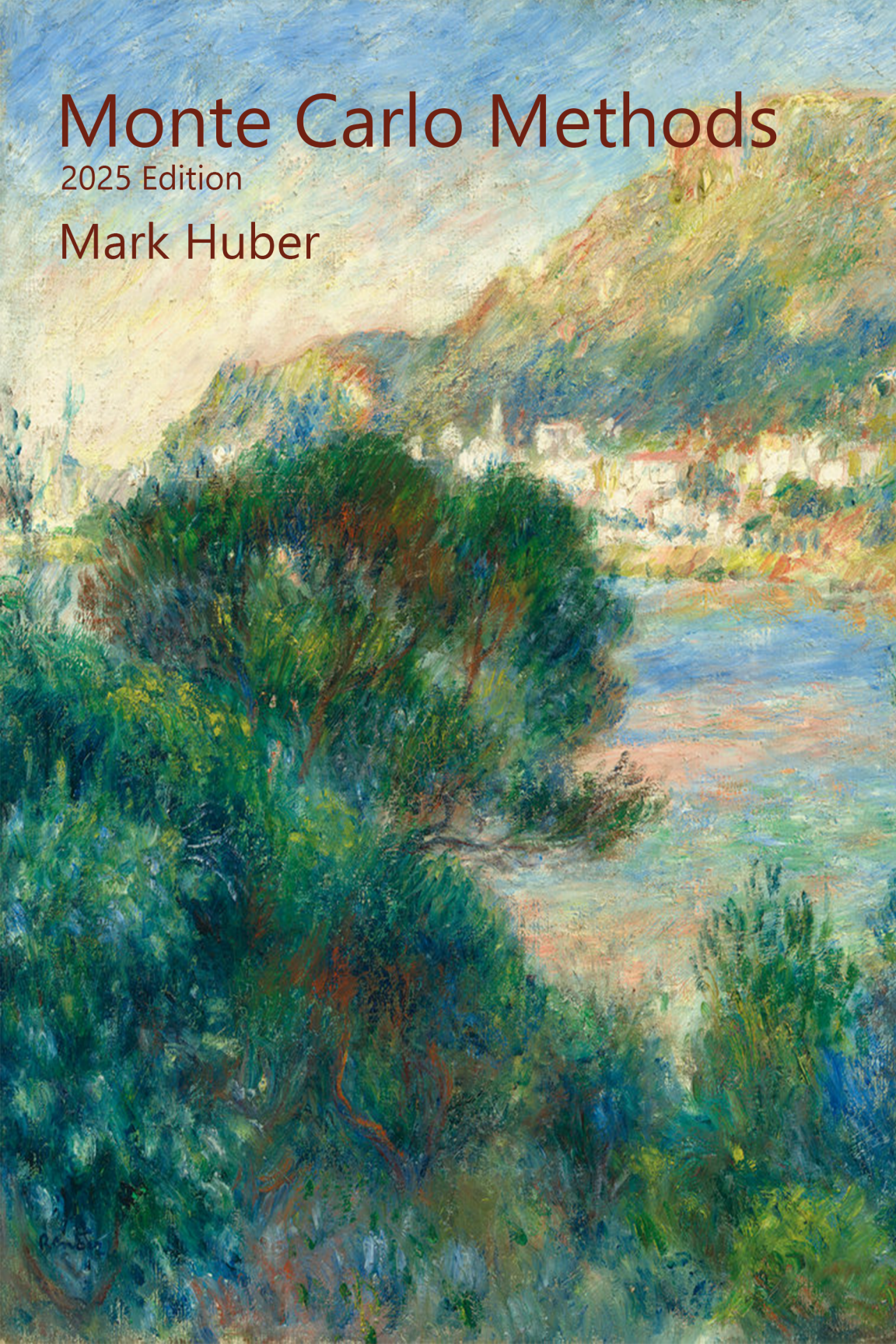
Monte Carlo Methods

2025 Edition

Mark Huber



Version 2025-01-10

Contents

Preface

These notes cover a one semester course in Monte Carlo methods for students who have had a calculus based probability course.

A *Monte Carlo method* is any computational technique that utilizes random choices. Typically these random choices are using to make the algorithm faster than a deterministic one. These methods are essential for the estimation of high dimensional sums or integrals that arise in physics, statistics (both frequentist and Bayesian), #P complete problems, centrality measures in graphs, and evaluation of the prices of derivatives in mathematical finance.

The course introduces students to the use of R, but does not assume prior programming knowledge.

Background The preparation needed for this course is a typical undergraduate course in probability. In particular, knowledge of expected value together with exponential and uniform distributions is a must. The text *Probability: Theory and Exploration* is OpenAccess and free to use, and covers all the ideas from Probability needed for the course. For those needing a less comprehensive review, the first Appendix provides a rapid introduction to probability.

Why are all the numerical answers in problems and examples given to 4 significant digits?

In my homework assignments to students I require that all noninteger answers be presented to four significant digits. There are several reasons why I do this.

The first is that it makes answers uniform. I do not have to worry if $1/(3+\sqrt{2}) = (3-\sqrt{2})/5$ or not if the answer given is 0.2265. The second is that it emphasizes to students that in most problems in applied mathematics the exact numbers are uncertain. The number $1/3$ is specific and exact, but not actually encountered outside of toy problems. Third, it builds numerical literacy (or numeracy as it is sometimes called.) Seeing that $\exp(-2) \approx 13.53\%$ is a useful thing, as it gives that much desired reality check on the answers provided.

Part I

One dimensional Monte Carlo

Chapter 1

Introduction to Monte Carlo

Question of the day

Estimate

$$\int_0^1 x^2 \, dx$$

by using random choices.

Summary

Monte Carlo methods are algorithms that use random choices. The basic Monte Carlo technique to estimate a number a is as follows.

1. Construct a random variable X such that $\mathbb{E}(X) = a$.
2. Take independent identically distributed samples $X_1, \ldots, X_n$ where each $X_i \sim X$, and return
$$\hat{a} = \frac{X_1 + \cdots + X_n}{n}$$
as the estimate of a.

For bounded integrals, to estimate

$$I = \int_{x \in [0,1]} g(x) \, dx,$$

use U uniformly distributed over $[0, 1]$. Then $\mathbb{E}[g(U)] = I$.

1.1 What is Monte Carlo?

An integral such as

$$I = \int_0^1 x^2 \, dx$$

is deterministic. That is, there is exactly one true value of I. So at first glance, it would seem strange to intentionally add random choices into an algorithm for finding I. And for such a simple integral, that intuition would be exactly right.

But as integrals become more complicated and move to higher dimensions, the use of randomness in their evaluation becomes essential to developing efficient approximations. To distinguish a deterministic algorithm from one that inject randomness into decisions, call the ones that use randomness *Monte Carlo methods*.

Definition 1
A **Monte Carlo method** is an algorithm that utilizes random choices.

History 1
Monte Carlo algorithms go back thousands of years, but in the modern era credit for developing them into a commonly used method is typically given to Stanisław Ulam who conceived of them during work at Los Alamos National Laboratory. Ulam loved to gamble, and his friend Nicholas Metropolis gave these algorithms the moniker *Monte Carlo* because of the famous casino in Monte-Carlo, Monaco.

Figure 1.1: *Stanisław Ulam's Los Alamos desk exhibit.*

When computer scientists study an algorithm, they often find it useful to imagine that it is a game that is being played against is an opponent called *the adversary*. This adversary is trying to give inputs for the algorithm that make the algorithm run for as long as possible. By making choices randomly, however, it is often possible to foil the adversary most of the time and create an algorithm that runs more quickly than if the same choice is always made when given the same input.

1.2 Integrals

One way to see this phenomenon in action is to consider an algorithm for approximating integrals numerically. Suppose the goal is to calculate the value of an integral or a sum in high dimensions. Deterministic algorithms work well for one-dimensional integrals, but the problem grows much harder as the dimension of the problem increases.

Here, in order to keep things simple, start by looking at a one dimensional integral:

$$a = \int_{x=0}^{1} x^2 \, dx.$$

Any integral consists of three pieces:

1. **Limits**: In the example, the limits are $x \in [0, 1]$.
2. **Integrand**: The function being integrated, in this case x^2.
3. **Differential**: Here we use dx to indicate that this is to be calculated using a Riemann integral.

The acronym LID can be used as a mnemonic to remember these three parts of an integral.

It is helpful to be able to write our integrals as going from negative infinity up to positive infinity. Outside the original limits of the integral, the value of the integrand should be zero. An *indicator function* can be used to do this.

Define the indicator function as follows.

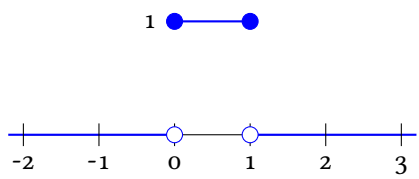
Definition 2
Using T for a true statement and F for a false statement, the **indicator function** $\mathbb{I} : \{\mathsf{F}, \mathsf{T}\} \to \{0, 1\}$ is defined as

$$\mathbb{I}(\mathsf{T}) = 1$$
$$\mathbb{I}(\mathsf{F}) = 0.$$

The indicator function can be used to describe functions that have jumps. For instance, the function

$$f(x) = \mathbb{I}(x \in [0, 1])$$

evaluates to 1 when $x \in [0, 1]$, and is o everywhere else. So its function looks like



You might recognize this function as the *density* or *pdf* of a random variable that is uniform over $[0, 1]$.

Since $f(x) = \mathbb{I}(x \in [0, 1])$ equals 0 outside $[0, 1]$, anything multiplied by it will also be 0 outside $[0, 1]$. Anything in $[0, 1]$ is unchanged when multiplied by $\mathbb{I}(x \in [0, 1])$. Also, when the limits are omitted, the convention is that the limits are over the whole space of the variable of integration. That means

$$\int_{x=0}^{1} x^2 \, dx = \int_{x=-\infty}^{\infty} x^2 \mathbb{I}(x \in [0, 1]) \, dx = \int x^2 \mathbb{I}(x \in [0, 1]) \, dx.$$

In other words, an indicator function can be used to incorporate the limits of integration into the function being integrated (the integrand.) In the other direction, you can take an indicator function out of the integrand by placing it in the limits.

Okay, but what does that actually accomplish? Remember when you learned how to find the expected value of a random variable? Recall that $\mathbb{E}[Y]$ stands for the *mean* (also known as the *expectation*, *expected value*, and *average*) of the random variable Y.

Fact 1
For X a continuous random variable with density f_X and h a (measurable) function,

$$\mathbb{E}[g(X)] = \int_{x \in \mathbb{R}} g(x) f_X(x) \, dx.$$

For X a discrete random variable with density f_X and h a (measurable) function,

$$\mathbb{E}[g(X)] = \sum_{i \in \mathbb{R}} g(i) f_X(i).$$

Recall $\mathbb{I}(x \in [0, 1])$ is the density of a random variable X that is uniform over $[0, 1]$. Write $X \sim \mathsf{Unif}([0, 1])$. Then

$$\mathbb{E}[X^2] = \int_{x \in \mathbb{R}} x^2 \mathbb{I}(x \in [0, 1]) \, dx.$$

This means that the value of the integral is an expected value of a random variable. So now the question becomes: how can $\mathbb{E}[X^2]$ be estimated?

1.3 The Strong Law of Large Numbers

Consider a stream $X_1, X_2, \ldots$ of random variables that have the same distribution as X. We call such a stream *independent, identically distributed* (iid for short) random variables.

Then the Strong Law of Large Numbers says that with probability 1, the sample average of the first n values of the stream will converge to the mean of X provided $\mathbb{E}[X]$ exists and is finite.

Theorem 1 (Strong Law of Large Numbers)
Let X have finite expectation, and $X_1, X_2, \ldots \sim X$ be an iid sequence. Then

$$\mathbb{P}\left(\frac{X_1 + \cdots + X_n}{n} = \mathbb{E}[X]\right) = 1.$$

So one way to estimate $\mathbb{E}[X^2]$ is to draw a bunch of numbers from the stream, and take the sample average.

Definition 3
The **naive Monte Carlo method** for estimating a is to generate $X_1, \ldots, X_n$ where each X_i has mean a, and use

$$\hat{a} = \frac{X_1 + \cdots + X_n}{n}$$

as the estimate.

1.4 Using R

It's no coincidence that Monte Carlo methods gained prominence around the time that the first digital computers were invented. It turns out that to make Monte Carlo methods effective, you need a *lot* of random variables, and computers are a great way to make that happen.

In this text, I'll be giving code examples using the statistical computing language R. Since R is open source, you can download it for free to your computer or laptop from https://www.r-project.org. In the next chapter I'll discuss how to use R and a companion program called RStudio to build more complicated pieces of code.

For now, however, you can run R code directly through your web browser using various services. One such is called Programiz, and can be accessed at https://www.programiz.com/r/online-compiler/. The code will be typed into a text editor on the left, and then after the blue "Run" button is pressed, the output will be displayed on the right. This is a public server, so do not run any code that you wish to keep private on this service.

To generate random uniform variables over $[0, 1]$ we can use the `runif()` command in R. All the random number generating functions in R have the same form, an `r` followed by the name of the distribution. The first parameter of the function specifies how many uniforms we need.

To store the results, the assignment operator `<-` is used. The command

```
runif(1)
```

returned `[1] 0.484434` when run for this work, but if you run the command the result could be different! The `[1]` means that the output starts with the first number. The number of random values generated was controlled by the number inside the parentheses after `runif`. To get more random numbers, increase the number.

The command

```
runif(4)
```

gives output

```
[1] 0.5453125 0.3496551 0.4990681 0.2960938
```

when run.

The `mean()` command (aka function) takes the sample average of a vector. That is, it adds up the values then divides by the number of values. Putting this together, the following generates 1000 iid uniforms, then takes the sample average. To feed these numbers to the `mean()` function, use the *pipe* operator `|>`.

```
runif(1000) |> mean()
```

Now let's estimate $\int_0^1 x^2 \, dx$ using the following code:

```
runif(1000)^2 |> mean()
```

The `^2` notation means to square the random numbers used. The output result should be close to 0.3333333, but not exactly.

Each time you run the code, your answer will change slightly. This is because each time you run the code, the algorithm is making slightly different random choices. To force the algorithm to make the same random choice each time, you can use the `set.seed()` function.

```
set.seed(123456)
runif(1000)^2 |> mean()
```

With the seed set in this way, when I ran the code I always received the answer

```
[1] 0.3290711
```

The `[1]` means that the first number of output is `0.3290711`. No matter how many times the "Run" button is pressed, the result is the same.

The random numbers in most computing languages are the result of applying a complex function to an initial seed. By resetting the seed with the function `set.seed()`, you can get the same results every time you call any function that generates from a probability distribution.

A few notes.

- No computer can actually generate a number that is uniform over $[0, 1]$ because that would require an infinite amount of memory to hold the digits. Instead, it will generate numbers uniform over the finite set of possible numbers represented by the computer using 32 bits of information. This is called a *single-precision floating point number*. Some languages generate 64 bits of information. This is called a *double-precision floating point number*, or *double* for short.

- Using more uniforms gives more accurate results. Trying

  ```
  runif(10^6)^2 |> mean()
  ```

three times gives

$$0.3329048,\ 0.3337699,\ 0.3334153,$$

which are more tightly clustered around the true answer of $1/3$ than when only 1000 samples were used. In later chapters the error in the Monte Carlo method will be discussed in detail, along with approaches to minimizing error.

- Once a random variable has been generated by a computer, it becomes a *random variate*.

Problems

1.1: Given U a uniform over $[0, 1]$, that is, a standard uniform, create a random variable Y that is a function of U such that

$$\mathbb{E}[Y] = \int_0^1 x^3 \, dx.$$

1.2: Given a standard uniform U, create a random variable W that is a function of U such that

$$\mathbb{E}[W] = \int_0^1 e^x \, dx.$$

1.3: Consider the integral

$$I = \int_{s=0}^2 s^2 \, ds.$$

a) Find a change of variables for s so that the limit of the integral runs from 0 to 1.

b) Find a function h such that for $U \sim \mathsf{Unif}([0, 1])$, $\mathbb{E}[h(U)] = I$.

1.4: Create a function h such that for $U \sim \mathsf{Unif}([0, 1])$,

$$h(U) = \int_{-\tau/4}^{\tau/4} \cos(t) \, dt.$$

It might be helpful to first change the integral through a change of variables for t.

