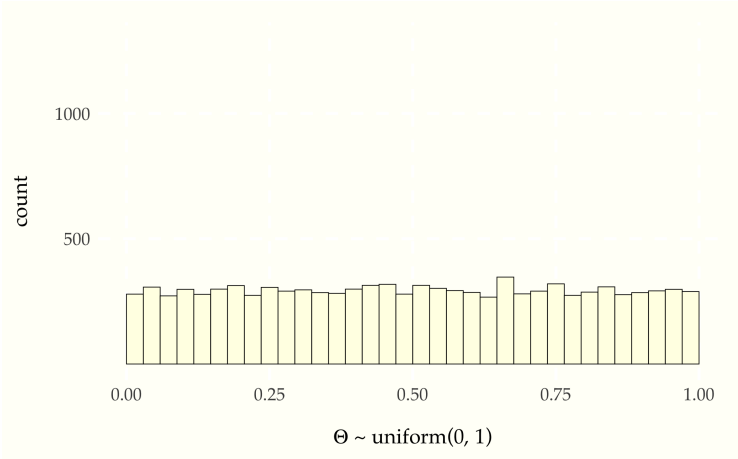


Figure 43: Histogram of 10 000 simulated draws of $\Theta \sim \text{uniform}(0, 1)$.

and then the transform to log odds $\Phi = \text{logit}(\Theta)$,

Even though the probability variable $\Theta \sim \text{uniform}(0, 1)$ is uniform by construction, the log odds variable $\Phi = \text{logit}(\Theta)$ is not distributed uniformly.

A further feature of the log odds plot is that the distribution of values is symmetric around zero. Zero on the log odds scale corresponds to 0.5 on the probability scale,[126] i.e.,

$$0 = \text{logit}(0.5),$$

or equivalently,

$$\text{logit}^{-1}(0) = 0.5.$$

[126] Recall that the inverse log odds function is defined by

$$\text{logit}^{-1}(u) = \frac{1}{1 + \exp(-u)}.$$

This function is called the *logistic sigmoid* in engineering circles. Inverses satisfy for $u \in \mathbb{R}$,

$$\text{logit}(\text{logit}^{-1}(u)) = u$$

and $v \in (0, 1)$,

$$\text{logit}^{-1}(\text{logit}(v)) = v.$$

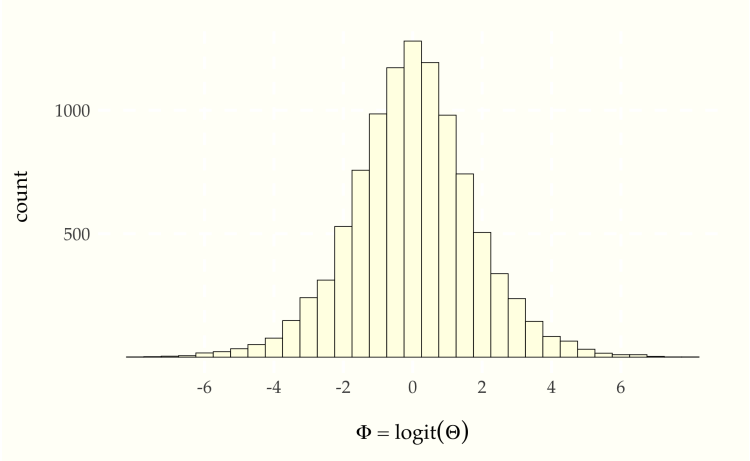


Figure 44: Histogram of 10 000 simulated draws of $\Theta \sim \text{uniform}(0, 1)$ transformed to the log odds scale by $\Phi = \text{logit}(\Theta)$.

Unboundedness and symmetry around zero make log odds quite convenient statistically and will resurface in categorical regressions.

The third relevant feature of the log odds plot is that almost all of the values are within ± 6 of the origin. This is not surprising given that we took 10 000 draws and

$$\text{logit}^{-1}(-6) = 0.0025$$

and

$$\text{logit}^{-1}(6) = 0.9975$$

on the probability scale.

We can also do what we did for uniform distributions and plot the cumulative distribution based on simulation; we need merely insert the log-odds transform.

```
for (m in 1:M)
  theta[m] <- logit(uniform_rng(0, 360))
theta_ascending <- sort(theta)
prob <- (1:M) / M
```

We again plot with $M = 1\,000$ simulated values.

The result is an S-shaped function whose values lie between 0 and 1, with asymptotes at one as θ approaches ∞ and at zero as θ approaches $-\infty$. The argument of 0 has a value of 0.5.

The cumulative distribution function of this distribution is well known and has a closed analytic form based on the inverse of the log odds transform,

$$F_\Theta(\theta) \;=\; \text{logit}^{-1}(\theta) \;=\; \frac{1}{1 + \exp(-\theta)}.$$

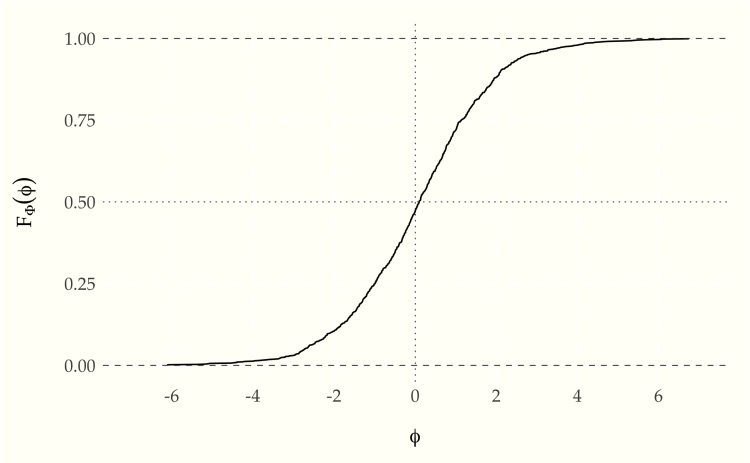


Figure 45: Plot of the cumulative distribution function of a random variable $\Phi = \text{logit}(\Theta)$ representing the the log odds transform of a uniformly distributed random variable $\Theta \sim \text{uniform}(0, 1)$. The curve it picks out is S-shaped. The asymptotes at 0 and 1 are indicated with dashed lines; the symmetries around 0 on the x-axis and 0.5 on the y-axis are picked out with dotted lines.

The inverse log odds function is itself known as the *logistic sigmoid* function.[127]

[127] A name presumably derived from its shape and the propensity of mathematicians, like doctors, to prefer Greek terminology—the Greek letter "σ" (sigma) corresponds to the Roman letter "S".

Expectation and variance of continuous random variables

Just as with discrete random variables, the expectation of a continuous random variable Y is defined as a weighted average of its values. Only this time, the weights are defined by the probability density function rather than by the probability mass function. Because Y takes on continuous values, we'll need calculus to compute the weighted average.

$$\mathbb{E}[Y] = \int_{-\infty}^{\infty} y \times p_Y(y) \, \mathrm{d}y.$$

Integrals of this general form should be read as a weighted average. It averages the value of y with weights equal to the density $p_Y(y)$ of y.[128]

[128] Sometimes physicists will rearrange integral notation to reflect this and write

$$\mathbb{E}[f(y)] = \int \mathrm{d}y \; p_Y(y) \times f(y)$$

or even

$$\mathbb{E}[f(y)] = \int p_Y(\mathrm{d}y) \times f(y).$$

Variances are calculated just as they were for discrete variables, as

$$\text{var}[Y] \;=\; \mathbb{E}\left[(Y - \mathbb{E}[Y])\right].$$

Let's check this with some simulation by estimating the mean and variance of our running example. Suppose we have a a random variable $\Phi = \text{logit}(\Theta)$, where $\Theta \sim \text{uniform}(0, 1)$. We can estimate the expectation and variance of Φ by simulating and calculating means and variances of the simulated values,

```
set.seed(1234)
phi <- rep(NA, M)
for (m in 1:M)
  phi[m] = logit(uniform_rng(0, 1))
```

```
E_Phi = sum(phi) / M
var_Phi = sum((phi - E_Phi)^2) / M
print 'Estimated E[Phi] = ' E_Phi
      '; var[Phi] = ' var_Phi
      '; sd[Phi] = ' sqrt(var_Phi)
```

Let's run that for $M = 1\,000\,000$ and see what we get.

```
Estimated  E[Phi] = -0.00;  var[Phi] = 3.30;  sd[Phi] = 1.82
```

The true value of the expectation $\mathbb{E}[Y]$ is zero, and the true value of the variance is $\frac{\pi^2}{3} \approx 3.29$.[129]

[129] The true mean and variance for the logistic distribution can be calculated analytically. See the final section on this chapter for the analytic derivativation of the probability density function. The density must be integrated to analytically calcuate the mean and variance, though the result for the mean also arises from symmetry.

From histograms to densities

There is no equivalent of a probability mass function for continuous random variables. Instead, there is a probability density function, which in simulation terms may usefully be thought of as a limit of a histogram as the number of draws increases and the width of bins shrinks. Letting the number of simulations grow from 10 to 1 000 000, we see the limiting behavior of the histograms.

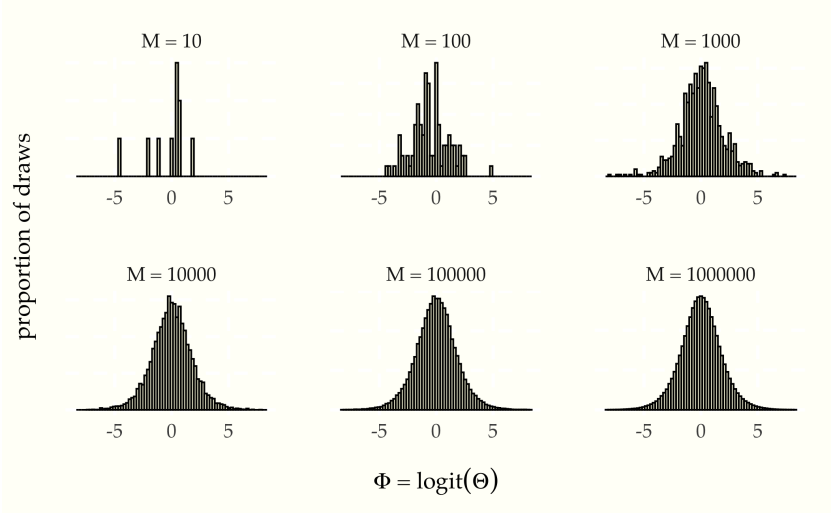


Figure 46: Histograms of M simulated draws of $\Theta \sim \text{uniform}(0, 1)$ transformed to the log odds scale by $\Phi = \text{logit}(\Theta)$. The limiting behavior is shown in the bell-shaped curve in the lower right based on 1 000 000 draws.

In a histogram, a bin's height is proportional to the number of simulations that landed in that bin. Because each bin is the same width, a bin's area (given by its width time its height) must also be proportional to the number of simulations that landed in that bin.

With simulation, the estimate of a probability landing in a bin is just the proportion of simulate values that land in the bin. Thus we can think of the area of a histogram's bar as an estimate of the probability a value will fall in that bin.

Because the bins are exclusive (a number can't fall in two bins), the probability of landing in either of two bins is proportional to the sum of their areas. This notion extends to intervals, where the estimated probability of the random variable falling between -2 and 2 is just the proportion of area between those two values in the histogram of simulations. Similarly, we can take a simulation-based estimate of $\Pr[\Theta \leq \theta]$ for any θ as the proportion of simulated values that are less than or equal to θ. This is just the area to the left of the θ.

As the number of draws M increases, the estimated bin probabilities become closer and closer to the true values. Now we are going to look at the limiting continuous behavior. Put a point in the middle of the top of each histogram bar and connect them with lines. With a finite number of bins, that makes a jagged pointwise linear function. As the number of bins increases and the number of draws per bin increases, the function gets smoother and smoother. In the limit as $M \to \infty$, it approaches a smooth function. That smooth function is called the *probability density function* of the random variable. Let's see what that limiting function looks like with $M = 1\,000\,000$ draws.

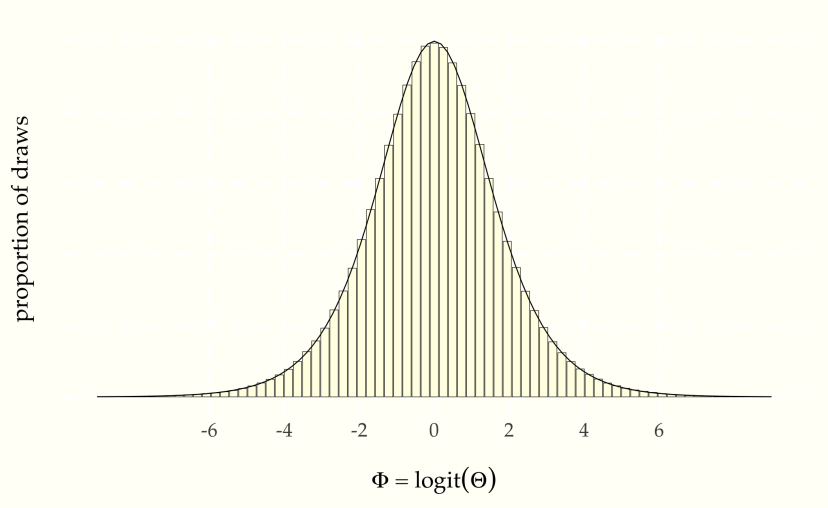


Figure 47: Histogram of $M = 1\,000\,000$ simulations of $\Theta \sim \text{uniform}(0,1)$ transformed to $\Phi = \text{logit}(\Theta)$. The black line connects the tops of the histogram bins. In the limit, as the number of draws and bins approach infinity, the connecting line approaches the probability density function for the variable being simulated.

A detour through calculus

We have seen that the probability of a variable falling in an interval is estimated by proportion of the overall histogram area falls in the interval—that is, the sum of the histogram areas in the interval. What we want to do is let the number of bins and number of draws continue to increase to get ever better approximations. When we let the number of bins increase toward infinity, we have a familiar limit from integral calculus.

If $p_Y(y)$ is the continuous density function we get as the limit of the histogram, then the probability that Y falls between a and b is given by the proportion of area between a and b in the function $p_Y(y)$. This is the key insight for understanding density functions and continuous random variables. For bounded intervals, we have

$$\Pr[a \leq Y \leq b] \ \propto \ \int_a^b \ p_Y(y) \, \mathrm{d}y.$$

To make our lives easier and avoid writing the proportional-to symbol ($\propto$) everywhere, we will make the conventional assumption that our density functions like p_Y are *normalized*. This means that the total area under their curve is one,

$$\int_{-\infty}^{\infty} p_Y(y) \, \mathrm{d}y \ = \ 1.$$

Because they are based on the limits of histograms, which are counts, we will also meet the standard requirement placed on density functions that they be positive, so that for all $y \in \mathbb{R}$,

$$p_Y(y) \geq 0.$$

With these assumptions in place, we now define interval probabilities using definite integration over density functions,

$$\Pr[a \leq Y \leq b] \ = \ \int_a^b \ p_Y(y) \, \mathrm{d}y.$$

For simple upper bounds, we just integrate from negative infinity,

$$\Pr[Y \leq b] \ = \ \int_{-\infty}^b \ p_Y(y) \, \mathrm{d}y.$$

This reveals the relation between the cumulative distribution function $F_Y) = \Pr[Y \leq b]$ and the probability density function p_Y

$$F_Y(b) \ = \ \int_{-\infty}^b \ p_Y(y) \, \mathrm{d}y.$$

Working the other way around, it reveals that the probability density function is just the derivative of the cumulative distribution function,

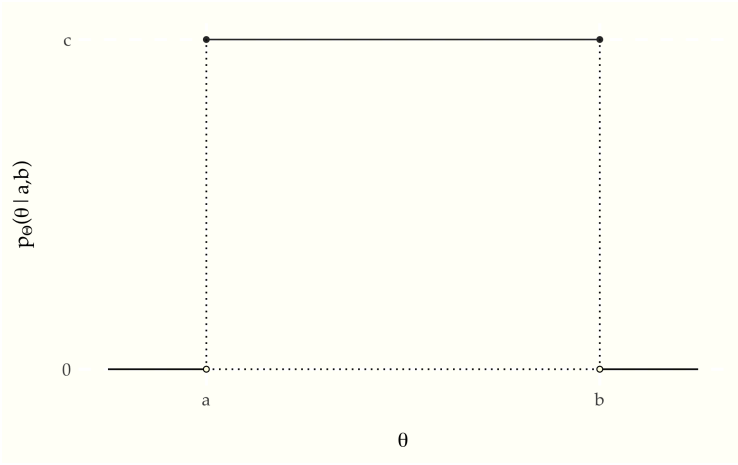
$$p_Y(b) = \frac{\mathrm{d}}{\mathrm{d}y} F_Y(y) \Big|_{y=b}.$$

Thus the units of a probability density function are change in cumulative probability, not probability. Density functions must be integrated to get back to units of probability.

The uniform density function

We've already seen the histograms for variables $\Theta \sim \text{uniform}(0, 1)$ distributed uniformly from zero to one. With an increasing numbers of draws, the histograms flatten out. With more draws the histograms will level out even more until the density becomes a straight line. This means that the probability density function of a uniformly distributed random variable is constant.[130] That is, if $\Theta \sim \text{uniform}(a, b)$, then $p_\Theta(\theta) = c$ for some constant c. Let's see what that looks like so the solution for c becomes evident.

[130] Another way to reach the same conclusion is by calculus. We worked out from first principles that the cumulative distribution function is linear if uniformity means equal probability of landing in any interval of the same size. The derivative of a linear function is constant, so the density for a uniform distribution must be constant.



The plot shows the area from a to b under c to be $(b - a) \times c$. Given that we require the area to be one, that is, $(b - a) \times c = 1$, we can work out c by dividing both sides by $b - a$,

$$c = \frac{1}{b - a}.$$

Putting this into density notation, if $\Theta \sim \text{uniform}(a, b)$, then

$$p_\Theta(\theta) = \text{uniform}(\theta \mid a, b),$$

where we have now worked out that

$$\text{uniform}(\theta \mid a, b) = \frac{1}{b - a}.$$

That is, the density does not depend on y—it is constant and the same for every possible value of θ.[131]

[131] For convenience, we can assume the impossible values of θ have density zero.

Back to simulation

The traditional bottleneck to performing statistics beyond the data collection was wrangling integral calculus to provide analytic results or approximations for a given applied problem. Today, very general

numerical solvers absolve us of the heavy lifting of calculus and replace it with wrangling computer code for simulations. This lets us solve much harder problems directly.

Let's actually solve the integral we mentioned in the last section, namely the probabilty that a log odds variable will land between -2 and 2.

```
success = 0
for (m in 1:M)
  Phi[m] = logit(uniform_rng(0, 1))
  if (-2 < Phi[m] & Phi[m] < 2)
    success += 1
print 'Pr[-2 < Phi < 2] = ' success / M
```

Let's run that for $M = 100\,000$ simulation draws and see what we get,

```
 Pr[-2 < Phi < 2] = 0.76
```

What is perhaps more remarkable than not requiring calculus is that we don't even require the formula for the density functin p_Φ— we only need to be able to simulate random instantiations of the random variable in question.

Laws of probability for densities

Whether a random variable Y is continuous or discrete, its cumulative distribution function F_Y is defined by

$$F_Y(y) = \Pr[Y \leq y].$$

Using simulation, if Y is a continuous random variable, its probability density function p_Y is the limit of the histogram of simulation draws. Using calculus, the density p_Y of a continuous random variable Y is defined as the derivative of the cumulative distribution function F_Y,[132]

$$p_Y = \mathrm{d}F_Y.$$

Joint cumulative distibution functions for a pair of continuous random variables X, Y are defined as expected,

$$F_{X,Y}(x, y) = \Pr[X \leq x \text{ and } Y \leq y],$$

and may be easily extended to more variables. With simulation, cumulative distribution functions may be recreated by sorting the simulated values and normalizing.

[132] Differential notation avoids the fiddly notation arising from bound vaiables, e.g.,

$$p_Y(y) \;=\; \frac{\mathrm{d}}{\mathrm{d}y} F_Y(y).$$

With multivariate functions, the derivative operator is replaced with the gradient operator ∇.

Joint densities for a pair X, Y of continuous random variables are defined by differentiating the joint cumulative distribution twice,[133]

$$p_{X,Y} = \partial^2 F_{X,Y}.$$

Marginal densities p_X may now be defined in terms of the joint density $p_{X,Y}$ by integration,[134]

$$p_X(x) = \int_{-\infty}^{\infty} p_{X,Y}(x,y)\,\mathrm{d}y.$$

With simulation, if we can simulate $x^{(m)}, y^{(m)}$ jointly, then we can simulate $x^{(m)}$ by simply dropping $y^{(m)}$.

If we can simulate from Y, we can compute $p_X(x)$ for a given value of x by averaging,

$$p_X(x) \;\approx\; \frac{1}{M} \sum_{m \in 1:M} p_{X,Y}(x, y^{(m)}).$$

Conditional densities $p_{X|Y}$ are defined by dividing the joint density $p_{X,Y}$ by the marginal density p_X,

$$p_{X|Y}(x \mid y) \;=\; \frac{p_{X,Y}(x,y)}{p_Y(y)}.$$

Conditional densities $p_{X|Y}(x \mid y)$ may be handled by simulation for specific values of y.

Equivalently, we can see this as a definition of the joint density in terms of the conditional and marginal,

$$p_{X,Y}(x,y) \;=\; p_{X|Y}(x \mid y) \times p_Y(y).$$

With simulation, this is often the strategy to generate simulations from the joint distribution—first simulate from Y, then simulate X given Y.

A convenient form of marginalization uses this definition,

$$p_X(x) = \int_{-\infty}^{\infty} p_{X|Y}(x,y) \times p_Y(y)\,\mathrm{d}y.$$

Continuous random variables X and Y are independent if their densities factor, so that for all x, y,

$$p_{X,Y}(x,y) \;=\; p_X(x) \times p_Y(y),$$

or equivalently,

$$p_{X|Y}(x \mid y) \;=\; p_X(x).$$

Expectations for continuous random variables are defined using integration to calculate the average of y weighted by the density $p_Y(y)$,

[133] With bound variables,

$$p_{X,Y}(x,y) = \frac{\partial^2}{\partial x \partial y} F_{X,Y}(x,y).$$

[134] If we had a convenient integral operator, we could avoid the bound variable fiddling. As written, in the traditional style, it is muddied that the integral just averages over y treating x as a variable bound by the function definition notation.

$$\mathbb{E}[Y] \;=\; \int_{-\infty}^{\infty} y \times p_Y(y)\,\mathrm{d}y.$$

In moving from discrete to continuous variables, we have merely switched the definition from summation to integration. Luckily, calculation by simulation need not change—we will still be calculating expectations by averaging over simulated values. If we can simulate $y^{(m)}$ according to $p_Y(y)$ for $m \in 1:M$, our simulation-based estimate is

$$\mathbb{E}[f(y)] \;\approx\; \frac{1}{M}\sum_{m=1}^{M} f\left(y^{(m)}\right).$$

This estimate becomes exact as $M \to \infty$.

Variances are defined in terms of expectation, just as before,

$$\mathrm{var}[Y] \;=\; \mathbb{E}\left[(Y - \mathbb{E}[Y])^2\right] \;=\; \mathbb{E}[Y^2] - (\mathbb{E}[Y])^2.$$

Variances can be estimated through simulation like any other expectation.[135]

[135] The sample variance computed by standard software divides by $M-1$ to correct for the bias introduced by using the sample mean to estimate variance. The maximum likelihood estimate resulting from dividing by M is biased to underestimate variance with finite samples; asymptotically, it provides the correct result, because the $\frac{M}{M-1}$ correction factor approaches one as as M increases.

Jacobians and changes of variables

When we moved from a random variable $\Theta \sim \mathrm{uniform}(0,1)$ to a variable $\Phi = \mathrm{logit}(\Theta)$, we made a class *change of variables*. That means we can use calculus to compute the probability density function. But let's do it in full generality.

We'll start by assuming we have a random variable X with a known density function $p_X(x)$. Assume further we have a smooth and invertible function f and define a new random variable $Y = f(X)$. The density of Y is then given by the rather daunting formula

$$p_Y(y) \;=\; p_X(f^{-1}(y)) \;\times\; \left|\, \frac{\mathrm{d}}{\mathrm{d}u} f^{-1}(u)\Big|_{u=y} \,\right|.$$

We're going to work through this in pieces using our running example. To keep the puzzle pieces straight, let $X = \Theta \sim \mathrm{uniform}(0,1)$ be our uniform probability variable and $Y = \Phi = \mathrm{logit}(\Theta)$ be the transformed variable on the log odds scale. Our goal is to calculate the density of Φ given that we know the density of Θ and the transform from Θ to Φ. We begin by noting that

$$\mathrm{logit}^{-1}(y) = \frac{1}{1+\exp(-y)}.$$

So to evaluate $p_\Phi(\phi)$, we first need to evaluate $p_\Theta(\mathrm{logit}^{-1}(\phi))$. We know this term will evaluate to 1, because $p_\Theta(\theta) = 1$ for every θ. So clearly just inverting and plugging in isn't enough.

We also need to account for the change in variables from Θ to Φ. This is where the Jacobian term comes into the equation—that's everything past the $\times$ sign. The Jacobian is the absolute value of the derivative of the inverse transform evaluated at the value in question. For our running example, we can work out through the chain rule that

$$\frac{\mathrm{d}}{\mathrm{d}u}\text{logit}^{-1}(u) \;=\; \text{logit}^{-1}(u) \times \left(1 - \text{logit}^{-1}(u)\right).$$

So if we plug in $u = \phi$ here, and put all the pieces back together, we get

$$p_\Phi(\phi) \;=\; \text{logit}^{-1}(\phi) \times \left(1 - \text{logit}^{-1}(\phi)\right).$$

This distribution is known as the standard logistic distribution,

$$\text{logistic}(\phi \mid 0, 1) \;=\; \text{logit}^{-1}(\phi) \times \left(1 - \text{logit}^{-1}(\phi)\right).$$

Thus after all the dust has settled, we know that if $\Theta \sim \text{uniform}(0, 1)$ and $\Phi = \text{logit}(\Theta)$, then $\Phi \sim \text{logistic}(0, 1)$.[136]

[136] The meaning of the parameters 0 and 1 will be explained in the next section.

Exponential distributions

We have already seen that if we take a uniformaly distributed variable,

$$U \sim \text{uniform}(0, 1),$$

and log-odds transform it to

$$V = \text{logit}(U),$$

the resulting variable has a standard logistic distribution,

$$V \sim \text{logistic}(0, 1).$$

In this section, we consider a negative log transform,

$$Y = -\log U,$$

We know that if $U \in (0, 1)$, then $\log U \in (-\infty, 0)$, and hence

$$Y = -\log U \in (0, \infty).$$

Let's plot a histogram of simulated values of Y to see what its density looks like. The simulation is trivial.

```
u = uniform_rng(0, 1)
y = -log(u)
```

We'll generate $M = 10^6$ draws and calculate some summary statistics.

```
mean(y) = 0.50
sd(y) = 0.29
central 95 pct interval = (0.03, 0.98)
min = 0.00;  max = 1.00
```

It's clear that the variable has a mean and standard deviation of one, but is highly right skewed.

