MATH5835M Statistical Computing

Matthew Aldridge

2025-09-30

Table of contents

About MATH5835

Organisation of MATH5835

This module is **MATH5835M Statistical Computing**.

This module lasts for 11 weeks from 29 September to 12 December 2025. The exam will take place sometime between 12 and 23 January 2026.

The module leader, the lecturer, and the main author of these notes is Dr Matthew Aldridge. (You can call me "Matt", "Matthew", or "Dr Aldridge", pronounced "*old*-ridge".) My email address is m.aldridge@leeds.ac.uk, although I much prefer questions in person at office hours (see below) rather than by email.

The HTML webpage is the best way to view the course material. There is also a **PDF version**, although I have been much less careful about the presentation of this material, and it does not include the problem sheets.

Lectures

The main way you will learn new material for this module is by attending lectures. There are three lectures per week:

- Mondays at 1400
- Thursdays at 1200
- Fridays at 1000

all in in Roger Stevens LT 14.

I recommend taking your own notes during the lecture. I will put brief summary notes from the lectures on this website, but they will not reflect all the details I say out loud and write on the whiteboard. Lectures will go through material quite quickly and the material may be quite difficult, so it's likely you'll want to spend time reading through your notes after the lecture. Lectures should be recorded on the lecture capture system; I find it very difficult to read the whiteboard in these videos, but if you unavoidably miss a lecture, for example due to illness, you may find they are better than nothing.

In Weeks 3, 5, 7, 9 and 11, the Thursday lecture will operate as a "problems class" – see more on this below.

Attendance at lectures in compulsory. You should record your attendance using the UniLeeds app and the QR code on the wall in the 15 minutes before the lecture or the 15 minutes after the lecture (but not during the lecture).

Problem sheets and problem classes

Mathematics and statistics are "doing" subjects! To help you learn material for the module and to help you prepare for the exam, I will provide 5 unassessed problem sheets. These are for you to work through in your own time to help you learn; they are *not* formally assessed. You are welcome to discuss work on the problem sheets with colleagues and friends, although my recommendation would be to write-up your "last, best" attempt neatly by yourself.

There will be an optional opportunity to submit one or two questions from the problem sheet to me in advance of the problems class for some brief informal feedback on your work. See the problem sheets for details.

You should work through each problem sheet in preparation for the problems class in the Thursday lecture of Week 3, 5, 7, 9 and 11. In the problems class, you should be ready to explain your answers to questions you managed to solve, discuss your progress on questions you partially solved, and ask for help on questions you got stuck on.

You can also ask for extra help or feedback at office hours (see below).

Coursework

There will be one piece of assessed coursework, which will make up 20% of your module mark. You can read more about the coursework here.

The coursework will be in the form of a worksheet. The worksheet will have some questions, mostly computational but also mathematical, and you will have to write a report containing your answers and computations.

The assessed coursework will be introduced in the **computer practical** sessions in Week 9.

The deadline for the coursework will be the penultimate day of the Autumn term, **Thursday 12 December** at 1400. Feedback and marks will be returned on Monday 13 January, the first day of the Spring term.

Office hours

I will run a n optional "office hours" drop-in session each week for feedback and consultation. You can come along if you want to talk to me about anything on the module, including if you'd like more feedback on your attempts at problem sheet questions. (For extremely short queries, you can approach me before or after lectures, but my response will often be: "Come to my office hours, and we can discuss it there!")

Office hours will happen on **Thursdays from 1300 to 1400** – so directly after the Thursday lecture / problems class – in my office, which is **EC Stoner 9.10n** in "Maths Research Deck" area on the 9th floor of the EC Stoner building. (One way to the Maths Research Deck is via the doors directly opposite the main entrance to the School of Mathematics; you can also get there from Staircase 1 on the Level 10 "red route" through EC Stoner, next to the Maths Satellite.) If you cannot make this time, contact me for an alternative arrangement.

Exam

There will be one exam, which will make up 80% of your module mark.

The exam will be in the January 2026 exam period (12–23 January); the date and time will be announced in December. The exam will be in person and on campus.

The exam will last 2 hours and 30 minutes. The exam will consist of 4 questions, all compulsory. You will be allowed to use a permitted calculator in the exam.

Content of MATH5835

Necessary background

I recommend that students should have completed at least two undergraduate level courses in probability or statistics – although confidence and proficiency in basic material is more important than very deep knowledge of more complicated topics.

For Leeds undergraduates, MATH2715 Statistical Methods is an official prerequisite (please get in touch with me if you are/were a Leeds undergraduate and have not taken MATH2715), although confidence and proficiency in the more basic material of MATH1710 & MATH1712 Probability and Statistics 1 & 2 is probably more important.

Some knowledge I will assume:

- **Probability:** Basic rules of probability; random variables, both continuous and discrete; "famous" distributions (especially the normal distribution and the continuous uniform distribution); expectation, variance, covariance, correlation; law of large numbers and central limit theorem.
- **Statistics:** Estimation of parameters; bias and error; sample mean and sample variance

This module will also include an material on Markov chains. I won't assume any pre-existing knowledge of this, and I will introduce all new material we need, but students who have studied Markov chains before (for example in the Leeds module MATH2750 Introduction to Markov Processes) may find a couple of lectures here are merely a reminder of things they already know.

The lectures will include examples using the **R** program language. The coursework and problem sheets will require use of R. The exam, while just a "pencil and paper" exam, will require understanding and writing short portions of R code. We will assume basic R capability – that you can enter R commands, store R objects using the `<-` assignment, and perform basic arithmetic with numbers and vectors. Other concepts will be introduced as necessary. If you want to use R on your own device, I recommend downloading (if you have not already) the R programming language and the program RStudio. (These lecture notes were written in R using RStudio.)

Syllabus

We plan to cover the following topics in the module:

- **Monte Carlo estimation:** definition and examples; bias and error; variance reduction techniques: control variates, antithetic variables, importance sampling. [9 lectures]
- **Random number generation:** pseudo-random number generation using linear congruential generators; inverse transform method; rejection sampling [7 lectures]
- **Markov chain Monte Carlo** (MCMC)**:** [7 lectures]
  - Introduction to Markov chains in discrete and continuous space
  - Metropolis–Hastings algorithm: definition; examples; MCMC in practice; MCMC for Bayesian statistics
- **Resampling methods:** Empirical distribution; plug-in estimation; bootstrap statistics; bootstrap estimation [4 lectures]
- Frequently-asked questions [1 lecture]

Together with the 5 problems classes, this makes 33 lectures.

Book

The following book is strongly recommended for the module:

- J Voss, *An Introduction to Statistical Computing: A simulation-based approach*, Wiley Series in Computational Statistics, Wiley, 2014

The library has electronic access to this book (and two paper copies).

Dr Voss is a lecturer in the School of Mathematics and the University of Leeds, and has taught MATH5835 many times. *An Introduction to Statistical Computing* grew out of his lecture notes for this module, so the book is ideally suited for this module. My lectures will follow this book closely – specifically:

- Monte Carlo estimation: Sections 3.1–3.3
- Random number generation: Sections 1.1–1.4
- Markov chain Monte Carlo: Section 2.3 and Sections 4.1–4.3
- Bootstrap: Section 5.2

For a second look at material, for preparatory reading, for optional extended reading, or for extra exercises, this book comes with my highest recommendation!