1.5: Given an integral

$$I = \int_0^1 \exp(-\sqrt{x}) \, dx,$$

what function $h(u)$ has $\mathbb{E}[h(U)] = I$ for U a standard uniform?

1.6: Given an integral

$$I = \int_0^1 1/\sqrt{1 + x} \, dx$$

what function $g(u)$ has $\mathbb{E}[g(U)] = I$ for U a standard uniform?

1.7: Write R code to estimate

$$I = \int_0^1 \exp(-\sqrt{x})\, dx$$

with seed 123456 using 10^6 samples. Hint: look up (or use your favorite search engine) to find out the functions in R to take the square root function and apply the exponential function to numbers.

1.8: Write R code to estimate

$$I = \int_0^1 1/\sqrt{1+x}\, dx$$

with seed 123456 using 10^6 samples.

1.9: If $U \sim \text{Unif}([0,1])$, what is $\mathbb{E}[\sqrt{U}]$?

1.10: If $U \sim \text{Unif}([0,1])$, what is $\mathbb{E}[1/(1+U)]$?

Chapter 2

Scripts and R Markdown files

Question of the day

What is R, and how can it be used effectively to run Monte Carlo simulations?

Summary

- **R** is a free statistical computing environment that is well-suited to building Monte Carlo algorithms.
- **RStudio** is a free Integrated Development Environment for building code in R.
- A **script** is a set of commands that can be executed by R. Use a `.R` file extension for scripts.
- An **R Markdown** file can be easily transformed into a standard, professional looking document that includes R code execution. Use a `.Rmd` file extension for these files.
- Inside RStudio, an R Markdown file acts as a **notebook**, where code chunks can be executed individually, and the results displayed.

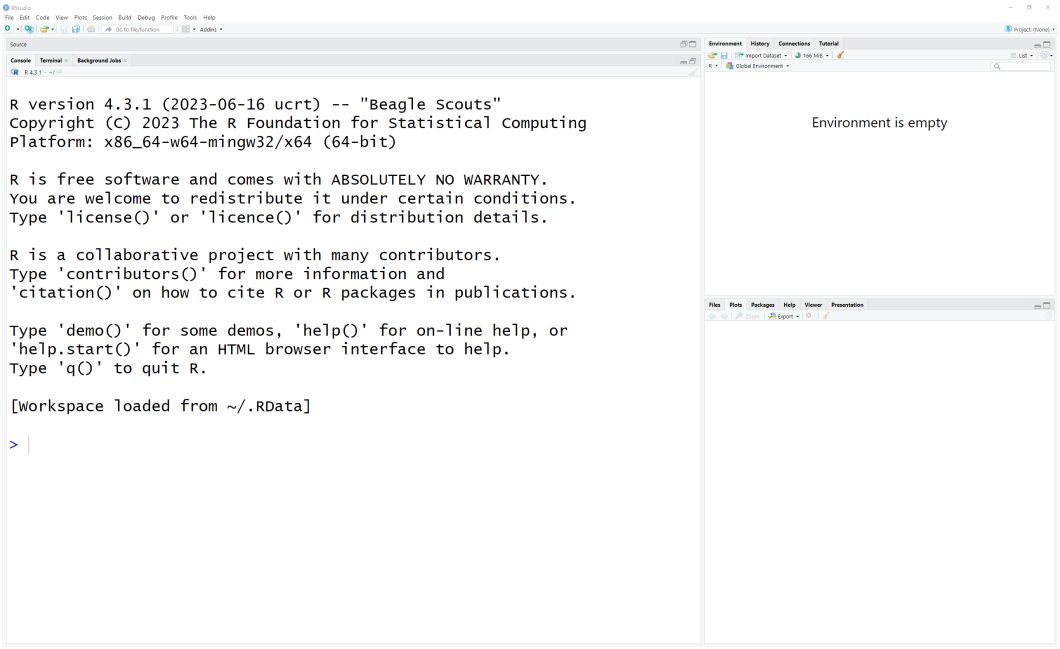
R is a *statistical environment for computing*, which means that it is a programming language that has the capacity to run scripts and many functions intended to help in running Monte Carlo algorithms. In addition, it has a free Integrated Development Environment (IDE) called *RStudio* that includes *R Markdown*. This allows the user to create a publishable work that includes code, text, and mathematics using the LaTeXsystem.

RStudio was created by a company called Posit, and the best way to find out how to download it is to use a search engine with the query `ownloa` RStudio . As of December 2024, this takes you to https://posit.co/download/rstudio-desktop/ that includes buttons on the web page for first downloading R and then downloading RStudio. Because this is run by a company that link will probably change over time.

Once you install first R and then RStudio on your computer, you are ready to start learning how to use R to compute!

2.1 The Console

When you first start RStudio, it will look something like this:



The various rectangular areas on the screen are called **panes**. The pane on the left is (by default) set to the tab labeled *console*. In the console, you can type in commands that perform computations or assign values to variables.

For instance, if you type

```
4 + 5
```

in the console and hit Enter, you will see

```
[1] 9
```

The `[1]` indicates that there was one output from your computation. The result of the computation was `9`.

Assignment to variables can be done with the `<-` operator.

```
x <- 4
```

Once a variable (like `x`) has been assigned a value, then every time R sees that variable name, it substitutes the value the variable has been assigned. So

```
x + 5
```

also returns

```
[1] 9
```

when executed.

2.2 Scripts

Often we wish to give multiple commands to R. The simplest way to organize these commands is to use a *script*.

Definition 4
In R, a **script** is a collection of commands that you want R to execute.

In RStudio, a file for a script can be created with the menu command:

File ▶ New File ▶ R Script.

By default, RStudio will open up a window in the upper left portion of its area. Commands can then be typed into this area. For instance, suppose I put

```
x <- 4
y <- 5
x + y
```

Note that nothing happens, except the lines of the file get numbered 1, 2, and 3 (in the gray area to the left of the code) as you type the lines.

To tell R to execute these lines, use the `source` function. There is a very helpful shortcut for activating this function. Above the pane where the code displays there is a checkbox with the label 'Source On Save'. When this box is checked, every time the file is saved, the script will automatically be executed in the console.

Give it a try! First, make sure that the checkbox is checked. Then use the menu command

File ▶ Save

to save the file (there is also a shortcut for this command that is operating system dependent.)

Note that upon saving the file, in the console below the `source` command has been given.

Check that the commands were executed by typing

```
x
y
```

directly into the console.

R fact 1
If the file extension is unspecified when saving a file, the default is `.R`. This convention for R script files will be sued throughout this work.

It seemed like the `x+y` command in the script did not do anything! This is because commands that would normally print in the console do not print when executed as part of a script. A value can be forced to print in a script using the `print` command. Try changing line 3 of your script to

```
print(x+y)
```

and source the script again. This time you should get output

```
[1] 9
```

just as if you had typed `x + y` into the console.

Why you should use scripts Scripts act as a scientific record of your analysis. They record *exactly* what you did and how you got your simulated data.

Now, if someone else (say a researcher following your work) is trying to extend your analysis or apply it to another area, they do not have to guess what exactly you did. They can see each step exactly. That is why it is important to keep track of your procedures precisely as a script or similar form.

2.3 R Markdown

One of the most important aspects of Monte Carlo simulations is having the ability to communicate what you learned from your simulation, and the ability to show how you found it to others. Scripts are a good start to this process.

You want your communications to have the following good properties.

1. **Complete.** Someone reading your work should be able to replicate what you did.
2. **Compatible.** You want to use a standard format, HTML, PDF, or Markdown to communicate your results so that they can be viewed by the widest possible not so tech savvy audience.
3. **Professional.** You want output that is both neat, well organized, and looks good.

These three goals are usually accomplished using a *markup language*. These are a set of commands that allow you to *emphasize* words, add a bit of color, and start new sections, subsections, and paragraphs. Markup languages also can be used to create a list of bullet points or numbered points.

The most commonly used markup language today is HTML, which stands for *Hypertext Markup Language* and is the language webpages are usually written in.

In mathematics and the sciences, another commonly used markup language is LaTeX, because it is very good at typesetting documents that include mathematics. This ebook was typeset using LaTeX.

Most word processors have an internal markup language, but since the user usually cannot see it, they cannot directly make changes. The advantage of a markup language is that you can specify what you want to happen in a general sense, and then the language takes care of the details. For instance, if you say you want a new chapter, the markup language will take care of the numbering and table of contents for you without the need for you to intervene.

2.4 Markdown

That being said, markup languages can be cumbersome to use with a lot of extra commands embedded in the file. For this reason, John Gruber created a light markup language that emphasized ease of use and readability over the ability to do any possible thing. The result was *Markdown*. (Get it? Mark**down** is a lighter version of a mark**up** language. That's computer science humor for you in a nutshell.)

The Markdown language has been implemented in many different formats, the one that we will use here is the version implemented by R, called R Markdown. If you want to learn more about how R Studio incorporates R Markdown, go to https://rmarkdown.rstudio.com/

We can start a new R markdown file with

File ▶ New File ▶ R Markdown

which will create a new document similar to the way that we created a new script.

R fact 2
The default file extension for R Markdown files is `.Rmd`.

Here's an example of an R markdown file that contains code

```
---
title: "Our script in R Markdown"
author: "Mark Huber"
date: "November 15, 2018"
output:
  html_document:
    number_sections: true
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is a document written in R Markdown. Notice that we
    started a new section very simply by putting a # character.
    To create a subsection, we can use ##. More examples,
    templates, and instructions for R Markdown can be found at <
    http://rmarkdown.rstudio.com>.

This is a subsection

To take an R Markdown document in R Studio and create a new
    document, use the **Knit** button. This button takes the
    Markdown document and converts it to a different format or
```

```
    markup language.  Because in the preamble (called the YAML (
    YAML is not a markup language) heading) it says output:  html
    _document, the final output created by knit will be in HTML
    for this document.

## Putting R code inside an R Markdown document

When you click the **Knit** button a document will be generated
    that includes both text as well as the output of any embedded
     R code chunks within the document. R commands are embedded
    as a **code chunk** like this:

'''{r code}
x <- 4
y <- 5
print(x + y)
'''
```

Some key points:

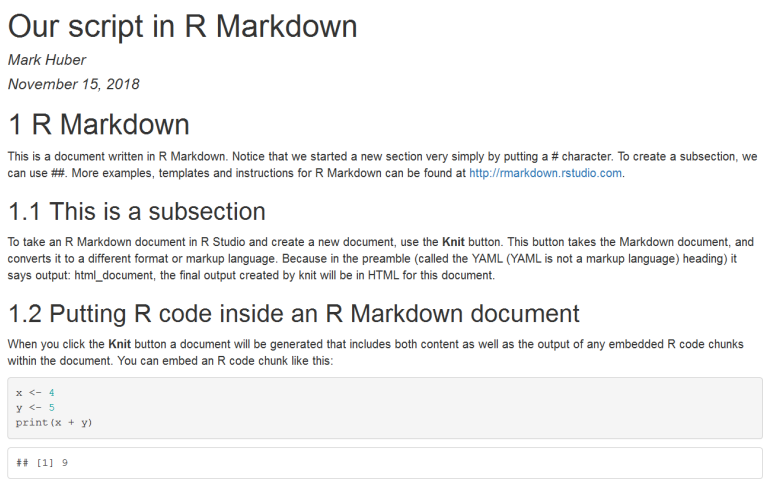
1. The heading at the beginning marked out by `---` is called a YAML header. YAML stands for YAML Ain't Markup Language. This is an example of a recursive acronym. The contents of the header such as `title` and `author` should be self-explanatory. YAML is not a markup language, instead, it is considered a *data serialization language*.

2. Use # to start a new section.

3. Use ## to start a new subsection

4. Use ``` ``` ``` to mark out blocks of code. The `echo=FALSE` parameter indicates that the code block should not be listed in the output, but the results of running the command should be.

Note that there is a button above the file called **Knit**. Press this button to turn the R Markdown file into an HTML file.



Our script in R Markdown

Mark Huber

November 15, 2018

1 R Markdown

This is a document written in R Markdown. Notice that we started a new section very simply by putting a # character. To create a subsection, we can use ##. More examples, templates and instructions for R Markdown can be found at http://rmarkdown.rstudio.com.

1.1 This is a subsection

To take an R Markdown document in R Studio and create a new document, use the **Knit** button. This button takes the Markdown document, and converts it to a different format or markup language. Because in the preamble (called the YAML (YAML is not a markup language) heading) it says output: html_document, the final output created by knit will be in HTML for this document.

1.2 Putting R code inside an R Markdown document

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
x <- 4
y <- 5
print(x + y)
```

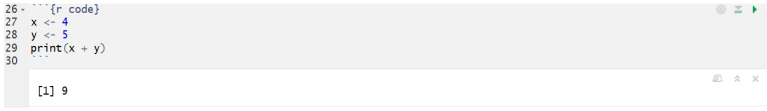
```
## [1] 9
```

2.5 *Using .Rmd files as a notebook*

Going back to the .Rmd file in RStudio, notice that any code chunk has a little green arrow next to it.

```
26 ```{r code}
27 x <- 4
28 y <- 5
29 print(x + y)
30 ```
```

Pressing that green arrow immediately copies the contents of the code chunk to the console and executes each line one at a time. The output is displayed in the R Markdown file right below the code chunk.



```
26 ```{r code}
27 x <- 4
28 y <- 5
29 print(x + y)
30 ```

[1] 9
```

This ability to execute code chunks inside a document makes the file a *notebook*.

The symbol just to the left of the little green arrow will execute the current code chunk and all the chunks above it, which can be helpful if it depends on previous definitions.

For simple analyses, there is no need to write scripts, only R Markdown files.

2.6 LaTeX

When you are writing papers and descriptions in social or physical sciences, you often need to add in mathematics equations and definitions. These can be added through the use of LaTeX.

Here we will not be using a complete LaTeX document, instead, we will embed simple commands into our R Markdown documents. For instance, suppose we added to our previous document the following code.

```
## LaTeX examples

This is an example of inline mathematics:  \( a^2+b^2=c^2 \).
   This is an example of *inline mathematics*, the mathematics
   is presented in the middle of a line of text.