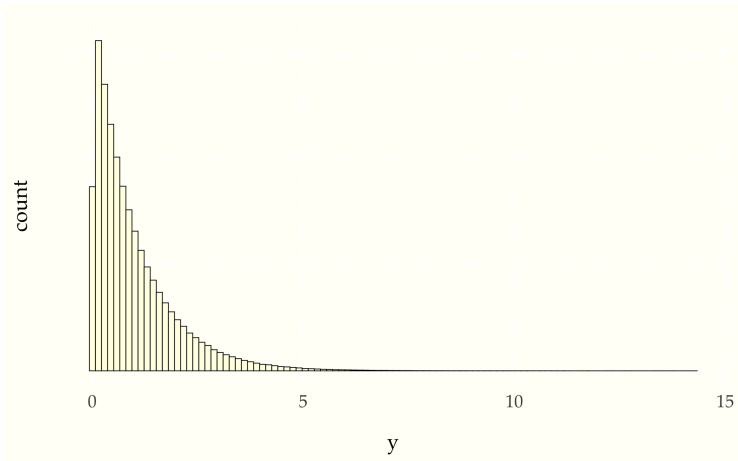


Figure 48: Histogram of $M = 10^6$ draws from $U \sim \text{uniform}(0, 1)$ transformed to $Y = -\log U$. The mean and standard deviation are 1, but the distribution is highly right skewed.

While the histogram plot lets us visualize the density, we can also derive the density p_Y from the uniform density p_U given the transform

$$Y = f(U) = -\log U.$$

First, we calculate the inverse transform,

$$\begin{aligned} Y &= -\log U \\ -Y &= \log U \\ \exp(-Y) &= U \end{aligned}$$

so that

$$f^{-1}(Y) = \exp(-U).$$

We'll need the derivative of this inverse for the Jacobian,[137]

$$\begin{aligned} \tfrac{\mathrm{d}}{\mathrm{d}y} f^{-1}(y) &= \tfrac{\mathrm{d}}{\mathrm{d}y} \exp(-y) \\ &= -\exp(-y). \end{aligned}$$

We can now derive the density of $Y = f(U) = -\log U$, where

[137] We need to employ the chain rule here,

$$\frac{\mathrm{d}}{\mathrm{d}x} a(b(x)) = \left(\frac{\mathrm{d}}{\mathrm{d}u} a(u) \Big|_{u = b(x)} \right) \cdot \left(\frac{\mathrm{d}}{\mathrm{d}x} b(x) \right),$$

with $a(u) = \exp(u)$ and $b(x) = -x$.

$U \sim \text{uniform}(0,1)$,

$$\begin{aligned} p_Y(y) &= p_U(f^{-1}(y)) \cdot \left| \tfrac{\mathrm{d}}{\mathrm{d}y} \exp(-y) \right| \\ &= \text{uniform}(f^{-1}(y) \mid 0,1) \cdot |-\exp(-y)| \\ &= \exp(-y). \end{aligned}$$

The result, as simple as it looks, is a properly normalized density.[138]

This distribution is popular enough to get its own name, the exponential distribution, the standard version of which is conventionally defined by

$$\text{exponential}(y \mid 1) = \exp(-y).$$

As for other distributions, we will write $Y \sim \text{exponential}(1)$ if $p_Y(y) = \text{exponential}(y) = \exp(-y)$.

We will generalize the standard exponential by allowing it to be scaled. Unlike the normal distribution, which scales a standard normal by multiplying it by a scale parameter, it is convential to divide a standard exponential variate by a rate parameter to get a general exponential variate. Suppose $V \sim \text{exponential}(1)$ and we define a new variable $Y = V/\lambda$ for some rate $\lambda > 0$.[139] This gives us a general exponential variate, $Y \sim \text{exponential}(\lambda)$.

To define the general exponential density, we just apply the Jacobian formula again, keeping in mind that our transform is

$$h(V) = V/\lambda,$$

which has an inverse

$$h^{-1}(Y) = \lambda \cdot Y.$$

Plugging this into the formula for a change of variables, we get

$$\begin{aligned} p_Y(y) &= p_V\left(h^{-1}(y)\right) \cdot \left| \tfrac{\mathrm{d}}{\mathrm{d}y} h^{-1}(y) \right| \\ &= p_V(\lambda \cdot y) \cdot \left| \tfrac{\mathrm{d}}{\mathrm{d}y} \lambda \cdot y \right| \\ &= \exp(-\lambda \cdot y) \cdot \lambda. \end{aligned}$$

This gives us the final form of the exponential distribution,

$$\text{exponential}(y \mid \lambda) = \lambda \cdot \exp(-\lambda \cdot y).$$

By construction, we know that if the standard distribution has a mean of 1, then the transformed version has a mean of $1/\lambda$. Similarly, the trasnformed version also has a standard deviation of $1/\lambda$.[140]

[138] Given the derivative

$$\frac{\mathrm{d}}{\mathrm{d}y} -\exp(-y) = \exp(-y),$$

the basic rule for computing definite integrals,

$$\begin{aligned} \textstyle\int_a^b f(x)\,\mathrm{d}x &= \textstyle\int f(x)\,\mathrm{d}x \Big|_{x=b} - \int f(x)\,\mathrm{d}x \Big|_{x=a} \\ &= -\exp(-x)\Big|_{x=b} - -\exp(-x)\Big|_{x=a} \\ &= -\exp(-b) + \exp(-a). \end{aligned}$$

Plugging in $a = 0$ and $b = \infty$ (the latter is really taking a limit), gives us

$$-\exp(-\infty) + \exp(-0) = 0 + 1 = 1.$$

[139] A rate parameter divides a variable in contrast to a rate parameter, which multiplies a parameter.

[140] The inverse cumulative distribution function for the standard exponential is just the function $-\log u$ used to construct the variable. It is thus simple symbolically to derive the cumulative distribution function and the generalized version with a non-unit rate.

Statistical Inference and Inverse Problems

Deductive inference works from facts toward conclusions deterministically. For example, if I tell you that all men are mortal and that Socrates is a man, you can deductively conclude that Socrates is mortal. *Inductive inference*, on the other hand, is a bit more slippery to define, as it works from observations back to facts. That is, if we think of the facts as governing or generating the observations, then induction is a kind of inverse inference. *Statistical inference* is a kind of inductive inference that is specifically formulated as an inverse problem.

Laplace's birth ratio model

The roots of statistical inference lie not in games of chance, but in the realm of public health. Pierre-Simon Laplace was investigating the rate of child births by sex in France in an attempt to predict future population sizes.[141] Laplace reports the following number of live births, gathered from thirty departments of France between 1800 and 1802 was as follows.

sex	live births
male	110 312
female	105 287

[141] Pierre-Simon Laplace. 1812. *Essai philosophique sur les probabilités*. H. Remy. p. lvi of the Introduction. Annotated English translation of the 1825 Fifth Edition: Andrew I. Dale, 1995. *Philosophical Essay on Probabilities*. Springer-Verlag.

Laplace assumed each birth is independent and each has probability $\Theta \in [0, 1]$ of being a boy. Letting Y be the number of male births and N be the total number of births, Laplace assumed the model

$$Y \sim \text{binomial}(N, \Theta).$$

In other words, his data-generating distribution had the probability mass function[142]

$$p_{Y|\Theta}(y \mid \theta) \;=\; \text{binomial}(y \mid N, \theta).$$

[142] The constant N that appears in the full binomial notation is suppressed in the density notation $p_{Y|\Theta}$—it is common to suppress constants in the notation to make the relationship between the modeled data Y and parameters Θ 3easier to scan.

Because it employs a binomial distribution, this model assumes that the sex of each baby is independent, with probability θ of being a

boy. This may or may not be a good approximation to reality. Part of our job is going to be to check the assumptions like this built into our models.

We know how to generate Y given values for the parameter Θ, but we are now faced with the *inverse problem* of drawing inferences about Θ based on observations about Y.

What is a model?

We say that this simple formula is a *model* in the sense that it is not the actual birth process, but rather a mathematical construct meant to reflect properties of the birth process. In this sense, it's like Isaac Newton's model of the planetary motions using differential equations.[143] The equations are not the planets, just descriptions of how they move in response to gravitational and other forces.

Models like Newton's allow us to predict certain things, such as the motion of the planets, the tides, and balls dropped from towers. But they typically only approximate the full process being modeled in some way. Even Newton's model, which is fabulously accurate at predictions at observable scales, is only an approximation to the finer-grained models of motion and gravity introduced by Albert Einstein.[144] which itself was only a special case of the more general theory of relativity.[145] Each successive model is better than the last in that it's better at prediction, more general, or more elegant—science does not progress based on a single criterion for improving models.

The reproductive process is complex, and many factors may impinge on the sex of a baby being born. Part of our job as scientists is to check the assumptions of our models and refine them as necessary. This needs to be done relative to the goal of the model. If the goal of this simple reproductive model is only to predict the prevalence of male births at a national scale, then a simple, direct prevalence model with a single parameter like the one we have introduced may be sufficient.

To conclude, when we say "model", all we have in mind is some mathematical construct taken to represent some aspect of reality. Whether a model is useful is a pragmatic question which must be judged relative to its intended application.

What is a random variable?

As in all statistical modeling, Laplace treated the observed number of male births Y as a random variable. This assumes a form of counterfactual reasoning whereby we assume the world might have been some other way than it actually turned out to be.

[143] Isaac Newton. 1687. *Philosophiae Naturalis Principia Mathematica*. Translated as I. Bernard Cohen and Anne Whitman. 1999. *The Principia: Mathematical Principles of Natural Philosophy.* University of California Press.

[144] Einstein, Albert. 1907. Über das Relativitätsprinzip und die aus demselben gezogenen Folgerungen. (English translation: On the relativity principle and the conclusions drawn from it.) *Jahrbuch der Radioaktivität und Elektronik* 4:411–462.

[145] Einstein, Albert. 1916. The foundation of the general theory of relativity. *Annalen Phys.* 14:769–822.

As in most statistical models, Laplace treated N as a constant. In many cases, the denominator of binary events is not itself a constant, but is itself a random variable determined by factors of the environment. For instance, the number of attempts an athlete on a sports team get depends on the ability of that athlete and the number of reviews a movie receives depends on its popularity.

As originally formulated by Thomas Bayes,[146] Laplace also treated Θ as a random variable. That is, Laplace wanted to infer, based on observation and measurement, that the probability that Θ's value was in a certain range. Specifically, Laplace was curious about the question of whether the male birth rate is higher, which can be expressed in probabilistic terms by the event probability $\Pr[\Theta > 0.5]$.

[146] Bayes, T., 1763. LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFRS. *Philosophical Transactions of the Royal Society*, pp. 370–418.

Laplace's inverse problem

Given a total of N births, we have introduced random variables for

- the observed data of Y male births, and
- the probability Θ that a live birth will result in a boy.

We also have the actual observed number of male births, y. That is, we know the value of the random variable Y. Given our observed data, we can ask two obvious questions, namely

- What is the probability of a boy being born?
- Is it more likely that a boy is born than a girl?

Given that Θ is the male birth rate, the first question is asking about the value of Θ. To provide a probabilistic answer, we want to look at the distribution of Θ given that we observe the actual data $Y = y$, which has the density $p_{\Theta|Y}(\theta \mid y)$. We can summarize this distribution probabilistically using intervals, for instance by reporting the central 95% interval probability,

$$\Pr\left[0.025 \leq \Theta \leq 0.975 \;\middle|\; Y = y\right].$$

The second question, namely whether boys are more likely to be born, is true if $\Theta > \frac{1}{2}$. The probability of this event is

$$\Pr\left[\Theta > \frac{1}{2} \;\middle|\; Y = y\right].$$

If we can estimate this event probability, we can answer Laplace's second question.[147]

[147] The quality of the answer will be determined by the quality of the data and the quality of the model.

Bayes's rule to solve the inverse problem

The model we have is a *generative model*[148]—it works from a parameter value θ to the observed data y through a *sampling distribution* with probability function $p_{Y|\Theta}(y \mid \theta)$. What we need to solve our inference problems is the *posterior density* $p_{\Theta|Y}(\theta \mid y)$. Bayes realized that the posterior could be defined in terms of the sampling distribution as

$$\begin{aligned} p_{\Theta|Y}(\theta \mid y) &= \frac{p_{Y|\Theta}(y \mid \theta) \times p_{\Theta}(\theta)}{p_Y(y)} \\ &\propto p_{Y|\Theta}(y \mid \theta) \times p_{\Theta}(\theta). \end{aligned}$$

All of our sampling algorithms will work with densities known only up to a proportion.

[148] Also known as a *forward model* or a *mechanistic model* by scientists.

The prior distribution

This still leaves the not inconsequential matter of how to determine $p_{\Theta}(\theta)$, the density of the so-called *prior distribution* of Θ. The prior distribution encapsulates what we know about the parameters Θ before observing the actual data y. This prior knowledge may be derived in many different ways.

- We may have prior knowledge from physical constraints. For example, if we are modeling the concentration of a molecule in a solution, the concentration must be positive.
- We may have prior knowledge of the basic scale of the answer from prior scientific knowledge. For example, if we are modeling human growth, we know that heights above two meters are rare and heights above three meters are unheard of.
- We may have prior knowledge from directly related prior experiments. For example, if we are doing a Phase II drug trial, we will have data from the Phase I trial, or we may have data from Europe if we are doing a trial in the United States.
- We may have experiments from indirectly related trials. For example, if we are modeling football player abilities, we have years and years of data from prior seasons. We know nobody is going to average 10 points a game—it's just not done.

Because we are working probabilistically, our prior knowledge will itself be modeled with a probability distribution, say with density $p_{\Theta}(\theta)$. The prior distribution may depend on parameters, which may be constants or may themselves be unknown. This may seem like an awfully strong imposition to have to express prior knowledge as a density. If we can express our knowledge well and sharply in a distribution, we will have an *informative prior.* Luckily, because we

are only building approximate models of reality, the prior knowledge model does not need to be perfect. We usually err on the side of underpowering the prior a bit compared to what we really know, imposing only *weakly informative priors*, such as those that determine scales, but not exact boundaries of parameters.[149]

We will have a lot to say about prior knowledge later in the book, but for now we can follow Laplace in adopting a uniform prior for the rate of male births,

$$\Theta \sim \text{uniform}(0, 1).$$

In other words, we assume the prior density is given by

$$p_\Theta(\theta) = \text{uniform}(\Theta \mid 0, 1).$$

Here, the bounds zero and one, expressed as constant parameters of the uniform distribution, are logical constraints imposed by the fact that the random variable Θ denotes a probability.

Other than the logical bounds, this uniform prior distribution is saying a value in the range 0.01 to 0.05 is as likely as one in 0.48 to 0.52. This is a very weak prior indeed compared to what we know about births. Nevertheless, it will suffice for this first analysis.

[149] The notion of a truly *uninformative prior* is much trickier, because to be truly uninformative, a prior must be scale free.

The proportional posterior

With a prior and likelihood,[150] we have our full joint model in hand,

$$p_{Y|\Theta}(y \mid N, \theta) \times p_\Theta(\theta) \;=\; \text{binomial}(y \mid N, \theta) \times \text{uniform}(\theta \mid 0, 1).$$

We have carried along the constant N so we don't forget it, but it simply appears on the right of the conditioning bar on both sides of the equation.

The sampling distribution $p(y \mid \theta)$ is considered as a density for y given a value of θ. If we instead fix y and view $p(y \mid \theta)$ as a function of θ, it is called the *likelihood function*. As a function, the likelihood function is not itself a density. Nevertheless, it is crucial in posterior inference.

With Bayes's rule, we know the posterior is proportional to the prior times the likelihood,

$$\underbrace{p_{\Theta|Y}(\theta \mid y)}_{\text{posterior}} \;\propto\; \underbrace{p_{Y|\Theta}(y \mid \theta)}_{\text{likelihood}} \times \underbrace{p_\Theta(\theta)}_{\text{prior}}.$$

Given the definitions of the relevant probability functions,[151] we have

[150] Remember, the likelihood is just the sampling distribution $p_{Y|\Theta}(y \mid \theta)$ viewed as a function of θ for fixed y.

[151] For reference, these are the likelihood

$$\text{binomial}(y \mid N, \theta) \;\propto\; \theta^y \times (1 - \theta)^{N-y}$$

and the prior

$$\text{uniform}(\theta \mid 0, 1) \;=\; 1.$$

$$\begin{aligned} p_{\Theta|Y}(\theta \mid y, N) &\propto \text{binomial}(y \mid N, \theta) \times \text{uniform}(\theta \mid 0, 1) \\ &\propto \theta^y \times (1 - \theta)^{N-y} \end{aligned}$$

To summarize, we know the posterior $p_{\Theta|Y}$ up to a proportion, but are still missing the normalizing constant so that it integrates to one.[152]

[152] We return to the normalizer later when we discuss the beta distribution.

Sampling from the posterior

Now that we have a formula for the posterior up to a proportion, we are in business for sampling from the posterior. All of the sampling algorithms in common use require the density only up to a proportion.

For now, we will simply assume a method exists to draw a sample $\theta^{(1)}, \cdots, \theta^{(M)}$ where each $\theta^{(m)}$ is drawn from the posterior $p_{\Theta|y}(\theta \mid y)$ given the observed data y.

When we do begin to employ general samplers, they are going to require specifications of our models that are exact enough to be programmed. Rather than relying on narrative explanation, we'll use a pseudocode for models that can be easily translated for an assortment of posterior samplers.[153]

[153] This specification is sufficient for coding a sampler in BUGS, Edward, emcee, Greta, JAGS, NIMBLE, PyMC, Pyro, or Stan.

Name	simple binomial	
Data	$N \in \mathbb{N}$	
	$y_n \in \{0, 1\}$	for $n \in 1 : N$
Parameters	$\theta \in (0, 1)$	
Prior	$\theta \sim \text{uniform}(0, 1)$	
Likelihood	$y_n \sim \text{binomial}(N, \theta)$	for $n \in 1 : N$

Simulating data

Rather than starting with Laplace's data, which will present computational problems, we will start with some simulated data. We simulate data for a model by simulating the parameters from the prior, then simulating the data from the parameters. That is, we run the model in the forward direction from prior to parameters to data. This is usually how we construct the models in the first place, so this should be a natural step. In pseudocode, this is a two-liner.

```
theta = uniform_rng(0, 1)
y = binomial_rng(N, theta)
print 'theta = ' theta ';  y = ' y
```

Before we can actually simulate, we need to set the constants, because they don't have priors. Here, we'll just take $N = 10$ for pedagogical convenience. Let's run it a few times and see what we get.

```
theta = 0.29;  y = 4
theta = 0.41;  y = 6
theta = 0.94;  y = 10
theta = 0.53;  y = 3
theta = 0.55;  y = 6
```

The values simulated for θ are not round numbers, so we know that we won't satisfy $y = N \times \theta$, the expected value of a random variable Y such that $Y \sim \text{binomial}(N, \theta)$. From an estimation perspective, we won't have $\theta = y/N$, either. So the question becomes what are reasonable values for θ based on our observation of y? That's precisely the posterior, so let's proceed to sampling from that. We'll just assume we have a function that samples from the posterior of a model with a given name when passed the data for the model. Here, the data consists of the values of y and N, and we will run $M = 1\,000$ iterations.

```
N = 10
y = 3
theta[1:M] = posterior_sample('simple binomial', y, N)

print 'theta = ' theta[1:10] '...'
```

Let's run that and see what a few posterior draws look like.

```
theta =
0.41  0.16  0.23  0.27  0.56  0.39  0.40  0.35  0.25  0.42
```

It's hard to glean much from the draws. What it does tell us is that the posterior in the range we expect it to be in—near 0.3, because the data was $y = 3$ boys in $N = 10$ births. The first thing we want to do with any posterior is check that it's reasonable.

For visualizing draws of a single variable, such as the proportion of boy births θ, histograms are handy.

Let's up M to 1 000 000 and double the number of bins to get a better look at the posterior density. [154]

Histograms have their limitations. The distribution is slightly asymmetric, with a longer tail to the right than to the left, but asymmetry can be difficult to detect visually until it is more extreme than here. Asymmetric distributions are said to be *skewed*, either to the right or left, depending on which tail is longer.[155] It's also hard to tell the exact location of the posterior mean and median visually.

[154] A sample size $M > 100$ is rarely necessary for calculating estimates, event probabilities, or other expectations conditioned on data. For histograms, many draws are required to ensure low relative error in every bin so that the resulting histogram is smooth.

[155] The formal measurement of the *skew* of a random variable Y is just another expectation that may be estimated via simulation,

$$\text{skew}[Y] = \mathbb{E}\left[\left(\frac{Y - \mathbb{E}[Y]}{\text{sd}[Y]}\right)^3\right].$$

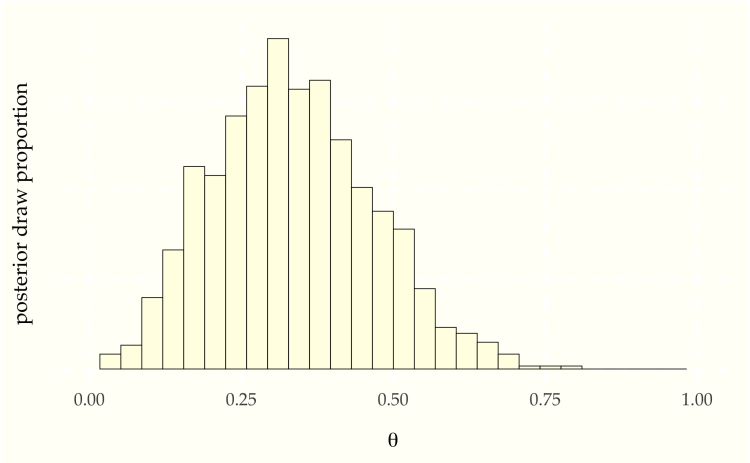


Figure 49: Histogram of one thousand draws from the posterior $p(\theta \mid y)$. With thirty bins, the histogram appears ragged, but conveys the rough shape and location of the posterior.

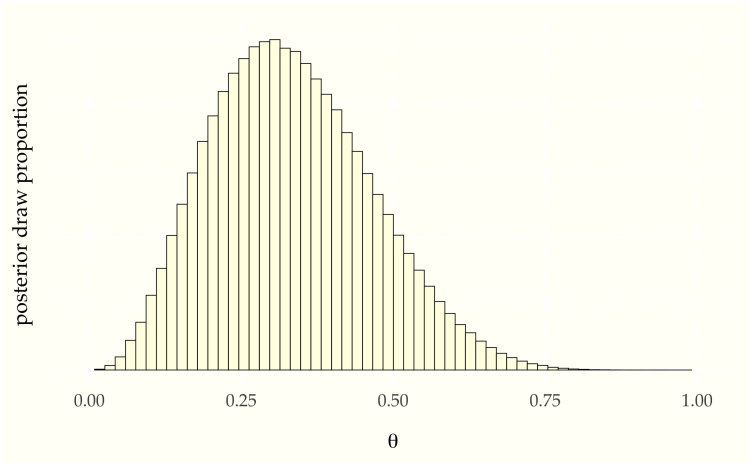


Figure 50: Histogram of one million draws from the posterior $p(\theta \mid y)$. A *much* larger M is required to get a fine-grained view of the whole posterior distribution than is required for an accurate summary statistic.

Posterior summary statistics

We often want to look at summaries of the posterior, the posterior mean, standard deviation, and quantiles being the most commonly used in practice. These are all easily calculated based on the sample draws.

Calculating the posterior mean and standard deviation are as simple as calling built-in mean and standard deviation functions,

```
print 'estimated posterior mean = ' mean(theta) '
print 'estimated posterior sd = ' sd(theta) '
```

Let's see what we get.

```
 estimated posterior mean = 0.33
 estimated posterior   sd = 0.13
```

The posterior mean and standard deviation are excellent marginal summary statistics for posterior quantities that have a roughly normal distribution.[156] If the posterior distribution has very broad or narrow tails or is highly skewed, standard deviation and mean are less useful.

[156] Most posterior distributions we will consider approach normality as more data is observed.

We can estimate quantiles just as easily, assuming we have built-in functions to compute quantiles.

```
print 'estimated posterior median = ' quantile(theta, 0.5)
print 'estimated posterior central 80 pct interval = '
      quantiles(theta, { 0.1, 0.9 })
```

Running this produces the following.[157]

[157] The median is slightly lower than the mean, as they will be in right skewed distributions.

```
 estimated posterior median = 0.32
 estimated posterior central 90 pct interval = (0.17, 0.51)
```

The posterior simulations and summaries answer Laplace's question about the value of θ, i.e., the proportion of boys born, at least relative to this tiny data set.

We have reported a central 90% interval here. It is a 90% interval in the sense that it is 90% probable to contain the value (relative to the model, as always). We have located that interval centrally in the sense that it runs from the 5% quantile to the 95% quantile.

There is nothing privileged about the width or location of a posterior interval. A value is as likely to be in a posterior interval from the 1% quantile to the 91% quantile, or from the 10% quantile to the 100% quantile. The width is chosen to be convenient to reason about. With a 90% interval, we know roughly nine out of ten values will fall within it, and choosing a central interval gives us an idea of the central part of the distribution.

Estimating event probabilities

To answer the question about whether boys are more prevalent than girls, we need to estimate $\Pr[\theta > 0.5]$, which is straightforward with simulation. As usual, we just count the number of times that the simulated value $\theta^{(m)} > 0.5$ and divide by the number of simulations M,

```
print 'estimated Pr[theta > 0.5] = ' sum(theta > 0.5) / M
```

Running this, we see that with 3 boys in 10 births, the probability boys represent more than 50% of the live births is estimated, relative to the model, to be

```
estimated Pr[theta > 0.5] = 0.11
```

Now let's overlay the median and central 90% interval.

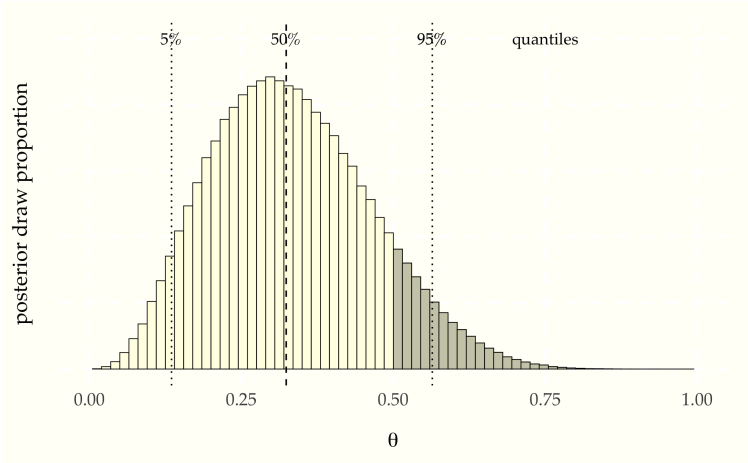


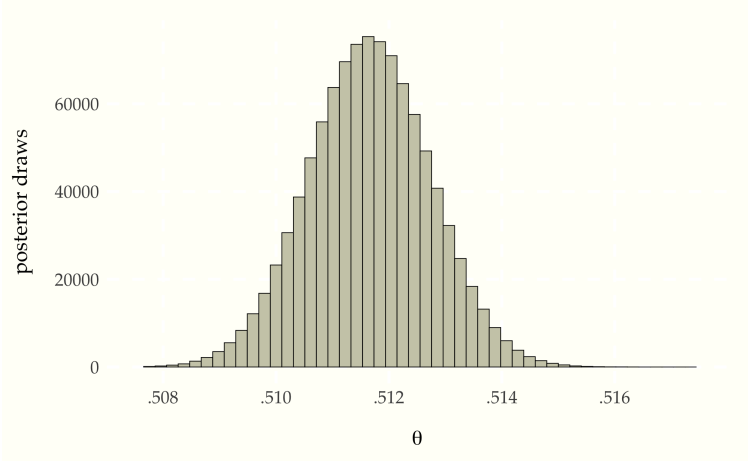
Figure 51: Histogram of 1 000 000 draws from the posterior

$$p(\theta \mid y, N) \propto \text{binomial}(y \mid N, \theta),$$

given $N = 10, y = 3$. The median (50 percent quantile) is indicated with a dashed line and the boundaries of the central 90 percent interval (5 percent and 95 percent quantiles) are picked out with dotted lines. The proportion of the total area shaded to the right of 0.5 represents the posterior probability that $\theta > 0.5$, which is about 11 percent.