Part I

Monte Carlo estimation

1 Introduction to Monte Carlo

Today, we'll start the first main topic of the module, which is called "Monte Carlo estimation". But first, a bit about the subject as a whole.

1.1 What is statistical computing?

"Statistical computing" – or "computational statistics" – refers to the branch of statistics that involves not attacking statistical problems merely with a pencil and paper, but rather by combining human ingenuity with the immense calculating powers of computers.

One of the big ideas here is **simulation**. Simulation is the idea that we can understand the properties of a random model not by cleverly working out the properties using theory – this is usually impossible for anything but the simplest "toy models" – but rather by running the model many times on a computer. From these many simulations, we can observe and measure things like the typical (or "expected") behaviour, the spread (or "variance") of the behaviour, and other things. This concept of simulation is at the heart of the module MATH5835M Statistical Computing.

In particular, we will look at **Monte Carlo** estimation. Monte Carlo is about estimating a parameter, expectation or probability related to a random variable by taking many samples of that random variable, then computing a relevant sample mean from those samples. We will study Monte Carlo in its standard "basic" form, then look at ways we can make Monte Carlo estimation more accurate (Lectures 1–9).

To run a simulation – for example, when performing Monte Carlo estimation – one needs random numbers with the correct distribution. **Random number generation** (Lectures 10–16) will be an important part of this module. We will look first at how to generate randomness of any sort, and then how to manipulate that randomness into the shape of the distributions we want.

Sometimes, it's not possible to generate perfectly independent samples from exactly the distribution you want. But we can use the output of a process called a "Markov chain" to get "fairly independent" samples from nearly the distribution we want. When we perform Monte Carlo estimation with the output of a Markov chain, this is called **Markov chain Monte Carlo**

(MCMC) (Lectures 17–23). MCMC has become a vital part of modern Bayesian statistical analysis.

The final section of the module is about dealing with data. Choosing a random piece of data from a given dataset is a lot like generating a random number from a given distribution, and similar Monte Carlo estimation ideas can be used to find out about that data. We think of a dataset as being a sample from a population, and sampling again from that dataset is known as **resampling** (Lecture 24–27). The most important method of finding out about a population by using resampling from a dataset is called the "bootstrap", and we will study the bootstrap in detail.

MATH5835M Statistical Computing is a *mathematics* module that will concentrate on the *mathematical* ideas that underpin statistical computing. It is not a programming module that will go deeply into the practical issues of the most efficient possible coding of the algorithms we study. But we will want to investigate the behaviour of the methods we learn about and to explore their properties, so will be computer programming to help us do that. We will be using the statistical programming language R, (although one could just as easily have used Python or other similar languages). As my PhD supervisor often told me: "You don't really understand a mathematical algorithm until you've coded it up yourself."

1.2 What is Monte Carlo estimation?

Let X be a random variable. We recall the **expectation** $\mathbb{E}X$ of X. If X is discrete with probability mass function (PMF) p, then the expectation of X is

$$\mathbb{E}X = \sum_x x\, p(x);$$

while if X is continuous with probability density function (PDF) f, then the expectation is

$$\mathbb{E}X = \int_{-\infty}^{+\infty} x\, f(x)\, \mathrm{d}x.$$

More generally, the expectation of a function ϕ of X is

$$\mathbb{E}\,\phi(X) = \begin{cases} \displaystyle\sum_x \phi(x)\, p(x) & \text{for } X \text{ discrete} \\ \displaystyle\int_{-\infty}^{+\infty} \phi(x)\, f(x)\, \mathrm{d}x & \text{for } X \text{ continuous.} \end{cases}$$

(This matches with the "plain" expectation when $\phi(x) = x$.)

But how do we actually *calculate* an expectation like one of these? If X is discrete and can only take a small, finite number of values, then we can simply add up the sum $\sum_x \phi(x)\, p(x)$. But otherwise, we just have to hope that ϕ and p or f are sufficiently "nice" that we can

manage to work out the sum/integral using a pencil and paper (and our brain). But while this is often possible in the sort of "toy example" one comes across in maths or statistics lectures, this is very rare in "real life" problems.

Monte Carlo estimation is the idea that we can get an approximate answer for $\mathbb{E}X$ or $\mathbb{E}\,\phi(X)$ if we have access to lots of samples from X. If we have access to $X_1, X_2 \dots, X_n$, independent and identically distributed (IID) samples with the same distribution as X, then we already know that the mean

$$\overline{X} = \frac{1}{n}(X_1 + X_2 + \cdots + X_n) = \frac{1}{n}\sum_{i=1}^{n} X_i$$

can be used to estimate the expectation $\mathbb{E}X$. We know that $\overline{X}$ is usually close to the expectation $\mathbb{E}X$, at least if if the number of samples n is large; this is justified by the "law of large numbers", which says that $\overline{X} \to \mathbb{E}X$ as $n \to \infty$.

Similarly, we can use

$$\frac{1}{n}(\phi(X_1) + \phi(X_2) + \cdots + \phi(X_n)) = \frac{1}{n}\sum_{i=1}^{n} \phi(X_i)$$

to estimate $\mathbb{E}\,\phi(X)$. The law of large numbers again says that this estimate tends to the correct value $\mathbb{E}\,\phi(X)$ as $n \to \infty$.

In this module we will write that $X_1, X_2, \dots, X_n$ is a "**random sample** from X" to mean that $X_1, X_2, \dots, X_n$ are IID with the same distribution as X.

Definition 1.1. Let X be a random variable, ϕ a function, and write $\theta = \mathbb{E}\,\phi(X)$. Then the **Monte Carlo estimator** $\hat{\theta}_n^{\mathrm{MC}}$ of θ is

$$\hat{\theta}_n^{\mathrm{MC}} = \frac{1}{n}\sum_{i=1}^{n} \phi(X_i),$$

where $X_1, X_2, \dots, X_n$ are a random sample from X.

While general ideas for estimating using simulation go back a long time, the modern theory of Monte Carlo estimation was developed by the physicists Stanislaw Ulam and John von Neumann. Ulam (who was Polish) and von Neumann (who was Hungarian) moved to the US in the early 1940s to work on the Manhattan project to build the atomic bomb (as made famous by the film *Oppenheimer*). Later in the 1940s, they worked together in the Los Alamos National Laboratory continuing their research on nuclear physics generally and nuclear weapons more specifically, where they used simulations on early computers to help them numerically solve difficult mathematical and physical problems.

The name "Monte Carlo" was chosen because the use of randomness to solve such problems reminded them of gamblers in the casinos of Monte Carlo, Monaco. Ulam and von Neumann also worked closely with another colleague Nicholas Metropolis, whose work we will study later in this module.

1.3 Examples

Let's see some simple examples of Monte Carlo estimation using R.