The second kind of mathematics is *display mathematics*, which
   is written like
\[
a^2 + b^2 = c^2.
\]
This is the same statement, but now it appears on a separate
   line of the output document.
```

The result looks like

1.4 LaTeX examples

This is an example of inline mathematics: $a^2 + b^2 = c^2$. This is an example of *inline mathematics*, the mathematics is presented in the middle of a line of text.

The second kind of mathematics is *display mathematics*, which is written like

$$a^2 + b^2 = c^2.$$

This is the same statement, but now it appears on its own line of the document.

Problems

2.1: Two useful symbols in LaTeX are `_` for subscripts and `^` for superscripts. For instance, `x^3` becomes x^3, and `x_7` becomes x_7. Use LaTeX and braces { and } to make the following.

a. x^2.

b. x^{-3}.

c. x_4.

d. x_{a+b}.

2.2: Write LaTeX code to do the following.

a. Create x^{-2}.

b. Create x^{y+x}.

c. Create x_{-2}.

d. Create x_{y+x}.

2.3: The `\frac` command in LaTeX creates fractions. For instance, `\frac{3}{4}` produces

$$\frac{3}{4}$$

Write LaTeX code to create

$$\frac{x_1 + x_2 + x_3}{3} = 7.3.$$

2.4: Write LaTeX code to make the following line of mathematics:

$$\frac{34 + 66}{100} = 1.$$

2.5: The `\int` command makes integrals in LaTeX. For instance, `\int_0^5 x^2 dx = 125 /3` could be used to make

$$\int_0^5 x^2 \, dx = 125/3.$$

Write LaTeX code to make the following output

$$\int_{-1}^{1} 1 - x^2 \, dx.$$

2.6: Using the fact that `\sqrt{x}` returns $\sqrt{x}$, write LaTeX code to create the following output:

$$\int_0^{15} \sqrt{x} \, dx.$$

Chapter 3

Essential Probability

Question of the day

Suppose X has density $f_X(s) = 2s\mathbb{I}(s \in [0,1])$. What is $\mathbb{P}(X \in [0.6, 0.8])$?

Summary

- A **measure** of a set gives the size of the set. The two most common measures used in probability are **Lebesgue measure**, applied to subsets of n dimensional real space, and **counting measure**, applied to discrete sets.
- A **random variable** X is a variable about which the user has extra information, so for some measurable sets A, it is possible to calculate the probability that X falls into the set A.
- A function f_X is a **density** of a random variable X with respect to measure μ if for all measurable sets A,
$$\mathbb{P}(X \in A) = \int_A f_X(s)\, d\mu.$$
- A function g_X is an **unnormalized density** of a random variable $X \in \Omega$ if for $I = \int_\Omega g_X(s)\, d\mu$, g_X/I is a density of X. Call I the **normalizing constant** for the density.
- For real-valued random variables X, the **cumulative distribution function** (aka cdf) of X is a function such that for all real a, $\mathbb{P}(X \leq a) = \text{cdf}_X(a)$. If the cdf is continuous, the random variable is **continuous**. If the cdf is flat with jumps, then the random variable is **discrete**.

Unlike a deterministic method for estimating an integral like the trapezoid rule, Monte Carlo algorithms will usually return a different answer every time you run the simulation. This is because the answer is a *random variable*. In this chapter, the properties of random variables will be explored. In addition, other facts from probability that will be useful to the construction of Monte Carlo Algorithms will be introduced.

3.1 Random variables

So what is a random variable? A regular variable can be anything inside a set. For instance, $x \in \mathbb{R}$ denotes that the value of the variable x is completely unknown, other than it is a real number.

The idea for a random variable like X is that the user has extra information. The value of X is not completely unknown, but a statement like $X \in [5, 10]$ has a probability of being true that is known to the user. That is, random variables are really variables with extra information attached.

To understand that information, it is necessary to know the *density* of a random variable with respect to a *measure*.

3.2 Measures

All real-valued random variables are either *continuous*, *discrete*, or *mixed*.

- **Continuous** means that the chance the random variable equals any particular value is 0. In mathematical notation: $(\forall a \in \mathbb{R})(\mathbb{P}(X = a) = 0)$.
- **Discrete** means that the random variable takes on one of a countable set of values with probability 1. In notation: $(\exists\{x_1, x_2, \ldots\})(\mathbb{P}(X \in \{x_1, x_2, \ldots\} = 1)$
- **Mixed** means that there is a probability that the random variable is continuous. If not, it is discrete. In notation: $(\exists C \text{ cont}, D \text{ discrete}, B \sim \text{Bern}(p))(X = CB + D(1 - B))$

A **measure** tells us the size of a set. The two measures that we use most often in probability are *Lebesgue measure* and *counting measure*.

Lebesgue measure This is for continuous sets, and we use ℓ to denote it. In one dimension, it tells us the length of an interval, in two dimensions, it is the area of the set, in three the volume, and so on. The Lebesgue measure of a set can be found by integrating the function that is the constant 1 over the set, or the indicator function that is 1 when the point falls into the set.

$$\ell(A) = \int_A 1 \, dA = \int_\Omega \mathbb{I}(s \in A) \, ds.$$

Example 1

$$\ell([3, 7]) = \int_{s \in \mathbb{R}} \mathbb{I}(s \in [3, 7]) \, ds = \int_{s \in [3,7]} 1 \, ds = s|_3^7 = 7 - 3 = 4.$$

Counting measure This is for discrete sets, and we use $\#$ to denote it. This counts the number of elements in the set. For A a set,

$$\#(A) = \sum_a \mathbb{I}(a \in A) = \sum_{a \in A} 1 = \sum_a \mathbb{I}(a \in A)$$

Example 2

$$\#(\{a,b,c\}) = \sum_i \mathbb{I}(i \in \{a,b,c\}) = \sum_{i \in \{a,b,c\}} 1 = 1+1+1 = 3.$$

An integral of the form $\int_A f(s)\,ds$ is said to be *with respect to* or *against* Lebesgue measure. Integrals with respect to counting measure can also be built. They turn out to just be sums, which makes them much easier to calculate in practice (but strangely, tougher to handle analytically.)

Notation 1
Writing sums as integrals with respect to counting measure can be done as follows.

$$\sum_i f(i) = \int_i f(i)\,d\#.$$

3.3 Densities

A *density* of a random variable can be used together with integrals to find probabilities associated with the random variable.

Definition 5
A random variable X has **density** (aka **probability density function** aka **pdf**) with respect to measure μ if for $(X \in A)$ a measurable statement,

$$\mathbb{P}(X \in A) = \int_A f_X(s)\,d\mu.$$

This means that if X is a continuous random variable,

$$\mathbb{P}(X \in A) = \int_{s \in A} f_X(s)\,ds,$$

and if X is a discrete random variable

$$\mathbb{P}(X \in A) = \sum_{i \in A} f_X(i).$$

Example 3
Question of the day Here X is continuous, with density $f_X(s) = 2s\mathbb{I}(s \in [0,1])$ with respect to Lebesgue measure. So

$$\begin{aligned}\mathbb{P}(X \in [0.6, 0.8]) &= \int_{s \in [0.6,0.8]} 2s\mathbb{I}(s \in [0,1])\, ds \\ &= \int_{0.6}^{0.8} 2s\, ds = s^2|_{0.6}^{0.8} = 0.64 - 0.36 = \boxed{0.2800}\end{aligned}$$

In the example, since $s \in [0.6, 0.8]$ in the integral, the indicator function $\mathbb{I}(s \in [0,1])$ did not do anything. However, in general, an indicator function can be used to change the limits of integration.

Fact 2
For any real-valued integrand,

$$\int_A f_X(s)\mathbb{I}(s \in B)\, d\mu = \int_{A \cap B} f_X(s)\, d\mu.$$

Example 4

$$\int_3^6 x^2\, dx = \int_{\mathbb{R}} x^2\mathbb{I}(x \in [3,6])\, dx.$$

Often we talk about an *unnormalized density*.

Definition 6
Say that g is an **unnormalized density** over Ω with respect to measure μ if g is a nonnegative function and

$$I = \int_\Omega g\, d\mu$$

is a positive, finite number not equal to 1. Call I the **normalizing constant** of the density, and g/I the **normalized density**.

Example 5
What is the normalized density associated with unnormalized density $g(x) = x^2\mathbb{I}(x \in [3,6])$?

Answer The integral of the unnormalized density is

$$\int_3^6 x^2\,dx = \frac{x^3}{3}\Big|_3^6 = \frac{216-27}{3} = 63,$$

so the normalized density is

$$\boxed{f_X(x) = \frac{x^2\mathbb{I}(x \in [3,6])}{63}.}$$

For historical reasons, letters towards the end of the alphabet (r, s, t, x, y) tend to be used when dealing with integrals against Lebesgue measure. Letters in the middle third of the alphabet (i, j, ℓ, k) are typically used when dealing with integrals against counting measure.

3.4 Cumulative distribution functions

Definition 7
For a real-valued random variable X, let

$$\text{cdf}_X(a) = F_X(a) = \mathbb{P}(X \le a)$$

be the **cumulative distribution function** or **cdf** of X.

If a random variable has a density, calculate the cdf using an integral.

Fact 3
For a random variable X with density f_X with respect to μ,

$$\text{cdf}_X(a) = \int_{-\infty}^a f_X(s)\,d\mu.$$

From the rules of probability, the cdf has several nice properties.

Fact 4
For a random variable X, cdf_X is

1: a function whose image is in $[0, 1]$,

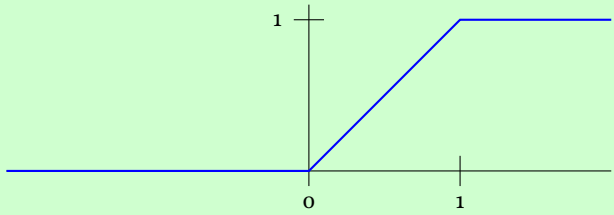
2: right continuous,

3: increasing.

Example 6
For $U \sim \text{Unif}([0, 1])$, the cdf of U is

$$\text{cdf}_U(a) = a \cdot \mathbb{I}(a \in [0, 1]) + 1 \cdot \mathbb{I}(a > 1).$$

This has a graph that looks like



Note that the cdf of U can also be written in piecewise fashion using multiple lines:

$$\text{cdf}_U(a) = \begin{cases} a & a \in [0, 1] \\ 1 & a > 1 \end{cases}$$

Indicator functions can be used to write this function in a single line.

$$\text{cdf}_U(a) = a\mathbb{I}(a \in [0, 1]) + \mathbb{I}(a > 1).$$

When the cdf is a continuous function, say that the random variable is **continuous**.

Definition 8
If cdf_X is a continuous function, then X is a **continuous random variable**.

Example 7
For $U \sim \text{Unif}(\{1, 2, 3, 4\})$,

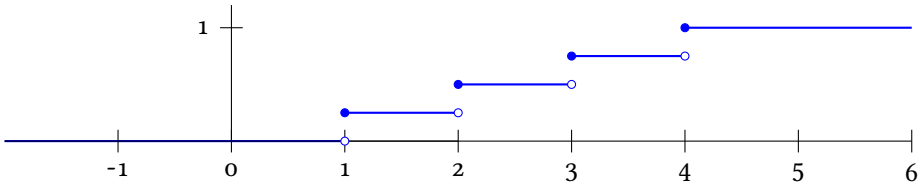
$$\text{cdf}_U(a) = (1/4)\mathbb{I}(a \in [1, 2)) + (2/4)\mathbb{I}(a \in [2, 3)) + (3/4)\mathbb{I}(a \in [3, 4)) + 1 \cdot \mathbb{I}(a \geq 4).$$

Because the cdf is increasing, we can also write this as

$$\text{cdf}_U(a) = (1/4)\mathbb{I}(a \geq 1) + (1/4)\mathbb{I}(a \geq 2) + (1/4)\mathbb{I}(a \geq 3) + (1/4)\mathbb{I}(a \geq 4).$$

From this form, it is much easier to see that U has a $1/4$ chance of equalling each of 1, 2, 3, or 4.

The graph of the cdf looks like:



Note that it is flat in some places and has jumps in others. Call random variables with this type of cdf *discrete.*

Definition 9
Suppose there exist $p_1, p_2, \ldots$ and $a_1, a_2, \ldots$ such that

$$\text{cdf}_X(a) = \sum_{i=1}^{\infty} p_i \mathbb{I}(a \geq a_i).$$

Then X is a **discrete** random variable.

Before getting to that, however, it will be helpful in this course to write Riemann integrals and regular sums both as integrals. That way only one statement about things like expected value is needed instead of two. To accomplish this requires a touch of measure theory. There are two measures that probability uses the most: Lebesgue measure and counting measure.

Lebesgue measure is like length in one dimension, area in two dimensions, volume in three dimensions, and so on. It will be denoted with ℓ here. The following is a nice fact about Lebesgue measure.

Fact 5
For $A \subset \mathbb{R}^n$, if the Riemann integral

$$\int_{x \in A} 1 \, dx = \int_{x \in \mathbb{R}^n} \mathbb{I}(x \in A) \, dx$$

exists, then this equals the Lebesgue measure of A, $\ell(A)$.

Example 8
Find the Lebesgue measure of the interval $[3, 6]$.

Answer Because the interval is one dimensional (so $[3, 6] \subseteq \mathbb{R}$), the Lebesgue measure is the length of the interval $6 - 3 = 3$. Alternatively, we could have used the integral to get the same result:

$$\ell([3, 6]) = \int_3^6 1 \, dx = x|_3^6 = 6 - 3 = \boxed{3}.$$

Counting measure (often denoted $\#$) just counts the number of elements in a set. So $\#\{a, b, c\} = 3$ and $\#(\emptyset) = 0$. Lebesgue measure of a continuous set is found by using Riemann integrals over that set of the constant 1 function. Similarly, counting measure equals a sum over the elements of a set of the number 1.

Fact 6
For a set A,

$$\#(A) = \sum_{a \in A} 1 = \sum_a \mathbb{I}(a \in A).$$

Example 9
Find $\#(\{a, b, c\})$.

Answer The sum form is

$$\begin{aligned} \#(\{a, b, c\}) &= \sum_i \mathbb{I}(i \in \{a, b, c\}) \\ &= \mathbb{I}(a \in \{a, b, c\}) + \mathbb{I}(b \in \{a, b, c\}) + \mathbb{I}(c \in \{a, b, c\}) \\ &= 1 + 1 + 1 = \boxed{3}. \end{aligned}$$

By combining these facts and generalizing to integrands more complicated than indicator functions, a general integral can be defined.

Fact 7
The integral

$$\int_x g(x)\, d\mu$$

equals $\int_x g(x)\, dx$ when μ is Lebesgue measure and the Riemann integral exists, and it equals $\sum_x g(x)$ when μ is counting measure.

Problems

3.1: Find the following.

a) $\#(\{a, b, c, d\})$.

b) $\#(\{2, 4, 6, \ldots, 100\})$.

c) $\ell([3, 6])$.

d) $\ell((-16, 16))$.

3.2: Find the following.

a) $\#(\{3, 6, 9, \ldots, 100\})$

b) $\ell([6.5, 2.1])$

c) $\ell([0, x])$

3.3: For W with density $12s^2(1 - s)\mathbb{I}(s \in [0, 1])$, what is $\mathbb{P}(W \geq 1/2)$?

3.4: For Y with density $4\exp(-4s)\mathbb{I}(s \geq 0)$, what is $\mathbb{P}(Y \in [-2, 2])$?

3.5: Given X has unnormalized density $g(s) = \exp(-3s)\mathbb{I}(s \geq 0)$, find the normalized density of X.

3.6: For X with unnormalized density $g(s) = \mathbb{I}(s \in [-3, 3])$, what is the normalized density?

Chapter 4

Importance Sampling

Question of the day

Estimate

$$I = \int_0^\infty \sqrt{x} \exp(-x)\, dx$$

using random draws from $X \sim \mathsf{Exp}(1)$.

Summary

Suppose that $I = \int_A g(x)\, d\mu$ and that X is a random variable with density f_X which is positive for all $x \in A$ where $g(x) \neq 0$. Let

$$h(x) = \frac{g(x)}{f_X(x)}.$$

- $\mathbb{E}[h(X)] = I$.
- Using iid draws from X in order to estimate I is called **importance sampling**, or IS for short.
- For a random variable X with mean a, and $\hat{a}$ the sample average of n iid draws from X, the standard deviation of $\hat{a}$ is $\mathrm{SD}(X)/\sqrt{n}$.
- The value of $\mathrm{SD}(X)$ can be estimated with the **sample standard average**.

Recall that $X \sim \mathsf{Exp}(1)$ if it has density $f_X(x) = \exp(-x)\mathbb{I}(x \geq 0)$. Also remember that for a function $h(X)$, the expected value can be found using:

$$\mathbb{E}[h(X)] = \int_{\mathbb{R}} h(x) f_X(x)\, dx = \int_{\mathbb{R}} h(x) \exp(-x)\mathbb{I}(x \geq 0)\, dx = \int_0^\infty h(x) \exp(-x)\, dx.$$

So the limits of the integral for $h(X)$ match the integral in the Question of the day. The only question is how to choose the function $h(x)$ to match the integral value. This is a simple algebra problem.

$$h(x)\exp(-x) = \sqrt{x}\exp(-x) \Rightarrow h(x) = \sqrt{x}.$$

In other words, for $X \sim \text{Exp}(1)$,

$$\mathbb{E}[\sqrt{X}] = \int_0^\infty \sqrt{x}\exp(-x)\,dx = I.$$

This integral value I turns out to be $0.8862269\ldots$. Using R to estimate the integral with the basic Monte Carlo method can be done as follows.