Laplace's data

What happens if we use Laplace's data, rather than our small data set, which had roughly 110 thousand male births and 105 thousand female? Let's take some draws from the posterior $p(\theta \mid y, N)$ where $y = 110\,312$ boys out of $N = 110\,312 + 105\,287$ total births. We'll take $M = 1\,000\,000$ simulations $\theta^{(1)}, \ldots, \theta^{(M)}$ here because they are cheap and we would like low sampling error.



The mean of the posterior sample is approximately 0.511, or a slightly higher than 51% chance of a male birth. The central 90% posterior interval calculated from quantiles of the sample is $(0.510, 0.513)$.

What about the event probability that boy births are more likely than girl births, i.e., $\Pr[\theta > 0.5]$? If we make our usual calculation, taking draws $\theta^{(1)}, \ldots, \theta^{(M)}$ from the posterior and look at the proportion for which $\theta^{(m)} > 0.5$, the result is 1. No decimal places, just 1. If we look at the draws, the minimum value of $\theta^{(m)}$ in $1\,000,000$ draws was approximately 0.506. The proportion of draws for which $\theta^{(m)} > 0.5$ is thus 100%, which forms our estimate for $\Pr[\theta > 0.5]$.

As we have seen before, simulation-based estimates provide probabilistic guarantees about absolute tolerances. With 100 000 draws, we are sure that the answer is 1.0000 to within plus or minus 0.0001 or less.[158] We know the answer must be strictly less than one. Using some analytic techniques,[159] the true estimate to within 27 decimal places is

$$\Pr[\theta > 0.5] = 1 - 10^{-27}.$$

Thus Laplace was certain that the probability of a boy being born was higher than that of a girl being born.

[158] Tolerances can be calculated using the central limit theorem, which we will define properly when we introduce the normal distribution later.

[159] The cumulataive distribution function of the posterior, which is known to be the beta distribution

$$p(\theta \mid y, N) = \text{beta}(\theta \mid y + 1, N - y + 1).$$

Inference for and comparison of multiple variables

The first example of Laplace's is simple in that it has only a single parameter of interest, θ, the probability of a male birth. Now we will consider a very similar model with two variables, so that we can do some posterior comparisons. We will consider some simple review data for two New York City-based Mexican restaurants. The first contender is Downtown Bakery II, an East Village Mexican restaurnat has $Y_1 = 114$ out of $N_1 = 235$ 5-star reviews on Yelp, La Delicias

Mexicanas, in Spanish Harlem, has $Y_1 = 24$ out of $N_2 = 51$ 5-start reviews. Our question is, which is more likely to deliver a 5-star experience? In terms of proportion of 5-star votes, they are close, with Downtown Bakery garnering 49% 5-star reviews and La Delicias only 47%. Knowing how noisy binomial data is, this is too close to call.

We'll model each restaurant independently for $n \in 1:2$ as

$$Y_n \sim \text{binomial}(N, \Theta_n)$$

with independent uniform priors for $n \in 1:2$ as

$$\Theta_n \sim \text{uniform}(0, 1).$$

We can now draw $\theta^{(1)}, \ldots, \theta^{(M)}$ simulations from the posterior $p_{\Theta|Y,N}(\theta \mid y, N)$ as usual.

The main event is whether $\theta_1 > \theta_2$—we want to know if the probability of getting a five-star review is higher at Downtown Bakery than La Delicias. All we need to do is look at the posterior mean of the indicator function $\text{I}[\theta_1 > \theta_2]$. The calculus gets more complicated—a double integral is now required because there are two variables. The simulation-based estimate, on the other hand, proceeds as before, counting proportion of draws in which the event is simulated to occur.

$$\begin{aligned} \Pr[\theta_1 > \theta_2 \mid y, N] &= \int_0^1 \int_0^1 \text{I}[\theta_1 > \theta_2] \times p(\theta_1, \theta_2 \mid y, N) \, \mathrm{d}\theta_1 \, \mathrm{d}\theta_2 \\ &\approx \tfrac{1}{M} \textstyle\sum_{m=1}^{M} \text{I}[\theta_1^{(m)} > \theta_2^{(m)}]. \end{aligned}$$

In pseudocode, this is just

```
success = 0
for (m in 1:M)
  draw theta(m) from posterior p(theta | y, N)
  if (theta(m)[1] > theta(m)[2])
    success += 1
print 'Pr[theta[1] > theta[2] | y, M] = ' success / M
```

Let's run that with $M = 10\,000$ simulations and see what we get:

```
 Pr[theta[1] > theta[2] | y, M] = 0.52
```

Only about a 52% chance that Downtown Bakery is the better bet for a 5-star meal.[160]

To get a sense of the posterior, we can construct a histogram of posterior draws of $\Delta = \Theta_1 - \Theta_2$.

[160] As much as this diner loves Downtown Bakery, the nod for food, ambience, and the existence of beer goes to La Delicias Mexicanas.

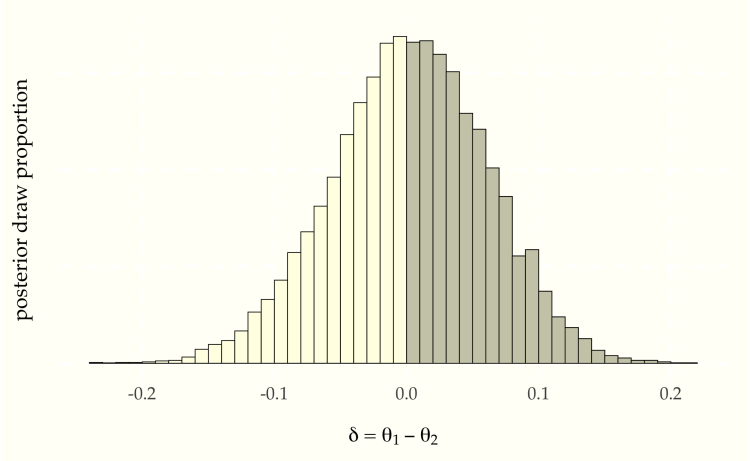


Figure 52: Histogram of posterior differences between probabilty of Downtown Bakery getting a 5-star review (θ_1) and that of La Delicias Mexicanas getting one (θ_2). The draws for which $\delta > 0$ (equivalently, $\theta_1 > \theta_2$) are shaded darker. The area of the darker region divided by the total area is the estimate of the probability that Downtown Bakery is more likely to get a 5-star review than La Delicias Mexicanas.

There is substantial uncertainty, and only 52% of the draws lie to the right of zero. That is,

$$\Pr[\theta_1 > \theta_2] \;=\; \Pr[\delta > 0] \;\approx\; 0.52.$$

Rejection Sampling

Inverse cumulative distribution generation

We have so far assumed we have a uniform random number generator that can sample from a uniform$(0, 1)$ distribution. That immediately lets us simulate from a uniform(a, b).[161] But what if we want to simulate realizations of a random variable Y whose density p_Y is not uniform?

If we happen to have a random variable Y for which we can compute the inverse $F_Y^{-1}(p)$ of the cumulative distribution function for $p \in (0, 1)$,[162] then we can simulate random realizations of Y as follows. First, simulate a uniform draw U,

$$u^{(m)} \sim \text{uniform}(0, 1),$$

then apply the inverse cumulative distribution function to turn it into a simulation of Y,

$$y^{(m)} = F^{-1}(u^{(m)}).$$

It turns out we've already seen an example of this strategy—it is how we we simulated from a logistic distribution right off the bat. The log odds transform is the inverse cumulative distribution function for the logistic distribution. That is, if

$$Y \sim \text{logistic}(0, 1),$$

then

$$F_Y(y) = \text{logit}^{-1}(y) = p$$

and hence

$$F_Y^{-1}(p) = \text{logit}(p) = y.$$

This lets us simulate Y by first simulating $U \sim \text{uniform}(0, 1)$, then setting $Y = \text{logit}(U)$. Both the log odds function and its inverse are easy to compute. Often, the inverse cumulative distribution function is an expensive computation.[163]

[161] If

$$U \sim \text{uniform}(0, 1)$$

then

$$a + U \times (b - a) \sim \text{uniform}(a, b).$$

[162] If $F_Y^{-1}(p) = y$ then $F^{-1}(y) = \Pr[Y \leq y] = p$.

[163] In general, the inverse cumulative distribution function $F_Y^{-1}(p)$ can be computed by solving

$$F_Y(u) = \int_{\infty}^{u} p_Y(y) \, \mathrm{d}y = p$$

for u, which can typically be accomplished numerically if not analytically.

Beta distribution

As a first non-trivial example, let's consider the beta distribution, which arises naturally as a posterior in binary models. Recall that when we have a binomial likelihood $\text{binomial}(y \mid N, \theta)$ and a uniform prior $\theta \sim \text{uniform}(0, 1)$, then the posterior density is

$$p(\theta \mid y) \ \propto \ \theta^y \times (1 - \theta)^{N-y}.$$

This turns out to be a beta distribution with parameters $\alpha = y + 1$ and $\beta = N - y + 1$. If a random variable $\Theta \in (0, 1)$ has a *beta distribution*, then its density p_Θ is

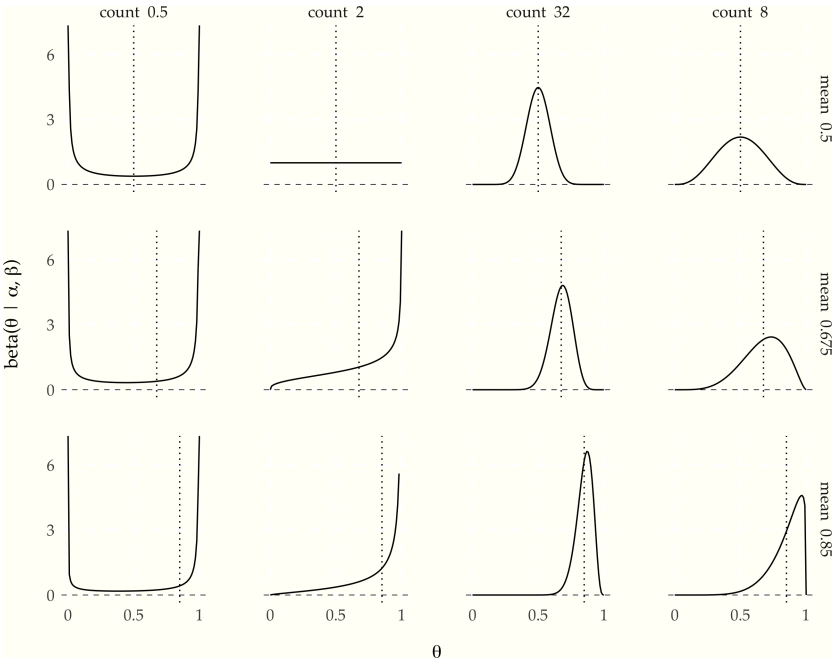
$$\text{beta}(\theta \mid \alpha, \beta) \ \propto \ \theta^{\alpha-1} \times (1 - \theta)^{\beta-1}$$

for some $\alpha, \beta > 0$.[164]

If $Y \sim \text{beta}(\alpha, \beta)$, then its expected value is

$$\mathbb{E}[Y] = \frac{\alpha}{\alpha + \beta}.$$

It is convenient to work with the beta distribution parameters in terms of the mean $\alpha/(\alpha + \beta)$ and the total count $\alpha + \beta$.[165] The higher the total count, lower the variance. For example, here is a plot of a few beta distributions organized by total count and mean.



For example, a total count of $\alpha + \beta = 8$ and mean of $\alpha/(\alpha + \beta) = 0.85$ corresponds to beta distribution parameters $\alpha = 8 \times 0.85 = 6.8$ and $\beta = 8 \times (1 - 0.85) = 1.2$.

[164] We prefer to present densities up to normalizing constants, because the normalizing constants are distracting—what matters is how the density changes with the variate. Here, the fully normalized distribution is

$$\text{beta}(\theta \mid \alpha, \beta) \ = \ \frac{1}{\text{B}(\alpha, \beta)} \, \theta^{\alpha-1} \times (1 - \theta)^{\beta-1},$$

where Euler's beta function, which gives its namesake distribution its name, is defined as the integral of the unnormalized beta density

$$\text{B}(\alpha, \beta) \ = \ \int_0^1 \theta^{\alpha-1} (1 - \theta)^{\beta-1} \, \text{d}\theta.$$

Hence, it's clear the beta density integrates to 1 for any α, β.

[165] Distributions have means, random variables have expectations. The mean of a distribution is the expectation of a random variable with that distribution, so the terms are often conflated.

Figure 53: Plots of the densities

The plot shows that when the mean is 0.5 and the count is 2 (i.e., $\alpha = \beta = 1$), the result is a uniform distribution. The algebra agrees,

$$\begin{aligned} \text{beta}(\theta \mid 1, 1) &\propto \theta^{1-1} \times (1-\theta)^{1-1} \\ &= 1 \\ &= \text{uniform}(\theta \mid 0, 1). \end{aligned}$$

The area under each of these curves, as drawn, is exactly one. The beta distributions with small total count ($\alpha + \beta$) concentrate most of their probability mass near the boundaries. As the count grows, the probability mass concentrates away from the boundaries and around the mean.[166]

Each of these distributions has a well-defined mean, as shown in row labels in the plot. But they do not all have well defined modes (maxima). For example, consider the beta distribution whose density is shown in the upper left example in the plot. It shows a U-shaped density for a $\text{beta}(0.25, 0.25)$ distribution, which corresponds to mean $0.5 = 0.25/(0.25 + 0.25)$ and total count $0.5 = 0.25 + 0.25$. As θ approaches either boundary, 0 or 1, the density grows without bound. There is simply no maximum value for the density.[167]

[166] Nevertheless, in higher dimensions, the curse of dimensionality rears its ugly head, and concentrates the total mass in the corners, even if each dimension is independently distributed and drawn from a distribution concentrated away from the edges in one dimension. The reason is the same—each dimension's value squared just pulls the expected distance of a draw further from the multidimensional mean.

[167] Despite the lack of a maximum, the area under the density is one. The region of very high density near the boundary becomes vanishingly narrow in order to keep the total area at one.

Uniform, bounded rejection sampling

How do we sample from a beta distribution? Rejection sampling is very simple algorithm to sample from general distributions for which we know how to compute the density. It is a good starter algorithm because it is easy to understand, but points the way toward more complex sampling algorithms we will consider later.

Let's start with the simplest possible kind of rejection sampling where we have a bounded distribution like a beta distribution. The values drawn from a beta distribution are bounded between 0 and 1 by construction. Furthermore, if $\alpha, \beta > 1$, as we will assume for the time being, then there is a maximum value for $\text{beta}(\theta \mid \alpha, \beta)$.[168]

For concreteness, let's start specifically with a $\text{beta}(6.8, 1.2)$ distribution, corresponding to the mean 0.85 and count of 8 case from the previous section. We observe that all values fall below 5, so we will create a box with height 5 and width of 1.[169] Next, we draw uniformly from the box, drawing a horizontal position $\theta^{(m)} \sim \text{uniform}(0, 1)$ and vertical position $u^{(m)} \sim \text{uniform}(0, 5)$. The points whose value for u falls below the density at the value for θ are retained.

More specifically, we keep the values $\theta^{(m)}$ where

$$u^{(m)} < \text{beta}(\theta^{(m)} \mid 6.8, 1.2).$$

[168] The maximum density occurs at the *mode* of the distribution, which for $\text{beta}(\theta \mid \alpha, \beta)$ is given by

$$\theta^* = \frac{\alpha - 1}{\alpha + \beta - 2}.$$

[169] For an unknown distribution, we could use an optimization algorithm to find the largest value.

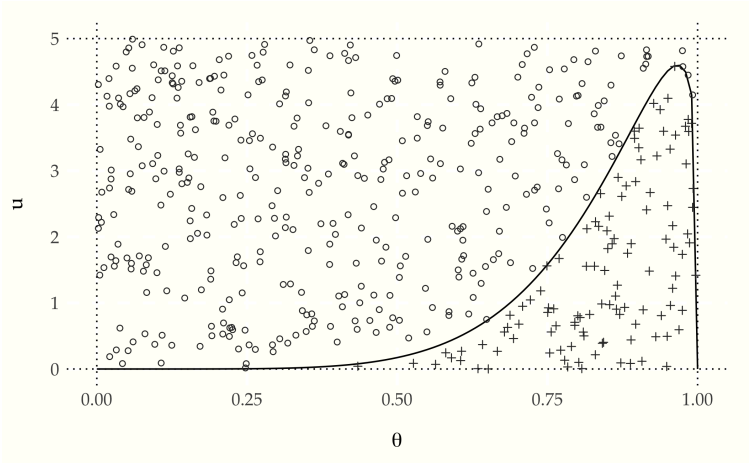


Figure 54: A simple instance of rejection sampling from a bounded beta(6.8, 1.2) distribution, whose density is shown as a solid line. Points (θ, u) are drawn uniformly from the rectangle, then accepted as a draw of θ if u falls below the density at θ. The accepted draws are rendered as plus signs and the rejected ones as circles. The acceptance rate here is roughly 20 percent.

The set of accepted draws that are retained are distributed uniformly in the area under the density plot. The probability of a draw from rejection sampling for variable U falling between points a and b is proportional to the area under the density curve between a and b, which is the correct probability by the definition of the probability density function p_U, which ensures

$$\Pr[a \leq U \leq b] = \int_a^b p_U(u) \, \mathrm{d}u.$$

If the intervals are right, the density is right.

Rejection sampling algorithm

To generate a single draw from beta(6.8, 1.2), we continually sample points (u, θ) uniformly until we find one where u falls below the density value for θ, then return the θ value. Because the variable u is only of use for sampling, it is called an *auxiliary variable*. Many sampling methods use auxiliary variables. The rejection sampling algorithm written out for our particular case is as follows.

```
while (true)
  u = uniform_rng(0, 5)
  theta = uniform_rng(0, 1)
  if (u < beta(theta | 6.8, 1.2))
    return theta
```

Let's run this algorithm for $M = 100\,000$ iterations and see what the histogram looks like.

This looks like it's making the appropriately distributed draws from the beta distribution.[170]

[170] After we introduce the normal distribution, we will develop a χ^2 test statistic for whether a given set of draws come from a specified distribution. For now, we perform the inaccurate test known informally as "χ by eye."

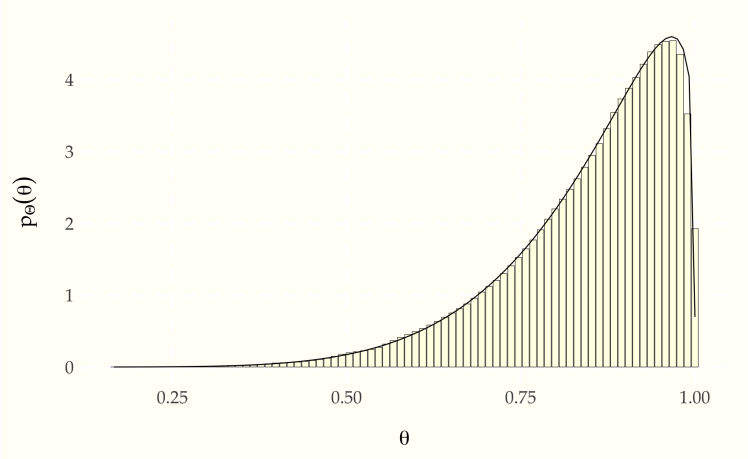


Figure 55: Histogram of $M = 100\,000$ Draws from beta$(6.8, 1.2)$ made via rejection sampling. The true density is plotted over the histogram as a line. The acceptance rate for draws was roughly 20 percent.

Acceptance, concentration, and the curse of dimensionality

As noted in the caption of the plot, the acceptance rate is only 20% for the uniform proposals. This acceptance rate can become arbitrarily bad with uniform proposals as the true distribution from which we want to sample concentrates around a single value. As such, rejection sampling's value is as a pedagogical example for introducing generic sampling and also as a component in more robust sampling algorithms.

The algorithm could also be extended to higher dimensions, but the problem of rejection becomes worse and worse due to the curse of dimensionality. If the posterior is concentrated into a reasonably small area (even as mildly in our beta$(6.8, 1.2)$ example) in each dimension, the chance of a random multidimensional sample being accepted in every dimension is only 0.2^N, which becomes vanishingly small for practical purposes even in 10 dimensions.[171]

[171] In ten dimensions, overall acceptance of a uniform draw would be 0.2^{10}, or a little less than one in ten million. While this may be possible with patience, another ten dimensions becomes completely unworkable, even with massive parallelism.

General rejection sampling

The more general form of rejection sampling takes a proposal from an arbitrary scaled density. In the example of the previous section, we used a uniform$(0, 1)$ distribution scaled by a factor of five. The acceptance procedure in the general case remains exactly the same.

In the general case, suppose we want to draw from a density $p(y)$. We'll need a density $q(y)$ from which we know how to simulate draws, and we'll need a constant c such that

$$c \times q(y) > p(y)$$

for all y.[172]

[172] This means the support of the proposal distribution must be at least as large as that of the target distribution.

The general rejection sampling algorithm for target density $p(y)$, proposal density $q(y)$ with constant c such that $c \times q(y) > p(y)$ for all y, is as follows:

```
while (true)
  y = q_rng(y)
  u = uniform(0, c * q(y))
  if (u < p(y)) return y
```

Our simplified algorithm in the previous section was just a special case where $q(y)$ is uniform over a bounded interval. The argument for correctness in the more general case is identical—draws are made proportional to the area under the density, which provides the correct distribution.

Posterior Predictive Inference

One of the primary reasons we fit models is to make predictions about the future. More specifically, we want to observe some data y and use it to predict future data $\tilde{y}$. Even more specifically, we'd like to understand the probability distribution of the future data $\tilde{y}$ given the observed data y.

We are going to assume that we are still working relative to a model whose sampling distribution has a density $p(y \mid \theta)$.[173] Thus if we knew the value of θ,[174] the distribution of $\tilde{y}$ would be given by $p(\tilde{y} \mid \theta)$. Here, we are assuming that the sampling distribution will be the same density for the original data $p(y \mid \theta)$ and the predictive data $p(\tilde{y} \mid \theta)$.

Unfortunately, we don't know the true value of θ. All we have to go on are the inferences we can make about θ given our model and observed data y. As we saw in the last chapter, this knowledge is encapsulated in a posterior distribution with density $p(\theta \mid y)$. So rather than making predictions $p(\tilde{y} \mid \theta)$ based on a single estimated value of θ, we are going to create a weighted average of predictions for every possible θ with weights determined by the posterior $p(\theta \mid y)$. Because θ is continuous, the averaging must proceed by integration. The result is the *posterior predictive distribution*, the density for which is defined by

$$p(\tilde{y} \mid y) \ = \ \int_{\Theta} p(\tilde{y} \mid \theta) \times p(\theta \mid y) \, \mathrm{d}\theta.$$

The variable Θ is now doing extra duty as the set of possible values for θ, that is the range of variables over which θ is averaged.

When estimating the probability of new data, there are two forms of uncertainty that need to be taken into account. The first is estimation uncertainty arising from not knowing the exact value of θ. This comes into play by averaging over the posterior $p(\theta \mid y)$. The second form of uncertainty is introduced by the sampling distribution $p(\tilde{y} \mid \theta)$. Even if we knew the precise value of θ, we would still not know the value of $\tilde{y}$ because it is not a deterministic function of θ. Let's write our formula again highlighting the two sources of

[173] From now on, we'll be dropping random variable subscripts on probabilty functions. The convention in applied statistics is to choose names for bound variables that allow the random variables to be determined by context. For example, we will write $p(\theta \mid y)$ for a posterior, taking it to mean $p_{\Theta|Y}(\theta \mid y)$.

[174] We are also overloading lowercase variables like θ to mean their random variable counterpart Θ when necessary, as it is here, where we should properly be saying that the random variable Θ takes on some known value θ.

uncertainty.

$$p(\tilde{y} \mid y) \;=\; \int_{\Theta} \underbrace{p(\tilde{y} \mid \theta)}_{\text{sampling uncertainty}} \times \underbrace{p(\theta \mid y)}_{\text{estimation uncertainty}} \, \mathrm{d}\theta.$$

Calculation via simulation

The probability mass function for the posterior predictive is defined by an integral that averages over the posterior, thus it can be estimated using simulations $\theta^{(1)}, \theta^{(M)}$ of the posterior $p(\theta \mid y)$ as

$$p(\tilde{y} \mid y) \;\approx\; \frac{1}{M} \sum_{m=1}^{M} p(\tilde{y} \mid \theta^{(m)}).$$

That is, we just take the average prediction over our sample of simulations.

Continuing our example from the previous chapter, we will put everything together and show how to compute posterior predictive densities. To review, we have y boys born out of N births, under the sampling distribution $y \sim \text{binomial}(N, \theta)$ and prior $\theta \sim \text{uniform}(0, 1)$. We have shown in the previous chapter how to take simulated draws $\theta^{(1)}, \ldots, \theta^{(M)}$ from the posterior distribution with density $p(\theta \mid y, N)$.

Now suppose we have new data $\tilde{y}$ from a trial of size $\tilde{N}$. We can estimate $p(\tilde{y} \mid y, \tilde{N})$ by instantiating the general formula above,

$$p(\tilde{y} \mid y, \tilde{N}) \;\approx\; \frac{1}{M} \sum_{m=1}^{M} \text{binomial}(\tilde{y} \mid \tilde{N}, \theta^{(m)}).$$

If we treat binomial as evaluating elementwise,[175] so that a collection of θ as input produces a collection as output, then we can write this using the elementwise definition of the binomial and a function to calculation the mean of a collection as

$$\begin{aligned}
p(\tilde{y} \mid y) &\approx \text{mean}\big(\text{binomial}\big(\tilde{y} \mid \tilde{N}, \theta\big)\big) \\
&\approx \text{mean}\Big(\text{binomial}\Big(\tilde{y} \mid \tilde{N}, \big(\theta^{(1)}, \ldots \theta^{(M)}\big)\Big)\Big) \\
&= \text{mean}\Big(\big(\text{binomial}\Big(\tilde{y} \mid \tilde{N}, \theta^{(1)}\Big), \ldots, \text{binomial}\Big(\tilde{y} \mid \tilde{N}, \theta^{(M)}\Big)\Big) \\
&= \tfrac{1}{M} \textstyle\sum_{m=1}^{M} \text{binomial}\Big(\tilde{y} \mid \tilde{N}, \theta^{(m)}\Big).
\end{aligned}$$

The second two lines illustrate how an elementwise definition is expanded, and the third shows how the compound function for the mean is applied.[176]

The pseudocode translates the mathematical definition directly.

[175] A function $f : \mathbb{R} \to \mathbb{R}$ can be extended *elementwise* to a function on sequences $f : \mathbb{R}^N \to \mathbb{R}^N$ by defining

$$f(x)[n] = f(x[n]).$$

For example, elementwise exponentiation satisfies

$$\exp((u, v, w)) = (\exp(a), \exp(v), \exp(w)).$$

[176] The definition of the mean function is

$$\text{mean}((u_1, \ldots, u_N)) \;=\; \frac{1}{N} \sum_{n=1}^{N} u_n.$$

```
for (m in 1:M)
  draw theta(m) from posterior p(theta | y, N)
  p[m] = binomial(y_pred | N_pred, theta(m))
print 'esimated p(y_pred | y) = ' mean(p)
```

With an elementwise version of the binomial probability mass function, this simplifies to the following.[177]

```
for (m in 1:M)
  draw theta(m) from posterior p(theta | y, N)
p = binomial(y_pred | N_pred, theta)
print 'esimated p(y_pred | y) = ' mean(p)
```

Let's continue with the small data example, where $y = 3$ and $N = 10$. We'll take $\tilde{N} = 20$ in order to make a prediction over the next twenty observations and calculate the probability for each possible $\tilde{y} \in 0 : \tilde{N}$. Then we'll plot them in a bar chart to inspect the distribution.