Example 1.1. Let's suppose we've forgotten the expectation of the exponential distribution $X \sim \text{Exp}(2)$ with rate 2. In this simple case, we could work out the answer using the PDF $f(x) = 2\text{e}^{-2x}$ as

$$\mathbb{E}X = \int_0^\infty x\, 2\text{e}^{-2x}\, \text{d}x$$

and, without too much difficulty, get the answer $\frac{1}{2}$. But instead, let's do this the Monte Carlo way.

In R, we can use the `rexp()` function to get IID samples from the exponential distribution: the full syntax is `rexp(n, rate)`, which gives `n` samples from an exponential distribution with rate `rate`. The following code takes the mean of $n = 100$ samples from the exponential distribution.

```
n <- 100
samples <- rexp(n, 2)
MCest <- (1 / n) * sum(samples)
MCest
```

```
[1] 0.4594331
```

So our Monte Carlo estimate is 0.4594, to 4 decimal places.

That's fairly close to the correct answer of $\frac{1}{2}$. But we should (hopefully) be able to get a more accurate estimation if we use more samples. We could also simplify the third line of our code by using the `mean()` function.

```
n <- 1e6
samples <- rexp(n, 2)
MCest <- mean(samples)
MCest
```

```
[1] 0.4996347
```

In the second line, `1e6` is R code for the scientific notation 1×10^6, or a million. I just picked this as "a big number, but where my code still only took a few seconds to run."

Our new Monte Carlo estimate is 0.4996, which is (probably) much closer to the true value of $\frac{1}{2}$.

By the way: all R code "chunks" displayed in the notes should work perfectly if you copy-and-paste them into RStudio. (Indeed, when I compile these lecture notes in RStudio, all the R code gets run on my computer – so I'm certain in must work correctly!) If you hover over a code chunk, a little "clipboard" icon should appear in the top-right, and clicking on that will copy it so you can paste it into RStudio. I strongly encourage playing about with the code as a good way to learn this material and explore further!

Example 1.2. Let's try another example. Let $X \sim \mathrm{N}(1, 2^2)$ be a normal distribution with mean 1 and standard deviation 2. Suppose we want to find out $\mathbb{E}(\sin X)$ (for some reason). While it *might* be possible to somehow calculate the integral

$$\mathbb{E}(\sin X) = \int_{-\infty}^{+\infty} (\sin x) \, \frac{1}{\sqrt{2\pi \times 2^2}} \exp\left(-\frac{(x-1)^2}{2 \times 2^2}\right) \, \mathrm{d}x,$$

that looks extremely difficult to me.

Instead, a Monte Carlo estimation of $\mathbb{E}(\sin X)$ is very straightforward: we just take the mean of the sine of a bunch of normally distributed random numbers. That is we get a random samples $X_1, X_2, \ldots, X_n$ from X; then take the mean of the values

$$\sin(X_1), \sin(X_2), \ldots, \sin(X_n).$$

(We must remember, though, when using the `rnorm()` function to generate normally distributed random variates, that the third argument is the *standard deviation*, here 2, *not* the variance, here $2^2 = 4$.)

```
n <- 1e6
samples <- rnorm(n, 1, 2)
MCest <- mean(sin(samples))
MCest
```

```
[1] 0.1131249
```

Our Monte Carlo estimate is 0.11312.

Next time: *We look at more examples of things we can estimate using the Monte Carlo method.*

Summary:

- Statistical computing is about solving statistical problems by combining human ingenuity with computing power.

- The Monte Carlo estimate of $\mathbb{E}\,\phi(X)$ is

$$\hat{\theta}_n^{\mathrm{MC}} = \frac{1}{n}\sum_{i=1}^{n}\phi(X_i),$$

  where $X_1, \ldots, X_n$ are IID random samples from X.

- Monte Carlo estimation typically gets more accurate as the number of samples n gets bigger.

Read more: Voss, *An Introduction to Statistical Computing*, Section 3.1 and Subsection 3.2.1.

2 Uses of Monte Carlo

Quick recap: Last time we defined the Monte Carlo estimator for an expectation of a function of a random variable $\theta = \mathbb{E}\,\phi(X)$ to be

$$\hat{\theta}_n^{\mathrm{MC}} = \frac{1}{n}(\phi(X_1) + \phi(X_2) + \cdots + \phi(X_n)) = \frac{1}{n}\sum_{i=1}^{n}\phi(X_i),$$

where $X_1, X_2, \ldots, X_n$ are independent random samples from X.

Today we look at two other things we can estimate using Monte Carlo simulation: probabilities, and integrals.

2.1 Monte Carlo for probabilities

What if we want to find a *probability*, rather than an expectation? What if we want $\mathbb{P}(X = x)$ for some x, or $\mathbb{P}(X \geq a)$ for some a, or, more generally, $\mathbb{P}(X \in A)$ for some set A?

The key thing that will help us here is the *indicator function*. The indicator function simply tells us whether an outcome x is in a set A or not.

Definition 2.1. Let A be a set. Then the **indicator function** $\mathbb{I}_A$ is defined by

$$\mathbb{I}_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A. \end{cases}$$

The set A could just be a single element $A = \{y\}$. In that case $\mathbb{I}_A(x)$ is 1 if $x = y$ and 0 if $x \neq y$. Or A could be a semi-infinite interval, like $A = [a, \infty)$. In that case $\mathbb{I}_A(x)$ is 1 if $x \geq a$ and 0 if $x < a$.

Why is this helpful? Well $\mathbb{I}_A$ is a function, so let's think about what the expectation $\mathbb{E}\,\mathbb{I}_A(X)$ would be for some random variable X. Since $\mathbb{I}_A$ can only take two values, 0 and 1, we have

$$\begin{aligned}\mathbb{E}\,\mathbb{I}_A(X) &= \sum_{y\in\{0,1\}} y\,\mathbb{P}(\mathbb{I}_A(X)=y)\\ &= 0\times\mathbb{P}(\mathbb{I}_A(X)=0)+1\times\mathbb{P}(\mathbb{I}_A(X)=1)\\ &= 0\times\mathbb{P}(X\notin A)+1\times\mathbb{P}(X\in A)\\ &= \mathbb{P}(X\in A).\end{aligned}$$

P(X A) . \end{align*} In line three, we used that $\mathbb{I}_A(X)=0$ if and only if $X\notin A$, and that $\mathbb{I}_A(X)=1$ if and only if $X\in A$.

So the expectation of an indicator function a set is the probability that X is in that set. This idea connects "expectations of functions" back to probabilities: if we want to find $\mathbb{P}(X\in A)$ we can find the expectation of $\mathbb{I}_A(X)$.

With this idea in hand, how do we estimate $\theta=\mathbb{P}(X\in A)$ using the Monte Carlo method? We write $\theta=\mathbb{E}\,\mathbb{I}_A(X)$. Then our Monte Carlo estimator is

$$\widehat{\theta}_n^{\mathrm{MC}}=\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}_A(X_i).$$

We remember that $\mathbb{I}_A(X_i)$ is 1 if $X_i\in A$ and 0 otherwise. So if we add up n of these, we count an extra +1 each time we have an $X_i\in A$. So $\sum_{i=1}^n\mathbb{I}_A(X_i)$ counts the total number of the X_i that are in A. So the Monte Carlo estimator can be written as

$$\widehat{\theta}_n^{\mathrm{MC}}=\frac{\#\text{ of }X_i\text{ that are in }A}{n}.$$

(I'm using # as shorthand for "the number of".)