```
> results <- rexp(10^6, 1)
> mean(sqrt(results))
[1] 0.8858147
```

For a target integral value I and random variable X, this method of finding a function h such that $\mathbb{E}[h(X)] = I$ is called *importance sampling*.

An important note is that importance sampling can only be used when the density f_X is positive whenever the integrand is positive.

Definition 10
Suppose that X is a random variable with density $f_X(x)$ such that $g(x) > 0$ implies $f_X(x) > 0$. Set

$$h(x) = \frac{g(x)}{f_X(x)},$$

when $f_X(x) > 0$, and to o otherwise. Then $\mathbb{E}[h(X)] = I$, where

$$I = \int_x g(x)\,dx \quad \text{or} \quad I = \sum_i g(i).$$

The technique of generating $X_1, \ldots, X_n$ iid X and then setting $\hat{a} = \bar{X} = (X_1 + \cdots X_n)/n$ is called **importance sampling**, or IS for short.

Example 10
Use IS to estimate

$$I = \int_0^\infty \exp(-x^{1.2})\, dx,$$

using exponential random variables with rate 1.

Answer An exponential random variable of rate 1 (write $X \sim \text{Exp}(1)$) has density $f_X(x) = \exp(-x)\mathbb{I}(x \geq 0)$. This means that

$$\frac{\exp(-x^{1.2})\mathbb{I}(x \geq 0)}{\exp(-x)\mathbb{I}(x \geq 0)} = \exp(-[x^{1.2} - x])\mathbb{I}(x \geq 0).$$

So for $X_1, X_2, \ldots, X_n$ iid $\text{Exp}(1)$,

$$\hat{I} = \frac{\exp(-[X_1^{1.2} - X_1]) + \cdots + \exp(-[X_n^{1.2} - X_n])}{n}$$

is an importance sampling estimate of I.

Example 11
How can I estimate $S = \sum_{i=1}^6 \sqrt{i}$ using draws from $X \sim \text{Unif}(\{1, 2, 3, 4, 5, 6\})$?

Answer Here the density of X (with respect to counting measure) is $f_X(i) = (1/6)\mathbb{I}(i \in \{1, \ldots, 6\})$. So $h(i) = 6\sqrt{i}$ works for the importance sampling function. Drawing $X_1, \ldots, X_n$ iid $\text{Unif}(\{1, \ldots, 6\})$ then yields

$$\hat{S} = 6(\sqrt{X_1} + \cdots + \sqrt{X_n})/n$$

as an estimate of S.

4.1 Error in the method

Recall the basic Monte Carlo method for estimating a value a.

1: Create a random variable X such that $\mathbb{E}[X] = a$.

2: Draw $X_1, X_2, \ldots, X_n$ iid from X.

3: Then $\hat{a} = (X_1 + \cdots + X_n)/n$ is an estimate of a.

The Strong Law of Large numbers guarantees that $\hat{a}$ will converge to a as n goes to infinity, what it does not say is how quickly that convergence occurs.

One common way of measuring how far a random variable is from its mean is the standard deviation of the random variable. Unfortunately, not all random variables have standard deviations! But if they do, then they obey certain rules that will be helpful here. The first rule tells us how the standard deviation scales.

Fact 8
Suppose X has standard deviation $\mathrm{SD}(X)$. Then for any $n > 0$,

$$\mathrm{SD}(X/n) = \mathrm{SD}(X)/n.$$

The second rule tells us how standard deviations add.

Fact 9
Suppose $X_1, \ldots, X_n$ are independent random variables with finite standard deviations.

$$\mathrm{SD}(X_1 + \cdots + X_n) = \sqrt{\mathrm{SD}(X_1)^2 + \cdots + \mathrm{SD}(X_n)^2}.$$

Combining these facts tells us how far away our estimate is from the true answer.

Fact 10
Suppose that $\mathbb{E}[X] = a$, $X_1, \ldots, X_n$ are iid X and $\hat{a} = (X_1 + \cdots + X_n)/n$. Then $\mathrm{SD}(\hat{a}) = \mathrm{SD}(X)/\sqrt{n}$.

Proof. Using our two rules

$$\begin{aligned}\mathrm{SD}(\hat{a}) &= \frac{1}{n}\mathrm{SD}(X_1 + \cdots + X_n) \\ &= \frac{\sqrt{\mathrm{SD}(X_1)^2 + \cdots + \mathrm{SD}(X_n)^2}}{n} \\ &= \frac{\sqrt{n\,\mathrm{SD}(X)^2}}{n} = \frac{\mathrm{SD}(X)}{\sqrt{n}}.\end{aligned}$$

□

This says that the error in a Monte Carlo simulation goes down as the square root of the number of samples. This is extremely slow! For comparison, the error in the trapezoidal method for estimating the value of a one dimensional integral goes down as the *square* of the number of samples. People use Monte Carlo methods not because the error rate is good (it's not) but because the same error rate applies, regardless of what dimension we are in.

4.2 Estimating the mean and standard deviation

In practical problems, it is a harder problem to calculate the error than the original integral! So instead, try to estimate the standard deviation from the data. Suppose that $X_1, \ldots, X_n$ is a set of iid draws from X.

The *sample standard deviation* is a formula from statistics:

$$\hat{\sigma} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2}.$$

For a vector of data `x` in R, this sample standard deviation can be computed with the **`sd`** command.

```
sd(x)
```

Recall that in R a command like **`rexp`**`(30, 2)` will generate 30 iid draws from an $\text{Exp}(2)$ distribution. Here the **`rexp`** function has a built-in parameter that gives the number of samples.

However, many functions in R have no such parameter. In this case, the `replicate` command can be used to repeatedly call a function.

For instance, the following function rolls three fair six sided dice, and adds the result together.

```
monte_carlo_draw <- function() return(sum(floor(runif(3) * 6)
    + 1))
```

The following then independently calls this function 1000 times. The resulting vector can then be used as part of a sample average to estimate the mean and standard deviation of the resulting estimate.

```
x <- replicate(1000, monte_carlo_draw())
mean(x)
sd(x) / sqrt(length(x))
```

Note that **`sd`**`(x)` gives an estimate of the standard deviation of x. So for the standard deviation of the sample average, we divide by $\sqrt{n}$. That is, the error in **`mean`**`(x)` is **`sd`**`(x)/sqrt(length(x))`. We write the final result as

$$0.3357 \pm 0.09367$$

Of course, 0.09367 gives the impression that this estimate is far more accurate than it really is. Rarely is more than the first digit able to be kept in this estimate. A better way of representing it is:

$$0.34 \pm 0.09.$$

In the next section we will show how to calculate the standard deviation exactly, and show how a proper choice of importance sampling function is essential to keeping the error small.

Problems

4.1: Suppose we want to estimate

$$I = \int_{x \in [0,2]} 3x^2 \, dx.$$

a) Using $U_2 \sim \text{Unif}([0,2])$, find h_2 such that $\mathbb{E}[h_2(U_2) = I]$.

b) Transform the integral using $s = x/2$ into an integral over $[0, 1]$, and then find h_1 such that $\mathbb{E}[h_1(U_1)] = I$ where $U_1 \sim \mathsf{Unif}([0, 1])$.

4.2: Suppose we want to estimate

$$S = \sum_{i=1}^{100} i^2.$$

Create a random variable X and function h such that $\mathbb{E}[h(X)] = S$.

4.3: Consider the integral

$$I = \int_{\mathbb{R}} \frac{1}{1 + x^4}\, dx.$$

a) Say $Y \sim \mathsf{Cauchy}$, so $f_Y(s) = [(\tau/2)(1 + s^2)]^{-1}$. Find a function $h(Y)$ such that $\mathbb{E}[h(Y)] = I$.

b) Use Wolfram Alpha to find the maximum value of $h(Y)$ for $Y \in \mathbb{R}$.

4.4: For the integral

$$I = \exp(-|x|),$$

given that X is a standard Cauchy random variable, find h such that $\mathbb{E}[h(Y)] = I$.

4.5: Suppose $\mathrm{SD}(X) = 3.2$ and $X_1, \ldots, X_{10}$ are iid as X. What is the standard deviation of

$$\frac{X_1 + \cdots + X_{10}}{10}?$$

4.6: Suppose that Y has standard deviation 4.2. What would be the standard deviation of the sample average of 24 iid draws from Y?

4.7: Estimate $\int_0^\infty \exp(-x^{3/2})\, dx$ using $T \sim \mathsf{Exp}(1)$ and importance sampling with 1000 samples.

4.8: Estimate $\int_0^\infty \exp(-2\sqrt{x})\, dx$ using $T \sim \mathsf{Exp}(1)$ and importance sampling with 1000 samples.

4.9: Write R code to estimate $\int_0^1 \exp(\sqrt{x})\, dx$ using 1000 draws. Estimate the error and report your answer in the form $a \pm b$.

4.10: Write R code to estimate $\int_0^1 1/(1 + \sqrt{x})\, dx$ using 1000 draws. Estimate the error and report your answer in the form $a \pm b$.

Chapter 5

Calculating the standard deviation exactly

Question of the day

Consider

$$I = \int_0^\infty \sqrt{x} \exp(-x)\, dx.$$

Let $X \sim \mathsf{Exp}(1)$ and $Y \sim \mathsf{Exp}(2)$. Then

$$\mathbb{E}[\sqrt{X}] = \mathbb{E}[\sqrt{Y} \exp(Y)] = I$$

give us two ways of using importance sampling to find I. Which of these ways is better?

Summary

- The **Law of the Unconscious Statistician** is

$$\mathbb{E}[h(X)] = \int_s h(s) f_X(s)\, d\mu.$$

- The **n-th moment** of a random variable X is $\mathbb{E}[X^n]$.
- The **variance** of a random variable X is the second moment minus the square of the first moment.
- The **standard deviation** of a random variable X is the square root of the variance.

Last time some of the rules for standard deviations were discussed. Here standard deviation will be defined precisely, and methods for calculating it exactly will be given. Start by showing how a mean of a function of a random variable can be turned into an integral.

Theorem 2 (Law of the unconscious statistician)
Suppose X has density $f_X(x)$ with respect to μ. Then

$$\mathbb{E}[h(X)] = \int_s h(s) f_X(s)\, d\mu.$$

This is called the Law of the Unconscious Statistician because the user just replaces each instance of X inside the expectation with an instance of s inside the integral without thinking about it. Consider the following example.

Example 12
If X has density f_X with respect to μ, what is $\mathbb{E}[X/(1+X)]$?

Answer By the Law of the Unconscious Statistician, this is

$$\mathbb{E}\left[\frac{X}{1+X}\right] = \int_{s\in\mathbb{R}} \frac{s}{1+s} \cdot f_X(s)\, ds.$$

Here the $X/(1+X)$ inside the mean became $s/(1+s)$ inside the integral. Multiply by the density of X, and integrate over all possible s, and you have written the mean as an integral!

A common way of measuring how far a random variable is from its mean is to use the *standard deviation*. This can be written in terms of the *moments* of a random variable.

Definition 11
The **ith moment** of a random variable is $\mathbb{E}[X^i]$.

If the first and second moments of a random variable exist, then the random variable has a *variance*.

Definition 12
A random variable X has **variance**

$$\mathbb{V}(X) = \mathbb{E}[(X - \mathbb{E}(X))^2].$$

When $\mathbb{V}(X) = \infty$, say the variance is infinite or does not exist.

Because of the square, variance will have units that are the square of the units of X. To get this back down to the same units, we take the square root.

Definition 13
The square root of the variance of a random variable is the **standard deviation**. That is,

$$\mathrm{SD}(X) = \sqrt{\mathbb{V}(X)}.$$

There is a formula for the variance that is somewhat easier to use in practice.

Fact 11
Suppose X has finite first and second moments. Then

$$\mathbb{V}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2.$$

Example 13
Find the variance of X with density $f_X(s) = 3\exp(-3s)\mathbb{I}(s \geq 0)$.

Answer Since the density is positive over $[0, \infty)$, we know that the density is with respect to Lebesgue measure. Therefore,

$$\mathbb{E}[X] = \int_{\mathbb{R}} s \cdot 3\exp(-3s)\mathbb{I}(s \geq 0)\, ds = \int_0^\infty 3s\exp(-3s)\, ds = 1/3$$
$$\mathbb{E}[X^2] = \int_{\mathbb{R}} s^2 \cdot 3\exp(-3s)\mathbb{I}(s \geq 0)\, ds = \int_0^\infty 3s^2\exp(-3s)\, ds = 2/9$$

Hence the variance is $(2/9) - (1/3)^2 = 1/9 = \boxed{0.1111\ldots}$.

Recall that the expectation operator is linear, meaning that when we shift and scale the random variable, we shift and scale the mean by the same amount:

$$\mathbb{E}[c_1 X + c_2] = c_1\mathbb{E}[X] + c_2.$$

Variance and standard deviation are **not** linear operators, but the effects of scaling and shifting on their values are known.

Fact 12
For a random variable X where c_1 and c_2 are real numbers

$$\mathbb{V}(c_1 X + c_2) = c_1^2\mathbb{V}(X)$$
$$\mathrm{SD}(c_1 X + c_2) = |c_1|\,\mathrm{SD}(X).$$

5.1 Calculating error for importance sampling

Suppose $f(x) > 0$ implies that $f_W(x) > 0$, then the importance sampling function h is

$$h(w) = \frac{f(w)}{f_W(w)}$$

where $f(w)$ is positive, and 0 if $f(w) = 0$.

Then the point of building the IS function h is that when it is applied to W, the mean of the resulting new random variable is exactly the integral being estimated.

$$\mathbb{E}[h(W)] = I = \int_{x:f_W(x)>0} \frac{f(x)}{f_W(x)} f_W(x)\,dx = \int_{\mathbb{R}} f(x)\,dx.$$

What about the standard deviation of $h(W)$? This will be

$$\mathrm{SD}(h(W)) = \sqrt{h(W)^2 - \mathbb{E}(h(W))^2} = \sqrt{h(W)^2 - I^2}.$$

There is nothing that can be done about the I^2 part, that is fixed, so the error is controlled by the size of the second moment of $h(W)$. That is, $\mathbb{E}(h(W)^2)$ controls how big the error will be. If this is close to I^2, then the error is small, but if it is larger, that spells trouble. Note

$$\mathbb{E}[h(W)^2] = \int_{x:f_W(x)>0} \left(\frac{f(x)}{f_W(x)}\right)^2 f_W(x)\,dx = \int_{\mathbb{R}} \frac{f(x)^2}{f_W(x)}\,dx$$

This means that if the density $f_W(x)$ is small compared to $f(x)$, then $f(x)/f_W(x)$ will be very large. It is even possible that if W has too small a density, that $\mathbb{E}[h(W)^2] = \infty$.

To understand how this works with IS, consider the question of the day. This problem gave two random variables (X and Y) from different distributions ($\mathsf{Exp}(1)$ and $\mathsf{Exp}(2)$) for use in estimating an integral.

The density for X is $\exp(-x)\mathbb{I}(x \geq 0)$, so

$$h_1(x) = \frac{\sqrt{x}\exp(-x)\mathbb{I}(x \geq 0)}{\exp(-x)\mathbb{I}(x \geq 0)} = \sqrt{x}\mathbb{I}(x \geq 0).$$

The density for Y is $2\exp(-2x)\mathbb{I}(x \geq 0)$, so

$$h_2(x) = \frac{\sqrt{x}\exp(-x)\mathbb{I}(x \geq 0)}{2\exp(-2x)\mathbb{I}(x \geq 0} = (1/2)\sqrt{x}\exp(x)\mathbb{I}(x \geq 0).$$

In both cases, the mean of the resulting random variables is I.

$$\mathbb{E}[h_1(X)] = \mathbb{E}[h_2(Y)] = I.$$

However, the standard deviations are very different!

$$\mathbb{E}[h_1(X)^2] = \mathbb{E}[(\sqrt{X})^2] = \mathbb{E}[X] = \int_{x\geq 0} x\exp(-x)\,dx = 1,$$

while

$$\mathbb{E}[h_2(Y)^2] = \mathbb{E}[(\sqrt{Y}\exp(Y))^2] = \mathbb{E}[Y\exp(2Y)]$$

Which evaluates to

$$\int_{x\geq 0} x\exp(2x)\exp(-x)\,dx = \int_{x\geq 0} x\exp(x)\,dx = \infty.$$

So the standard deviation using Y will be infinity! The reason it is better to use X than Y here is that the density of Y $\exp(-2x)$ goes to zero much faster than the integrand $\sqrt{x}\exp(-x)$.

Another way of thinking about this: Since Y has rate 2, it rarely samples from large values like 4.1. When it does, the value has to be weighted to $\sqrt{4.1}\exp(4.1) = 122.17\ldots$. So we have a small chance of seeing large values that affect the average a lot. That's why the variance is infinite.

The best random variables for IS are ones where the tail goes to zero at roughly the same rate as the integrand that we are targeting. The closer we get to the true answer, the less variance we will have in our result.

Problems

5.1: Let $Y \sim \text{Exp}(2)$.

a. Set up the integral for the fourth moment of Y.

b. Solve the integral using WolframAlpha.

5.2: Suppose X has density

$$\frac{8}{\sqrt{\tau}}\sqrt{x}\exp(-x)\mathbb{I}(x \geq 0).$$

a. Set up the integral for the fifth moment of X.

b. Solve the integral using WolframAlpha.

5.3: Suppose X has mean 4.1 and standard deviation 1.2, and $X_1, \ldots, X_{10}$ are iid X. Let $Y = (X_1 + \cdots + X_{10})/10$

a) Find $\mathbb{E}[Y]$.

b) Find $\text{SD}[Y]$.

5.4: Let W have mean -2.3 and standard deviation of 4. For $W_1, \ldots, W_{12}$ iid W,

a) Find $\mathbb{E}[W]$.

b) Find $\text{SD}(W)$.

5.5: Let $U \sim \text{Unif}([0,1])$ (so it has density $f_U(u) = \mathbb{I}(u \in [0,1])$.) Find the ith moment of U.

5.6: Find the variance of $U \sim \text{Unif}([0,1])$.

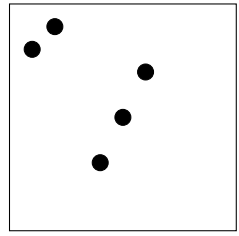
Chapter 6

Estimating Probabilities

Question of the day

Suppose bacteria are growing in a petri dish at the following locations in a petri dish normalized to lie in a unit square:

$$(0.4, 0.3), (0.5, 0.5), (0.1, 0.8), (0.2, 0.9), (0.6, 0.7)$$



Is there enough evidence to reject the null hypothesis that the bacteria locations are independently uniform over the dish?

Summary

Probabilities are just the expected value of indicator functions. To find $\mathbb{P}(X \in A)$, use

$$\mathbb{P}(X \in A) = \mathbb{E}[\mathbb{I}(X \in A)].$$

An example of where Monte Carlo is used to find probabilities is in p-values.

- Given a set of data, a **null hypothesis** is a statistical (probabilistic) model of how the data was generated.
- A **test statistic** is a function of the data that can be used to determine if a set of data comes from a model.
- The p-**value** for a particular dataset, test statistic, and statistical model is the probability that a draw from the statistical model is greater than the test statistic applied to the data.

Suppose the goal is to estimate the probability that an event occurs. For instance, suppose I want to know $\mathbb{P}(X \leq 4)$ where $X \sim \text{Unif}(\{1, 2, 3, 4, 5, 6\})$.

Suppose the die is rolled eight times, with the result:

$$4, 4, 3, 5, 5, 6, 2, 3.$$

Replace each value with a 1 if it is at most 4, and a 0 otherwise. The sequence is then

$$1, 1, 1, 0, 0, 0, 1, 1.$$

The sample average of these numbers is $5/8$, which is an estimate of the probability that the roll was at most 4.

Recall that we can use the indicator function to turn our X_i values in $\{1, 2, \ldots, 6\}$ into 1's and 0's. If we make

$$B_i = \mathbb{I}(X_i \leq 4),$$

then

$$(X_1, \ldots, X_8) = (4, 4, 3, 5, 5, 6, 2, 3)$$

makes

$$(B_1, \ldots, B_i) = (1, 1, 1, 0, 0, 0, 1, 1).$$

A random variable that is either 0 or 1 is called a *Bernoulli* random variable.

Definition 14
If a random variable $B \in \{0, 1\}$, then say that B has a **Bernoulli distribution** with parameter p (write $B \sim \text{Bern}(p)$) if $\mathbb{P}(B = 1) = p$.

This arises, for instance, in the problem of finding *p-values* in frequentist statistics.

6.1 Calculating a test statistic

Now consider the question of the day. The goal is to test some spatial data to see if a particular model is reasonable. The statistical model in the question to be tested says that the points are iid drawn uniformly from the unit square. This statistical model is called the *null hypothesis*. In Fisher-style frequentist statistics, the plan is to use the data to decide if there is enough evidence to reject the null hypothesis. To test if this is true, it is necessary to have something called a *test statistic*.

Definition 15
A **test statistic** is any function of the data which is used to determine if there is enough evidence to reject a hypothesis.

In the case of the Question of the Day, there are several ways to construct a test statistic. One simple way is to just add the distances between all pairs of points. Call this test statistic S. If the points in $\mathbb{R}^2$ are $p_1, \ldots, p_n$, then the test statistic can be described as:

$$S = \sum_{p_i, p_j} \text{dist}(p_i, p_j)$$

In R, the first step is to input the data. The *combine* command, `c`, can be used to turn values into vectors. For instance, the x and y coordinates can be input into R as follows.