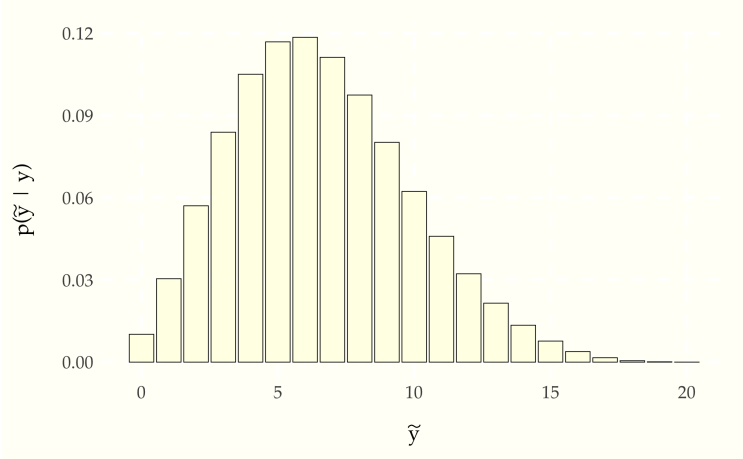


Figure 56: Probablity mass function for the posterior predictive distribution of the number of boys, $\theta^* = \frac{y}{N} = 0.3$, into the sampling distribution, to yield $\text{binomial}(\tilde{y} \mid \tilde{N}, \theta^*)$. Compared to the full posterior taking into account estimation uncertinaty in θ, this plug-in estimate makes it seem very unlikely there will be more boys than girls born in the next 20 births.

Because we have only observed N outcomes, the posterior is not very concentrated—it is consistent with a wide range of possible outcomes. Contrast this situation to using the binomial directly to do prediction by setting $\theta = 0.3$, the observed proportion of boys.[178]

The probability mass in the predictions is much more concentrated around the observed proportion of boys. Ignoring the estimation uncertainty in θ captured by the full posterior leads to inferences that are overly concentrated compared to the full probabilistic conditioning on observed data. As we will see in subsequent chapters, such predictions are not well calibrated for future data assuming the model is correct—they place too much certainty in the observed proportion of boys in a small sample.

[177] Simple can mean several things—we're measuring code simplicity, not necessarily how simple it is to undersetand. Code simplicity involves several things, among them shallower nesting of statments and fewer indexed expressions, as in this example. As another example, the calculation of the mean could be done manually, but it is simpler to use a library function. Even if there's no library function, it simplifies code to break complex operations down into well-named functions. Building up a personal library of such functions is the first step toward becoming a developer.

[178] Inference with a single point estimate of one or more parameters is common in live, real-time applications, where the cost of averaging over the posterior may be prohibitive.

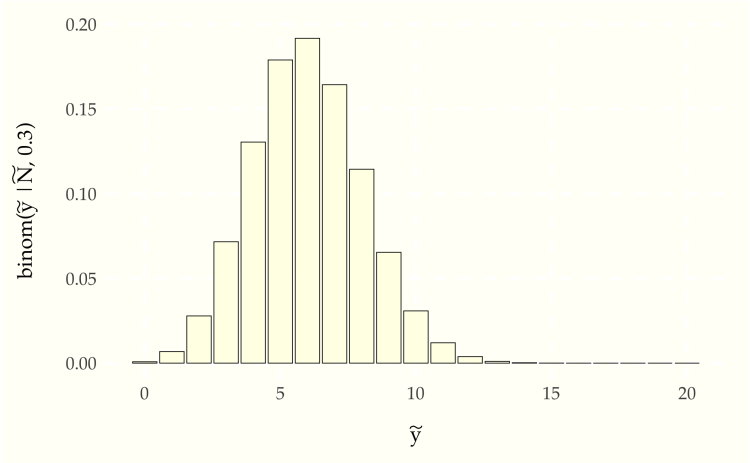


Figure 57: Probability mass function for predictions made by plugging the proportion of boy births observed in the data $\tilde{y}$ in a subsequent group of $\tilde{N} = 20$ births based on observing 3 boys in 10 births with no prior information. Although it is peaked near the 30 percent level observed in the data, it would not be that surprising to see more boys than girls in the next 20 births.

The full posterior automatically adjusts for the size of the sample. Consider the following plot, in which the same proportion of boys is provided (30%), but the sample size continues to grow (quadrupling each time).

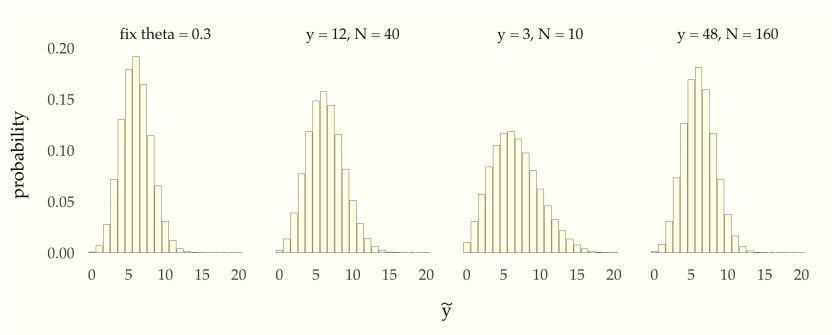


Figure 58: Illustration of convergence of posterior to binomial prediction based on proportion of boys as number of observations y and N grows, with proportion $\frac{y}{N}$ fixed. The central limit theorem tells us that each quadrupling of the data cuts the uncertainty in parameter estimation in half until all that is left is the sampling uncertainty in the binomial, as represented in the final plot, where $\theta = 0.3$ is fixed.

As the data size grows, the posterior predictive distribution approaches the predictions derived from just plugging the proportion of boys observed (30%) in the sampling distribution. After 160 observations, the posterior predictive distribution is only a percent or two away from the predictions that do not take into account estimation uncertainty. Whether that matters or not will depend on the application.[179]

[179] If leveraged bets or lives are at stake, a 1% discrepancy in predictions can have enormous consequences.

Numerically stable expectations on the log scale

We often run into the problem of underflow when computing densities or probabilities.[180] To get around that problem we compute on the log scale, using $\log p(y \mid \theta)$ rather than doing calculations on the original scale $p(y \mid \theta)$.

But we have to be careful with averaging non-linear operations.

[180] Underflow is the result of operations which produce real numbers ϵ smaller than the smallest number that can be represented.

The averaging that we do with simulation-based estimates does not distribute. For most u and v,[181]

$$\log u + \log v \neq \log(u + v).$$

As a result, the log of an average is not equal to the average of a log,

$$\frac{1}{M} \sum_{m=1}^{M} \log u_m \neq \log \left(\frac{1}{M} \sum_{m=1}^{M} u_m \right).$$

All is not lost, however. We will rewrite our desired result as

$$\begin{aligned} \log \tfrac{1}{M} \textstyle\sum_{m=1}^{M} p(\tilde{y} \mid \theta^{(m)}) &= \log \tfrac{1}{M} + \log \textstyle\sum_{m=1}^{M} p(\tilde{y} \mid \theta^{(m)}) \\ &= -\log M + \log \textstyle\sum_{m=1}^{M} \exp\left(\log p(\tilde{y} \mid \theta^{(m)})\right) \\ &= \text{log\_sum\_exp}(\log p(\tilde{y} \mid \theta)) - \log M. \end{aligned}$$

Extending our example, we can work on the log scale to stablize calculations with larger N by calculating

$$\begin{aligned} \log p(\tilde{y} \mid y) &\approx \log \textstyle\sum_{m=1}^{M} \exp\left(\log \text{binomial}(\tilde{y} \mid \tilde{N}, \theta^{(m)})\right). \\ &= \text{log\_sum\_exp}(\log \text{binomial}(\tilde{y} \mid \tilde{N}, \theta)). \end{aligned}$$

As with the algorithm on the original scale, the pseudocode is straightforward given a means to draw $\theta^{(m)}$ from the posterior $p(\theta \mid y, N)$.[182]

```
for (m in 1:M)
  draw theta(m) from posterior p(theta | y, N)
  lp[m] = log(binomial(y_pred | N_pred, theta(m))
print 'log p(y_pred | y) = ' log_sum_exp(lp)
```

Thus we can calculate posterior predictive densities on the log scale using log scale density calculations throughout to prevent overflow and underflow in intermediate calculations or in the final result.

Posterior predictive densities as expectations

Working with expectations and conditional expectations is natural for posterior inference, but initially requires some mental gymnastics to interpret all the implicit bindings. Using expectation notation, the posterior predictive distribution can be defined as

$$p(\tilde{y} \mid y) \;=\; \mathbb{E}[p(\tilde{y} \mid \theta) \mid y].$$

[181] An exception is $u = v = 2$.

[182] Typically, software will have the binomial implemented on the log scale directly, so it won't be necessary to take the log of the standard scale version.

We are now overloading lower case variables to do double duty as their upper-case counterparts. Rendered with full random variable indexing, the expectation and its definition are

$$\begin{aligned} p_{\tilde{Y}|Y}(\tilde{y} \mid y) &= \mathbb{E}\Big[p_{\tilde{Y}|\Theta}(\tilde{y} \mid \Theta) \mid Y = y\Big] \\ &= \int_T p_{\tilde{Y}|\Theta}(\tilde{y} \mid \theta) \times p_{\Theta|Y}(\theta \mid y) \, \mathrm{d}\theta, \end{aligned}$$

where T is domain of integration for θ, i.e., possible values for Θ. The twist is that the function whose expectation is being taken is now a density $p(\tilde{y} \mid \theta)$ in which θ shows up as a conditioning variable for the data $\tilde{y}$ being predicted.

The trick to understanding expectations is in understanding which variables are bound—all other random variables are averaged out to define the expectation. The value of both the observed data y and the data $\tilde{y}$ for which we are computing predictions are fixed by the function definition and are not free variables in the expectation. The random variable θ is not bound anywhere, and hence it is averaged in the calculation.

Pseudorandom Number Generators

We have been assuming so far that we have pseudorandom number generators that work in some sense to simulate true randomness. This chapter clarifies what that means and shows how we gain confidence in our random number generator's randomness through testing various hypotheses that it's random against the actual results produced.

Linear congruential generator

To provide a feel for what a (weak) pseudorandom number generator looks like, we'll start with a very simple one. Starting from an initial random number *seed* s, the *linear congruential* pseudorandom number generator does a linear transform and then reduces the number modulo M to generate an integer in the range $0, 1, \ldots, M-1$.[183] The sequence of random numbers generated, $y_0, y_2, \ldots$, is defined inductively based on an integer seed m, integer multiplier α, and integer increment β, by the base case

$$y_0 = s \bmod M$$

and inductive case

$$y_{n+1} = (\alpha \times y_n + \beta) \bmod M.$$

[183] The *modulus operator* is defined so that $m \bmod n$ is the remainder after dividing m by n. For example, $5 \bmod 2 = 1$ and $6 \bmod 3 = 0$.

The pseudocode for generating the next random number given the previous random number is a one-liner.[184]

```
next_prng(seed, M, alpha, beta)
  return (alpha * last_y + beta) mod M
```

[184] Such a function is typically set up as an iterator with either object-encapsulated or static storge of its arguments.

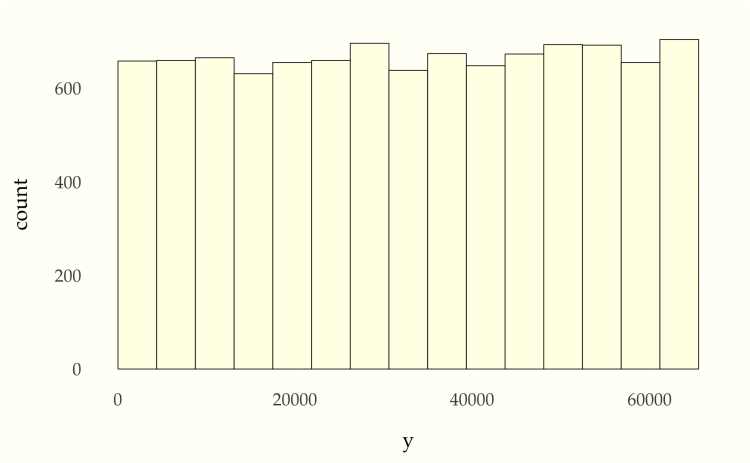
We will conveniently choose $M = 2^16$ to provide 16-bit integer results.[185]

To make sure this works to rotate through all M possible values, we need β to be relatively prime to M. So we'll just set $a = 123$ and and $b = 127$ and start with a seed of $s = 1234$.

[185] We are implementing these algorithms in R, which has the serious limitation of restricting user integer values to short, 32-bit values.

```
20827  5934  9103  5674  42659  4294
 3991 32258 35691 64734  32543  5210
51123 62326 64039 12594  41851 35982
34991 44170 59075 57382  45751 56930
```

Let's go ahead and generate 10 000 draws and plot a histogram.[186]



That looks good, but as we suggested when we first introduced pseudorandom number generators, we'll be able to do better than a simple χ-by-eye test.

Converting to a uniform$(0,1)$ sampler

We can now take our sampler for an integer range and divide by the range to produce a continuous sampler. To generate a continuous sample from uniform$(0,1)$, we just generate a discrete uniform draw from 0 to M and divide by M.[187]

```
uniform01_prng()
  seed = next_prng(seed, M, alpha, beta)
  return seed / M
```

Sometimes, the boundary values of zero and one are avoided by returning `(seed + 1) / (M + 1)`.

Mean and variance tests

One simple test for a pseudorandom number generator is whether it produces the right means and variances. We know that for a uniform distribution between 0 and 1 that the mean is $\frac{1}{2}$ and its variance is $\frac{1}{12}$.[188]

Lets see what our mean and variances are from 10 000 draws from our new function `uniform01_prng()`.

[186] For this to look roughly uniform, as here, it's important to set the boundary of the first bin at 0, set the limit to be exactly 0 to $2^{16} - 1$, and to make all the bins the same width. We achieve the latter by choosing the number of bins to be 16, a multiple of the total number of possible outcomes, $2^{16} - 1$.

[187] Our pseudocode is assuming we can store the seed value internally to the function and that `M`, `alpha`, and `beta` are all fixed from the outside.

[188] The mean follows by symmetry and the variance is derived by solving

$$\int_0^1 \left(y - \frac{1}{2}\right)^2 \, \mathrm{d}y = \frac{1}{12}.$$

```
    sample mean = 0.504
sample variance = 0.084
```

Because $\frac{1}{12} \approx 0.083$, it looks like we're in the right ballpark. How do we turn these into proper tests, though? We can't just eybeall the results every time.

The central limit theorem gives us the ability to characterize the behavior of sample averages, which is what both mean and variance estimates are.[189] For example, in the simple case of the mean, we know that the variance of a single draw is $\frac{1}{12}$, so the variance of the average of N draws will be $\frac{1}{N \times 12}$, and the standard deviation is $(N \times 12)^{-\frac{1}{2}} = \sqrt{\frac{1}{N \times 12}}$.

[189] Depending on the normality of the variables being averaged, the approximation is often reasonable starting from as few as ten draws and is usually very good with one hundred draws.

We can use the variance of the mean estimate to formulate a probabilistic test that the mean is truly distributed as normal $\left(0.5, (N \times 12)^{-\frac{1}{2}}\right)$. Thus we would expect 99.9% of the draws to be within three standad deviations, or $\pm 3 \times (N \times 12)^{-\frac{1}{2}}$. With the 10 000 draws we took, this is approximately 0.0029$, so it looks like the first run of the mean test passed to within a single standard deviation.

Probabilistic tests and false positives

We are unfortunately left with a test where we expect a false positive failure one in a thousand runs. That may be fine if we're only running the test once or twice. But if we're distributing this software to thousands of people or regularly running our tests during development, this is going to cause headaches. The simple recourse we have is to sharpen the test. We expect the 99.997% of the draws to fall within four standard deviations of the mean. We might be able to live with the resulting one in thirty thousand failure rate. But we may need to sharpen the test even further given our test conditions. The problem with sharpening the test too far is that we can miss bugs in less significant digits.

Another thing we can do for more power is run the test 100 times and look at the distribution of means. That distribution should itself be normal and can be tested. This is possible, and can lead to better tests, but starts to get computationally prohibitive for large number of draws per test.

The problem of requiring probabilistic tests for pobabilistic programs pervades all of statistical software engineering. There's no easy solution.

Binomial interval tests

If we are simulating a random variable Y, we know the probability that it lies between a and b is given by[190]

$$\begin{aligned} \Pr[a < Y < b] &= \int_a^b p_Y(y)\,\mathrm{d}y \\ &= F_Y(b) - F_Y(a). \end{aligned}$$

[190] Recall that we don't need to worry about the points on either end, so can use less-than and less-than-or-equal interchangeably in these formulas.

We can use this as the basis for a statistical test because it lets us know the proportion of draws that are expected to fall in any given interval.

Suppose we take M draws $y^{(1)}, \ldots, y^{(M)}$ from our pseudorandom number generator. Our test statistic is the number of draws $Z_{(a,b)}$ that fall in the interval (a, b), i.e.,

$$z_{(a,b)} = \sum_{m=1}^{M} \mathrm{I}[a < y^{(m)} < b].$$

If the pseudorandom number generator is producing draws with the proper distribution, then the test statistic is distributed as

$$z_{(a,b)} \sim \text{binomial}(M, \Pr[a < Y < b]).$$

We'll perform a central test, rejecting the null hypothesis with p-value α if the test statistic falls outside the central $1 - \alpha$ interval, i.e., if either

$$F_{Z_{(a,b)}}(z_{(a,b)}) < \frac{\alpha}{2}$$

or[191]

$$F_{Z_{(a,b)}}(z_{(a,b)}) > 1 - \frac{\alpha}{2}.$$

[191] The second condition is rendered parallel to the first with the complementary cumulative distribution function, as

$$F^{\complement}_{Z_{(a,b)}}(z_{(a,b)}) < \frac{\alpha}{2}.$$

To illustrate how this works, let's take our uniform pseudorandom number generator and test it on the interval $(0, 0.5)$.[192]

[192] 50% intervals are convenient for testing—more extreme probabilities lead to smaller expected counts either inside or outside the interval.

Suppose we are given the putatively uniformly distributed values $y^{(1)}, \ldots, y^{(M)}$ for testing. We can write a simple program to generate a p-value for the central hypothesis test as follows.[193]

[193] In general, the ternary *conditional operator* is defined so that `cond ? a : b` evaluates to `a` if `cond` evaluates to true, and to `b` otherwise.

```
m = sum(y < 0.5)
a = binomial_cdf(m | M, 0.5)
print 'm = ' m ' out of M = ' M
print 'reject uniformity with p-value '
      (a < 0.5) ? a : (1 - a)
```

The first line counts the number of draws $y^{(m)}$ for which $y^{(m)} < 0.5$. The second line computes the cumulative distribution function

value. The third line manages the tail issues—if the value is less than one half it's in the lower tail and we return it as is; otherwise it is in the upper tail and we return the distance from one.

Let's see what happens when we run it.

```
m = 4938 out of M = 10000
reject unformity with p-value 0.11
```

So our smapler passes this simple test. Ideally we'd want to test it with a range of seeds to make sure passing isn't seed dependent as it may be in these simple pseudorandom number generators.

Chi-squared interval tests

A more fine-grained interval test that can handle multiple intervals simultaneously relies on a normal approximation to the binomial. We know that as N grows, binomial(N, θ) approaches a normal distribution.[194]

The chi-squared test is going to require the domain of the variable being tested to be divided into K exclusive and exhausive regions. We'll assume that's done by dividing the domain into intervals with dividing points points $a_1, \ldots, a_{K-1}$, producing the folllowing intervals.[195]

$$\begin{array}{rcl} A_1 & = & (-\infty, a_1) \\ A_2 & = & (a_1, a_2) \\ \vdots & \vdots & \vdots \\ A_K & = & (a_{K-1}, \infty) \end{array}$$

Now we let z_k be the count of the number of draws falling in the interval A_k. The expected number of draws falling into interval A_k will be defined as

$$E_k = M \times \Pr[Y \in A_k].$$

We are going to base the test on the following test statistic.[196]

$$X^2 \ = \ \sum_{k=1}^{K} \frac{(z_k - E_k)^2}{E_k}.$$

Because the E_k is a constant and z_k is approximately normal, $z_k - E_k$ will also be approximately normal, and thus $(z_k - E_k)^2$ will be roughly a squared normal, so that the sum of K such terms will be roughy a chi-squared distribution with $K - 1$ degrees of freedom.[197] That means that if the null hypothesis of the distribution of pseudo-random numbers being uniform is correct, then we can set up the

[194] The normal distribution it approaches has the same mean and standard deviation, i.e., binomial(N, θ) is well approximated by normal$(N \times \theta, \sqrt{N \times \theta \times (1 - \theta)})$ as N grows.

[195] As before, we needn't worry about points on the boundaries as they have probability zero.

[196] This was first introduced by Karl Pearson in Pearson, K., 1900. X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175.

[197] This follows from the definition of the chi-squared distribution. Specifically, if

$$Y_1, \ldots, Y_N \sim \text{normal}(0, 1),$$

then

$$(Y_1^2 + \cdots + Y_K^2) \sim \text{chi\_squared}(K - 1)$$

has a *chi-squared distribution* with $K - 1$ degrees of freedom. The subtraction of E_k and division by $\sqrt{E_k}$ normalizes the distribution so that approximately

$$\frac{z_k - E_k}{\sqrt{E_k}} \sim \text{normal}(0, 1)$$

same kind of central hypothesis test as for the ibnomial test. Specifically, we'll reject the null hypothesis that the pseudorandom number generator is behaving properly at significance level α if

$$\text{chi\_squared\_cdf}(X_2, K-1) < \frac{\alpha}{2}$$

or if

$$\text{chi\_squared\_cdf}(X_2, K-1) > 1 - \frac{\alpha}{2}.$$

That is, we reject the null hypothesis at signifiance level α if the test statistic X^2 is outside of the central $1 - \alpha$ interval of the chi-squared distribution.

There are lots of choices when using the chi-squared test, like how many bins to use and how to space them. Generally, we want the expected number of elements in each bin to be good enough that the normal approximation is approximation, the traditionally suggested minimum for which is five or so. If we're testing a pseudorandom number generator, the only bound is compute time, so we'll usually have many more expected elements per bin than five. It's common to see equal probability bins used, generating them with an inverse cumulative distribution function where available.

Floating Point Arithmetic

Contemporary[198] computers use floating-point representations of real numbers and thus perform arithmetic on floating-point representations.

[198] Contemporary being 2019 as of this writing.

What is a floating point number?

A *bit* is the smallest discrete representational unit in a computer—it takes on a value of 0 or 1.[199] Floating point numbers are most commonly represented using 32 or 64 bits, which are known as *single precision* and *double precision* respectively.[200]

[199] A *byte* consists of a sequence of 8 bits. The natural computing unit is a *word*, the size of which is hardware dependent, but most computers nowadays (it's still 2019) use 64-bit, or 8-byte words.

[200] Modern machine learning floating point representations go as low as 8 bits and high-precision applied mathematical calculations may use 1024 bits or more.

Finite, not-a-number, and infinite values

A floating point number consists of a fixed number of bits for a *significand a* and a fixed number of bits for the *exponent b* to represent the the real number $a \times 2^b$. The significand determines the precision of results and the exponent the range of possible results.[201] The significand may be negative in order to represent negative values. The exponent may be negative in order to represent fractions. Both the significand and exponent are represented using a single bit for a sign and the remaining bits for the value in binary representation. Standard double-precision (i.e., 64 bit) representations allocate 53 bits for the significand and 11 bits for the exponent.[202]

[201] This exponent causes the decimal place to float, giving the representation its name.

The standard also sets aside three special values, *not a number* for ill-defined results, e.g., dividing zero by zero, *positive infinity* for infinite results, e.g., dividing one by zero, and *negative infinity* for negative infinity, e.g., dividing negative one by zero.[203] These are all specified to have the expected behavior with arithmetic and comparison operators and built-in functions.[204]

[202] Zuras, D., Cowlishaw, M., Aiken, A., Applegate, M., Bailey, D., Bass, S., Bhandarkar, D., Bhat, M., Bindel, D., Boldo, S. and Canon, S., 2008. *IEEE Standard 754-2008 (floating-point arithmetic).

[203] Technically, there are two forms of not-a-number and often two forms of zero, but these distinctions are irrelevant in statistical applications.

[204] Not-a-number typically propagates, but there are some subtle interactions of not-a-number and comparison operators because comparisons return integers in most languages.

Literals

Floating point numbers are written in computer programs using *literals*, which can be integers such as `314`, floating point numbers

such as `3.14`, and scientific notation such as `0.314e+1`. Scientific notation uses decimal notation, where `e+n` denotes multiplication by 10^n and `e-n` by 10^{-n}. Multiplication by 10^n shifts the decimal place n places to the right; multiplication by 10^{-n} shifts left by n places. For example, `0.00314e+3`, `314e+2`, and `3.14` are just different literals representing the same real number, 3.14.

Machine precision and rounding

If we add two positive numbers, we get a third number that's larger than each of them. Not so in floating point. Evaluating

```
1 == 1 + 10^-16
```

produces the surprising result

```
TRUE
```

When we add 1 and 10^{-16}, we get 1. This is not how arithmetic is supposed to work. We should get 1.0000000000000001.

The problem turns out to be that there's only so close to 1 we can get with a floating point number. Unlike with actual real numbers, where there is always another number between any two non-identical numbers, floating point numbers come with discrete granularity. The closest we can get to one is determined by the number of non-sign sigificand bits, or $2^{-52} \approx 2.2 \times 10^{-16}$. This is known as the *machine precision* of the representation. Writing out in decimal rather than binary, the largest number smaller than one is

$$1 - 2^{-52} = 0.\underbrace{999999999999999}_{15 \text{ nines}}7,$$

whereas the smallest number greater than one is

$$1 + 2^{-52} \approx 1.\underbrace{000000000000000}_{15 \text{ zeros}}2.$$

This numerical granularity and subsequent rounding of expressions like `1 + 1e-20` to one is a serious problem that we have to fight in all of our simulation code.

Underflow and overflow

The most common problem we run into with statistical computing with floating point is underflow. If we try to represent something like a probability, we quickly run out of representational power. Simply consider evaluating $N = 2\,000$ Bernoulli draws with a 50% chance of success,

```
p = 1
for (n in 1:N)
  p *= bernoulli(y[n] | 0.5)
print 'prob = ' p
```

The result should be 0.5^{2000}. What do we get?

```
 prob = 0
```

The result is exactly zero.[205] What's going on?

[205] It's not just rounding in the printing.

Just as there's a smallest number greater than one, there's a smallest number greater than zero. That's the smallest positive floating point number. This number is defined by taking the largest magnitude negative number for the exponent and the smallest number available for the significand. For the double-precision floating point in common use, the number is about 10^{-300}.[206]

[206] 10^{-322} can be represented, but 10^{-323} underflows to zero.