Although we've had to do a bit of work to get here, this a totally logical outcome! The right-hand side here is the proportion of the samples for which $X_i\in A$. And if we want to estimate the probability something happens, looking at the proportion of times it happens in a random sample is very much the "intuitive" estimate to take. And that intuitive estimate is indeed the Monte Carlo estimate!

Example 2.1. *Let $Z\sim\mathrm{N}(0,1)$ be a standard normal distribution. Estimate $\mathbb{P}(Z>2)$.*

This is a question that it is impossible to answer exactly using a pencil and paper: there's no closed form for

$$\mathbb{P}(Z>2)=\int_2^\infty\frac{1}{\sqrt{2\pi}}\,\mathrm{e}^{-z^2/2}\,\mathrm{d}z,$$

so we'll have to use an estimation method.

The Monte Carlo estimate means taking a random sample $Z_1,Z_2,\ldots,Z_n$ of standard normals, and calculating what proportion of them are greater than 2. In R, we can do this as follows.

```
n <- 1e6
samples <- rnorm(n)
MCest <- mean(samples > 2)
MCest
```

```
[1] 0.022716
```

In the second line, we could have written `rnorm(n, 0, 1)`. But, if you don't give the parameters `mean` and `sd` to the function `rnorm()`, R just assumes you want the standard normal with `mean = 0` and `sd = 1`.

We can check our answer: R's inbuilt `pnorm()` function estimates probabilities for the normal distribution (using a method that, in this specific case, is much quicker and more accurate than Monte Carlo estimation). The true answer is very close to

```
pnorm(2, lower.tail = FALSE)
```

```
[1] 0.02275013
```

so our estimate was pretty good.

We should explain the third line in the code we used for the Monte Carlo estimation `mean(samples > 2)`. In R, some statements can be answered "true" or "false": these are often statements involving equality `==` (that's a *double* equals sign) or inequalities like `<`, `<=`, `>=`, `>`, for example. So `5 > 2` is `TRUE` but `3 == 7` is `FALSE`. These can be applied "component by component" to vectors. So, for example, testing which numbers from 1 to 10 are greater than or equal to 7, we get

```
1:10 >= 7
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

six `FALSE`s (for 1 to 6) followed by four `TRUE`s (for 7 to 10).

We can also use `&` ("and") and `|` ("or") in true/false statements like these.

But R also knows to treat `TRUE` like the number 1 and `FALSE` like the number 0. (This is just like the concept of the indicator function we've been discussing.) So if we add up some `TRUE`s and `FALSE`s, R simply counts how many `TRUE`s there are

```
sum(1:10 >= 7)
```

```
[1] 4
```

So in our Monte Carlo estimation code, `samples > 2` was a vector of `TRUE`s and `FALSE`s, depending on whether each sample was greater than 2 or not, then `mean(samples > 2)` took the *proportion* of the samples that were greater than 2.

2.2 Monte Carlo for integrals

There's another thing – a non-statistics thing – that Monte Carlo estimation is useful for. We can use Monte Carlo estimation to approximate integrals that are too hard to do by hand.

This might seem surprising. Estimating the expectation of (a function of) a random variable seems a naturally statistical thing to do. But an integral is just a straight maths problem – there's not any randomness at all. But actually, integrals and expectations are very similar things.

Let's think of an integral: say,

$$\int_a^b h(x)\,\mathrm{d}x,$$

the integral of some function h (the "integrand") between the limits a and b. Now let's compare that to the integral $\mathbb{E}\,\phi(X)$ of a continuous random variable that we can estimate using Monte Carlo estimation,

$$\mathbb{E}\,\phi(X) = \int_{-\infty}^{\infty} \phi(x)\,f(x)\,\mathrm{d}x.$$

Matching things up, we can see that we if we were to a function ϕ and a PDF f such that

$$\phi(x)\,f(x) = \begin{cases} 0 & x < a \\ h(x) & a \leq x \leq b \\ 0 & x > b, \end{cases} \tag{2.1}$$

then we would have

$$\mathbb{E}\,\phi(X) = \int_{-\infty}^{\infty} \phi(x)\,f(x)\,\mathrm{d}x = \int_a^b h(x)\,\mathrm{d}x,$$

so the value of the expectation would be precisely the value of the integral we're after. Then we could use Monte Carlo to estimate that expectation/integral.

There are lots of choices of ϕ and f that would satisfy this the condition in Equation 2.1. But a "common-sense" choice that often works is to pick f to be the PDF of X, a continuous

uniform distribution on the interval $[a, b]$. (This certainly works when a and b are finite, anyway.) Recall that the continuous uniform distribution means that X has PDF

$$f(x) = \begin{cases} 0 & x < a \\ \dfrac{1}{b-a} & a \leq x \leq b \\ 0 & x > b. \end{cases}$$

Comparing this equation with Equation 2.1, we then have to choose

$$\phi(x) = \frac{h(x)}{f(x)} = (b-a)h(x).$$

Putting this all together, we have

$$\mathbb{E}\,\phi(X) = \int_{-\infty}^{+\infty} \phi(x)\, f(x)\,\mathrm{d}x = \int_a^b (b-a)h(x)\,\frac{1}{b-a}\,\mathrm{d}x = \int_a^b h(x)\,\mathrm{d}x,$$

as required. This can then be estimated using the Monte Carlo method.

Definition 2.2. Consider an integral $\theta = \int_a^b h(x)\,\mathrm{d}x$. Let f be the probability density function of a random variable X and let ϕ be function such that Equation 2.1 holds. Then the **Monte Carlo estimator** $\widehat{\theta}_n^{\mathrm{MC}}$ of the integral θ is

$$\widehat{\theta}_n^{\mathrm{MC}} = \frac{1}{n}\sum_{i=1}^{n} \phi(X_i),$$

where $X_1, X_2, \ldots, X_n$ are a random sample from X.

Example 2.2. Suppose we want to approximate the integral

$$\int_0^2 x^{1.6}(2-x)^{0.7}\,\mathrm{d}x.$$

Since this is an integral on the finite interval $[0, 2]$, it would seem to make sense to pick X to be uniform on $[0, 2]$. This means we should take

$$\phi(x) = \frac{h(x)}{f(x)} = (2-0)h(x) = 2\,x^{1.6}(2-x)^{0.7}.$$

We can then approximate this integral in R using the Monte Carlo estimator

$$\int_0^2 x^{1.6}(2-x)^{0.7}\,\mathrm{d}x = \mathbb{E}\,\phi(X) \approx \frac{1}{n}\sum_{i=1}^{n} 2\,X_i^{1.6}(2-X_i)^{0.7}.$$

```r
n <- 1e6
integrand <- function(x) x^1.6 * (2 - x)^0.7
a <- 0
b <- 2
samples <- runif(n, a, b)
mean((b - a) * integrand(samples))
```