```
x1 <- c(0.4, 0.5, 0.1, 0.2, 0.6)
x2 <- c(0.3, 0.5, 0.8, 0.9, 0.7)
```

These can then be combined into a *matrix* of values with the `cbind` command.

```
cbind(x1, x2)
```

The matrix created

$$\begin{matrix} 0.4 & 0.3 \\ 0.5 & 0.5 \\ 0.1 & 0.8 \\ 0.2 & 0.9 \\ 0.6 & 0.7 \end{matrix}$$

has 2 columns and 5 rows.

To use a function on every row of the matrix, the `apply` command can be used. Consider the following:

```
apply(cbind(x1, x2), 1, sum)
```

which returns

```
0.7 1.0 0.9 1.1 1.3
```

The second parameter value 1 in `apply` indicates that the function should be applied to every row. So these five numbers are the sums of the two columns for each of the five rows.

If instead the function is to be applied to every column, use 2 as the second parameter value.

The function `apply` can be used with a custom function. The following returns `x1 - x2` for each row of the matrix.

```
apply(cbind(x1, x2), 1, function(x) return(x[1] - x[2]))
```

which returns

```
0.1  0.0 -0.7 -0.7 -0.1
```

To calculate the test statistic, it is necessary to calculate the distance between a point and all the points recorded as rows of the matrix. The distance between two points is

$$\text{dist}((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

The following function does this computation.

```
sumdist <- function(pt, mat) {
  return(
    sum(
      apply(mat, 1, function(x) return(sqrt((x[1] - pt[1])^2 +
                                            (x[2] - pt[2])^2))
      )
    )
  )
}
```

Test this out:

```
A <- cbind(x1, x2)
sumdist(c(0.4, 0.3), A)
```

which returns

```
1.886371
```

as the answer.

So now, use this function with `apply` to get the sum over *all* pairs of points. Since `sumdist` takes two arguments, it is also necessary to tell it that the `mat` argument is `A`.

```
sum(apply(A, 1, sumdist, mat = A))
```

which returns

```
8.41703
```

There's still one problem: this adds up all pairs of points twice, so it is necessary to divide by two to get our test statistic. Putting it all together gives the following function.

```
testS <-
  function(A) return(sum(apply(A, 1, sumdist, mat = A)) / 2)
```

Then

```
  testdata <- testS(cbind(x1, x2))
  testdata
```

returns

```
4.208515
```

which is the value of our test statistic for this data.

6.2 Estimating a p-value

The next question is what should be done with this number? Is 4.208515 large? Is it small? One way to measure this is with a p-value, which is the probability that a draw from the statistical model has test statistic value greater than the test statistic applied to the actual data.

Definition 16
The **p-value** for a set of data, a statistical model, and a test statistic is the probability that the test statistic applied to a draw from the statistical model is greater than the test statistic applied to the data.

If this p-value is small, then that is typically taken as evidence *against* the statistical model being correct.

To understand this, one way to start is to look at what the distribution of the sum of these distances looks like.

Suppose the model that we are trying to test is that the points are uniformly distributed over the unit square. Then we can draw points P uniformly from the unit square and then calculate the sum of the distances between them.

```
generateS <- function() {
  # Draw 5 uniform random points
  u1 <- runif(5)
  u2 <- runif(5)
  # Find their S value
  return(testS(cbind(u1, u2)))
}
```

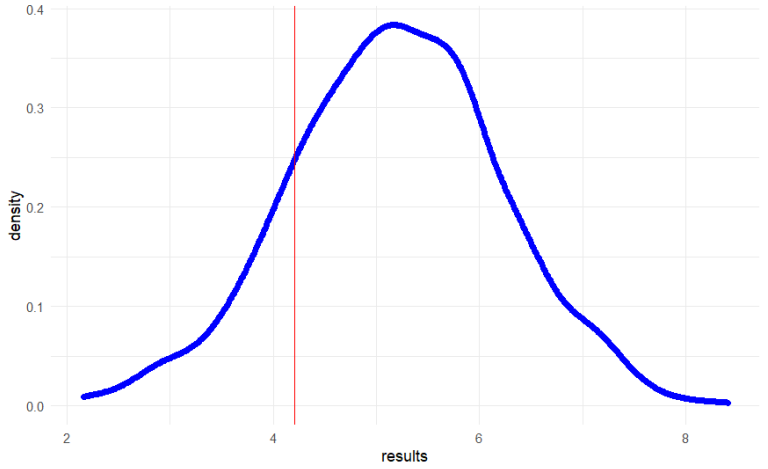
Now generate many iid draws from the distribution of the test statistic and use the kernel density estimate to get an idea of how the test statistic behaves under the null hypothesis.

```
n <- 1000
results <- replicate(n, generateS() > testdata)
mean(results)
sd(results) / sqrt(results)
```

Often the null hypothesis is rejected when the p-value is below 5%. Here the p-value is about 83%, so there is not enough evidence to indicate that the data does not come from the null hypothesis.

A plot of these results can be made as follows.

```
ggplot() +
  geom_density(aes(results), color = "blue", lwd = 2) +
  theme_minimal() +
  geom_vline(aes(xintercept = testdata), color = "red")
```



Then the p-value estimate is the percentage of values that falls to the right of the red line.

6.3 Another example

Suppose that the statistical model for values W_1, W_2, W_3, W_4 is that they are iid uniform over $[0, 1]$. Since each has average value 0.5, a test statistic might be the sum of the absolute values of the difference between the value and 0.5.

A function to calculate this test statistic for a vector `w` is as follows.

```
testW <- function(w) return(sum(abs(w - 0.5)))
```

For data $(0.3, 0.9, 0.1, 0.2)$ the test statistic is found by:

```
w <- c(0.3, 0.9, 0.1, 0.2)
testdata <- testW(w)
```

The p-value could then be estimated as:

```
n <- 1000
results <- replicate(n, testW(runif(4)) > testdata)
mean(results)
sd(results) / sqrt(length(results))
```

The result of my run was about 0.15 ± 0.1.

6.4 Monte Carlo for big data

Suppose that I have a quadrillion data points,

$$X_1, X_2, \ldots, X_{10^{15}},$$

all of which are real numbers between 0 and 1 inclusive, and I am interested in knowing what percentage of them are greater than 0.6. Call this percentage p.

Now p could be found exactly:

$$p = \frac{1}{10^{15}} \sum_{i=1}^{10^{15}} \mathbb{I}(X_i > 6).$$

Of course, if the data is spread out over secondary storage, this might take an unacceptably long time.

Consider drawing $I \sim \text{Unif}(\{1, 2, \ldots, 10^{15}\})$, then

$$p = \mathbb{E}[\mathbb{I}(X_D > 0.6)].$$

This allows tight estimation of Big Data values without pulling out every item.

Problems

6.1: Write a Monte Carlo algorithm in R to estimate $\mathbb{P}(U_1 + \cdots + U_6 \geq 5)$ where the U_i are iid $\text{Unif}([0, 1])$.

6.2: Write code in R to estimate $\mathbb{P}(T_1 + \cdots + T_{10} < 4)$ where the T_i are iid $\text{Exp}(2)$ using the `rexp` function.

6.3: Suppose we model $U_1, \ldots, U_n$ given parameter θ as iid $\text{Unif}([0, \theta])$. Given $U_1, \ldots, U_n$, our test statistic is

$$T = \min\{U_1, \ldots, U_n\}$$

We are trying to test if $\theta = 10$, and we consider high values of T to be evidence against this hypothesis.

a) If $n = 8$, what is the p-value for a test statistic of 4? (Calculate this value exactly as a probability.)

b) Use R to test your last answer to the last part by writing a function to generate variates from T, and then use Monte Carlo to estimate the p-value.

6.4: Suppose we model $T_1, \ldots, T_n$ given parameter λ as iid $\text{Exp}(\lambda)$. Given data $T_1, \ldots, T_n$, our test statistic is $S = T_1 + \cdots + T_n$. We are trying to test if $\lambda = 2$, and we reject if S is too small.

If $n = 20$, what is the p-value for a test statistic of 6.1.

Chapter 7

The Inverse Transform Method

Question of the day

Suppose X has density

$$f_X(s) = 2s\mathbb{I}(s \in [0,1]).$$

How do I draw random variates from this density?

Summary

- The **inverse transform method**, or ITM works as follows. Let $U \sim \mathsf{Unif}([0,1])$. Given a random variable X, let Y be the smallest value of a such that

  $$\mathbb{P}(X \leq a) \geq U.$$

  Then Y has the same distribution as X.

- The **pseudoinverse** of a cdf F_X is

  $$F_X^{-1}(b) = \inf\{a : F_X(a) \geq b\}.$$

- If X is a real-valued random variable, then $X \sim F_X^{-1}(U)$ where $U \sim \mathsf{Unif}([0,1])$.

7.1 Introduction

Consider how to generate a random variable X that comes from a particular distribution. First consider how to deal with one dimensional random variables. Call these 1-D distributions for short.

As usual, $U \sim \mathsf{Unif}([0,1])$ means that U is a uniform random variable over the interval $[0,1]$. It turns out that by taking a function and applying it to U, it is possible to generate draws from every real-valued random variable!

For example,

$$X = \mathbb{I}(U \in [0,1/3)) + 2\mathbb{I}(U \in [1/3,2/3)) + 3\mathbb{I}(U \in [2/3,1])$$

results in $X \sim \mathsf{Unif}(\{1, 2, 3\})$. Or

$$W = (1/2)\ln(1 - U)$$

results in $W \sim \mathsf{Exp}(2)$.

The above examples are instances of the *Inverse Transform Method*, which is a general way for turning a uniform random variable over $[0, 1]$ into any real-valued random variable.

7.2 Random variate generation on computers

If you are using any modern computing language, there are packages that will allow you to sample from the common distributions. For instance, R allows generation from many different distributions. The format is to use `r` followed by the distribution name. So for instance, `runif` draws from the continuous uniform distribution. Other examples include `rcauchy`, `rnorm`, `rbeta`, `rgamma`, `rexp`, and `rpois`

If it was necessary to work from scratch to generate random draws from every distribution, Monte Carlo simulation would be incredibly difficult. Fortunately, it turns out uniform random variables can be used to generate from all 1-D distributions. Every other distribution can be generated from by taking a computable function of a uniform random variable.

Every major language has a built-in function for generating uniforms over $[0, 1]$.

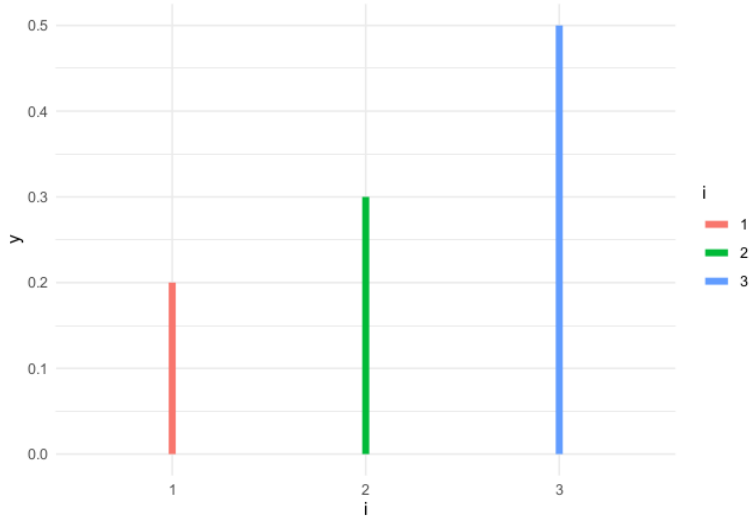
C/C++	`rand()`
R	`runif(1)`
MATLAB	`rand()` or `random()`
Python	`random()`

7.3 Inverse Transform Method for discrete random variables

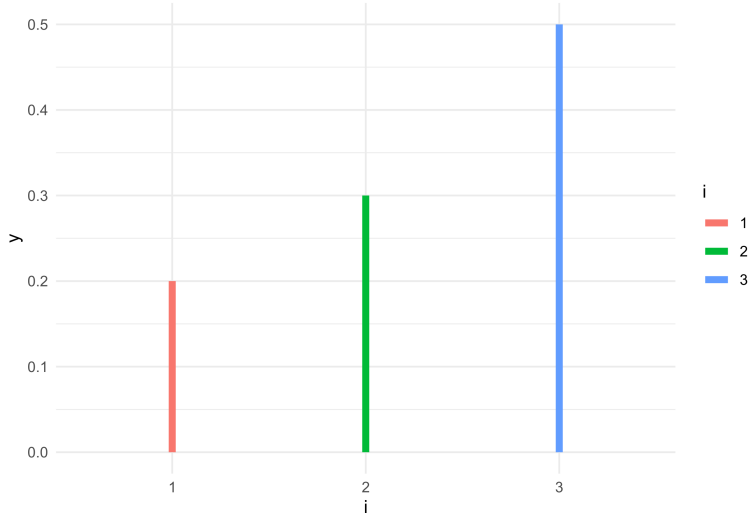
Consider the following distribution:

$$\mathbb{P}(X = 1) = 0.2,\ \mathbb{P}(X = 2) = 0.3,\ \mathbb{P}(X = 3) = 0.5.$$

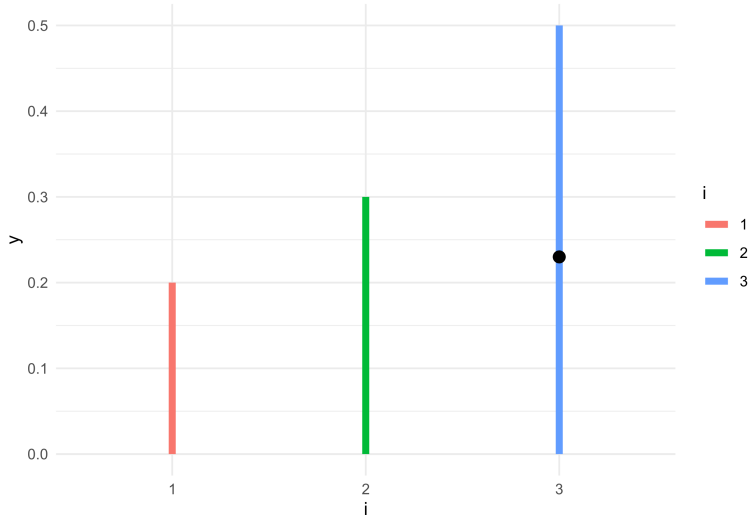
This can be represented pictorially using line segments.



Now suppose that a draw from X is made, and then given X, a draw from Y is made uniformly over the line segment associated with it. So $[Y|X] \sim \mathsf{Unif}([0, f_X(X)])$.



For instance, if $X = 3$, then Y is uniform from 0 to 0.5, and might be $Y = 0.23$, in which case the graph looks like:



Such an (X, Y) point will be uniform over the three line segments.

More generally, the following fact holds.

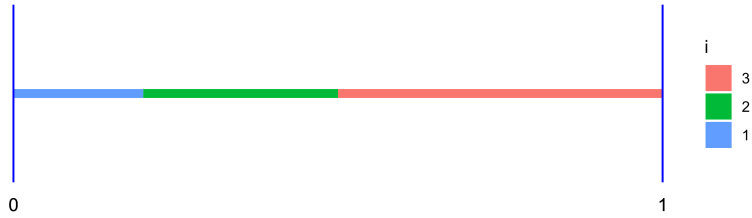
If X has density f_X, and $[Y|X] \sim \mathsf{Unif}([0, f_X(X)])$, then $(X, Y) \sim \mathsf{Unif}(\{(x, y) : 0 \leq y \leq f_X(x)\})$.

The joint density of (X, Y) will be

$$\begin{aligned} f_{(X,Y)}(x, y) &= f_X(x) f_{Y|X=x}(y) \\ &= f_X(x) \frac{1}{f_X(x)} \mathbb{I}(y \in [0, f_X(x)]) \\ &= \mathbb{I}(0 \leq y \leq f_X(x)), \end{aligned}$$

which is the uniform density over $\{(x, y) : 0 \leq y \leq f_X(x)\}$.

So how does that help us to generate from X? Well, suppose the three line segments were laid end to end:



Then it is easy to draw uniformly from the line segment of length 1, just draw U uniformly over $[0, 1]$. Then see which line segment the point lands on, and use that to figure out X.

This gives rise to the following function for generating variates from X.

$$h(U) = 1 \cdot \mathbb{I}(U \in [0, 0.2)) + 2 \cdot \mathbb{I}(U \in [0.2, 0.5)) + 3 \cdot \mathbb{I}(U \in [0.5, 1])$$

The generalization of this procedure for discrete random variables is called the *Inverse Transform Method*, and operates as follows.

1. Generate U uniformly over $[0, 1]$.
2. Return as output the smallest number a such that $\mathbb{P}(X \leq a) \geq U$.

7.4 Inverse Transform Method for continuous random variables

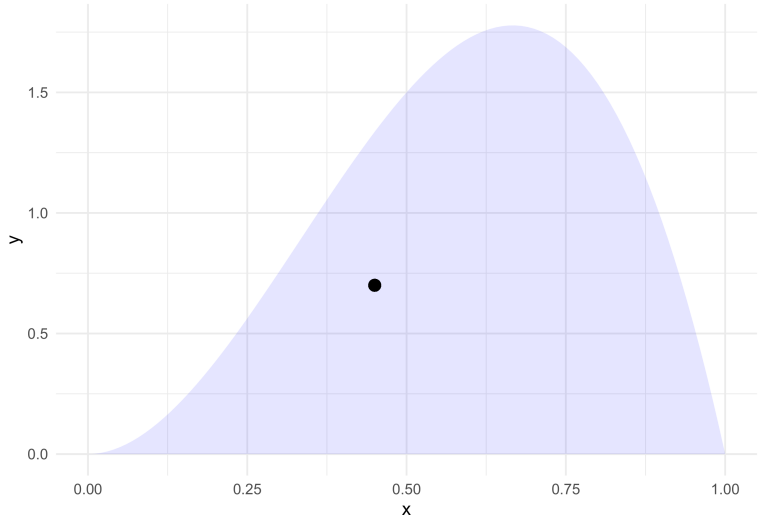
Okay, so that works if X is discrete, what about if X is a continuous random variable? Here is cool thing: the IVM still works exactly as defined above for continuous random variables!

To see why, first note that our first fact still holds. That is, for X with density f_X, and $[Y|X] \sim \mathsf{Unif}([0, f_X(X)])$, then (X, Y) is uniform over

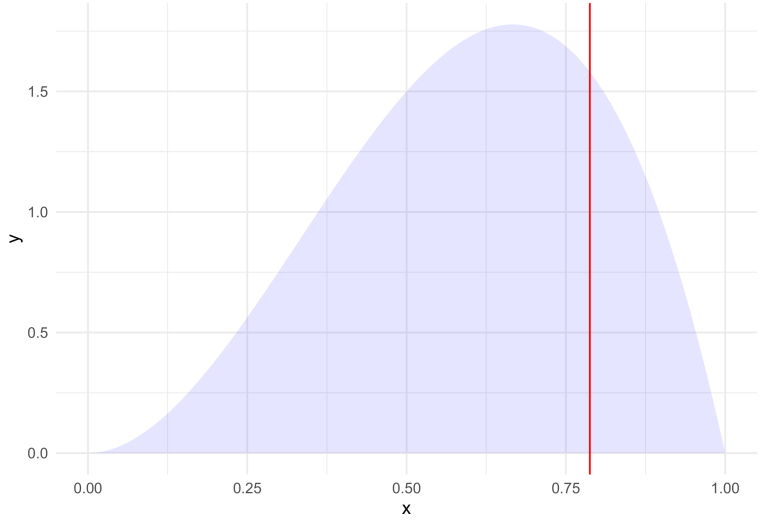
$$\{(x, y) : 0 \leq y \leq f_X(x)\}.$$

In other words, to draw a value X from a density, first draw (X, Y) from the area underneath the density, then throw away the Y coordinate to get the result.

For instance, if the density was $12x^2(1 - x)\mathbb{I}(x \in [0, 1])$, then the goal is to sample uniformly from the shaded region.



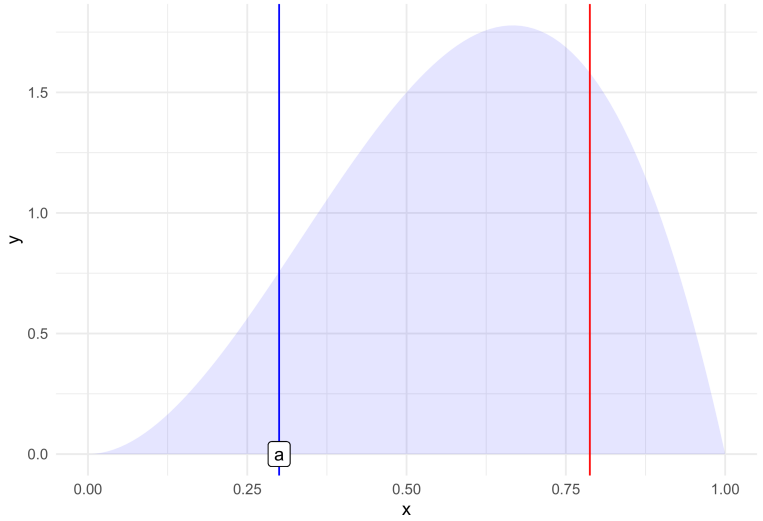
Because this is a density, the area of the shaded region under the density equals 1. In the next picture, 80% of the area under the shaded region is to left of the vertical red line.



By the way, the $x_{0.8} = 0.7876\ldots$ coordinate of this vertical line can be found by giving the following command to WolframAlpha at www.wolframalpha.com:

```
solve integral 12*x^2*(1-x) from 0 to b is equal to 0.8
```

Now suppose that $U \sim \mathsf{Unif}([0,1])$ is drawn, and a is the smallest number such that $\mathbb{P}(X \leq a) \geq U$. Because the density is continuous, it is possible to get $\mathbb{P}(X \leq a) = U$. Then the picture looks like this, where the area under the shaded region to the left of the blue vertical line is exactly U.



Note that $\mathbb{P}(a < x_{0.8}) = \mathbb{P}(U \leq 0.8) = 0.8$. Moreover, there was nothing really special about the value 0.8 there. So the way a was chosen, it has exactly the same distribution as X. That is, $a \sim X$. This means that the ITM works equally well for discrete and continuous random variables!

This can be used to answer the question of the day as follows.

Question of the Day

Note that for $a \in (0, 1)$,

$$\begin{aligned}\mathbb{P}(X \leq a) &= \int_{-\infty}^{a} f_X(s)\, ds \\ &= \int_{-\infty}^{a} 2s\mathbb{I}(s \in [0,1])\, ds \\ &= \int_{0}^{a} 2s\, ds = s^2|_0^a = a^2.\end{aligned}$$

So if $A^2 = U$ then $A = \sqrt{U}$.

So the algorithm for generating from density f_X is

1. Draw $U \leftarrow \mathsf{Unif}([0,1])$
2. Return $\sqrt{U}$

Formally, the theorem establishing the validity of the ITM can be stated as follows.

Theorem 3 (Inverse Transform Method)
Suppose $U \sim \mathsf{Unif}([0,1))$. Let A be the smallest value such that $\mathbb{P}(X \leq A) \geq U$. Then $X \sim A$.

Proof. The plan is to show that $(\forall a \in \mathbb{R})(\mathbb{P}(X \leq a) = \mathbb{P}(A \leq a))$. Start by proving that $(U \leq \mathbb{P}(X \leq a)) \Leftrightarrow (A \leq a)$.

Let $a \in \mathbb{R}$. Suppose $U \leq \mathbb{P}(X \leq a)$. Then A is the smallest number such that $U \leq \mathbb{P}(X \leq A)$. So $A \leq a$ in this case.

Now suppose that $A \leq a$. Then $(X \leq A) \to (X \leq a)$, so $\mathbb{P}(X \leq A) \leq \mathbb{P}(X \leq a)$. But $\mathbb{P}(X \leq A) \geq U$, so $U \leq \mathbb{P}(X \leq a)$.

We have shown that $A \leq a$ if and only if $U \leq \mathbb{P}(X \leq a)$. Taking the probability of these two expressions gives

$$\mathbb{P}(A \leq a) = \mathbb{P}(U \leq \mathbb{P}(X \leq a)) = \mathbb{P}(X \leq a).$$

(The last equality follows from the fact that for any $b \in [0, 1]$, $\mathbb{P}(U \leq b) = b$.) □

7.5 The pseudoinverse of a function

Another way to write the Inverse Transform method is using a *pseudoinverse* of the cdf of the random variable.

The **pseudoinverse** of a right continuous increasing function f is

$$f^{-1}(b) = \inf\{a : f(a) \geq b\}.$$

The $\inf$ stands for infimum, and means the best (largest) lower bound for a set of values. But you really need to know is that where the function f is strictly increasing, then f has a regular inverse and the pseudoinverse is the same function there. However, when f is flat or has jumps (like the cdf for discrete random variables), then the pseudoinverse is the location of the gap, or the smallest value of the flat part.

Consider the following examples.

- Say $F_X(a) = [1-\exp(-a)]\mathbb{I}(a \geq 0)$. Then if $a > 0$, then $F_X(a)$ is strictly increasing, so to the pseudoinverse is just the regular inverse:

$$1 - \exp(-a) = b \Rightarrow a = -\ln(1 - b).$$

$$\begin{aligned} F^{-1}(b) &= -\infty & b &< 0 \\ F^{-1}(b) &= -\ln(1 - b) & b &> 0. \end{aligned}$$

  Since $\mathbb{P}(U < 0) = 0$, ITM becomes,

$$-\ln(1 - U) \sim \mathsf{Exp}(1).$$

- Suppose

$$\mathrm{cdf}_W(a) = 0.3\mathbb{I}(a \geq 1) + 0.7\mathbb{I}(a \geq 2).$$

This cdf has a jump of size 0.3 at $a = 1$, and a jump of size 0.7 at $a = 2$. So the pseudoinverse is

$$F^{-1}(b) = 0 \text{ for } b < 0$$
$$F^{-1}(b) = 1 \text{ for } 0 \leq b < 0.3$$
$$F^{-1}(b) = 2 \text{ for } 0.3 \leq b < 1$$

Hence the ITM method for generating from W is to draw $U \sim \mathsf{Unif}([0,1])$, and use

$$W = 1 \cdot \mathbb{I}(U \in [0, 0.3)) + 2 \cdot \mathbb{I}(U \in [0.3, 1])$$

In terms of the pseudoinverse, the ITM theorem becomes the following.

Fact 13
For $U \sim \mathsf{Unif}([0,1])$, $\text{cdf}_X^{-1}(U) \sim X$.

7.6 The Fundamental Theorem of Simulation

The fact that a uniform can be transformed into any real-valued variable is called the *Fundamental Theorem of Simulation.*

Theorem 4 (Fundamental Theorem of Simulation)
Let $U \sim \mathsf{Unif}([0,1])$. Then for any computable random variable X there is a computable function g such that $X \sim g(U)$.

Why is this theorem so important to simulation as to be called fundamental? Because this theorem means that only the ability to generate uniforms over $[0,1]$ is truly necessary to generate from any distribution. It is nice to have functions like `rexp` and `rbeta`, but truly only `runif` is needed. That means that there needs to be only one method of generating pseudorandom variables for uniforms, and not a different method for every distribution.

Problems

7.1: Suppose $\mathbb{P}(A = -1) = 0.6$ and $\mathbb{P}(A = 1) = 0.4$. Find a function $h(u)$ such that $h(U) \sim A$ when $U \sim \mathsf{Unif}([0,1])$.

7.2: Consider the following random variable W:

i	$\mathbb{P}(W = i)$
1	0.6
2	0.2
3	0.2

Find a function $g(u)$ such that $g(U) \sim W$, when $U \sim \mathsf{Unif}([0,1])$.

7.3: For $U \sim \text{Unif}([-1,1])$, find the density of U^2.

7.4: Use the inverse transform method to find g such that $g(U)$ has density $(2/3)a^{-1/3}\mathbb{I}(a \in [0,1])$.

7.5: Find a function g such that $g(U)$ has density

$$f(s) = (1/s^2)\mathbb{I}(s \geq 1).$$

assuming $U \sim \text{Unif}([0,1])$.

7.6: Find a function g such that $g(U)$ (for $U \sim \text{Unif}([0,1])$) has density $f(s) = (1/4)s^3\mathbb{I}(s \in [0,2])$.

Chapter 8

Acceptance/Rejection

Question of the day

Suppose that an iid stream of random variables $X_1, X_2, \ldots$ that have the $\mathsf{Exp}(2)$ distribution. Create an algorithm for sampling from Y which has the distribution of X but conditioned to lie in $[1, 2]$. Write $Y \sim [X|X \in [1, 2]]$.

Summary

The **Acceptance Rejection** (AR for short) protocol is a method for drawing from distributions conditioned on having some property. Let $X \sim \mu$ have the desired distribution, and B be a set of values so that $X \in B$ is the desired property. Then

Acceptance-Rejection	*Input:* μ, B
1)	Repeat
2)	Draw $X \leftarrow \mu$
3)	Until $X \in B$
4)	Return X

has output $[X|X \in B]$.

Acceptance rejection (AR) is one of those algorithms that seems too good to be true, in that the algorithm is often the first thing tried for real problems.

For instance, suppose that you have access to a fair six sided die. That is, you can generate uniformly from $\{1, 2, 3, 4, 5, 6\}$ as many times as you like. Now suppose your goal is to draw uniformly from $\{1, 2, 3, 4\}$. A natural algorithm would be to roll the die. See if the roll falls into $\{1, 2, 3, 4\}$. If it does, return that roll as the final answer. If not, just reroll the die. This is the essence of AR.

In the question of the day, the user has access to a stream of random variables $X_1, X_2, \ldots$ such that each X_i is independent and has an $\mathsf{Exp}(2)$ distribution. The goal is to get a draw from $\mathsf{Exp}(2)$ conditioned to lie in $[1, 2]$.

An AR algorithm first draws X_1 and then checks its value. If X_1 is in B, stop and output X_1. (Say that the algorithm *accepts* X_1.) Otherwise, the algorithm *rejects* X_1 and then moves on to X_2. If X_2 is in B, stop and output X_2. Otherwise, reject and move to X_3, and so on.

There are several ways to write this algorithm in pseudocode. One way that does away with the subscripts on the X uses *recursion*.

Definition 17
A **recursive** algorithm calls itself while running (possibly with different parameters).

A classic example of a recursive algorithm is finding the factorial of a nonnegative integer. This factorial function is usually denoted as $n! = n \cdot (n-1) \cdot (n-2) \cdots 1$.

Factorial *Input:* n
1) If $n = 0$ then output 1
2) Else, output $n \cdot$ **Factorial**$(n-1)$

The recursive form of AR looks like this.

Acceptance-Rejection *Input:* μ, B
1) Draw $X \leftarrow \mu$
2) If $X \in B$, then return X
3) Else let $X \leftarrow$ **Acceptance-Rejection**, return X

Example 14
For $X \sim \mathsf{Exp}(2)$, generate from $[X|X \in [1,2]]$.

qotd
1) Draw $X \leftarrow \mathsf{Exp}(2)$
2) If $1 \le X \le 2$ then return X
3) Else $X \leftarrow$ **qotd**, return X

In practice, calling functions recursively is relatively slow because of the overhead of setting up a function in most computing languages. Therefore, *repeat* or *while* loops give an alternate way to write this pseudocode.

qotd_loop
1) Repeat
2) Draw $X \leftarrow \mathsf{Exp}(2)$
3) Until $1 \le X \le 2$
4) Return X

In R, this could be implemented in the following code.