Working on the log scale

Becuase of the everpresent threat of underflow in statistical calculations, we almost always work on the log scale. Let's try that calculation again, only now, rather than calculating

$$\begin{aligned} p(y \mid \theta = 0.5) &= \prod_{n=1}^{2000} p(y_n \mid \theta = 0.5) \\ &= \prod_{n=1}^{2000} \text{bernoulli}(y_n \mid 0.5) \\ &= \prod_{n=1}^{2000} 0.5 \\ &= 0.5^{2000}, \end{aligned}$$

we'll be calculating the much less troublesome

$$\begin{aligned} \log p(y \mid \theta = 0.5) &= \log \prod_{n=1}^{2000} p(y_n \mid \theta = 0.5) \\ &= \sum_{n=1}^{2000} \log p(y_n \mid 0.5) \\ &= \sum_{n=1}^{2000} \log \text{bernoulli}(y_n \mid 0.5) \\ &= \sum_{n=1}^{2000} \log 0.5 \\ &= 2000 \times \log 0.5. \end{aligned}$$

We can verify that it works by coding it up.

```
log_p = 0
for (n in 1:N)
  log_p += bernoulli(y[n] | 0.5)
print 'prob = ' p
```

We have replaced the variable `p` representing the probability with a variable `log_p` representing the log probability. Where `p` was initialized to 1, `log_p` is initialized to $\log 1 = 0.$. Where the probability of

each case was multiplied into the total, the log probability is added to the total. Let's see what happens with $N = 2000$.

```
prob = -1386.3
```

The result is indeed $2000 \times \log 0.5 \approx -1386.29$, as expected. Now we're in no danger of overflow even with a very large N.

Logarithms of sums of exponentiated terms

When we take a logarithm, it is well known that it converts multiplication into addition, so that

$$\log (u \times v) \ = \log u + \log v.$$

But what if we have $\log u$ and $\log v$ and want to produce $\log(u + v)$? It works out to

$$\log (u + v) \ = \ \log \left(\exp(\log u) + \exp(\log v)\right).$$

In words, it takes the logarithm of the sum of exponentiations. This may seem like a problematic amount of work, but there's an opportunity here, as well. By rearranging terms, we have

$$\log(\exp(a) + \exp(b)) \ = \ \max(a, b) + \log(\exp(a - \max(a, b)) + \exp(b - \max(a, b))).$$

This may seem like even more work, but when we write it out in code, it's less daunting and serves the important purpose of preventing overflow or underflow.

```
log_sum_exp(u, v)
  c = max(u, v)
  return c + log(exp(u - c) + exp(v - c))
```

Because c is computed as the maximum of u and v, we know that $u - c \leq 0$ and $v - c \leq 0$. As a result, the exponentiations will not overflow. They might underflow to zero, but that doesn't cause a problem, because the maximum is preserved by being brought out front, with all of its precision intact.

If we have a vector, it works the same way, only the vectorized pseudocode is neater. If u is an input vector, then we can compute the log sum of exponentials as

```
log_sum_exp(u)
  c = max(u)
  return c + log(exp(u - c))
```

This is how we calculate the average of a sequence of values whose logarithms are known.

```
log_mean(log_u)
  M = size(log_u)
  c = max(log_u)
  return -log(M) + c + log(exp(log_u - c))
```

Failure of basic arithmetic laws and comparison

Because of the need to round to a fixed precision result, floating point arithmetic does not satisfy basic transitivity or distributivity laws of arithmetic.[207]

One surprising artifact of this is rounding, where we can have $u + v = u$, even for strictly positive values of u and v. For example, $u = 1$ and $v = 10^{-20}$ have this property, as do $u = 0$ and $v = 10^{-350}$.

The upshot is that we have to be very careful when comparing two floating point numbers, because even though pure mathematics might guarantee the equality of two expressions, they may not evaluate to the same result in floating point arithmetic. Instead, we need to compare floating-point numbers within tolerances.

With absolute tolerances, we replace exact comparisons with comparisons such as `abs(u - v) < 1e-10`. This tests that `u` and `v` have values within 10^{-10} of each other. This isn't much use if the numbers are themselves much larger or much smaller than 10^{-10}.

There are many ways to code relative tolerances. One common approach is to use `2 * abs(u - v) / (abs(u) + abs(v)) < 1e-10`. For example, the numers 10^{-11} and 10^{-12} are within an absolute tolerance of 10^{-10} of each other,

$$\begin{aligned} \left|10^{-11} - 10^{-12}\right| &= 9 \times 10^{-12} \\ &< 10^{-10} \end{aligned}$$

but they are not within a relative tolerance of 10^{-10},

$$\begin{aligned} \frac{2 \times \left|10^{-11} - 10^{-12}\right|}{\left|10^{-11}\right| + \left|10^{-12}\right|} &\approx 1.63 \\ &> 10^{-10}. \end{aligned}$$

Loss of precision

Once we have precision, we have to work very hard not to lose it. Even simple operations like subtraction are frought with peril. When taking differences of two very close numbers, there can be an arbitrary amount of loss of precision.[208] As a simple example, consider

[207] In general, we cannot rely on any of

$$u + (v + w) = (u + v) + w,$$

$$u \times (v \times w) = (u \times v) \times w, \text{ or}$$

,

$$u \times (v + w) = u \times v = u \times w.$$

[208] When the loss of precision is great relative to the total precision, this is called *catastrophic cancellation*.

$$\begin{array}{rl} & 0.7595127881504595 \\ - & 0.7595127881504413 \\ \hline & 0.0000000000000182 \end{array}$$

We started with two numbers with 15 digits of precision and are somehow left with only 3 digits of precision. In statistical computing, this problem arises in calculating variances, which involve differences of variates from the mean—when variance is low relative to the mean, catastrophic cancellation may arise.

Normal Distribution

Limit of binomials

In 1733, Abraham de Moivre noticed that as the number of trials in a binomial distribution grew, the resulting distribution became nearly symmetric and bell-shaped.[209] With 10 trials and a probability of success of 0.9 in each trial, the distribution is clearly asymmetric (i.e., skewed).

[209] De Moivre, A., 1733. *Approximatio ad summam terminorum binomii*.

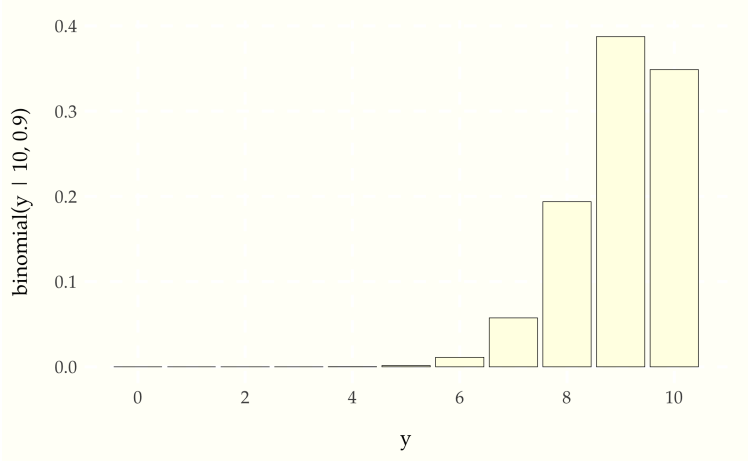


Figure 59: Probability of number of successes in $N = 10$ independent trials, each with a 90 percent chance of success. The result is binomial$(y \mid 10, 0.9)$ by construction. With only ten trials, the distribution is highly asymmetric, with skew (longer tails) to the left.

But when we increase the number of trials by a factor of 100 to $N = 1\,000$ without changing the 0.9 probability of success, the result is a nearly symmetric bell shape.

de Moivre found that the normal density function (to be developed below), despite coming from a continuous distribution, provided a tight approximation to the binomial probablity mass function as N becomes large.[210]

[210] This is because the interval between integers is 1, so that

$$\begin{aligned} 1 &= \int_{-\infty}^{\infty} p(x)\mathrm{d}x \\ &\approx \sum_{x=-\infty}^{\infty} p(x). \end{aligned}$$

The normal motivated by least square errors

In 1809, Carl Friedrich Gauss published his treatise on the motion of planets, in which he derives the normal distribution from the

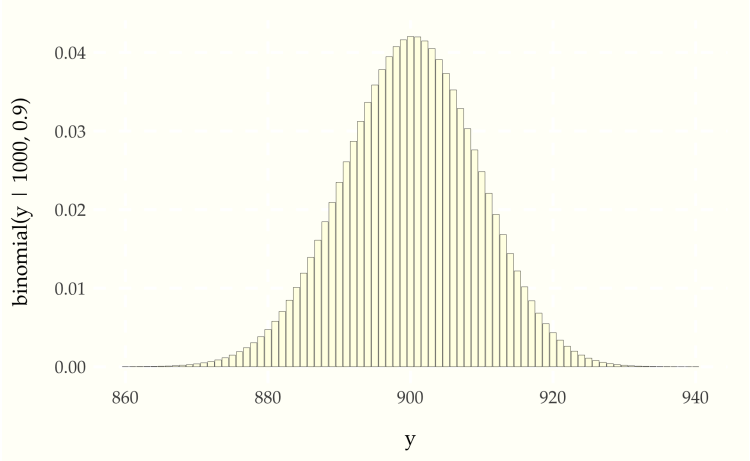


Figure 60: Probability of number of successes in $N = 1000$ independent trials, each with a 90 percent chance of success. The familiar bell-shaped curve arises as N grows.

method of least squares.[211] Gauss was faced with a sequence of noisy measurements $y_1, \ldots, y_N$ from some distribution and wanted to combine them by taking their average,

$$\bar{y} = \frac{1}{N} \sum_{n=1}^{N} y_n.$$

Gauss realized that the average $\bar{y}$ minimizes the sum of square differences from the observed values,[212]

$$\bar{y} = \arg\min_y \sum_{n=1}^{N} (y_n - y)^2 .$$

Gauss reasoned backward to the normal distribution, reasoning that for the average to be a good estimator, the distribution of the errors $y_n - y$ must have this same quadratic property. Working on the log scale, Gauss was looking for a distribution whose density functions would have roughly the property that

$$\log p(y) = -y^2 + \text{const.}$$

We have to work on the log scale here in order for the resulting density function to be normalizable.[213]

In order for the standard deviation and variance to work out to unity, it is convenient to work with half th squared error. Therefore, the standard normal distribution is defined on the log scale up to an additive constant that doesn't depend on y by

$$\log \text{normal}(y) = -\frac{1}{2} y^2 + \text{const.}$$

This ensures that if $Z \sim \text{normal}()$, then $\mathbb{E}[Z] = 0$ and $\text{var}[Z] = \text{sd}[Z] = 1$.

[211] Gauss, Carolo Friderico, 1809. *Theoria Motus Corporum Coelestium in sectionibus conicis solem ambientium.* Sumtibus Frid. Perthes et IH Besser, Hamburgi. English translation: Theory of Motion of the Celestial Bodies Moving in Conic Sections Around the Sun.

[212] The expression $\arg\min_y f(y)$ returns the value of y that minimizes $f(y)$, e.g.,

$$\arg\min_y (y-3)^2 = 3$$

.

[213] Normalizability requires

$$\begin{aligned} \int_{-\infty}^{\infty} p(y)\,\mathrm{d}y &= \int_{-\infty}^{\infty} \exp(\text{const}) \times \exp(-y^2)\,\mathrm{d}y \\ &= \exp(\text{const}) \times \int_{-\infty}^{\infty} \exp(-y^2)\,\mathrm{d}y \end{aligned}$$

to be finite.

Converting back the linear scale gives us the kernel of the normal distribution,

$$\text{normal}(y) \propto \exp\left(-\frac{1}{2}y^2\right).$$

The *kernel* of a probabilty function defines it as a function of parameters and variate outcome up to a proportion. Most of the algebra in statistics only requires probability functions up to a proportion, so it's usually simpler to drop the normalizing constants.[214]

Including the normalizing constants, the normal distribution is[215]

$$\text{normal}(y) \;=\; \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}y^2\right).$$

Now we have a distribution where the value y that maximizes the density of the independent error terms $y - y_n$ is the one that minimizes square error,

$$\begin{aligned}
\arg\max_y \textstyle\prod_{n=1}^N \text{normal}(y - y_n) &= \arg\max_y \log \textstyle\prod_{n=1}^N \text{normal}(y - y_n) \\
&= \arg\max_y \textstyle\sum_{n=1}^N \log \text{normal}(y - y_n) \\
&= \arg\max_y \textstyle\sum_{n=1}^N -\frac{1}{2}\,(y - y_n)^2 \\
&= \arg\max_y -\frac{1}{2}\textstyle\sum_{n=1}^N (y - y_n)^2 \\
&= \arg\min_y \frac{1}{2}\textstyle\sum_{n=1}^N (y - y_n)^2 \\
&= \arg\min_y \textstyle\sum_{n=1}^N (y - y_n)^2 \\
&= \bar{y}.
\end{aligned}$$

Adding location and scale parameters

The standard normal distribution has an expectaiton of zero and standard deviation one. We can add a scale parameter $\sigma > 0$ to multiply the standard deviation and a location parameter μ so that if $Y \sim \text{normal}(\mu, \sigma)$, then $\mathbb{E}[Y] = \mu$ and $\text{sd}[Y] = \sigma$.

Starting with a standard normal variate,

$$Z \sim \text{normal}(),$$

we can scale it by σ and shift by μ to get a new variable

$$Y = \mu + \sigma \times Z,$$

for which

$$Y \sim \text{normal}(\mu, \sigma).$$

[214] It's also faster computationally to drop normalizing constant, which often involve complicated expensive special function evaluation.

[215] The constant factor is defined by

$$\int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}y^2\right) \;=\; \sqrt{2\pi}.$$

Dealing with the required change of variables for the inverse transform[216]

$$Z \ = \ \frac{Y - \mu}{\sigma},$$

lets us derive the general normal density function for location parameter μ and scale parameter $\sigma > 0$ as[217]

$$\text{normal}(y \mid \mu, \sigma) \ = \ \frac{1}{\sqrt{2\pi}} \, \frac{1}{\sigma} \, \exp\left(-\frac{1}{2} \left(\frac{y - \mu}{\sigma} \right)^2 \right).$$

Presented this way, the formula makes clear that the normal density function is a product of three factors,

- a factor deriving from the standard normal distribution, $\text{normal}(z) = \exp(-\frac{1}{2}z^2)$, applied to the inverse of the location-scale transform, $\frac{y-\mu}{\sigma}$.
- a normalizing factor of $\frac{1}{\sigma}$ that depends on the scale,
- a constant normalizing factor of $\frac{1}{\sqrt{2\pi}}$, and

Central limit theorem

Laplace proved the central limit theorem in 1812, which established the sense in which the normal distribution is normal. The theorem tells us that if we take a sequence of independent, finite expectation, finite variance random variables and add them, the sum approaches a normal distribution. It goes even further and tells us which normal distribution, based on the expectations and variances. Let's state it mathematically as a proper theorem.

Central Limit Theorem (Laplace 1812). Suppose

$$Y = Y_1, Y_2, \ldots, Y_N$$

is a sequence of independent, identically distributed random variables with

$$\mathbb{E}[Y_n] = \mu$$

and

$$\text{sd}[Y_n] = \sigma.$$

Define the average of Y as a new random variable

$$Z = \frac{1}{N} \sum_{n=1}^{N} Y_n.$$

[216] This inverse transform has its own name, the *z-transform*, and in general may be written as

$$Z = \frac{Y - \mathbb{E}[Y]}{\text{sd}[Y]}.$$

and used to standardize any random variable Y with a finite expectation and variance. The resulting variable Z has $\mathbb{E}[Z] = 0$ and $\text{sd}[Z] = 1$ by construction.

[217] It's a standard Jacobian calculation for transform

$$Y = f(Z) = \mu + \sigma \times Z,$$

$$Z = f^{-1}(Y) = \frac{Y - \mu}{\sigma},$$

and

$$p_Z(z) = \text{normal}(),$$

from which the Jacobian calculation gives us

$$\begin{aligned} p_Y(y) &= \text{normal}(f^{-1}(y)) \times \left| \tfrac{\text{d}}{\text{d}y'} f^{-1}(y') \right|_{y'=y} \Big| \\ &= \text{normal}\left(\tfrac{y-\mu}{\sigma} \right) \times \tfrac{1}{\sigma} \\ &= \tfrac{1}{\sqrt{2\pi}} \, \tfrac{1}{\sigma} \, \exp\left(\left(\tfrac{y-\mu}{\sigma} \right)^2 \right). \end{aligned}$$

As $N \to \infty$, the average has a normal distribution with the same location as the Y_n and a scale reduced by a factor of $\sqrt{\frac{1}{\sqrt{N}}}$.

$$p_Z(z) \to \text{normal}\left(z \;\middle|\; \mu, \tfrac{1}{\sqrt{N}} \times \sigma\right).$$

The theorem can be generalized to variables that do not have the same distribution as long as they are independent and have finite expectations and variances.

Many natural phenomena, such as adult human height in either sex, are the result of a number of relatively small, additive effects, the combination of which leads to normality no matter how the additive effects are distributed. Practically speaking, this is why the normal distribution makes sense as a representation of many natural phenomena.

Lognormal distribution

The normal distribution arises naturally as the the distribution of a random variable resulting from a sum of effects. What if effects are multiplicative rather than additive, so that it is the product,

$$Y = V_1 \times \cdots \times V_N$$

of positive effects $V_n > 0$. Because the effects V_n are positive, we can work on the log scale, where

$$\log Y = \log V_1 + \cdots \log V_N.$$

In this case, $\log Y$ should have a roughly normal distribution, being the sum of N additive terms. If $\log Y \sim \text{normal}(\mu, \sigma)$, then Y has what is called a *lognormal distribution*. The lognormal density function can be calculated by accounting for the change of variables,[218] so that

$$\begin{aligned} p_Y(y \mid \mu, \sigma) &= \text{lognormal}(y \mid \mu, \sigma) \\ &= \frac{1}{y} \times \text{normal}(\log y \mid \mu, \sigma) \\ &= \frac{1}{y} \frac{1}{\sqrt{2\pi}} \frac{1}{\sigma} \exp\left(-\frac{1}{2}\left(\frac{\log y - \mu}{\sigma}\right)^2\right). \end{aligned}$$

To see the heart of what's going on without the location and scale complicating matters, the kernel of the lognormal distribution derived from the standard normal is just

$$\text{lognormal}(y \mid 0, 1) \;\propto\; \frac{1}{y} \exp\left(-\frac{1}{2}(\log y)^2\right).$$

[218] To calculate the Jacobian for the change of variables, note that if $Z \sim \text{normal}(\mu, \sigma)$ and $Y = \exp(Z)$, then

$$\begin{aligned} p_Y(y \mid \mu, \sigma) &= p_Z(\exp^{-1}(y) \mid \mu, \sigma) \times \left|\tfrac{\mathrm{d}}{\mathrm{d}y'} \exp^{-1}(y')\right|_y \\ &= \text{normal}(\log y \mid \mu, \sigma) \times \left|\tfrac{\mathrm{d}}{\mathrm{d}y'} \log y'\right|_{y'=y} \\ &= \text{normal}(\log y \mid \mu, \sigma) \times \tfrac{1}{y}. \end{aligned}$$

Simulating from a normal distribution

To simulate values of a normally distributed random variable

$$Y \sim \text{normal}(\mu, \sigma),$$

it suffices to simulate a standard normal variate

$$Z \sim \text{normal}(0, 1)$$

and let

$$Y = \mu + \sigma \times Z.$$

The simplest way to approximately generate from the normal distribution is to invoke the central limit theorem and generate enough draws from a uniform distribution that the result is roughly normal. This relies on simulation, so it is neither accurate nor fast.[219]

A clever and efficient way to generate standard normal variates relies on solving a seemingly harder problem, generating two independent standard normal variates. By working with two independent normal variates (X, Y), we can work in polar coordinates and generate an angle and radius (Θ, R), where

$$X = R \times \cos \Theta$$

and

$$Y = R \times \sin \Theta.$$

The other way around,

$$R = \sqrt{X^2 + Y^2}$$

and

$$\Theta = \arctan\left(\frac{Y}{X}\right).$$

Our strategy is to generate the polar coordinates (Θ, R) and transform them to (X, Y) with independent standard normal distributions. We can generate a random angle uniformly in radians with

$$\Theta \sim \text{uniform}(0, 2\pi).$$

The tricky part is generating the radius, which we will do by simulating a uniform variate

$$U \sim \text{uniform}(0, 1)$$

[219] It does correspond to Francis Galton's *quincunx*, a physical machine into which marbles are dropped to fall onto a grid of pegs which push them randomly to the right or the left. This random walk produces distances after a number of steps that are roughly normally distributed. The approximate improves as the number of random moves is increased.

and transforming it into

$$R = \sqrt{-2 \log U}.$$

This relies on a distribution and properties of sums of squared uniform variables we have not yet introduced.[220]

[220] The technique hinges on the fact that

$$X^2 + Y^2 \sim \text{chi\_squared}(2)$$

and if

$$V \sim \text{uniform}(0, 1),$$

then

$$-2 \log V \sim \text{chi\_squared}(2),$$

so that for our normal variates,

$$R = \sqrt{-2 \log V}$$

.

Calibration and Sharpness

Forecasting the weather

Meteorologists make predictions about the weather every day. Some of these forecasts are probabilistic. For example, such a forecast might be an 80% chance of rain over the next 24 hours in some geographic area such as Dayton, Ohio.[221] We can't tell much about the meteorologist from a single forecast and single outcome—if it rains, we have a bit more faith in the forecaster and if it doesn't, a bit less faith.

[221] This is trickier to measure than it sounds, but let's suppose for now that we can reliably determine if it rained in a given area within a given period of time.

Suppose we have 100 days on which the meteorologist predicted an 80% chance of rain. Let's further suppose, for the sake of this thought experiment, that the chance of rain is independent on those 100 days.[222] How many days do we expect to be rainy? About 80. But we don't expect that number to be exact, because we are dealing in probabilities. If there really was an independent 80% chance of rain on 100 days, we would expect the distribution of rainy days to be $\text{binomial}(100, 0.8)$.

[222] In reality, weather in neighboring regions is related, as is weather hour to hour and day to day.

Another forecast we consider is temperature. Let's say we're forecasting the temperature at noon on February 1, 2019 on Bondi Beach in Sydney. A probabilistic forecast of temperature might take the form that there's a 90% chance the high temperature will be between 20.1 and 24.8 degrees celsius. We can consider calibration of such a forecast in terms of its event probabilities, just like the chance of rain. That 90% forecast is still very broad. A *sharper* forecast is one with a narrower interval, such as $(22.3, 23.4)$. Given two calibrated forecasts, a sharper forecaster is preferable as it provides more information. For binary events, like rain or not rain, we'd prefer the probability forecasts to be as close to 0 or 1 as possible.

EXAMPLE OF HOW THIS IS POSSIBLE

Calibration

With statistical models, forecasts take on the form of posterior probabilty distributions conditioned on some observed data.

Given a prior distribution $p(\theta)$ and sampling function $p(y \mid \theta)$, we observe data y and calculate a posterior distribution over the model's parameters with density $p(\theta \; midy)$. Although the posterior is a distribution, we calculate it by means of a finite number of simulated values $\theta^{(1)}, \ldots, \theta^{(M)}$.

A posterior distribution provides probability statements about the parameters θ conditioned on observing data y. For example, they provide interval probalities, such as $\Pr[\theta > 0.5 \mid y]$ or $\Pr[0.21 < \theta < 0.32 \mid y]$.

We would like these event probabilities to be calibrated in the sense that if $\Pr[\theta > 0.5]$, then there is a 50% chance that θ is greater than 0.5.

If our model has a prior distribution with density $p(\theta)$ and a sampling distribution with probability function $p(y \mid \theta)$, then the posterior density is $p(\theta \mid y)$.

The posterior distributiondensity $p(\theta \mid y)$ given data y and model parameters θ provides predictions about the value of θ

Inference provides posterior draws $\theta^{(m)}$ drawn according to the posterior density Given some observed data y and a model $p(y \mid \theta) \times p(\theta)$ with parameters θ, our inference

Regression to the Mean

After a band puts out a great first album, it's often followed by a less great second album. Bands seem to suffer from a kind of sophomore slump, in which they get worse after a first good effort.[223] Although journalists and other onlookers like to make up explanations, we'll see that there's a good statistical reason that this should not be so surprising or need so much explanation. They're just suffering what's known as regression to the mean.

[223] Used as an adjective, "sophomore" means the second of something.

Francis Galton and height

In 1886, Francis Galton was the first to notice the phenomenon and give it a name, when he moved from studying seeds to people, and wrote,[224]

> When Mid-Parents are taller than mediocrity, their Children tend to be shorter than they. When Mid Parents are shorter than mediocrity, their Children tend to be taller than they.

[224] Galton, Francis. 1886. Regression towards mediocrity in hereditary stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*. 15:246–263.

Translating to modern English, Galton was saying that tall parents are likely to have children shorter than they are and short parents likely to have children taller than themselves. Galton called this "regression toward mediocrity," with "mediocrity" denoting the median.[225]

Galton went further and identified the factor by which this regression occurred, with a greater difference occurring the larger the parents deviated above or below the median.

[225] In the kinds of roughly symmetric distributions with which Galton was working, the mean and median are roughly the same.

Binomial example

We're going to use simulation to demonstrate what regression to the mean looks like in a controlled situation. Let's suppose we have $N = 500$ robot athletes, all of whom have exactly the same chance $\theta = 0.3$ of success.[226]

A season's outcomes may be represented as a random variable $U = U_1, \ldots, U_N$, where $U_n \in 0 : M$. Now let's imagine we're going

[226] Imagine spiking serves, hitting baseballs, sinking putts, ringing horseshoes, bullseying darts, or curling stones to a mark; the chance of success is arbitrarily chosen to make the example concrete.

to simulate a season in which each each robot athlete gets exactly $M = 50$ chances,

$$U_n \sim \text{binomial}(M, \theta).$$

In the time off between seasons, the manager decides that robots who had below average performance would have a better chance of success with a fresh coat of bright orange paint. Assuming the paint has no effect on the robots, who are only following programming. The second season's outcome can be represented as a variable $V = V_1, \ldots, V_N$, where

$$V_n \sim \text{binomial}(M, \theta).$$

We can write a simulation as easily as

```
N = 500
M = 50
theta = 0.3
for (n in 1:N)
  u[n] = binomial_rng(M, theta)
  v[n] = binomial_rng(M, theta)
```

Here's a bar plot of the robots scores in the first season.

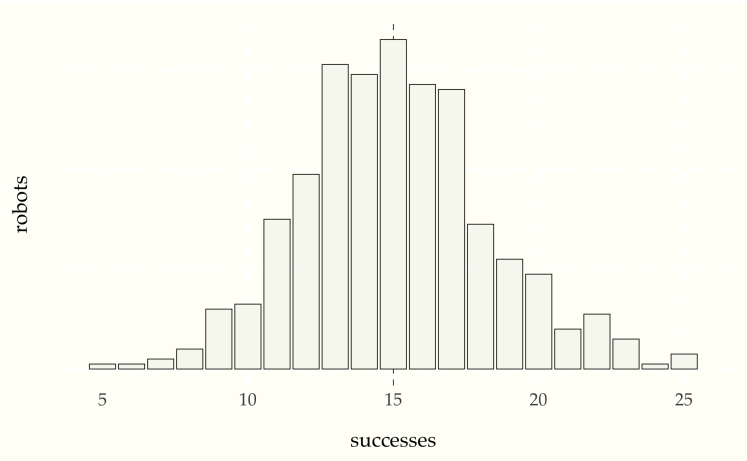


Figure 61: Histogram of first season robots for $N = 500$ given $M = 50$ attempts at a task for which they each have a 30 percent chance of success. The vertical dotted line is at the expected performance ($N \times \theta = 15$).

One way to evaluate whether the treatment had any effect is to look at the painted robots and see if their level of success improved on average after the paint job. This requires retrieving the difference in successes for each of the repainted robotos. We have assumed the robots that performed worse than average were repainted.

```
repainted = (u < 15)
improvement = v[repainted] - u[repainted]
print 'average improvement of repainted robots = ' mean(improvement)
```

The value of the expression `u < 15` is the sequence of indexes for values of `u` that are less than fifteen. Thus `u[repainted]` picks out those values that are less than fifteen in the original season, and `u[repainted]` the corresponding performances in the second season (which may be greater than fifteen).