```
[1] 1.444437
```

You have perhaps noticed that, here and elsewhere, I tend to split my R code up into lots of small bits, perhaps slightly unnecessarily. After all, those 6 lines of code could simply have been written as just 2 lines

```r
samples <- runif(1e6, 0, 2)
mean(2 * samples^1.6 * (2 - samples)^0.7)
```

There's nothing *wrong* with that. However, I find that code is easier to read if divided into small pieces. It also makes it easier to tinker with, if I want to use it to solve some similar but slightly different problem.

Example 2.3. Suppose we want to approximate the integral

$$\int_{-\infty}^{+\infty} \mathrm{e}^{-0.1|x|} \cos x \, \mathrm{d}x.$$

This one is an integral on the whole real line, so we can't take a uniform distribution. Maybe we should take $f(x)$ to be the PDF of a normal distribution, and then put

$$\phi(x) = \frac{h(x)}{f(x)} = \frac{\mathrm{e}^{-0.1|x|} \cos x}{f(x)}.$$

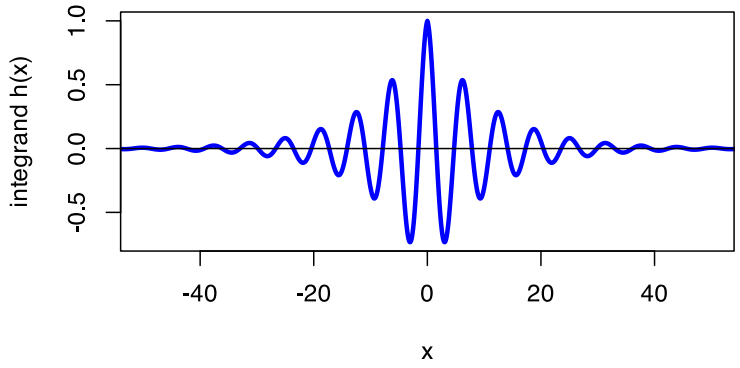
But which normal distribution should we take? Well, we're *allowed* to take any one – we will still get an accurate estimate in the limit as $n \to \infty$. But we'd like an estimator that gives accurate results at moderate-sized n, and picking a "good" distribution for X will help that.

We'll probably get the best results if we pick a distribution that is likely to mostly take values where $h(x)$ is big – or, rather, where the absolute value $|h(x)|$ is big, to be precise. That is because we don't want to "waste" too many samples where $h(x)$ is very small, because they don't contribute much to the integral. But we don't want to "miss" – or only sample very rarely – places where $h(x)$ is big, which contribute a lot to the integral.

Let's have a look at the graph of $h(x) = \mathrm{e}^{-0.1|x|} \cos x$.

```r
integrand <- function(x) exp(-0.1 * abs(x)) * cos(x)

curve(
  integrand, n = 1001, from = -55, to = 55,
  col = "blue", lwd = 3,
  xlab = "x", ylab = "integrand h(x)", xlim = c(-50,50)
)
abline(h = 0)
```



This suggests to me that a mean of 0 and a standard deviation of 20 might work quite well, since this will tend to take values in $[-40, 40]$ or so.

We will use R's function `dnorm()` for the probability density function of the normal distribution (which saves us from having to remember what that is).

```r
n <- 1e6
integrand <- function(x) exp(-0.1 * abs(x)) * cos(x)
pdf       <- function(x) dnorm(x, 0, 20)
phi       <- function(x) integrand(x) / pdf(x)