```
qotd <- function() {
  # Draw from exponential of rate 2 conditioned to lie in 1, 2
  accept <- FALSE
  while (!accept) {
    x <- rexp(1, 2)
    accept <- (x >= 1) & (x <= 2)
  }
  return(x)
}
```

A quick test to see if it is working properly:

```
  results <- replicate(1000, qotd())
  mean(results)
  sd(results) / sqrt(length(results))
```

This returned 1.351 ± 0.009 when I ran it. The true expected value of this random variable is

$$\frac{\int x \exp(-2)\mathbb{I}(x \in [1,2])\, dx}{\int \exp(-2)\mathbb{I}(x \in [1,2])\, dx} = 1.34348$$

so it is pretty close.

The AR method works for both continuous and discrete problems.

Example 15

Suppose the user has the ability to draw $X \sim \text{Unif}(\{1,2,3,4,5,6\})$, and wants to generate a random variable $Y \sim \text{Unif}(\{1,2,3,4,5\})$. It is easy to check that $Y \sim [X|X \in \{1,2,3,4,5\}]$. Let $i \in \{1,2,3,4,5\}$. Then

$$\begin{aligned}\mathbb{P}(X = i|X \in \{1,2,3,4,5\}) &= \frac{\mathbb{P}(X = i, X \in \{1,2,3,4,5\})}{\mathbb{P}(X \in \{1,2,3,4,5\})} \\ &= \frac{(1/6)\mathbb{I}(i \in \{1,2,3,4,5\})}{5/6} \\ &= \frac{1}{5}\mathbb{I}(i \in \{1,2,3,4,5\})\end{aligned}$$

which is just the uniform distribution over $\{1,2,3,4,5\}$.

Because generating from Y is the same as generating from X conditioned to lie in a set, acceptance rejection can be used.

Five sided die
1) Draw $X \leftarrow \text{Unif}(\{1,2,3,4,5,6\})$
2) If $X \leq 5$ then return X
3) Else $X \leftarrow$ **Five sided die**, return X

To why this algorithm works, consider $\mathbb{P}(X = 2)$. First, note that the algorithm terminates with probability 1. This is because for the algorithm to not terminate, all the rolls of X must be 6, 6, 6, ..., which happens with probability 0 for an infinite stream of rolls. So the algorithm terminates with probability 1. Either the original roll of the die is 2, or the algorithm calls itself recursively. Let Y be the output of the algorithm, W the draw of X in line 1, and R the draw of X in line 3. Then for $i \in \{1, 2, 3, 4, 5\}$,

$$\begin{aligned}
\mathbb{P}(Y = i) &= \mathbb{P}(W = i) + \mathbb{P}(W = 6)\mathbb{P}(Y = i) \\
\mathbb{P}(Y = i) &= (1/6) + (1/6)\mathbb{P}(Y = i) \\
(5/6)\mathbb{P}(Y = i) &= 1/6 \\
\mathbb{P}(Y = i) &= 1/5.
\end{aligned}$$

More generally, the following theorem holds for the acceptance rejection algorithm.

Theorem 5 (Acceptance/Rejection)
Suppose for $X \sim \mu$, $\mathbb{P}(X \in B) > 0$. Then the following algorithm

Acceptance-Rejection *Input:* μ, B
1) Draw $X \leftarrow \mu$
2) If $X \in B$, then return X
3) Else let $X \leftarrow$ **Acceptance-Rejection**, return X

has output $Y \sim [X|X \in B]$ with probability 1.

Proof. Since $\mathbb{P}(X \in B) > 0$, the algorithm will terminate in finite time with probability 1.

Let C be a (measurable) subset of B. Then consider the probability that the output Y falls into C. Let W be the value of X in line 1, and R be the value of X in line 3.

$$\begin{aligned}
\mathbb{P}(Y \in C) &= \mathbb{P}(W \in C) + \mathbb{P}(W \notin B)\mathbb{P}(R \in C) \\
\mathbb{P}(Y \in C) &= \mathbb{P}(W \in C) + \mathbb{P}(W \notin B)\mathbb{P}(Y \in C) \\
(1 - \mathbb{P}(W \notin B))\mathbb{P}(Y \in C) &= \mathbb{P}(W \in C) \\
\mathbb{P}(Y \in C) &= \frac{\mathbb{P}(W \in C)}{\mathbb{P}(W \in B)} \\
&= \mathbb{P}(W \in C|W \in B).
\end{aligned}$$

Since W was a draw from $X \sim \mu$, the proof is complete. □

Notation 2
The acceptance rejection method is also sometimes called **hit-or-miss** because the algorithm keeps drawing from A until we hit B. Statisticians often refer to it as just the **rejection** algorithm, because they are often pessimistic folk.

Example 16
Create an algorithm to draw Unif$(\{1,\ldots,7\})$ given the ability to draw uniformly from $\{1,\ldots,10\}$.

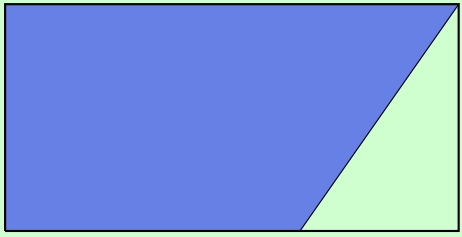
Answer Repeatedly draw from $\{1,\ldots,10\}$ until the result is at most 7.

Example 17
Suppose I want to shuffle a deck of cards in such a way that every King comes closer to the top of the deck than all the Aces.

Answer Keep generating perfect permutations of the deck until all the Kings are nearer to the top of the deck than all the Aces.

Example 18
Draw uniformly from the trapezoid connecting $(0,0)$, $(0,1)$, $(2,1)$, and $(1.3,0)$. Call this region B.

Answer Note that $B \subset A$, where A is the rectangle $[0,2] \times [0,1]$ (see the picture).



It is easy to sample from this rectangle with uniforms. Simply let U_1, U_2 be iid Unif$([0,1])$, then $2U_1 \sim$ Unif$([0,2])$, and $(2U_1, U_2)$ is uniform over A. To be in the blue region the point must be above the line that passes through $(1.3,0)$ and $(2,1)$. Solving for this line gives the following pseudocode.

Acceptance Rejection for rectangle
1) Repeat
2) Draw (U_1, U_2) iid Unif$([0,1])$
3) Until $(U_2 > 2U_1(10/7) - (13/7))$.

In R this can be done with

```
mc <- function() {
  # Draw from quadrilateral connecting (0,0),(0,1),(2,1),(1.3,0)
  accept <- FALSE
```

```
  while (!accept) {
    x <- runif(2)
    accept <- x[2] > 2 * x[1] * (10 / 7) - (13 / 7)
  }
  return(x)
}
```

8.1 Running time of AR

Consider again the stream $X_1, X_2, \ldots$ of iid random variables. Some of the X_t fall into B, and some do not. Let T be the smallest integer such that $X_T \in B$, then T is the infimum of the set $\{t : X_t \in B\}$. Then X_T is our output Y. That is,

$$Y = X_T,\ T = \inf\{t : X_t \in B\}.$$

Here T is a special type of random variable called a *stopping time*.

Definition 18
A random variable T is a **stopping time** with respect to a stream $X_1, X_2, \ldots$, if it is possible to determine if $T = t$ just by looking at $X_1, \ldots, X_t$.

The name means that at any time t, it is possible to decide whether to stop drawing from the X_i given the values of $X_1, \ldots, X_t$.

The value of T is the number of random variates drawn by the algorithm (including the last) until the result is accepted. This type of random variable, where an experiment that results in either success or failure is repeated until a success occurs, is called a *geometric* random variable. The parameter p is the probability that the experiment is a success.

Definition 19
Say that T is a **geometric** random variable with parameter p (write $T \sim \text{Geo}(p)$) if for all $t \in \{1, 2, \ldots\}$,

$$\mathbb{P}(T = t) = p(1-p)^{t-1}.$$

Because the distribution of T is known, its expected value and variance are also known.

Fact 14
For T the number of draws used by the AR algorithm is $\text{Geo}(\mathbb{P}(X \in B))$, so $\mathbb{E}[T] = 1/\mathbb{P}(X \in B)$, and $\mathbb{V}(T) = \mathbb{P}(X \notin B)/\mathbb{P}(X \in B)^2$.

Problems

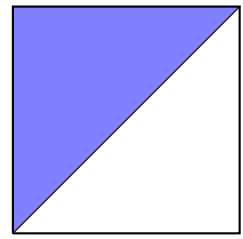
8.1: Suppose that the function `mc` draws numbers uniform from $\{-5, -4, \ldots, 4, 5\}$. Write an AR code in R that makes a function `mc` that draws uniformly from $\{-1, 0, 1\}$.

8.2: Suppose the function `testdraw` draws uniformly from the integers 1 through 10. Write a function `testdraw2` in R to draw uniformly from 1 through 7.

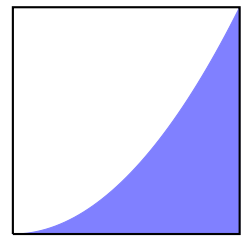
8.3: The function `rexp(1,2)` draws a single exponential random variable of rate 2. Use AR to build a function `mcexp` in R that draws from an exponential random variable of rate 2 conditioned to lie in $[0.5, 1.5]$.

8.4: Us AR to build a function `mcexp2` in R that draws from an exponential random variable of rate 1.5 conditioned to lie in $[0, 1)$.

8.5: Write pseudocode to draw $(X, Y) \sim \text{Unif}(\Omega)$, where Ω is the trangular region connecting $(0, 0), (0, 1), (1, 1)$.



8.6: Write pseudocode to draw $(X, Y) \sim \text{Unif}(A)$, where $A = \{(x, y) : 0 \leq y \leq x^2\}$.



Chapter 9

AR for densities

Question of the day

Suppose that X has unnormalized density

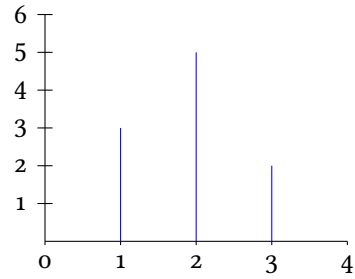
$$g_X(1) = 3,\ g_X(2) = 5,\ g_X(3) = 2.$$

Create an AR method for sampling from X.

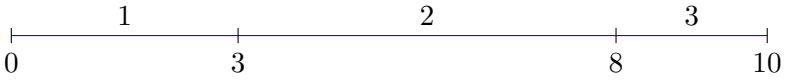
Summary

- For two function f and g with the same domain, write $f \geq g$ if for all elements x of the domain, $f(x) \geq g(x)$.
- Suppose that I have the ability to sample from unnormalized density g_Y and wish to sample from unnormalized density g_X where $g_Y \geq g_X$. The **Acceptance Rejection method for densities** is to take a draw Y from g_Y, then accept Y as a draw from X with probability $g_X(Y)/g_Y(Y)$.
- Note that if c is a constant that is at least $g_X(a)/g_Y(a)$ for all a, then g_Y and $c \cdot g_Y$ are the same unnormalized density, but $cg_Y \geq g_X$.
- The expected number of draws from g_Y used by AR for densities is $\mathbb{E}[g_Y(Y)/g_X(Y)]$ where $Y \sim g_Y$.
- Suppose $g \geq 0$ and the area of $A = \{(x, y) : 0 \leq y \leq g(x)\}$ is finite. Then for $(X, Y) \sim \mathsf{Unif}(A)$, $X \sim g$.
- For X with unnormalized density g_X and $[Y \mid X] \sim \mathsf{Unif}([0, g_X(X)])$, (X, Y) is a uniform draw over $\{(x, y) : 0 \leq y \leq g_X(x)\}$.

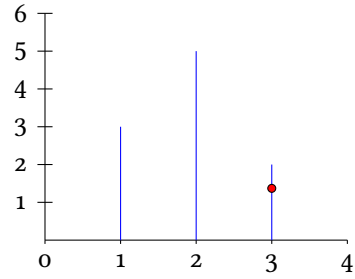
Start by taking a look at this unnormalized random variable. The density looks like this.



We can take these three bars and line them up together to get one long bar of length $3 + 5 + 2 = 10$, which we can then divide into pieces corresponding to the different values for X ($\{1, 2, 3\}$).

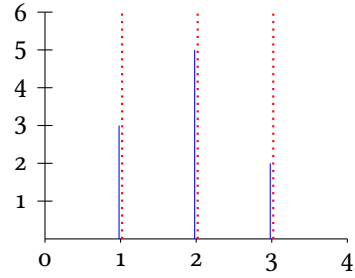


We can draw W uniformly from $[0, 10]$. If it falls into $[0, 3)$ return 1, if it falls into $[3, 8)$ return 2, and if it falls into $[8, 10]$ return 3. This is exactly what the Inverse Transform Method tells us to do. We can think of this as giving us a point uniformly drawn from the blue lines:



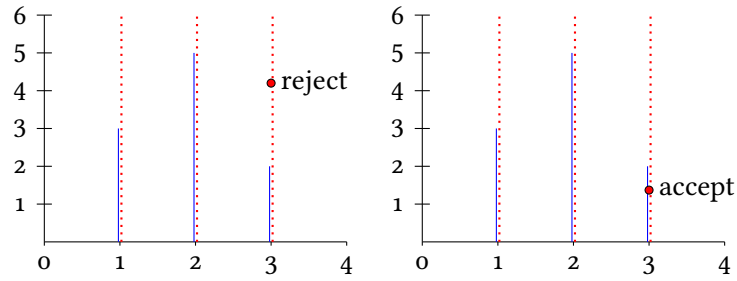
Note that we only care about the horizontal coordinate of the red dot, the vertical coordinate gives us no information.

If the lines had been the same height, things would have been easier.



To draw uniformly from the red dotted lines, just draw the horizontal Y coordinate uniformly over $\{1, 2, 3\}$, and then draw vertical coordinate W uniformly over $[0, 6]$.

When does (Y, W) also serve as an (X, W) point? Sometimes, this (Y, W) point falls off of the blue lines, in which case we reject it as a draw. But if the point works for both red and blue lines, then accept the point as a value for red.



Given $Y = 3$, then W must be in $[0, 2]$ to accept. Because W is uniform over $[0, 6]$, the chance that this occurs is $(2 - 0)/(6 - 0)$. This is the same as $\mathbb{P}(U \leq 2/6)$, where $U \sim \text{Unif}([0, 1])$. So when $Y = 3$, we accept when $U \leq g_X(3)/g_Y(3)$, and reject otherwise.

Expanding this idea to all the possible values of $Y \in \{1, 2, 3\}$ gives the following algorithm.

Acceptance Rejection for qotd	
1)	Repeat
2)	Draw Y uniformly from $\{1, 2, 3\}$
3)	Draw $U \leftarrow \text{Unif}([0, 1])$
4)	Until $U \leq g_X(Y)/g_Y(Y)$
5)	Return Y

9.1 The Acceptance Rejection Theorem for densities

Generalizing this approach gives us the Acceptance Rejection Theorem for densities. The earliest reference to the method that I can find is here [1]. In that paper Von Neumann does not make an overt claim to have created the method, but on the other hand he does not reference any earlier work either.

Theorem 6 (Acceptance Rejection for densities)
Suppose X and Y have unnormalized densities g_X and g_Y respectively where $g_Y \geq g_X$, and it is possible to generate draws from Y.

Then the following algorithm generates draws from X.

Acceptance Rejection for densities

1) Repeat
2) Draw Y from unnormalized density g_Y
3) Draw $U \leftarrow \mathsf{Unif}([0,1])$
4) Until $U \leq g_X(Y)/g_Y(Y)$
5) Return Y

The output of the algorithm will have unnormalized density g_X.

Proof. Let B be a measurable subset of $\mathbb{R}$, and X be the output of the algorithm. Let Z_Y be the normalizing constant for g_Y, and Z_X be the normalizing constant for g_X.

For $X \in B$, either the choice of Y at the start is in B and is accepted, or the choice of Y is not accepted, and subsequent choices puts the result in B. That is,

$$\mathbb{P}(X \in B) = \mathbb{P}\left(Y \in B, U \leq \frac{g_X(Y)}{g_Y(Y)}\right) + \mathbb{P}\left(U > \frac{g_X(Y)}{g_Y(Y)}\right)\mathbb{P}(X \in B)$$

$$\mathbb{P}\left(U \leq \frac{g_X(Y)}{g_Y(Y)}\right)\mathbb{P}(X \in B) = \mathbb{P}\left(Y \in B, U \leq \frac{g_X(Y)}{g_Y(Y)}\right)$$

$$\mathbb{P}(X \in B) = \frac{\mathbb{P}\left(Y \in B, U \leq \frac{g_X(Y)}{g_Y(Y)}\right)}{\mathbb{P}\left(U \leq \frac{g_X(Y)}{g_Y(Y)}\right)}$$

$$= \frac{\int_{b\in B} \frac{g_Y(b)}{Z_Y} \cdot \frac{g_X(b)}{g_Y(b)}\, d\mu}{\int_{b\in\Omega} \frac{g_Y(b)}{Z_Y} \cdot \frac{g_X(b)}{g_Y(b)}\, d\mu}$$

$$= \frac{\int_{b\in B} g_X(b)\, d\mu}{\int_{b\in\Omega} g_X(b)\, d\mu} = \frac{\int_{b\in B} g_X(b)\, d\mu}{Z_X} = \int_{b\in B} f_X(b)\, d\mu.$$

That means that the output has the desired density. □

Example 19
Draw $X \sim x^2\mathbb{I}(x \in [0,1])$.

Answer Note $x^2\mathbb{I}(x \in [0,1]) \leq 1$, so draw $X \sim \mathsf{Unif}([0,1])$ and $U \sim \mathsf{Unif}([0,1])$ independently. If $U \leq X^2$ then accept and return X, otherwise begin again.

Code in R for doing this is as follows.