Running this gives us

```
average improvement of repainted robots = 2.8
```

We can see that the new paint job really gave those underperforming robots a boost in spirit. Now let's see what happened to the robots who performed at or above average in the first season and were thus left with their original paint job.

```
oldpaint = (u >= 15)
decline = v[oldpaint] - u[oldpaint]
print 'average decline of unrepainted robots =' mean(decline)
```

Now let's run it.

```
average decline of unrepainted robots = -2.2
```

The average performance of the unrepainted robots went down. Maybe they were jealous of the repainted robots?

Regression to the mean explained

All that's really going on is an effect known as *regression to the mean*. Suppose a robot athlete had 10 successes in the first season. That's pretty unlucky, because it's well below the expectation of 15 successes. In general, if U_n anv V_n have the same distribution and

$$\mathbb{E}[U_n] \;=\; \mathbb{E}[V_n],$$

as it does in our robot athlete example, then if the result of the first season is $U_n = m$ successes, the average change expected in the second season is

$$\mathbb{E}[V_n - m] \;=\; \mathbb{E}[V_n] - m.$$

For example, a robot athlete with 10 successes in the first season still has a 30% chance of success and an expected number of successes equal to 15 in 50 attempts. Thus we expect an improvement of 5 successes in the second season. Similarly a robot with 18 successes in the first season is expected to decline in the second season by 3 successes. We can summarize the expected decline with a plot.

So while it may look like there is an improvement due to painting the underperforming robots, this is attributable entirely due to the regression to the mean effect.

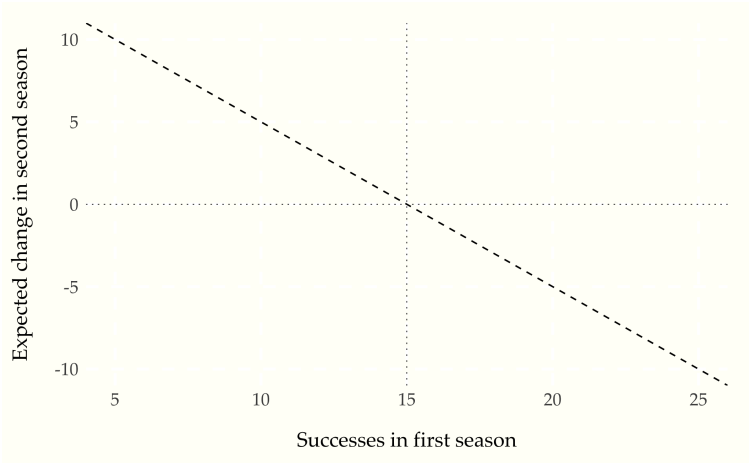


Figure 62: Expected change in performance in second season based on first season performance for robots with 30 percent chance of success in 50 attempts. The horizontal dotted line is at zero, or no change year over year, and the veritical line is at the expected number of successes. Any robot with fewer than the expected number of successes is expected to improve with no treatment, whereas one with more than the expexted number of successes is expected to decline.

Markov Chain Monte Carlo Methods

Monte Carlo Methods

Monte Carlo methods[227] simply use simulation to calculate numerical solutions to definite integrals, usually in high dimensions.[228] We have seen numerous examples in earlier chapters.

When we can simulate a sample of independent draws $\theta = \theta^{(1)}, \ldots, \theta^{(M)}$ according to the posterior $p(\theta \mid y)$, we can use them to calculate parameter estimates, event probabilities, and other expectations. The convergence of such estimates is governed by the central limit theorem and proceeds at an expected error rate of $\mathcal{O}(\frac{1}{\sqrt{M}})$.[229] This rate is manageable in most circumstances, but cannot be used for high precision applications because the number of iterations required for a given number of decimals of precision grows exponentially.[230]

[227] Monte Carlo methods are so-called because of the Monte Carlo Casino (shown below), located in the Principality of Monaco.

© Z.graber, own work, CC-BY-SA 3.0

[228] The book could've been subtitled "A Monte Carlo approach," but like putting "Bayesian" before "statistics", it's needlessly obscure nomenclature.

[229] The reduction in error is all relative to the scale of the quantity being sampled.

[230] Each additional decimal digit requires the error to be reduced by a factor of 10, and thus requires 100 times as much computation, because $\frac{1}{\sqrt{100}} = \frac{1}{10}$.

Markov chain Monte Carlo methods

For most statistical models we want to fit for applications, no methods exist for taking independent draws from the posterior. What we can do in most cases is create a Markov chain $\theta = \theta^{(1)}, \ldots, \theta^{(M)}$ of draws, the stationary distribution of which is the posterior $p(\theta \mid y)$ of interest. We then use the elements of the chain θ to estimate expectations and quantiles in the same way as the independent sample from the posterior. This is the basis of Markov chain Monte Carlo (MCMC) methods.

As we saw in the chapter on finite Markov chains, the rate of convergence varies depending on how much each draw $\theta^{(m+1)}$ depends on the previous draw $\theta^{(m)}$.[231]

[231] If the $\theta^{(m)}$ are independent, the central limit theorem governs the convergence.

Continuous-state Markov chains

The definitions we have already made for Markov chains apply to chains with continuous values. A random process $Y = Y_1, Y_2, \ldots, $ has discrete time steps but may have discrete or continuous

values or even multivariate values, which may themselves be discrete, continuous, or a mixture of the two. Such a process is a *Markov chain* if for all t, and all states $y_1, \ldots, y_{t+1}$,

$$p_{Y_{t+1}|Y_t}(y_{t+1} \mid y_t) \ = \ p_{Y_{t+1}|Y_t,Y_{t-1},\ldots,Y_1}(y_{t+1} \mid y_t, y_{t-1}, \ldots, y_1).$$

We are also assuming that our Markov chains are *time homogeneous* in that the conditional probability distribution of the next state is always the same. That means that for all t', the conditional distribution of the next element at t is the same as that at t',

$$p_{Y_{t+1}|Y_t} \ = \ p_{Y_{t'+1}|Y_{t'}}.$$

All of the Markov chains we will consider will have transitions that are time homogeneous.

To simplify notation, we will write p_t for $p_{Y_{t+1}|Y_t}$

Stationary distributions

Let's look at univariate continuous Markov chains first. Such a chain has real-valued states $Y_t \in \mathbb{R}$. Let

$$\tau(u, v) = p_{Y_{t+1}|Y_t}(v \mid u)$$

be the conditional distribution of the next state v if the current state is u, where $u, v \in \mathbb{R}$. We will call τ the transition distribution of the chain.

A density function $\pi(u)$ for $u \in \mathbb{R}$ is the density function of the *stationary distribution* for transition function $\tau(u, v)$ if

$$\pi(u) \ = \ \int_{\mathbb{R}} \pi(v) \times \tau(v, u) \, \mathrm{d}v.$$

In words, π is a stationary density if its value at a point u is the the average transition probability $\tau(v, u)$, where the average is weighted by the stationary density of v.

In order to sample from the posterior of a model with posterior density $p(\theta \mid y)$ with parameters θ conditioned on observed data y, we will construct Markov chains for which

$$\pi(\theta) = p(\theta \mid y).$$

We can also think of continuous Markov chains in terms of volumes. No matter how the space is partitioned into disjoint volumes, the probabilities of being in a volume and transitioning between volumes defines a discrete Markov chain.[232]

[232] The stationary probability of a volume $U \subseteq \mathbb{R}^N$ is just

$$\pi(U) = \int_U \pi(u) \mathrm{d}u.$$

and the probability of transitioning to volume $V \subseteq \mathbb{R}^N$ conditioned on being at state $u \in \mathbb{R}^N$ is given by

$$\tau(u, V) = \int_V \pi(v) \tau(u, v) \mathrm{d}v.$$

Reversibility

The notion of reversibility extends to continuous chains, where τ is said to be *reversible* with respect to π if for all u, v,

$$\pi(u) \times \tau(u, v) = \pi(v) \times \tau(v, u).$$

If the transition function of a Markov chain is reversible with respect to π, then π is the stationary distribution for the chain. The other way around, not every transition function with a stationary distribution is reversible.

Periodic chains

Continuous chains may exhibit periodicity just like discrete chains. The idea is that it revisits volumes in a predictable way. For example, consider a chain where the space has been divided into four non-overlapping regions A, B, C, D, and where

$$\begin{aligned} \Pr[Y_{t+1} \in B \mid Y_t \in A] &= 1 \\ \Pr[Y_{t+1} \in C \mid Y_t \in B] &= 1 \\ \Pr[Y_{t+1} \in D \mid Y_t \in C] &= 1 \\ \Pr[Y_{t+1} \in A \mid Y_t \in D] &= 1 \end{aligned}$$

We can easily simulate such a chain by taking A, B, C, D to be unit boxes in each quadrant of the real plane, starting in the positive (upper right) quadrant and moving clockwise. The Markov chain may be simulated as.

```
y[1] <- (uniform_rng(0, 1), uniform_rng(0, 1))
for (t in 2:T)
  if (y[t - 1] in A)
    y[t] = uniform_rng(B)
  else if (y[t - 1] in B)
    y[t] = uniform_rng(C)
  else if (y[t - 1] in C)
    y[t] = uniform_rng(D)
  else
    y[t] = uniform_rng(A)
```

Let's plot that for $T = 20$ steps.

We say that a Markov chain $Y = Y_1, Y_2, \ldots$ is *periodic* if there are volumes $A_1, \ldots, A_K \subseteq \mathbb{R}^N$ such that the chain moves deterministically from A_1 to A_2 to A_3 and finally from A_K back to A_1. In symbols, a chain is defined to be periodic if

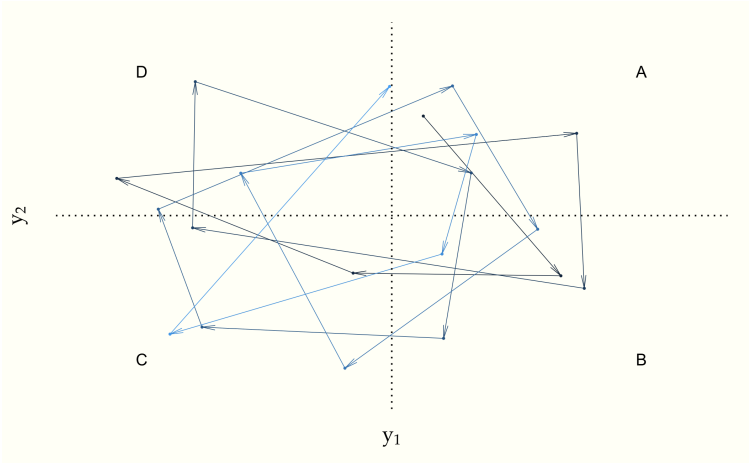


Figure 63: Plot of 20 steps of a periodic continuous Markov chain. Each value is drawn from a unit quadrant starting from the upper right (positive) quadrant and proceeding in a clockwise order.

$$\Pr[Y_{t+1} \in A_{k+1} \mid Y_t \in A_k] = 1 \text{ if } 1 \leq k < K$$

and

$$\Pr[Y_{t+1} \in A_1 \mid Y_t \in A_K] = 1.$$

Irreducibility

Roughly speaking, a Markov chain on a continuous space is *irreducible* if every subset of nonzero volume has a positive probability of eventually being visited from any other point in the space.[233]

[233] In general, a set $A \subseteq \mathbb{R}^N$ has hypervolume

$$\text{vol}(A) = \int_A 1 \, \mathrm{d}u = \int_{\mathbb{R}^N} \mathrm{I}[u \in A] \, \mathrm{d}u.$$

Ergodicity

A Markov chain is *ergodic* if it is aperiodic and irreducible. Ergodicity is important because it ensures that we converge to the stationary distribution if there is one.

Fundamental theorem of Markov chain Monte Carlo. If the discrete time, time-homogeneous Markov chain $Y = Y_1, Y_2, \ldots$ with $Y_t \in \mathbb{R}^N$ is ergodic and has stationary distribution π, then

$$\lim_{t \to \infty} \Pr[Y_t \in A \mid Y_1 = y] = \int_{\mathbb{R}^N} \mathrm{I}[y \in A] \times \pi(y) \, \mathrm{d}y.$$

Furthermore, for well-behaved functions f,[234]

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} f(Y_t) = \int_{\mathbb{R}^N} f(u) \times \pi(u) \, \mathrm{d}u.$$

[234] The well-behavedness required here is convergence of the absolute expectation,

$$\int_{\mathbb{R}^N} |f(y)| \times \pi(y) \, \mathrm{d}y < \infty.$$

Random Walk Metropolis

The Metropolis algorithm is a general purpose technique to sample from an arbitrary target density.[235] The algorithm constructs a Markov chain whose stationary distribution is equal to the target density, then runs the chain long enough to draw a sample from the target density.

[235] We are particularly interested in sampling from target posterior distributions in order to compute expectations corresponding to parameter estimates, event probabilities, predictions, and comparisons.

Random-walk Metropolis

Suppose $p(\theta)$ is the density of an N-dimensional target distribution over from which we want to sample.[236] For the sake of convenience, we will assume it has support on all real values, i.e., $p(\theta) > 0$ for all $\theta \in \mathbb{R}^N$.[237]

[236] Typically, this is a posterior $p(\theta \mid y)$ but we will suppress y here as it remains constant throughout.

[237] Most distributions with constrained support on $\mathbb{R}^N$ can be smoothly transformed to have support on all of $\mathbb{R}^N$.

The random walk Metropolis algorithm generates a Markov chain $\theta^{(1)}, \theta^{(2)}, \ldots$ whose stationary distribution is $p(\theta)$. The algorithm can start anywhere, so we will simply start it at the origin,[238]

[238] Later, we will have motivation to simulate multiple chains from diffuse starting locations, but for now, starting at the origin keeps things simple.

$\theta^{(1)} = 0$.

For each subsequent step $m + 1$, we first simulate a proposed jump,[239]

[239] The algorithm may be generalized to allow arbitrary jump proposal distributions. Popular choices include varying scale by dimension, multivariate normals to account for correlation, and longer tails.

$$\epsilon^{(m)} \sim \text{normal}(0, \sigma),$$

and a uniform variate,

$$u^{(m)} \sim \text{uniform}(0, 1),$$

to inform a probabilistic decision of whether or not to move from state $\theta^{(m)}$ to state $\theta^{(m)} + \epsilon^{(m)}$. With these simulated values in hand, the random walk Metropolis algorithm sets the next state to be

$$\theta^{(m+1)} = \begin{cases} \theta^{(m)} + \epsilon^{(m)} & \text{if } u^{(m)} < \dfrac{p\left(\theta^{(m)} + \epsilon^{(m)}\right)}{p\left(\theta^{(m)}\right)}, \\ \theta^{(m)} & \text{otherwise} \end{cases}$$

Given the definition, we always accept the proposed jump if it takes us to a state of higher density, i.e., if $p\left(\theta^{(m)} + \epsilon^{(m)}\right) > p\left(\theta^{(m)}\right)$.

If the jump takes us to a state of lower density, we accept the jump with probability equal to the density ratio of the proposed state and current state. This means the continuous-state Markov chain defined by the Metropolis algorithm can stay in the same state from time step to time step.[240]

[240] In keeping with our notation for simulation, we use m rather than t to index the "time" steps.

The pseudocode for the random walk Metropolis algorithm just follows the mathematical definition of the algorithm, using local variables for ϵ and u. As inputs, it takes the number M of steps to simulate, the jump scale σ, and the target density $p(\theta)$, and it returns the simulated Markov chain $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(M)}$.

```
theta(0) = 0
for m in 2:M
  for (n in 1:N)
    epsilon[n] = normal_rng(0, sigma)
  u = uniform_rng(0, 1)
  accept(m) = u < p(theta(m) + epsilon) / p(theta(m))
  if (accept(m))
    theta(m + 1) = theta(m) + epsilon
  else
    theta(m + 1) = theta(m)
accept_rate = sum(accept) / (M - 1)
```

The final step in the computation calculates the *acceptance rate*, which is the proportion of proposed jumps that were accepted in the particular execution of the algorithm.[241]

[241] This rate will vary based on the random number generator.

Because we may be working in high dimensions and with big data sets, we need to be careful to avoid arithmetic underflow by carrying out all computation on the log scale. Specifically, we assume that rather than a function to compute `p()`, we are given a function to compute `log p()`. Because both sides of the acceptance test inequality are positive and the logarithm function is monotonic, we can just take the log of both sides and define

```
  accept = log(u) < log(p(theta(m) + epsilon) / p(theta(m)))
```

In practice, we may only know the density up to an additive constant. We'll only be able to compute $\log q(\theta)$, where

$$\log q(\theta) = \log p(\theta) + \text{const.}$$

for some unknown constant that doesn't depend on θ. Although unregularized densities may be negative, we don't get in trouble with negative logarithms because the constants cancel out,

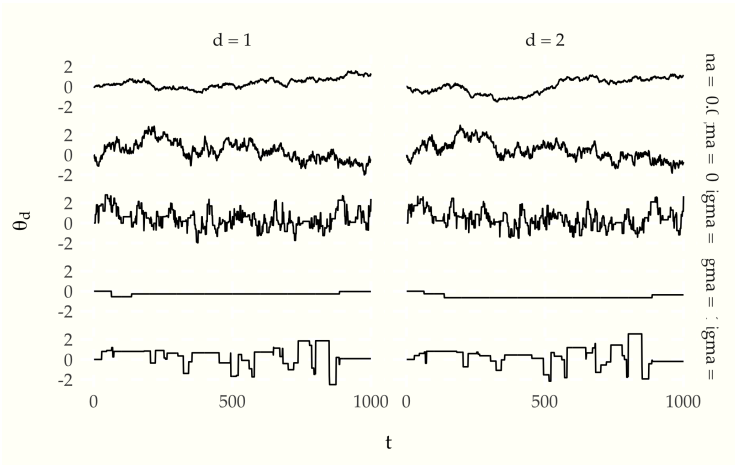
$$\begin{aligned}\left(\log p\left(\theta^{(m)}+\epsilon\right)+\text{const}\right)-\left(\log p\left(\theta^{(m)}\right)+\text{const}\right) &= \log p\left(\theta^{(m)}+\epsilon\right)-\log p\left(\theta^{(m)}\right)\\ &= \log\frac{p\left(\theta^{(m)}+\epsilon\right)}{p\left(\theta^{(m)}\right)}.\end{aligned}$$

Metropolis example

As an example, we can start with a two-dimensional density $p(\theta)$ in which the values $\theta = (\theta_1, \theta_2)$ are positively correlated,[242]

$$\log p(\theta) = -2.6\,\theta_1^2 \;-\; 2.6\,\theta_2^2 \;+\; 4.7\,\theta_1\,\theta_2 + \text{const.}$$

Let's see what we get when taking draws using the Metropolis algorithm. First, let's try $\sigma \in 0.125, 0.25, 0.5$ and start at $\theta(1) = (0, 0)$.[243]



Here's a scatterplot of independent draws.[244]

[242] The distribution is bivariate normal located at the origin, with unit scale and 0.9 correlation, i.e., with covariance matrix

$$\Sigma = \begin{bmatrix}1 & 0.9\\ 0.9 & 1\end{bmatrix}$$

and density

$$\begin{aligned}\log p(\theta) &= \log\text{ multi-normal}\left(\theta \;\middle|\; 0, \Sigma\right)\\ &\propto -\tfrac{1}{2}\,\theta^\top\,\Sigma^{-1}\,\theta \;+\; \text{const}\\ &\approx -2.6\times\theta_1^2 \;-\; 2.6\times\theta_2^2 \;+\; 4.7\times\theta_1\theta_2 + \text{con}\end{aligned}$$

[243] $(0, 0)$ is coincidentally the point at which the example density is maximized. This requires escaping from the point of maximum log density.

[244] Because it's bivariate normal, we can cheat and use off-the-shelf algorithms to generate independent draws.

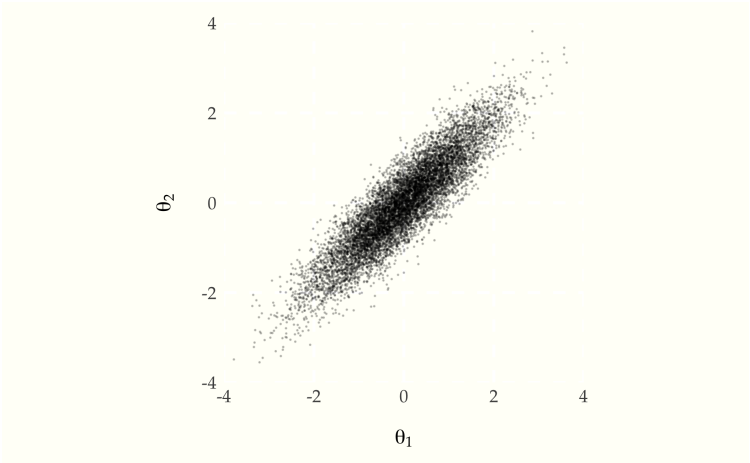


Figure 64: Scatterplot of draws from a bivariate normal with unit scale and correlation 0.9.

Monitoring Approximate Convergence

We know that if we run an ergodic Markov chain long enough, it will eventually approach stationarity. We can even use the entire chain for estimation if we run long enough because eny finite error will become infinitesimal in the asymptotic regime.

However, we do not live in the asymptotic regime. We need results that we can trust from finitely many draws. If we happened to know our chains were geometrically ergodic, so that we know they converged very quickly toward the stationry distribution, then our lives would be easy and we cold proceed with a single chain and as long as there weren't numerical problems, we could run it a long time and trust it.

We are also faced with distributions defined implicitly as posteriors, whose geometric ergodicity is difficult, if not impossible, to establish theoretically. Therefore, we need finite-sample diagnostics in order to test whether our chains have reached a stationary distribution which we can trust for estimation.

Running multiple chains

One way to diagnose non-convergence is to run multiple chains and monitor various statistics, which should match among the different chains. As an example, consider the two-dimensional Metropolis example. Starting four chains in the corners and running for various number of time steps shows how the chains approach stationarity.

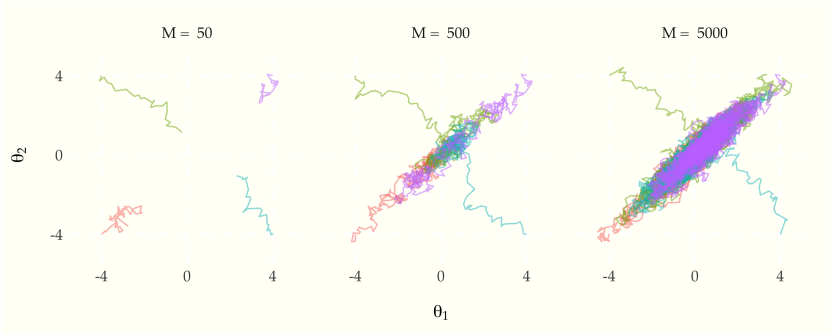


Figure 65: The evolution of four random walk Metropolis Markov chains, each started in a different corner of the plot. The target density is bivariate normal with correlation 0.9 and unit variance; the random walk step size is 0.2. After $M = 50$ iterations, the chains have not arrived at the typical set. After $M = 500$ iterations, the chains have each arrived in the typical set, but they have not had time to mix. After $M = 5000$ iterations, the chains are mixing well and have visited most of the target density.

Exponential and Poisson Distributions

In this chapter, we'll see how the exponential distribution can be used to model the waiting time between events. For example, waiting times might be used to model the time between salmon swimming upstream, the rate at which a user sends text messages, or the time between neutrinos arriving at a particle detector.

The Poisson distribution arises as the number of events in a fixed period of time, such as the number of salmon per hour or number of messages per day.[245]

[245] We also consider generalizations to spatial and volumetric processes, such as the number of graffiti tags in given area of a city or the number of whales in a volume of the ocean. These may even be combined to model, say, the number of whales in a volume of the ocean in a given month.

The Exponential distribution

The exponential distribution can be used to model the waiting time between events under two idealized assumptions. The first assumption is that the arrival process is homogeneous in time. No matter the time of day or night, arrivals happen at the same rate.[246] The second assumption is that the times between successive events is independent. It doesn't matter if it's been an hour since the last text message was sent or if one was just sent ten seconds ago, the probability another message will arrive in the next minute is the same. In other words, the expected waiting time for the next arrival doesn't change based on how long you've already been waiting.

[246] Obviously, this is too limiting for many applications, such as anything to do with human or animal activity; in such cases, these simple processes are used as building blocks in more complex spatio-temporal models.

A distribution of arrival times satisfying these properties is the exponential distribution. If $Y \sim \text{exponential}(1)$ has a standard exponential distribution, then its density has a particularly simple form[247]

$$p_Y(y) = \exp(-y).$$

[247] We can work backward from this density to the definite integral

$$\int_0^\infty \exp(-y) = 1.$$

For example, let's plot a few simulations of arrivals where the waiting time between arrivals is distributed according to a standard exponential distribution.[248]

[248] We will circle back and explain how to generate exponential variates shortly.

We can simulate a sequence of arrivals during λ time units, assuming the waiting time between arrivals is distributed as standard exponential. We start at time $t = 0$ and continue adding the waiting times $w_n \sim \text{exponential}(1)$ until we pass a time λ, at which point we return the arrival times we have accumulated.[249]

[249] The loop notation `1:infinity` is meant to indicate that `n` is unbounded. Such "forever" loops must be broken with internal logic such as the return in this case.

```
t = 0
for n in 1:infinity
  w[n] = exponential_rng(1)
  t += w[n]
  if (t > lambda)
    return y
  y[n] = t
```

Let's look at four realizations of this process up to time $\lambda = 10$.

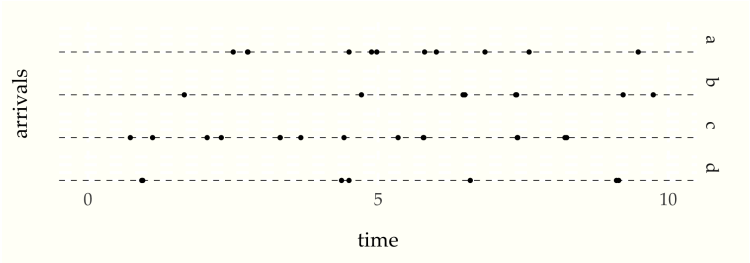


Figure 66: Four simulations of the first $\lambda = 10$ time units of an arrival process where waiting times are distributed as exponential(1).

A different number of arrivals occur in the different simulations.

Scaled exponential

As with any continuous distribution, the exponential may be scaled. As with scaling the normal distribution, this simply stretches or shrinks the distribution without changing its basic shape. In the normal distribution, the scale parameter σ multiplies a standard normal variate to get a scale σ variate. Instead of multiplying a standard exponential variate by a scale, it is traditional to divide it by a rate (an inverse scale).