samples <- rnorm(n, 0, 20)
mean(phi(samples))
```

```
[1] 0.2189452
```

Next time: *We will analyse the accuracy of these Monte Carlo estimates.*

Summary:

- The indicator $\mathbb{I}_A(x)$ function of a set A is 1 if $x \in A$ or 0 if $x \notin A$.
- We can estimate a probability $\mathbb{P}(X \in A)$ by using the Monte Carlo estimate for $\mathbb{E}\,\mathbb{I}_A(X)$.
- We can estimate an integral $\int h(x)\,\mathrm{d}x$ by using a Monte Carlo estimate with $\phi(x)\,f(x) = h(x)$.

Read more: Voss, *An Introduction to Statistical Computing*, Section 3.1 and Subsection 3.2.1.

3 Monte Carlo error I: theory

3.1 Estimation error

Today we are going to analysing the accuracy of Monte Carlo estimation. But before talking about Monte Carlo estimation specifically, let's first remind ourselves of some concepts about error in statistical estimation more generally. We will use the following definitions.

Definition 3.1. Let $\hat{\theta}$ be an estimator of a parameter θ. Then we have the following definitions of the estimator $\hat{\theta}$:

- The **bias** is $\text{bias}\,(\hat{\theta}) = \mathbb{E}(\hat{\theta} - \theta) = \mathbb{E}\hat{\theta} - \theta$.
- The **mean-square error** is $\text{MSE}\,(\hat{\theta}) = \mathbb{E}(\hat{\theta} - \theta)^2$.
- The **root-mean-square error** is the square-root of the mean-square error,

$$\text{RMSE}\,(\hat{\theta}) = \sqrt{\text{MSE}(\hat{\theta})} = \sqrt{\mathbb{E}(\hat{\theta} - \theta)^2}.$$

Usually, the main goal of estimation is to get the mean-square error of an estimate as small as possible. This is because the MSE measures by what distance we are missing on average. It can be easier to interpret what the root-mean-square error means, as the RMSE has the same units as the parameter being measured: if θ and $\hat{\theta}$ are in metres, say, then the MSE is in metres-squared, whereas the RMSE error is in metres again. If you minimise the MSE you also minimise the RMSE and vice versa.

It's nice to have an "unbiased" estimator – that is, one with bias 0. This is because bias measures any systematic error in a particular direction. However, unbiasedness by itself is not enough for an estimate to be good – we need low variance too. (Remember the old joke about the statistician who misses his first shot ten yards to the left, misses his second shot ten yards to the right, then claims to have "hit the target on average.")

(Remember also that "bias" is simply the word statisticians use for $\mathbb{E}(\hat{\theta} - \theta)$; we don't mean "bias" in the derogatory way it is sometimes used in political arguments, for example.)

You probably also remember the relationship between the mean-square error, the bias, and the variance:

Theorem 3.1. $\mathrm{MSE}\left(\hat{\theta}\right) = \mathrm{bias}\left(\hat{\theta}\right)^2 + \mathrm{Var}\left(\hat{\theta}\right)$.

Proof. The MSE is

$$\mathrm{MSE}\left(\hat{\theta}\right) = \mathbb{E}\left(\hat{\theta} - \theta\right)^2 = \mathbb{E}\left(\hat{\theta}^2 - 2\theta\hat{\theta} + \theta\right)^2 \tag{3.1}$$

$$= \mathbb{E}\,\hat{\theta}^2 - 2\theta\,\mathbb{E}\,\hat{\theta} + \theta^2, \tag{3.2}$$

where we have expanded the brackets and bought the expectation inside (remembering that θ is a constant). Since the variance can be written as $\mathrm{Var}(\hat{\theta}) = \mathbb{E}\,\hat{\theta}^2 - (\mathbb{E}\,\hat{\theta})^2$, we can use a cunning trick of both subtracting and adding $(\mathbb{E}\,\hat{\theta})^2$. This gives

$$\mathrm{MSE}\left(\hat{\theta}\right) = \mathbb{E}\,\hat{\theta}^2 - \left(\mathbb{E}\,\hat{\theta}\right)^2 + \left(\mathbb{E}\,\hat{\theta}\right)^2 - 2\theta\,\mathbb{E}\,\hat{\theta} + \theta^2 \tag{3.3}$$

$$= \mathrm{Var}\left(\hat{\theta}\right) + \left((\mathbb{E}\,\hat{\theta})^2 - 2\theta\,\mathbb{E}\,\hat{\theta} + \theta^2\right) \tag{3.4}$$

$$= \mathrm{Var}\left(\hat{\theta}\right) + \left(\mathbb{E}\,\hat{\theta} - \theta\right)^2 \tag{3.5}$$

$$= \mathrm{Var}\left(\hat{\theta}\right) + \mathrm{bias}(\hat{\theta})^2. \tag{3.6}$$

This proves the result. □

Since the bias contributes to the mean-square error, that's another reason to like estimator with low – or preferably zero – bias. But again, unbiasedness isn't enough by itself; we want low variance too. (There are some situations where there's a "bias–variance tradeoff", where allowing some bias reduces the variance and so can reduce the MSE. It turns out that Monte Carlo is not one of these cases, however.)

3.2 Error of Monte Carlo estimator: theory

In this lecture, we're going to be looking more carefully at the size of the errors made by the Monte Carlo estimator

$$\hat{\theta}_n^{\mathrm{MC}} = \frac{1}{n}(\phi(X_1) + \phi(X_2) + \cdots + \phi(X_n)) = \frac{1}{n}\sum_{i=1}^{n}\phi(X_i).$$

Our main result is the following.

Theorem 3.2. *Let X be a random variable, ϕ a function, and $\theta = \mathbb{E}\,\phi(X)$. Let*

$$\hat{\theta}_n^{\mathrm{MC}} = \frac{1}{n}\sum_{i=1}^{n}\phi(X_i)$$

be the Monte Carlo estimator of θ. Then:

1. *$\hat{\theta}_n^{\mathrm{MC}}$ is unbiased, in that $\mathrm{bias}\,(\hat{\theta}_n^{\mathrm{MC}}) = 0$.*
2. *The variance of of $\hat{\theta}_n^{\mathrm{MC}}$ is $\mathrm{Var}\,(\hat{\theta}_n^{\mathrm{MC}}) = \frac{1}{n}\mathrm{Var}\,(\phi(X))$.*
3. *The mean-square error of $\hat{\theta}_n^{\mathrm{MC}}$ is $\mathrm{MSE}\,(\hat{\theta}_n^{\mathrm{MC}}) = \frac{1}{n}\mathrm{Var}\,(\phi(X))$.*
4. *The root-mean-square error of $\hat{\theta}_n^{\mathrm{MC}}$ is*

$$\mathrm{RMSE}\,(\hat{\theta}_n^{\mathrm{MC}}) = \sqrt{\frac{1}{n}\mathrm{Var}\,(\phi(X))} = \frac{1}{\sqrt{n}}\,\mathrm{sd}\,(\phi(X)).$$

Before we get to the proof, let's recap some relevant probability.

Let $Y_1, Y_2, \ldots$ be IID random variables with common expectation $\mathbb{E}Y_1 = \mu$ and common variance $\mathrm{Var}(Y_1) = \sigma^2$. Consider the mean of the first n random variables,

$$\overline{Y}_n = \frac{1}{n}\sum_{i=1}^{n} Y_i.$$

Then the expectation of $\overline{Y}_n$ is

$$\mathbb{E}\overline{Y}_n = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} Y_i\right) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}Y_i = \frac{1}{n}\,n\,\mu = \mu.$$

The variance of $\overline{Y}_n$ is

$$\mathrm{Var}\,(\overline{Y}_n) = \mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n} Y_i\right) = \left(\frac{1}{n}\right)^2\sum_{i=1}^{n}\mathrm{Var}(Y_i) = \frac{1}{n^2}\,n\,\sigma^2 = \frac{\sigma^2}{n},$$

where, for this one, we used the independence of the random variables.

Proof. Apply the probability facts from above with $Y = \phi(X)$. This gives:

1. $\mathbb{E}\hat{\theta}_n^{\mathrm{MC}} = \mathbb{E}\overline{Y}_n = \mathbb{E}Y = \mathbb{E}\,\phi(X)$, so $\mathrm{bias}(\hat{\theta}_n^{\mathrm{MC}}) = \mathbb{E}\,\phi(X) - \mathbb{E}\,\phi(X) = 0$.
2. $\mathrm{Var}\,(\hat{\theta}_n^{\mathrm{MC}}) = \mathrm{Var}\,(\overline{Y}_n) = \frac{1}{n}\mathrm{Var}(Y) = \frac{1}{n}\mathrm{Var}\,(\phi(X))$.

3. Using Theorem 3.1,

$$\mathrm{MSE}(\widehat{\theta}_n^{\mathrm{MC}}) = \mathrm{bias}(\widehat{\theta}_n^{\mathrm{MC}})^2 + \mathrm{Var}(\widehat{\theta}_n^{\mathrm{MC}}) = 0^2 + \frac{1}{n}\,\mathrm{Var}\left(\phi(X)\right) = \frac{1}{n}\,\mathrm{Var}\left(\phi(X)\right).$$

4. Take the square root of part 3.

□

Let's think about MSE $\frac{1}{n}\,\mathrm{Var}(\phi(X))$. The variance terms is some fixed fact about the random variable X and the function ϕ. So as n gets bigger, $\frac{1}{n}$ gets smaller, so the MSE gets smaller, and the estimator gets more accurate. This goes back to what we said when we introduced the Monte Carlo estimator: we get a more accurate estimate by increasing n. More specifically, the MSE scales like $1/n$, or – perhaps a more useful result – the RMSE scales like $1/\sqrt{n}$. We'll come back to this in the next lecture.

3.3 Error of Monte Carlo estimator: practice

So when we form a Monte Carlo estimate $\widehat{\theta}_n^{\mathrm{MC}}$, we now know it will be unbiased. We'd also like to know it's mean-square and/or root-mean-square error too.

There's a problem here, though. The reason we are doing Monte Carlo estimation in the first place is that we *couldn't* calculate $\mathbb{E}\,\phi(X)$. So it seems very unlikely we'll be able to calculate the variance $\mathrm{Var}(\phi(X))$ either. So how will be able to assess the mean-square (or root-mean-square) error of our Monte Carlo estimator?

Well, we can't know it exactly. But we *can* estimate the variance from the samples we are already using: by taking the sample variance of the samples $\phi(x_i)$. That is, we can estimate the variance of the Monte Carlo estimator by the sample variance

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(\phi(X_i) - \widehat{\theta}_n^{\mathrm{MC}}\right)^2.$$

Then we can similarly estimate the mean-square and root-mean-square errors by

$$\mathrm{MSE} \approx \frac{1}{n}S^2 \qquad \text{and} \qquad \mathrm{RMSE} \approx \sqrt{\frac{1}{n}S^2} = \frac{1}{\sqrt{n}}\,S$$

respectively.

Example 3.1. Let's go back to the very first example in the module, Example 1.1, where we were trying to find the expectation of an Exp(2) random variable. We used this R code:

```
n <- 1e6
samples <- rexp(n, 2)
MCest <- mean(samples)
MCest
```

```
[1] 0.5006352
```

(Because Monte Carlo estimation is random, this won't be the *exact* same estimate we had before, of course.)

So if we want to investigate the error, we can use the sample variance of these samples. We will use the sample variance function `var()` to calculate the sample variance. In this simple case, the function is $\phi(x) = x$, so we need only use the variance of the samples themselves.