If $U \sim \text{exponential}(1)$ is a standard exponential variable, then $Y = U/\lambda$ is an exponential variable with rate λ (i.e., with scale $1/\lambda$), for which we write $Y \sim \text{exponential}(\lambda)$, and have the following[250]

$$\begin{aligned} p_Y(y \mid \lambda) &= \lambda \times p_U\left(\lambda \times y\right) \\ &= \lambda \times \exp\left(\lambda \times -y\right). \end{aligned}$$

[250] This is a straightforward derivation using the usual Jacobian formula, for which the adjustment for the inverse transform is

$$\left| \frac{\mathrm{d}}{\mathrm{d}y} \lambda \times y \right| = \lambda.$$

Simulating from the exponential distribution

Simulating a standard exponential variate is straightforward because of the simple form of the cumulative distribution function. If $Y \sim \text{exponential}(1)$, then the probability density function is $p_Y(y) = \exp(-y)$ and the cumulative distribution function $F_Y : [0, \infty) \to [0, 1)$ is given by

$$\begin{aligned} F_Y(y) &= \int_0^y \exp(-v')\mathrm{d}v \\ &= 1 - \exp(-y). \end{aligned}$$

This function is easily inverted to produce the inverse cumulative distribution function, $F_y^{-1} : [0,1) \to [0,\infty)$,

$$F_Y^{-1}(u) = -\log(1-u).$$

If we take $U \sim \text{uniform}(0,1)$, then

$$F_Y^{-1}(u) \sim \text{exponential}(1).$$

This is a very general trick for distributions for which we can compute the inverse cumulative distribution function. We first saw this used to generate logistic variates by log-odds transforming uniform variates.[251]

[251] The log-odds function, written $\text{logit}(u)$, is the inverse cumulative distribution function for the standard logistic distribution.

The pseudocode follows the math, so we can generate standard exponential variates as follows.

```
u <- uniform(0, 1)
y <- -log(1 - u)
```

It is traditional here to replace the term $\log 1 - u$ with the term $\log u$ because if $u \sim \text{uniform}(0,1)$, then we also know $1 - u \sim \text{uniform}(0,1)$. We can then generalize to the nonstandard exponential with rate (i.e., inverse scale) λ by dividing. This gives us the following exponential distribution pseudorandom number generator.

```
exponential_rng(lambda)
  u <- uniform(0, 1)
  y <- -log(u) / lambda
  return y
```

Memorylessness of exponential waiting times

Suppose we simulate a sequence standard exponential waiting times, $W_1, \ldots, W_N \sim \text{exponential}(1)$. Now what if we look at the distribution of all of the W and compare it to the distribution of just those $W > 1$ and those $W > 2.5$. To make sure we have the same number of draws so the histograms are scaled, we'll take 1000 of each situations by using simple rejection sampling.

```
for (m in 1:M)
  w[m] = -1
  while (w[m] < min)
    w[m] = exponential_rng(1)
```

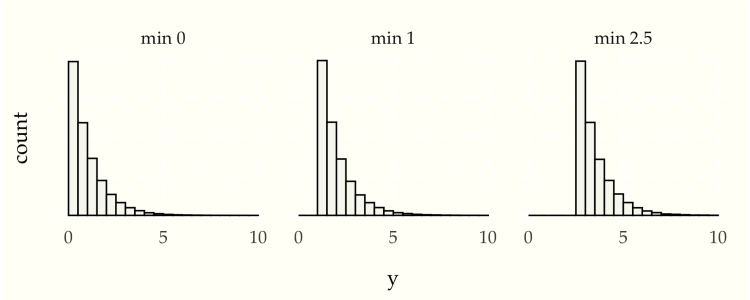


Figure 67: Plot of 10 000 draws from the standard exponential distribution (left) and discarding all draws below 1 (center) or 2.5 (right). Each histogram is the same, just shifted. This illustrates the memoryless of the exponential distribution as a model of waiting times—no matter how long you have already waited, the remaining wait time distribution is the same.

Let's plot histograms of 10 000 draws for $\text{min} = 0, 1, 2.5$.

The resulting histograms are almost identical because each condition has exactly the same distribution shifted over by the amount of wait time already experienced.

We can characterize this property in terms of the probability density function. If $Y \sim \text{exponential}(\lambda)$, for some fixed λ, then for any fixed $c > 0$, we have

$$p_Y(y \mid \lambda) \propto p_Y(y + c \mid \lambda).$$

The Poisson distribution

The Poisson distribution is a discrete distribution over counts $\mathbb{N} = 0, 1, 2, \ldots$ which can be defined as the number of arrivals that occur in a fixed unit λ of time when the waiting times w_n between arrivals are independently standard exponential distributed, $w_n \sim \text{exponential}(1)$.

If $Y \sim \text{Poisson}(\lambda)$, then we can simulate values of Y as follows.

```
poisson_rng(lambda)
  t = 0
  y = 0
  while (true)
    w = exponential_rng(1)
    t += w
    if (t > lambda)
      return y
    else
      y += 1
```

We start at time $t = 0$, with $y = 0$ arrivals, then continue simulating arrivals until we have past the total time λ, at which point we report the number of arrivals we have seen before the arrival that put us past time λ.

We can use this simulator to estimate the expectation and variance of a variable $Y \sim \text{Poisson}(\lambda)$, as follows

```
for (m in 1:M)
  y[m] = poisson_rng(lambda)
print 'estimated E[Y] = ' mean(y)
print 'estimated var[Y] = ' variance(y)
```

```
   estimated E[Y] = 9.98
   estimated var[Y] = 9.96
```

Poisson as limit of binomials

Another way to arrive at the Poisson distribution is as the limit of a sequence of binomial distributions. A random variable with a Poisson distribution is unbounded in that any value may arise.[252] A binomial variable on the other hand takes on values between 0 and N for some fixed N. But if we let that the total count N grows without bound while keeping the expected value at λ, the binomial approaches the Poisson distribution,

$$\text{Poisson}(y \mid \lambda) = \lim_{N \to \infty} \text{binomial}(N, \lambda / N).$$

[252] All but a few values will be wildly improbable, but still possible.

We can see what the binomial approximation looks like through simulation. We'll simulate $\text{binomial}(N, 5.5/N)$ for various N and compare with $\text{Poisson}(5.5)$.

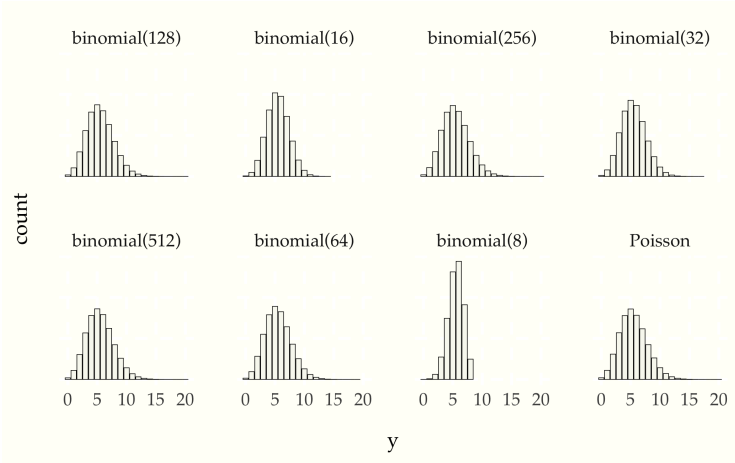


Figure 68: Histograms of 1 000 000 draws for a Poisson(5.5) and successively larger N binomial approximations.

Conjugate Priors

Uniform prior and binomial likelihood

Suppose we have a uniform prior for parameter $\theta \in (0, 1)$,

$$\theta \sim \text{uniform}(0, 1),$$

and combine it with a a likelihood for data $y \in 0 : N$,

$$y \sim \text{binomial}(N, \theta).$$

We know from Bayes's rule that the posterior is proportional to the likelihood times the prior. Writing this out in symbols,

$$\begin{aligned} p(\theta \mid y, N) &\propto \frac{p(y \mid \theta, N) \cdot p(\theta)}{p(y \mid N)} \\ &\propto p(y \mid \theta) \cdot p(\theta) \\ &= \text{binomial}(y \mid N, \theta) \cdot \text{uniform}(\theta \mid 0, 1) \\ &= \binom{N}{y} \cdot \theta^y \cdot (1 - \theta)^{N-y} \cdot 1 \\ &\propto \theta^y \cdot (1 - \theta)^{N-y}. \end{aligned}$$

Now that we know $p(\theta \mid y) \propto \theta^y \cdot (1 - \theta)^{N-y}$, we can follow Laplace in applying Euler's beta function to normalize,[253]

$$\begin{aligned} p(\theta \mid y, N) &= \frac{\theta^y \cdot (1 - \theta)^{N-y}}{\int_0^1 \theta^y \cdot (1 - \theta)^{N-y} \, \mathrm{d}\theta} \\ &= \frac{1}{\text{B}(y, N - y)} \cdot \theta^y \cdot (1 - \theta)^{N-y} \\ &= \frac{\Gamma(N)}{\Gamma(y) \cdot \Gamma(N - y)} \cdot \theta^y \cdot (1 - \theta)^{N-y}. \end{aligned}$$

Another way of arriving at the same result is to just work through Bayes's rule by brute force,

$$p(\theta \mid y) = \frac{p(y \mid \theta) \cdot p(\theta)}{\int_0^1 p(y \mid \theta) \cdot p(\theta) \, \mathrm{d}\theta}.$$

The integral in the denominator is just the beta function given above.

[253] Euler's beta function is defined by

$$\begin{aligned} \text{B}(\alpha, \beta) &= \int_0^1 u^{\alpha-1} \cdot (1 - u)^{\beta-1} \mathrm{d}u \\ &= \frac{\Gamma(\alpha) \cdot \Gamma(\beta)}{\Gamma(\alpha + \beta)}, \end{aligned}$$

where the gamma function is defined by

$$\Gamma(\gamma) = \int_0^\infty u^{\gamma-1} \cdot \exp(-u) \, \mathrm{d}u.$$

The gamma function generalizes the integer factorial function, satisfying

$$\Gamma(u + 1) = u!$$

for all integers $u \in \mathbb{N}$.

Beta prior and binomial likelihood produces a beta posterior

A beta$(1,1)$ distribution is identical to a uniform$(0,1)$ distribution, because

$$\begin{aligned}\text{beta}(\theta \mid 1,1) &\propto \theta^{1-1} \cdot (1-\theta)^{1-1} \\ &= 1 \\ &= \text{uniform}(\theta \mid 0,1).\end{aligned}$$

Now suppose we assume a beta prior with parameters $\alpha, \beta > 0$,

$$\theta \sim \text{beta}(\alpha, \beta).$$

When combined with a binomial likelihood,

$$y \sim \text{binomial}(N, \theta),$$

the result is a posterior of the following form

$$\begin{aligned}p(\theta \mid y, N) &\propto p(y \mid N, \theta) \cdot p(\theta) \\ &= \text{binomial}(y \mid N, \theta) \cdot \text{beta}(\theta \mid \alpha, \beta) \\ &\propto \left(\theta^{y} \cdot (1-\theta)^{N-y}\right) \cdot \left(\theta^{\alpha-1} \cdot (1-\theta)^{\beta-1}\right) \\ &= \theta^{y+\alpha-1} \cdot (1-\theta)^{N-y+\beta-1} \\ &\propto \text{beta}(y+\alpha, N-y+\beta).\end{aligned}$$

The rearrangement of terms in the penultimate step[254] lets us collect a result that matches the kernel of a beta distribution.

The takeaway message is that if the prior is a beta distribution and the likelihood is binomial, then the posterior is also a beta distribution.

[254] This uses the rule of exponents from algebra,

$$\theta^{u} \cdot \theta^{v} = \theta^{u=v}.$$

Conjugate priors

In general, a family $\mathcal{F}$ of priors is said to be *conjugate* to a family $\mathcal{G}$ of likelihood functions, if $p(\theta) \in \mathcal{F}$ and $p(y \mid \theta) \in \mathcal{G}$ imply that $p(\theta \mid y) \in \mathcal{F}$. We have already seen one example of conjugacy, with the family of beta priors and binomial likelihoods,

$$\begin{aligned}\mathcal{F} &= \{\, \text{beta}(\alpha, \beta) \mid \alpha, \beta > 0 \,\} \\ \mathcal{G} &= \{\, \text{binomial}(N, \theta) \mid N \in \mathbb{N},\ \theta \in (0,1) \,\}.\end{aligned}$$

It is no coincidence that the likelihood and prior in our conjugate example have matching forms,

prior	$\text{beta}(\theta \mid \alpha, \beta)$	$\propto$	$\theta^{\alpha-1}$	$\cdot$	$(1-\theta)^{\beta-1}$	$p(\theta)$
likelihood	$\text{binomial}(y \mid N, \theta)$	$\propto$	θ^{y}	$\cdot$	$(1-\theta)^{N-y}$	$p(y \mid \theta)$
posterior	$\text{beta}(\theta \mid y+\alpha,\ N-y+\beta)$	$\propto$	$\theta^{y+\alpha-1}$	$\cdot$	$(1-\theta)^{N-y+\beta-1}$	$p(\theta \mid y)$

Thinking of the exponents y and $N - y$ as success and failure counts respectively, we can think of the exponents $\alpha - 1$ and $\beta - 1$ as the prior number of succcesses and failures.[255] We then just add the prior successes and the likelihood successes to get $y + \alpha - 1$ posterior succesess; the prior failures work similarly, with $\beta - 1$ prior failures and $N - y$ observed failures producing a $N - y + \beta - 1$ posterior failure count.

[255] The prior counts are $\alpha - 1$ and $\beta - 1$ for a total prior count of $\alpha + beta - 2$. The reason for the subtraction is that $\text{beta}(1, 1)$ is the uniform distribution, so $\alpha = 1, \beta = 1$ corresponds to a prior count of zero.

Beta-Bernoulli conjugacy

The Bernoulli distribution is just a single trial binomial distribution. For a single binary observation $y \in \{0, 1\}$ and chance of success parameter θ,

$$\text{bernoulli}(y \mid \theta) \;=\; \text{binomial}(y \mid 1, \theta).$$

Therefore, the beta distribution must also be conjugate to the Bernoulli distribution—if the prior is a beta distribution and the likelihood is a Bernoulli distribution, then the posterior is also a beta distribution. This is evident if we recast the Bernoulli definition in the same form as the beta and binomial distributions,

$$\text{bernoulli}(y \mid \theta) \;=\; \theta^y \cdot (1 - \theta)^{1-y}.$$

Working through the algebra, if $p(\theta) = \text{beta}(\alpha, \beta)$ and the likelihood is $p(y \mid \theta) = \text{bernoulli}(y \mid \theta)$, then the posterior is $p(\theta \mid y) = \text{beta}(\alpha + y, \beta + 1 - y)$.

Said more simply, if the prior is $\text{beta}(\alpha, \beta)$ and we condition on a single binary observation y; if $y = 1$, the updated distribution is $\text{beta}(\alpha + 1, \beta)$; if $y = 0$, the updated distribution is $\text{beta}(\alpha, \beta + 1)$.

Chaining repeated observations

Suppose we do not have a single binary observation, but a whole sequence $y = y_1, \ldots, y_N$, where $y_n \in \{0, 1\}$. Now suppose we start with a prior $p(\theta) = \text{beta}(\theta \mid \alpha, \beta)$. Given no observations, our knowledge of the parameter θ is determined by the prior,

$$p(\theta) = \text{beta}(\theta \mid \alpha, \beta).$$

After the first observation, y_1, we can update our knowledge of θ conditioned on that observation to

$$p(\theta \mid y_1) \;=\; \text{beta}(\theta \mid \alpha + y_1, \beta + 1 - y_1).$$

What is our knowledge of θ after two observations, y_1, y_2? We can apply Bayes's rule, to see that

$$p(\theta \mid y_1, y_2) \propto p(\theta \mid y_1) \cdot p(y_2 \mid \theta).$$

Displayed this way, the distribution $p(\theta \mid y_1)$ is acting like a prior with respect to the likelihood for the single observation y_2. Substituting in the actual distributions, we have

$$\begin{aligned} p(\theta \mid y_1, y_2) &= p(\theta \mid y_1) \cdot p(y_2 \mid \theta) \\ &= \text{beta}(\theta \mid \alpha + y_1, \beta + 1 - y_1) \cdot \text{bernoulli}(y_2 \mid \theta) \\ &\propto \text{beta}(\theta \mid \alpha + y_1 + y_2, \beta + 1 - y_1 + 1 - y_2) \\ &= \text{beta}(\theta \mid \alpha + (y_1 + y_2), \beta + 2 - (y_1 + -y_2)) \end{aligned}$$

We can visualize these chained updates as follows.

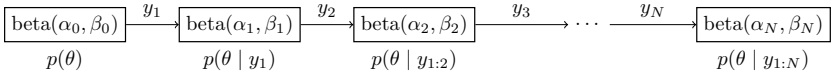


Figure 69: Progress of streaming updates with conjugate priors. There is an initial prior $\text{beta}(\alpha_0, \beta_0)$ and a stream of data $y_1, y_2, \ldots, y_N$. After each data point y_n is observed, the prior parameters $\alpha_{n-1}, \beta_{n-1}$ are updated to the posterior parameters α_n, β_n, which then acts as a prior for subsequent data.

Given a prior $\text{beta}(\alpha_0, \beta_0)$, we will let $\text{beta}(\alpha_n, \beta_n)$ to be the distribution for θ conditioned on observations

$$y_{1:n} = y_1, \ldots, y_n.$$

The values (α_n, β_n) are defined inductively. As a base case (α_0, β_0) are our initial prior parameters. Then inductively, after observing y_{n+1}, we have

$$(\alpha_{n+1}, \beta_{n+1}) \;=\; (\alpha_n + y_{n+1}, \beta_n + 1 - y_{n+1}).$$

Because y_n is binary, we can expand this definition by cases and also write

$$(\alpha_{n+1}, \beta_{n+1}) \;=\; \begin{cases} (\alpha_n + 1, \beta_n) & \text{if } y_{n+1} = 1, \text{ and} \\ (\alpha_n, \beta_n + 1) & \text{if } y_{n+1} = 0. \end{cases}$$

All that is happening is counting—we just add one to the success count if $y_n = 1$ and one to the failure count if $y_n = 0$. By the time we get to the end and have observed $y_{1:N}$, we have

$$(\alpha_n, \beta_n) \;=\; (\alpha_0 + \text{sum}(y),\; \beta_0 + N - \text{sum}(y)).$$

This is, not coincidentally, the same result we would have achieved had we started with the prior $\text{beta}(\alpha_0, \beta_0)$ and observed all $y_1, \ldots, y_N$ simultaneously. In that case, we could use the equivalence with the binomial,

$$\text{binomial}(\text{sum}(y) \mid N, \theta) \;\propto\; \theta^{\text{sum}(y)} \cdot (1 - \theta)^{N - \text{sum}(y)}.$$

Another way of deriving this result is by direct expansion

$$\begin{aligned}
p(\theta \mid y_1, \ldots, y_N) &\propto p(\theta) \cdot p(y_1, \ldots, y_N \mid \theta) \\
&= p(\theta) \cdot p(y_1 \mid \theta) \cdots p(y_N \mid \theta). \\
&= \text{beta}(\theta \mid \alpha_0, \beta_0) \cdot \textstyle\prod_{n=1}^{N} \text{bernoulli}(y_n \mid \theta). \\
&= \theta^{\alpha_0 - 1} \cdot (1 - \theta)^{\beta_0 - 1} \cdot \textstyle\prod_{n=1}^{N} \theta^{y_n} \cdot \theta^{1 - y_n}. \\
&= \theta^{\alpha_0 - 1} \cdot (1 - \theta)^{\beta_0 - 1} \cdot \theta^{\sum_{n=1}^{N} y_n} \cdot (1 - \theta)^{\sum_{n=1}^{N} 1 - y_n} \\
&= \left(\theta^{\alpha_0 - 1} \cdot (1 - \theta)^{\beta_0 - 1}\right) \cdot \left(\theta^{\text{sum}(y)} \cdot (1 - \theta)^{N - \text{sum}(y)}\right) \\
&= \theta^{\alpha_0 + \text{sum}(y) - 1} \cdot (1 - \theta)^{\beta_0 + N - \text{sum}(y) - 1} \\
&\propto \text{beta}(\theta \mid \alpha_0 + \text{sum}(y), \beta_0 + N - \text{sum}(y)).
\end{aligned}$$

Gamma-Poisson conjugacy

The same line of reasoning that led from a binomial distribution to a prior that behaved as prior data, we can reason from the Poisson distribution backward to a conjugate prior. Recall that for a count $y \in \mathbb{N}$ and rate $\lambda \in (0, \infty)$,

$$\text{poisson}(y \mid \lambda) \;=\; \frac{1}{y!} \cdot \lambda^y \cdot \exp(-\lambda).$$

Given this form, the gamma distribution provides a family of conjugate priors, parameters by a shape α and rate (inverse scale) β, as

$$\text{gamma}(\lambda \mid \alpha, \beta) \;=\; \frac{\beta^\alpha}{\Gamma(\alpha)} \cdot \lambda^{\alpha - 1} \cdot \exp(-\beta \cdot \lambda).$$

Now let's work out the conjugacy. Suppose we have a Poisson likelihood with rate parameter λ,

$$\begin{aligned}
p(y \mid \lambda) &= \text{poisson}(y \mid \lambda), \\
&= \tfrac{1}{y!} \cdot \lambda^y \cdot \exp(-\lambda)
\end{aligned}$$

with a gamma prior on the rate parameter,

$$\begin{aligned}
p(\lambda \mid \alpha, \beta) &= \text{gamma}(\lambda \mid \alpha, \beta) \\
&= \tfrac{\beta^\alpha}{\Gamma(\alpha)} \cdot \lambda^{\alpha - 1} \cdot \exp(-\beta \cdot \lambda).
\end{aligned}$$

The posterior is equal to the product of the likelihood and prior up to a proportion,

$$\begin{aligned}
p(\lambda \mid y, \alpha, \beta) &= p(y \mid \lambda) \cdot p(\lambda \mid \alpha, \beta) \\
&= \text{poisson}(y \mid \lambda) \cdot \text{gamma}(\lambda \mid \alpha, \beta) \\
&= \left(\tfrac{1}{y!} \cdot \lambda^y \cdot \exp(-\lambda)\right) \cdot \left(\tfrac{\beta^\alpha}{\Gamma(\alpha)} \cdot \lambda^{\alpha - 1} \cdot \exp(-\beta \cdot \lambda)\right) \\
&\propto \left(\lambda^y \cdot \exp(-\lambda)\right) \cdot \left(\lambda^{\alpha - 1} \cdot \exp(-\beta \cdot \lambda)\right) \\
&= \lambda^{y + \alpha - 1} \cdot \exp(-(\beta + 1) \cdot \lambda) \\
&\propto \text{gamma}(\alpha + y, \beta + 1).
\end{aligned}$$

Now suppose we have a sequence of count observations $y_1, \ldots, y_N \in \mathbb{N}$. Assuming a poisson$(\lambda)$ likelihood with prior gamma$(\lambda \mid \alpha_0, \beta_0)$, we can update with y_1 to produce a posterior

$$p(\lambda \mid y_1, \alpha_0, \beta_0) \;=\; \text{gamma}(\alpha_0 + y_1, \beta_0 + 1).$$

Using this as a prior for y_2, the posterior becomes

$$p(\lambda \mid y_1, y_2, \alpha_0, \beta_0) \;=\; \text{gamma}(\alpha_0 + y_1 + y_2, \beta_0 + 2).$$

Extending this logic, we can take a whole batch of observations $y_1, \ldots, y_N$ at once to produce a posterior,

$$p(\lambda \mid y_1, \ldots, y_N, \alpha_0, \beta_0) \;=\; \text{gamma}(\alpha_0 + \text{sum}(y), \beta_0 + N).$$

Worked out one step at a time, the posterior after observing $y_1, \ldots, y_n$ is

$$p(\lambda \mid y_1, \ldots, y_n, \alpha_0, \beta_0) \;=\; p(\lambda \mid \alpha_n, \beta_n)$$

where (α_0, β_0) are given as initial priors, and

$$(\alpha_{n+1}, \beta_{n+1}) \;=\; (\alpha_n + y_{n+1}, \beta_n + 1).$$

Thinking of the gamma prior in terms of Poisson data, gamma(α, β) represents a total of β prior observations, with a sum total of α. For example, if we have a single prior observation y, that corresponds to a gamma$(y, 1)$ prior; if we have three prior observations $12, 7, 9$, that'd be represented by a gamma$(26, 3)$ prior.

Typical Sets

In this chapter, we'll cover some important properties of sets of random draws from a distribution that provides important insight into the behavior of samplers, particularly in higher dimensions.

Most likely versus expected outcomes

Suppose we take a sequence $Y = Y_1, \ldots, Y_N$ of random variables which are independent and identically distributed (i.i.d.) according to

$$Y_n \sim \text{bernoulli}(\theta).$$

For concreteness, let's suppose $N = 100$ and $\theta = 0.8$.[256] Given the i.i.d. assumption, we can write the joint probability function as

$$\begin{aligned} p_Y(y) &= \prod_{n=1}^{100} p_{Y_n}(y_n) \\ &= \prod_{n=1}^{100} 0.8^{y_n} \cdot (1 - 0.8)^{1 - y_n} \\ &= 0.8^{\text{sum}(y)} \cdot 0.2^{\text{sum}(1-y)}. \end{aligned}$$

[256] That means there's an 80% chance $Y_n = 1$ and a 20% chance $Y_n = 0$.

What's the most likely value for Y? For each Y_n, the most probable result is success (i.e., 1). Because the Y_n are independent, that means the most probable joint result is all successes. In symbols, this run of all successes maximizes the joint probability mass function $p_Y(y)$,

$$\begin{aligned} y^* &= \arg\max_y p_Y(y). \\ &= (\underbrace{1, 1, \ldots, 1}_{100 \text{ times}}). \end{aligned}$$

Although a straight run of succesess is the most likely result, given that it requires one hundred consecutive successes with only an eighty percent chance each, the overall probability is still very low.

$$p_Y(\,(1, 1, \ldots, 1)\,) \;=\; 0.8^{100} \cdot 0.2^0 \;\approx\; 2.0 \cdot 10^{-10}.$$

In other words, we should be quite surprised to see one hundred straight successes.[257]

[257] Joe DiMaggio's record 56 consecutive baseball games with a hit is one of sports most unbreakable records. An eighty percent chance of success per game requires 4 at bats with a .333 batting average (i.e., chance of success), $(1 - 0.333)^4 \approx 0.2$.

The following plot demonstrates the range of total successes we're likely to see by simulating 100 trials and counting the number of successes. This whole process is repeated multiple times to produce the data for the histogram.

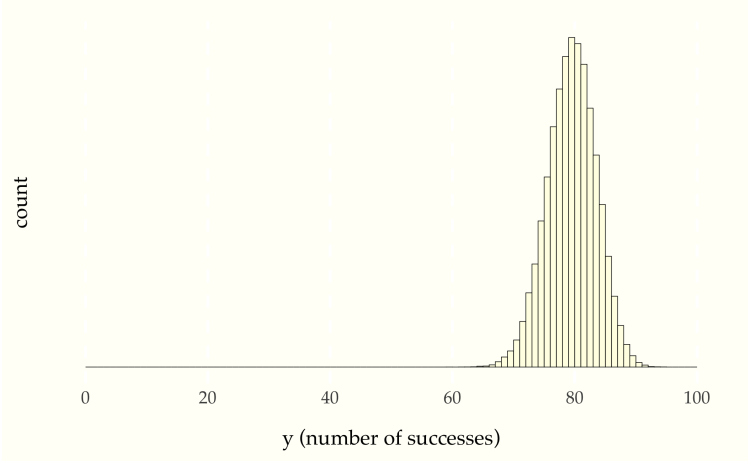


Figure 70: Histogram of 100 000 simulations of the number of successes in 100 independent binary trials with an 80 percent chance of success per trial. In symbols, this is a histogram of draws from $y \sim \text{binomial}(100, 0.8)$.