```
var_est <- var(samples)
MSEest  <- var_est / n
RMSEest <- sqrt(MSEest)
c(var_est, MSEest, RMSEest)
```

```
[1] 2.503570e-01 2.503570e-07 5.003569e-04
```

The first number is `var_est` $= 0.2504$, the sample variance of our $\phi(x_i)$s:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(\phi(x_i) - \hat{\theta}_n^{\mathrm{MC}}\right)^2.$$

This should be a good estimate of the true variance $\mathrm{Var}(\phi(X))$. (In fact, in this simple case, we know that $\mathrm{Var}(X) = \frac{1}{2^2} = 0.25$, so we know that the estimate is good.) In calculating this in the code, we used R's `var()` function, which calculates the sample variance of some values.

The second number is `MSEest` $= 2.504 \times 10^{-7}$, our estimate of the mean-square error. Since $\mathrm{MSE}(\hat{\theta}_n^{\mathrm{MC}}) = \frac{1}{n}\mathrm{Var}(\phi(X))$, then $\frac{1}{n}S^2$ should be a good estimate of the MSE.

The third number is `RMSEest` $= 5 \times 10^{-4}$ our estimate of the root-mean square error, which is simply the square-root of our estimate of the mean-square error.

Example 3.2. In Example 2.1, we were estimating $\mathbb{P}(Z > 2)$, where Z is a standard normal.

Our code was

```
n <- 1e6
samples <- rnorm(n)
MCest <- mean(samples > 2)
MCest
```

```
[1] 0.022458
```

So our root-mean-square error can be approximated as

```
MSEest <- var(samples > 2) / n
sqrt(MSEest)
```

```
[1] 0.0001481677
```

since `samples > 2` is the indicator function of whether $X_i > 2$ or not.

Next time: *We'll continue analysing Monte Carlo error, looking at confidence intervals and assessing how many samples to take..*

Summary:

- The Monte Carlo estimator is unbiased.
- The Monte Carlo estimator has mean-square error $\mathrm{Var}(\phi(X))/n$, so the root-mean-square error scales like $1/\sqrt{n}$.
- The mean-square error can be estimated by S^2/n, where S^2 is the sample variance of the $\phi(X_i)$.

Read more: Voss, *An Introduction to Statistical Computing*, Subsection 3.2.2.

4 Monte Carlo error II: practice

4.1 Recap

Let's recap where we've got to. We know that the Monte Carlo estimator for $\theta = \mathbb{E}\,\phi(X)$ is

$$\widehat{\theta}_n^{\mathrm{MC}} = \frac{1}{n}\sum_{i=1}^{n}\phi(X_i).$$

Last time, we saw that the Monte Carlo estimator is unbiased, and that its mean-square and root-mean-square errors are

$$\mathrm{MSE}\left(\widehat{\theta}_n^{\mathrm{MC}}\right) = \frac{1}{n}\,\mathrm{Var}\left(\phi(X)\right) \qquad \mathrm{RMSE}\left(\widehat{\theta}_n^{\mathrm{MC}}\right) = \sqrt{\frac{1}{n}\,\mathrm{Var}\left(\phi(X)\right)}.$$

We saw that these themselves can be estimated as S^2/n and $S/\sqrt{n}$ respectively, where S^2 is the sample variance of the $\phi(X_i)$s.

4.2 Confidence intervals

So far, we have described our error tolerance in terms of the MSE or RMSE. But we could have talked about "confidence intervals" or "margins of error" instead. This might be easier to understand for non-mathematicians, for whom "root-mean-square error" doesn't really mean anything.

Here, we will want to appeal to the central limit theorem approximation. A bit more probability revision: Let $Y_1, Y_2, ...$ be IID again, with expectation μ and variance σ^2. Write $\overline{Y}_n$ for the mean. We've already reminded ourselves of the law of large numbers, which says that $\overline{Y}_n \to \mu$ as $n \to infty$. Then in the last lecture we saw that $\mathbb{E}\overline{Y}_n = \mu$ and $\mathrm{Var}(\overline{Y}_n) = \sigma^2/n$. The **central limit theorem** says that the distribution of $\overline{Y}_n$ is approximately normally distributed with those parameters, so $\overline{Y}_n \approx \mathrm{N}(\mu, \sigma^2/n)$ when n is large. (This is an informal statement of the central limit theorem: you probably know some more formal ways to more precisely state it, but this will do for us.)

Recall that, in the normal distribution $\mathrm{N}(\mu, \sigma^2)$, we expect to be within 1.96 standard deviations of the mean with 95% probability. More generally, the interval $[\mu - q_{1-\alpha/2}\sigma, \mu + q_{1-\alpha/2}\sigma]$, where $q_{1-\alpha/2}$ is the $(1 - \frac{\alpha}{2})$-quantile of the normal distribution, contains the true value with probability approximately $1 - \alpha$.

We can form an approximate confidence interval for a Monte Carlo estimate using this idea. We have our Monte Carlo estimator $\hat{\theta}_n^{\mathrm{MC}}$ as our estimator of the μ parameter, and our estimator of the root-mean-square error $S/\sqrt{n}$ as our estimator of the σ parameter. So our confidence interval is estimated as

$$\left[\hat{\theta}_n^{\mathrm{MC}} - q_{1-\alpha/2}\frac{S}{\sqrt{n}},\ \hat{\theta}_n^{\mathrm{MC}} + q_{1-\alpha/2}\frac{S}{\sqrt{n}}\right].$$

Example 4.1. We continue the example of Example 2.1 and Example 3.2, where we were estimating $\mathbb{P}(Z > 2)$ for Z a standard normal.

```
n <- 1e6
samples <- rnorm(n)
MCest   <- mean(samples > 2)
RMSEest <- sqrt(var(samples > 2) / n)
MCest
```

```
[1] 0.023177
```

Our confidence interval is estimates as follows

```
alpha <- 0.05
quant <- qnorm(1 - alpha / 2)
c(MCest - quant * RMSEest, MCest + quant * RMSEest)
```

```
[1] 0.02288209 0.02347191
```

4.3 How many samples do I need?

In our examples we've picked the number of samples n for our estimator, then approximated the error based on that. But we could do things the other way around – fix an error tolerance that we're willing to deal with, then work out what sample size we need to achieve it.

We know that the root-mean-square error is

$$\mathrm{RMSE}\left(\hat{\theta}_n^{\mathrm{MC}}\right) = \sqrt{\frac{1}{n}\mathrm{Var}\left(\phi(X)\right)}$$

So if we want to get the RMSE down to ϵ, say, then this shows that we need

$$\epsilon = \sqrt{\frac{1}{n} \operatorname{Var}(\phi(X))}.$$

Squaring both sides, multiplying both sides by n, and dividing both sides by ϵ^2 gives

$$n = \frac{1}{\epsilon^2} \operatorname{Var}(\phi(X)).$$

So this tells us how many samples n we need. Except we still have a problem here, though. We (usually) don't know $\operatorname{Var}(\phi(X))$. But we can't even *estimate* $\operatorname{Var}(\phi(X))$ until we've already taken the samples. So it seems we're stuck.

But we can use this idea with a three-step process:

1. Run an initial "pilot" Monte Carlo algorithm with a small number of samples n. Use the results of the "pilot" to estimate the variance $S^2 \approx \operatorname{Var}(\phi(X))$. We want n small enough that this runs very quickly, but big enough that we get a reasonably OK estimate of the variance.
2. Pick a desired RMSE accuracy ϵ. We now know that we require roughly $N = S^2/\epsilon^2$ samples to get our desired accuracy.
3. Run the "real" Monte Carlo algorithm with this big number of samples N. We will put up with this being quite slow, because we know we're definitely going to get the error tolerance we need.

(We could potentially use further steps, where we now check the variance with the "real" big-N samples, and, if we learn we had underestimated in Step 1, take even more samples to correct for this.)

Example 4.2. Let's try this with Example 1.2 from before. We were trying to estimate $\mathbb{E}(\sin X)$, where $X \sim \mathrm{N}(1, 2^2)$.

We'll start with just $n = 1000$ samples, for our pilot study.

```
n_pilot <- 1000
samples <- rnorm(n_pilot, 1, 2)
var_est <- var(sin(samples))
var_est
```

```
[1] 0.4905153
```

This was very quick! We won't have got a super-accurate estimate of $\mathbb{E}\phi(X)$, but we have a reasonable idea of roughly what $\mathrm{Var}(\phi(X))$ is. This will allow us to pick out "real" sample size in order to get a root-mean-square error of 10^{-4}.

```
epsilon <- 1e-4
n_real  <- round(var_est / epsilon^2)
n_real
```

```
[1] 49051535
```

This tells us that we will need about 50 million samples! This is a lot, but now we know we're going to get the accuracy we want, so it's worth it. (In this particular case, 50 million samples will only take a few second on a modern computer. But generally, once we know our code works and we know how many samples we will need for the desired accuracy, this is the sort of thing that we could leave running overnight or whatever.)

```
samples <- rnorm(n_real, 1, 2)
MCest <- mean(sin(samples))
MCest
```

```
[1] 0.1139149
```

```
RMSEest <- sqrt(var(sin(samples)) / n_real)
RMSEest
```

```
[1] 9.964886e-05
```

This second step was quite slow (depending on the speed of the computer being used – it was only about 5 seconds on my pretty-new laptop, but slower on my ancient work desktop). But we see that we have indeed got our Monte Carlo estimate to (near enough) the desired accuracy.

Generally, if we want a more accurate Monte Carlo estimator, we can just take more samples. But the equation

$$n = \frac{1}{\epsilon^2}\,\mathrm{Var}\left(\phi(X)\right)$$

is actually quite bad news. To get an RMSE of ϵ we need order $1/\epsilon^2$ samples. That's not good. Think of it like this: to *double* the accuracy we need to *quadruple* the number of samples. Even worse: to get "one more decimal place of accuracy" means dividing ϵ by ten; but that means multiplying the number of samples by one hundred!

More samples take more time, and cost more energy and money. Wouldn't it be nice to have some better ways of increasing the accuracy of a Monte Carlo estimate besides just taking more and more samples?

Next time: *We begin our study of clever "variance reduction" methods for Monte Carlo estimation.*

Summary:

- We can approximate confidence intervals for a Monte Carlo estimate by using a normal approximation.
- To get the root-mean-square error below ϵ we need $n = \mathrm{Var}(\phi(X))/\epsilon^2$ samples.
- We can use a two-step process, where a small "pilot" Monte Carlo estimation allows us to work out how many samples we will need for the big "real" estimation.

Read more: Voss, *An Introduction to Statistical Computing*, Subsections 3.2.2–3.2.4.

5 Control variate

Summary:

- Variance reduction techniques attempt to improve on Monte Carlo estimation making the variance smaller.
- If we know $\eta = \mathbb{E}\,\psi(X)$, then the control variate Monte Carlo estimate is

$$\hat{\theta}_n^{\mathrm{CV}} = \frac{1}{n}\sum_{i=1}^{n}\left(\phi(X_i) - \psi(X_i)\right) + \eta.$$

- The mean-square error of the control variate Monte Carlo estimate is

$$\mathrm{MSE}\left(\hat{\theta}_n^{\mathrm{MC}}\right) = \frac{1}{n}\,\mathrm{Var}\left(\phi(X) - \psi(X)\right).$$

Read more: Voss, *An Introduction to Statistical Computing*, Subsection 3.3.3.

6 Antithetic variables I

Summary:

- Estimation is helped by combining individual estimates that are negatively correlated.
- For antithetic variables Monte Carlo estimation, we take pairs of non-independent variables (X, X'), to get the estimator

$$\hat{\theta}_n^{\mathrm{AV}} = \frac{1}{n} \sum_{i=1}^{n/2} (\phi(X_i) + \phi(X_i')).$$

On **Problem Sheet 1**, you should now be able to answer all questions. You should work through this problem sheet in advance of the problems class on *Thursday 17 October*.

Read more: Voss, *An Introduction to Statistical Computing*, Subsection 3.3.2.

Problem Sheet 1

This is Problem Sheet 1, which covers material from Lectures 1 to 6. You should work through all the questions on this problem sheet in advance of the problems class, which takes place in the lecture of **Thursday 16 October**. If you are stuck on any of the questions, you are welcome to discuss them with me in my office hours on Thursdays at 1300.

This problem sheet is to help you practice material from the module and to help you check your learning. It is *not* for formal assessment and does not count towards your module mark.

However, if, optionally, you would like some brief informal feedback on **Questions 4, 6 and 8** (marked), I am happy to provide some. If you want some brief feedback, you should submit your work electronically through Gradescope via the module's Minerva page by **1400 on Tuesday 14 October**. I will return some brief comments on your those two questions by the problems class on Thursday 16 October. Because this informal feedback, not part of the official assessment, I cannot accept late work for any reason – but I am always happy to discuss any of your work on any question in my office hours.

Many of these questions will require use of the R programming language (for example, by using the program RStudio).

Full solutions should be released on Friday 17 October.