What is the expected number of successes in one hundred independent trials where each has an eighty percent chance of success? Eighty, as the following derivation shows.

$$\begin{aligned} \mathbb{E}[Y_1 + \cdots + Y_{100}] &= \mathbb{E}[Y_1] + \cdots + \mathbb{E}[Y_{100}] \\ &= \underbrace{0.8 + \cdots + 0.8}_{100 \text{ times}} \\ &= 80. \end{aligned}$$

How do we reconcile the fact that we expect to see eighty successes when one hundred successes being the most likely outcome? By considering the fact that there is only a single sequence among the $2^{100} \approx 10^{30}$ possible outcomes that constitutes all successes, wheras there are a whole lot of sequneces Y with eighty successes and twenty failures. In fact, there are a whopping

$$\binom{100}{80} = \frac{100!}{80! \cdot (100 - 80)!} \approx 5 \cdot 10^{20}$$

possible ways to sequence eighty successes and twenty failures. Therefore, the total probability of seeing eighty successes is the product of the number of combinations with eighty successes (i.e., $\binom{100}{80}$) times the chance of any given outcome with eighty successes (i.e., $0.8^{80} \cdot 0.2^{20}$),

$$\begin{aligned} \Pr[\text{sum}(Y) = 80] &= \tbinom{100}{80} \cdot 0.8^{80} \cdot 0.2^{20} \\ &\approx 0.10 \end{aligned}$$

So while there is almost no chance of seeing the most likely outcome with all successes, there's nearly a ten percent chance of seeing an outcome with eighty successes.[258]

The probability of y successes in N independent trials with a θ chance of success is given by $\text{binomial}(y \mid N, \theta)$. Working this out for $N = 100$ and $\theta = 0.8$, we have,

$$\underbrace{\text{binomial}(80 \mid 100, 0.8)}_{\text{probability 80 successes total}} = \underbrace{\binom{100}{80}}_{\text{order 80 successes}} \cdot \underbrace{0.8^{80}}_{\text{80 specific successes}} \cdot \underbrace{0.2^{20}}_{\text{20 specific failures}}.$$

[258] This highlights a semantic point. Even though eighty successes represents the expectation of $\text{sum}(Y)$, there's still only a ten percent chance of observing exactly eighty successes. So we don't necessarily expect to see the expected value. This is even clearer with continuous distributions, where the probability of seeing the expected value is zero.

To visualize how the number of sequences and probability of each of the sequences interacts, we can plot everything. The plots on the left are on the linear scale and those on the right on the log scale.

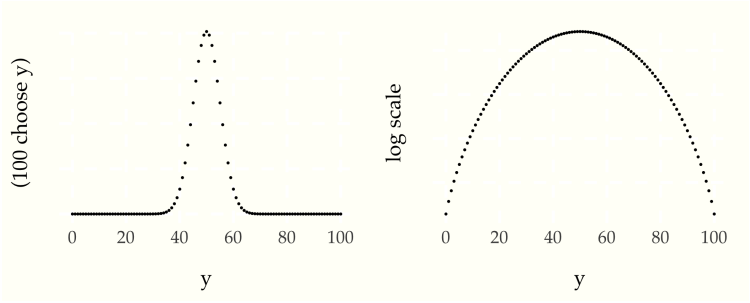


Figure 71: Total number of binary sequences of length one hundred in which exactly eighty values are one, i.e., $\binom{100}{y} = \frac{100!}{80! \cdot 20!}$.

Next, we make parallel plots of the probability of single specific sequence with y successes.[259]

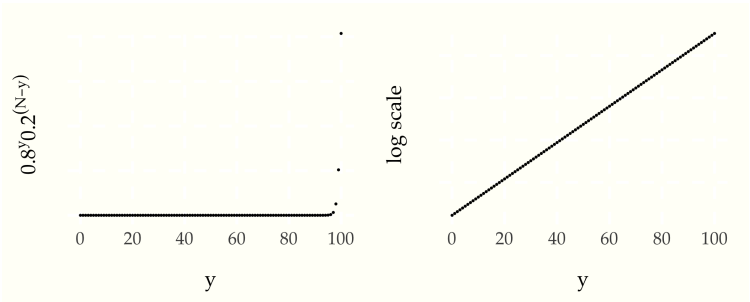


Figure 72: Probability of a specific single sequence which has y successes. The probability of a sequence of successes isn't dependent on the order of the successes, only their total number.

[259] For example, this might be the sequence where all the successes come first,

$$(\underbrace{1, 1, \ldots 1}_{y \text{ times}}, \underbrace{0, 0, \ldots, 0}_{N - y \text{ times}}).$$

On the linear scale, we multiply the number of possible sequences with y successes by the probability of a single sequence with y successes, to get the probability of drawing any sequence with y successes. On the log scale, we add the logarithm of these quantities to get the resulting log-scale plot.

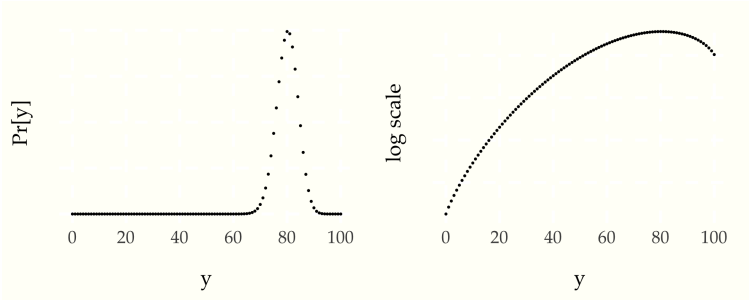


Figure 73: Probability of y successes in 100 independent trials each of which has an 80 percent chance of success. The result is the probability mass function for binomial$(y \mid 100, 0.8)$.

Discrete typical sets and log probability

Continuing our running example, we still suppose

$$Y_1, \ldots, Y_{100} \sim \text{bernoulli}(0.8),$$

so that

$$\text{sum}(Y) \sim \text{binomial}(100, 0.8).$$

Now let's consider a sequence of simulations of Y, $y^{(1)}, \ldots, y^{(M)}$. Each such $Y^{(m)}$ consists of 100 bernoulli(0.8) draws, $Y_1^{(m)}, \ldots, Y_{100}^{(m)}$. Let's look at a scatterplot of the log probabilities of the simulated Y, that is, a scatterplot of the joint log probability of all one hundred elements,

$$\begin{aligned} \log p_Y(y^{(m)}) &= \textstyle\sum_{n=1}^{100} \log p_{Y_n}(y_n^{(m)}) \\ &= \textstyle\sum_{n=1}^{100} \log \text{bernoulli}(y_n^{(m)} \mid 0.8). \end{aligned}$$

Here's the histogram.

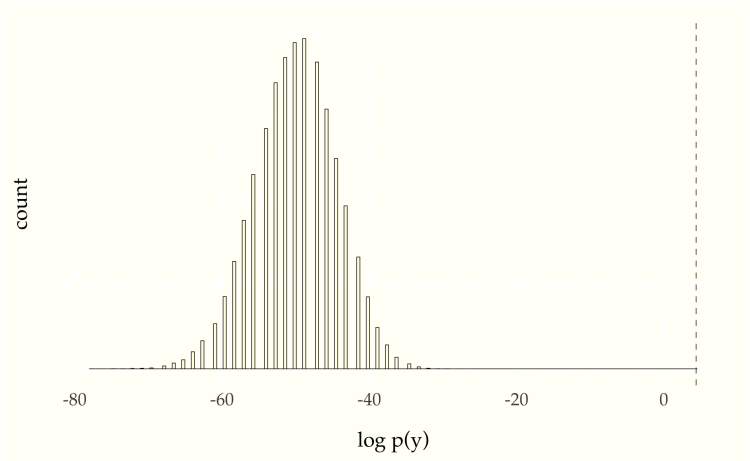


Figure 74: Histogram of $\log p_Y(y^{(m)})$ for simulations $y^{(m)}$ of one hundred binary trials with an eighty percent chance of success. The histogram is spikey because there are only a hundred trials, and thus a small number of possible outcomes. The number of observed outcomes is even smaller as they are concentrated around the expected value of 80 successes. The dashed line is at the log density of the mode (most likely value for Y), which consists of 100 successes. The observed draws have log probability between -80 and -20, whereas the mode has a log probability greater than 0.

The takeaway message here is that the log probability of the observed draws fall in a narrow interval that's bounded far away from the log probability of the most probabiltiy element (the sequence consisting of all successes).

Entropy

Information theory was developed as a means to deal with coding of signals across noisy channels, such as telegraph signals in a discrete case or or telephone signals in a continuous case. We will largely be using information theory to give us an alternative to variance to describe the variability (entropy) of a distribution and two alternatives to correlation to describe the similarity of two random variables (mutual information) and two distributions (divergence).

Entropy

The entropy of a univariate random variable Y is defined as the expectation of its negative log probability function,

$$\mathrm{H}[Y] \;=\; \mathbb{E}[-\log p_Y(Y)].$$

The logarithm can actually be computed in any base. It is traditional to work in base 2 for discrete probabilities and with natural logarithms (base e) for continuous (or mixed) quantities.[260] In base 2, the units are called *bits*, for reasons that will be clear shortly; in base e, the units are called *nats*.

[260] We use the notation $\log_2$ for base 2 logarithms and continue using simply log for natural logarithms.

Entropy in the discrete case

For a discrete distribution, p_Y is a probability mass function and the expectation expands to a sum over all possible integer values $y \in \mathbb{Z}$,

$$\begin{aligned} \mathrm{H}[Y] &= \mathbb{E}\left[\log_2 p_Y(y)\right] \\ &= \textstyle\sum_{y\in\mathbb{Z}} p_Y(y) \cdot \log_2 p_Y(y). \end{aligned}$$

In finite cases, the entropy is particularly simple to compute. For example, suppose $Y \sim \text{bernoulli}(\theta)$. Then

$$\mathrm{H}[Y] \;=\; \theta \cdot \log\theta + (1-\theta)\cdot\log(1-\theta).$$

Let's look at the case where $Y \sim \text{Bernoulli}\left(\frac{1}{2}\right)$. There is a 50% chance that $Y = 1$ and a 50% chance that $Y = 0$. The entropy is

calculated as derived above,

$$\begin{aligned} \mathrm{H}[Y] &= -\tfrac{1}{2} \cdot \log_2 \tfrac{1}{2} - \tfrac{1}{2} \cdot \log_2 \tfrac{1}{2} \\ &= -\log_2 \tfrac{1}{2} \\ &= -\log_2 2^{-1} \\ &= --\log_2 2 \\ &= \log_2 2 \\ &= 1. \end{aligned}$$

The entropy of $Y \sim \text{Bernoulli}\left(\frac{1}{2}\right)$ is one bit. This provides a natural scale for entropy—a single coin flip is a single bit of information.[261]

Now consider a categorical variable $Z \sim \text{categorical}(\theta)$, where $\theta = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)$. What is the entropy of Z?[262]

$$\begin{aligned} \mathrm{H}[Z] &= \textstyle\sum_{n=1}^{4} -\frac{1}{4} \log_2 \frac{1}{4} \\ &= -\log_2 2^{-2} \\ &= 2. \end{aligned}$$

As we might have expected, the entropy of rolling a four-sided die is $\log_2 4 = 2$ bits. That's because we can use two computer bits to encode a four-way outcome, with codes `00`, `01`, `10`, and `11`. The entropy of the outcome of rolling an eight-side die is $\log_2 8 = 3$ bits, with codes `000`, `001`, `010`, `011`, `100`, `101` `110`, `111`.

What is the entropy of two coin flips, $Y_1, Y_2 \sim \text{Bernoulli}\left(\frac{1}{2}\right)$? We just combine them into a joint variable $Z = (Y_1, Y_2)$ and proceed as usual, taking

$$\mathrm{H}[Z] = \mathbb{E}[-\log_2 p_Z(z)].$$

The expectation is computed as usual, by enumerating possible values for Z, which looks like

$$\begin{aligned} \mathrm{H}[Z] &= \textstyle\sum_{y_1=0}^{1} \sum_{y_2=0}^{1} p_{Y_1,Y_2}(y_1, y_2) \cdot \log p_{Y_1,Y_2}(y_1, y_2) \\ &= -\tfrac{1}{4} \cdot \log_2 \tfrac{1}{4} - \tfrac{1}{4} \cdot \log_2 \tfrac{1}{4} - \tfrac{1}{4} \cdot \log_2 \tfrac{1}{4} - \tfrac{1}{4} \cdot \log_2 \tfrac{1}{4} \\ &= -\log_2 4^{-1} \\ &= 2. \end{aligned}$$

So far, we've only considered balanced outcomes, as in a coin toss. Now suppose that $Y \sim \text{bernoulli}(\theta)$. The following plot shows the entropy $\mathrm{H}[Y]$ as a functiono f θ.

The maximum entropy for $Y \sim \text{bernoulli}(\theta)$ occurs at $\theta = 0.5$, when there is a 50% chance of each outcome. This represents the maximum amount of uncertainty possible. At the other extreme, $\theta = 0$ and $\theta = 1$ correspond to complete certainty in the outcome and

[261] Information theory was originally applied to coding discrete symbols for transmission. Morse Code, for example, is a way to code the Latin alphabet using binary symbols (conventionally written $\cdot$ and $-$). The genetic code is a way of coding amino acids using sequences of three base pairs, each of which can take on four possible values (conventionally written `a`, `c`, `g`, and `t`). There is no computer code which can faithfully encode the result a sequence of N coin flips that uses fewer than N bits on average.

[262] You may want to take a guess based on the notion of bits required to encode outcomes.

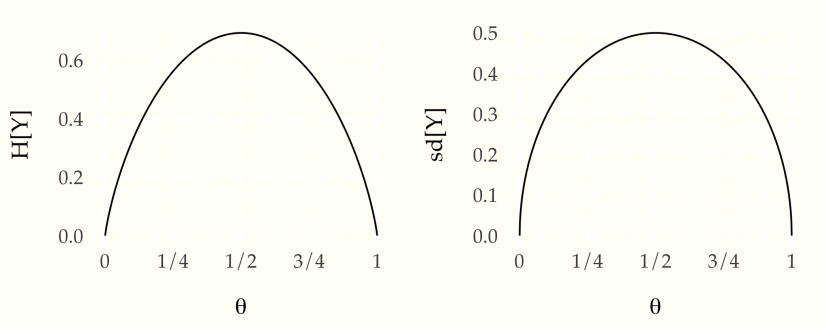


Figure 75: Left) entropy of a variable $Y \sim \text{bernoulli}(\theta)$ as a function of θ. Right) standard deviation of the same variable as a function of θ. The shapes are not identical, but share the same mode of $\theta = 0.5$ and mimima at $\theta = 0$ and $\theta = 1$.

thus the entropy is zero. Other values are in between. At $\theta = 0.9$, for example, the variable is much more likely to be one than zero.

Differential entropy

In the continuous case, p_Y is a probabiltiy density function and the expectation expands to an integral over all possible real values $y \in \mathbb{R}$,

$$\begin{aligned} \text{H}[Y] &= \mathbb{E}[-\log p_Y(y)] \\ &= \int_{\mathbb{R}} -\log p_Y(y) \cdot p_Y(y) \, \text{d}y. \end{aligned}$$

Because density values may be greater than one, differential entropy can be negative with very narrowly distributed variables, which makes it a less than desirable general measure of information.

Calculating entropy with simulation

Because entropy is defined as an expectation, it can be computed using simulation. Suppose $y^{(1)}, \ldots, y^{(M)}$ drawn according to $p_Y(y)$. The entropy of Y is then calculated by plugging the draws in for the random variable Y and averaging,

$$\begin{aligned} \text{H}[Y] &= \mathbb{E}[\log p_Y(y)] \\ &\approx \frac{1}{M} \sum_{m=1}^{M} \log p_Y\left(y^{(m)}\right). \end{aligned}$$

For example, let's calculate the entropy of a normally distributed random variable $Y \sim \text{normal}(\mu, \sigma)$ as a function of σ.[263] We just draw $y^{(m)} \sim \text{normal}(0, \sigma)$ and calculate the average $\log \text{normal}(y^{(m)} \mid 0, \sigma)$. value.

The plot shows that differential entropy increases sublinearly as the scale σ increases. The plot also shows that for values of σ, the differential entropy is negative. Discrete entropy, in contrast, is always positive.

[263] As with standard deviation, the location parameter μ does not affect the entropy of a normally distributed variable.

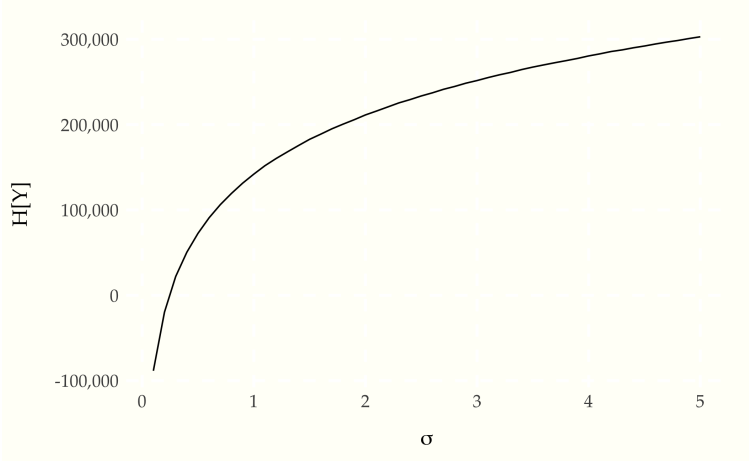


Figure 76: Differential entropy of a normally distributed variable $Y \sim \text{normal}(\mu, \sigma)$ as a function of its scale σ.

Maximum entropy distributions

Suppose we have a normally distributed variable with expectation μ and standard deviation σ,

$$U \sim \text{normal}(\mu, \sigma).$$

Now suppose we have another random variable V and only know that it also has an expectation of μ and standard deviation of σ,

$$\mathbb{E}[V] = \mu \quad \text{and} \quad \text{sd}[V] = \sigma.$$

Then we know that

$$H[U] \geq H[V],$$

with equality only if V is also normally distributed, $V \sim \text{normal}(\mu, \sigma)$. Another way of saying this is that among all possible random variables with a given expectation and standard deviation, a normally distributed one has the maximum entropy.

Conditional and joint entropy

Entropy may also be defined conditionally and jointly, just like distributions. We just take expectations of the relevant negative log probability functions.

The joint entropy of X, Y is defined as the expectation of the negative log joint probability function,

$$\text{H}[X, Y] \ = \ \mathbb{E}[\log p_{X,Y}(X, Y)].$$

For example, if U and V are both discrete, their joint entropy is defined to be

$$\text{H}[U, V] \ = \ \sum_{u \in \mathbb{Z}} \sum_{v \in \mathbb{Z}} - \log p_{U,V}(u, v) \cdot p_{U,V}(u, v).$$

Similarly, the conditional entropy is the expectation of the log conditional,

$$\mathrm{H}[Y \mid X] \;=\; \mathbb{E}[\log p_{Y|X}(Y \mid X)].$$

The expectation is read in the usual way, by averaging over all of the random variables, here X and Y. For example, if U and V are both discrete,

$$\mathrm{H}[V \mid U] \;=\; \sum_{u \in \mathbb{Z}} \sum_{v \in \mathbb{Z}} -\log p_{U,V}(u \mid v) \cdot p_{U,V}(u, v).$$

As usual, the expectation of an expression is calculated by averaging over all of the random variables in the expression weighted by their probability function. So whether we take the expectation of $-\log p_{U,V}(u, v)$ or $-\log p_{U|V}(u \mid v)$, we average all outcomes weighted by $p_{U,V}(u, v)$.

Conditional and joint entropy satisfy the usual rules of probability,[264]

$$\mathrm{H}[X, Y] \;=\; \mathrm{H}[X] + \mathrm{H}[Y \mid X]$$

Entropies add rather than multiply because they are on the log probability (or density) scale.

[264] The derivation is purely algebraic,

$$\begin{aligned}
\mathrm{H}[X] + \mathrm{H}[Y \mid X] &= \textstyle\sum_x -\log p(x) \cdot p(x) + \sum_x \sum_y -\log \\
&= \textstyle\sum_x -\log p(x) \cdot p(x) + \sum_x \sum_y -\log \\
&= \textstyle\sum_x -\log p(x) \cdot p(x) + \sum_x \left(\sum_y - \right. \\
&= \textstyle\sum_x \left(-\log p(x) + \left(\sum_y -\log p(y \right. \right. \\
&= \textstyle\sum_x \sum_y (-\log p(x) \cdot p(y \mid x) - \log \\
&= \textstyle\sum_x \sum_y -\log p(x, y) \cdot p(x, y) \\
&= \mathrm{H}[X, Y].
\end{aligned}$$

The only tricky step is pulling the summation over y out in the third to last step, which is made possible because

$$\sum_y -\log p(x) \cdot p(y \mid x) \;=\; -\log p(x).$$

Cross Entropy and Divergence

Entropy is a property of random variables that quantifies the amount of information knowing the variable's distribution provides about the variable. Cross-entropy, on the other hand, quantifies how much information knowing a different distribution provides about a variable. Divergence, also known as relative entropy, quantifies how much information we lose in going from the variable's distribution to another distribution.

Because they involve multiple distributions, cross entropy and divergence are defined over distributions rather than over random variables. We are going to take the somewhat unorthodox approach of motivating the definitions from entropy and random variables.

Cross entropy

If Y is a random variable, its entropy is defined as the expected value of its negative log density,

$$\mathrm{H}[Y] \;=\; \mathbb{E}[-\log p_Y(Y)].$$

But what if we use a density function $q(y)$ other than $p_Y(y)$? The quantity

$$\mathrm{X}[Y, q] \;=\; \mathbb{E}[-\log q(Y)]$$

is known as the *cross entropy*.[265]

Cross entropy has the same units as entropy. It reduces to entropy when $q = p_Y$,

$$\begin{aligned} \mathrm{H}[Y] &= \mathrm{X}[Y, p_Y] \\ &= \mathbb{E}[-\log p_Y(Y)]. \end{aligned}$$

A result known as *Gibbs inequality* states that cross entropy is always at least as large as entropy, with equality holding only for the true probability function. In symbols, the Gibbs inequality states that

$$\mathrm{H}[Y] \leq \mathrm{X}[Y, q],$$

with equality holding only if $q = p_Y$.

[265] Cross-entropy is not a widely used notion among information theory, which takes divergence as primitive. As such, there's not a conventional unambiguous notation for it. We use $\mathrm{X}[Y, q]$ here, which is unconventional in being defined in terms of a random variable Y and probability function q.

If we have a vector quantity $Y = Y_1, \ldots, Y_N$, we can scale the cross entropy to a *cross-entropy rate* by dividing by N. This lets us more easily compare results across data-set sizes N.

Estimating cross-entropy with simulation

If we can sample $y^{(1)}, \ldots, y^{(M)}$ according to $p_Y(y)$ and we can evaluate $\log q(y)$, then we can estimate cross-entropy as

$$\mathrm{X}[Y, q] \approx \frac{1}{M} \sum_{m=1}^{M} -\log q\left(y^{(m)}\right).$$

Evaluation via cross-entropy

Cross-entropy is an appealing evaluation tool in practice for both practical and theoretical reasons. Assume we have observed some data $y_1, \ldots, y_N$ which we are modeling with a distribution $q(y)$.[266]

If we assume the y_n are drawn from the distribution of interest at random, we can use them to estimate the cross entropy of a model $q(y)$ as

$$\mathrm{X}[Y, q] \approx \frac{1}{N} \sum_{n=1}^{N} -\log q(y_n).$$

For example, q might be the posterior predictive distribution of a parametric model estimated using observed data and prior knowledge.

We generally assume the lower the cross cross entropy, the better the model $q(y)$ approximates the true data generating process $p_Y(y)$.

[266] The modeling distribution $q(y)$ often has some parametric form involving unknown parameters, such as a linear or logistic regression. It may also involve predictors x_n.

Divergence

Having defined cross-entropy, the divergence of a random variable Y to a distribution with probability function $q(y)$ can be defined as the difference between the cross-entropy and the entropy. Because of Gibbs inequality, this value must be non-negative, and is only zero when $q(y)$ is the true generating probability function $p_Y(y)$.

Divergence is typically defined in terms of two probability functions p and q rather than between a random variable and a probability function. The *Kullback-Leibler divergence* from p to q is defined by

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}[p \,||\, q] &= \textstyle\sum_{y \in \mathbb{Z}} p(y) \cdot \log \frac{p(y)}{q(y)} \\ &= \textstyle\sum_{y \in \mathbb{Z}} p(y) \cdot (\log p(y) - \log q(y)) \\ &= \textstyle\sum_{y \in \mathbb{Z}} p(y) \cdot \log p(y) - \sum_{y \in \mathbb{Z}} p(y) \cdot \log q(y). \end{aligned}$$

If p and q are continuous, then

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}[p \,||\, q] &= \int_{\mathbb{R}} p(y) \cdot \log \tfrac{p(y)}{q(y)} \, \mathrm{d}y \\ &= \int_{\mathbb{R}} p(y) \cdot (\log p(y) - \log q(y)) \, \mathrm{d}y \\ &= \int_{\mathbb{R}} p(y) \cdot \log p(y) \, \mathrm{d}y - \int_{\mathbb{R}} p(y) \cdot \log q(y) \, \mathrm{d}y \end{aligned}$$

Suppose Y is a random variable such that $p_Y(y) = p(y)$. Then whether Y is discrete or continuous (or a mixture), we can express the KL-divergence as an expectation,

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}[p_Y \,||\, q] &= \mathbb{E}[\log p(Y) - \log q(Y)]. \\ &= \mathbb{E}[\log p(Y)] - \mathbb{E}[\log q(Y)]. \\ &= -\mathrm{H}[Y] - \mathbb{E}[\log q(Y)]. \\ &= \mathbb{E}[-\log q(Y)] - \mathrm{H}[Y]. \\ &= \mathrm{X}[Y, q] - \mathrm{H}[Y]. \end{aligned}$$

The final form is revealed as the difference between the cross entropy of Y to q and the entropy of \$Y.

Computing divergence with simulation

If we can draw a sample $y^{(1)}, \ldots, y^{(M)}$ according to $p_Y(y)$, and we can compute $\log p_Y(y)$ and $\log q(y)$, then we can compute the KL-divergence from p_Y to q via simulation as

$$\mathrm{D}_{\mathrm{KL}}[p_Y \,||\, q] \approx \frac{1}{M} \sum_{m=1}^{M} -\log p_Y(y^{(m)}) + \log q(y^{(m)}).$$

Divergence and cross-entropy with probability functions

The standard way to present divergence takes two arbitrary probability functions p and q and defines the discrete case as

$$\mathrm{D}_{\mathrm{KL}}[p \,||\, q] = \sum_{y \in \mathbb{Z}} -p(y) \cdot \log \frac{q(y)}{p(y)}.$$

and in the continuous case as

$$\mathrm{D}_{\mathrm{KL}}[p \,||\, q] = \int_{\mathbb{R}} -p(y) \cdot \log \frac{q(y)}{p(y)} \, \mathrm{d}y.$$

Asymmetry of divergence

Divergence is an asymmetric notion. $\mathrm{D}_{\mathrm{KL}}[p \,||\, q]$ is the divergence from p to q, whereas $\mathrm{D}_{\mathrm{KL}}[q \,||\, p]$ is the divergence from q to p. These are not usually equal.

The directionality of the definition can be visualized by considering a bivariate normal distribution with unit scale and 0.9 correlation,

$$p(y) \;=\; \text{multinormal}\left(y \;\middle|\; 0, \begin{bmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{bmatrix}\right),$$

along with another distribution $q(y \mid \sigma)$ that has diagonal covariance with scale σ,[267]

$$q(y \mid \sigma) \;=\; \text{multinormal}\left(y \;\middle|\; 0, \begin{bmatrix} \sigma & 0 \\ 0 & \sigma \end{bmatrix}\right).$$

[267] This second distribution can be defined as the product of two standard normals,

$$q(y \mid \sigma) \;=\; \text{normal}(y_1 \mid 0, \sigma) \cdot \text{normal}(y_2 \mid 0, \sigma).$$