```
ardensity_ex <- function() {
  acceptflag <- FALSE
  while (!acceptflag) {
    x <- runif(1)
    u <- runif(1)
    acceptflag <- u < x^2
  }
  return(x)
}
```

9.2 AR for arbitrary unnormalized densities

The theorem requires that $g_y \geq g_X$, but what if that is not true?

Because g_Y is an unnormalized density for Y, for any positive constant c, cg_Y is still an unnormalized density for Y. In particular, if

$$c = \max_s \frac{g_X(s)}{g_Y(s)},$$

then $cg_Y \geq g_X$ by the definition of maximum. This gives rise to an AR algorithm that works for arbitrary unnormalized densities g_Y and g_X.

Fact 15
Suppose g_Y and g_X are arbitrary unnormalized densities, and for all s,

$$c \geq \frac{g_X(s)}{g_Y(s)}.$$

Then the output of

Acceptance Rejection for densities

1) Repeat
2) Draw Y from unnormalized density g_Y
3) Draw $U \leftarrow \text{Unif}([0,1])$
4) Until $U \leq g_X(Y)/[cg_Y(Y)]$
5) Return Y

has unnormalized density g_X.

The running time of the algorithm will be fastest when $c = \max_s g_X(s)/g_Y(s)$, but any c that upper bounds the maximum will do.

9.3 Using Acceptance Rejection on unbounded random variables

By combining the Inverse Transform Method and Acceptance Rejection we can generate from unbounded random variables such as standard normals.

Using Cauchy draws to simulate normals For instance, recall that a Cauchy random variable has unnormalized density

$$g_1(x) = \frac{1}{1+x^2}.$$

A normal random variable has unnormalized density

$$g_2(x) = \exp(-x^2/2).$$

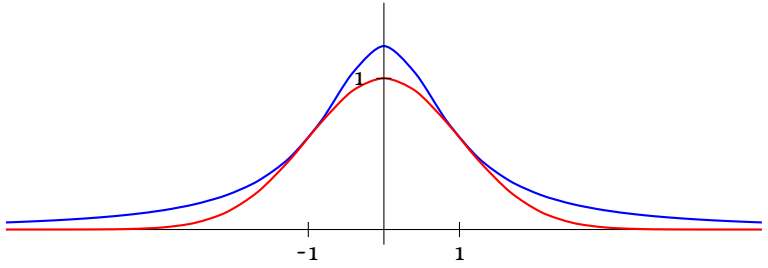
Unfortunately, we cannot use AR yet because for some values of x, $g_1(x) < g_2(x)$, and for other values $g_2(x) < g_1(x)$. However, we can find a constant c such that $cg_1(x) \geq g_2(x)$ for all x. To find this, we solve

$$c = \max\left\{\frac{g_2(x)}{g_1(x)}\right\}.$$

To find the maximum, we find the derivative of the ratio.

$$\begin{aligned}\left[\frac{g_2(x)}{g_1(x)}\right]' &= \left[\exp(-x^2/2)(1+x^2)\right]' \\ &= -x\exp(-x^2/2)(1+x^2) + \exp(-x^2/2)(2x) = \exp(-x^2/2)(x-x^3) \\ &= \exp(-x^2/2)x(1-x^2).\end{aligned}$$

So this is positive for $x < -1$, negative for $x \in (-1, 0)$, positive for $x \in (0, 1)$, and negative again for $x > 1$. That implies that the function has a local maximum at -1 and at 1. Plugging -1 and 1 into $g_2(x)/g_1(x)$ gives the same result: $2\exp(-1/2)$.



This makes

$$\frac{g_2(x)}{cg_1(x)} = (\exp(1/2)/2)\exp(-x^2/2)(1+x^2).$$

Next we need a way to draw from the Cauchy. To do that, we find the cdf, and use the Inverse Transform Method. First, we need to normalize the density.

$$\int_{-\infty}^{\infty} \frac{1}{1+x^2}\,dx = \arctan(x)|_{-\infty}^{\infty} = \frac{\tau}{4} - \left(-\frac{\tau}{4}\right) = \frac{\tau}{2}.$$

So the normalized density is $(2/\tau)(1+x^2)^{-1}$. This gives us the cdf:

$$\int_{-\infty}^{a} \frac{2}{\tau} \cdot \frac{1}{1+x^2}\, dx = (2/\tau)\arctan(x)|_{-\infty}^{a} = (2/\tau)\arctan(a) - (1/2).$$

Solving $U = (2/\tau)\arctan(a) - (1/2)$ gives

$$X = \tan((\tau/2)(U + 1/2)) \sim \text{Cauchy}.$$

Putting this together gives the following algorithm.

Drawing normals using Cauchys	
1)	Repeat
2)	Draw U_1, U_2 iid from Unif$([0,1])$
3)	$X \leftarrow \tan((\tau/2)(U_1 - 1/2))$
4)	Until $U_2 \le (\exp(1/2)/2)\exp(-X^2/2)(1+X^2)$

9.4 Running time

The following shows how to calculate the chance of accepting a draw from one density for another.

Fact 16
The chance of accepting in **Acceptance Rejection for densities** is

$$\mathbb{E}\left[\frac{g_X(Y)}{g_Y(Y)}\right].$$

Proof. The chance of accepting is

$$\mathbb{P}(U \le g_X(Y)/g_Y(Y)) = \mathbb{E}[\mathbb{I}(U \le g_X(Y)/g_Y(Y))].$$

Note that for any fixed value $y \in [0,1]$:

$$\mathbb{P}(U \le g_X(y)/g_Y(y)) = \mathbb{E}[\mathbb{I}(U \le g_X(y)/g_Y(y))] = g_X(y)/g_Y(y).$$

The Fundamental Theorem of Probability tells us that for any random variables $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$. So

$$\begin{aligned}\mathbb{P}(U \le g_X(Y)/g_Y(Y)) &= \mathbb{E}[\mathbb{E}[\mathbb{I}(U \le g_X(Y)/g_Y(Y))|Y]] \\ &= \mathbb{E}[g_X(Y)/g_Y(Y)],\end{aligned}$$

which completes the proof! □

Let's use this theorem to say how many Cauchy random variables we need for one normal.

Example 20
If $g_Y(s) = 1/(1+s^2)$ and $g_X(s) = (\exp(1/2)/2)\exp(-s^2/2)$, what is the probability of accepting?

Answer We want

$$\begin{aligned}\mathbb{E}\left[\frac{g_X(Y)}{g_Y(Y)}\right] &= \int \frac{g_X(s)}{g_Y(s)} f_Y(s)\, ds \\ &= \int \frac{(\exp(1/2)/2)\exp(-s^2/2)}{1/(1+s^2)} (2/\tau) \frac{1}{1+s^2}\, ds \\ &= \frac{\exp(1/2)}{\tau} \int \exp(-s^2/2)\, ds \\ &= \frac{\exp(1/2)}{\sqrt{\tau}} \approx \boxed{0.6577\ldots}.\end{aligned}$$

9.5 *The uniform perspective*

In some sense, densities do not exist. They are just special cases of uniform distributions.

Fact 17
Suppose X has unnormalized density g_X with respect to either counting or Lebesgue measure. Suppose Y has distribution given X of

$$[Y|X] \sim \mathsf{Unif}([0, g_X(X)]).$$

Then $(X,Y) \sim \mathsf{Unif}(\{(x,y) : 0 \le y \le g_X(x)\})$.

So we can always write a random variable X with a density as part of a uniform random pair (X,Y).

Proof. If X has unnormalized density g_X, and $[Y|X]$ has density

$$f_{Y|X=x}(y) = \frac{1}{g_X(x)} \mathbb{I}(y \in [0, g_X(x)])$$

gives joint density

$$\begin{aligned}f_{(X,Y)}(x,y) &\propto g_X(x) f_{Y|X=x}(y) \\ &= g_X(x) \frac{1}{g_X(x)} \mathbb{I}(y \in [0, g_X(x)]) \\ &= \mathbb{I}(y \in [0, g_X(x)]).\end{aligned}$$

Since the joint density is proportional to an indicator function, the distribution must be uniform over the region where the indicator function equals 1. □

This also works in the opposite direction.

Fact 18
Suppose $(X, Y) \sim \text{Unif}(\{(x, y) : x \in A, 0 \leq y \leq g(x)\})$ for a nonnegative g. Then $X \sim g$.

Proof. Suppose $(X, Y) \sim \text{Unif}(\{(x, y) : x \in A, 0 \leq y \leq g(x)\})$. Then for a set $B \subseteq A$,

$$\mathbb{P}(X \in A) = \frac{\mu(\{(x, y) : x \in B, 0 \leq y \leq g(x)\})}{\mu(\{(x, y) : x \in A, 0 \leq y \leq g(x)\})} = \frac{\int_{x \in B} g(x)\, d\mu}{\int_{x \in A} g(x)\, d\mu}.$$

This is exactly what it means for X to have (possibly unnormalized) density g. □

Problems

9.1: The following code uses the inverse transform method to draw a sample uniformly from

$$f_X(x) = \frac{1}{x^2}\mathbb{I}(x \geq 1)$$

```
mc <- function() {
  u <- runif(1)
  return(1/u)
}
```

Write R code that uses this function (together with AR) to draw a sample from unnormalized density

$$g_X(x) = \frac{1}{x^{2.5}}\mathbb{I}(x \geq 1)$$

9.2: a) Write an AR code that draws samples from the unnormalized density

$$g_Y(y) = \frac{1}{y^{2.3}}\mathbb{I}(y \in [1, 2]).$$

b) Use 10^6 samples from your function to estimate $\mathbb{E}[Y]$.

9.3: Suppose that I have the ability to draw Y from the following density

$$g_Y(i) = i\mathbb{I}(i \in \{1, 2, 3\}),$$

and U from $\text{Unif}([0, 1])$.

a) Create the most efficient AR algorithm for drawing from

$$g_X(i) = [(1 + i)/2]\mathbb{I}(i \in \{1, 2, 3\}).$$

b) What is the expected number of times through the repeat loops for your AR algorithm?

9.4: Suppose I wish to sample from the following density

$$g_X(s) = \sqrt{|x|}\mathbb{I}(x \in [-2, 2])$$

a) Create an AR algorithm for generating samples.

b) On average, how many times do you run through the repeat loop before accepting?

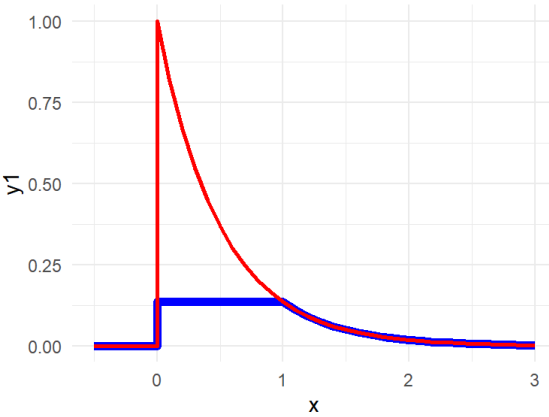
9.5: Consider the following unnormalized density:

$$f_W(w) = \exp(-2)\mathbb{I}(w \in [0, 1]) + \exp(-2w)\mathbb{I}(w > 1).$$

Now consider the unnormalized density

$$f_T(t) = \exp(-2t)\mathbb{I}(t \geq 0).$$

Note that for all t, $f_T(t) \geq f_W(t)$. This looks as follows.



a) Consider a draw (X, Y) from underneath the density f_T. What is the probability that this falls underneath the density f_W?

b) Write pseudocode for an AR algorithm that draws from W using draws from T.

9.6: Write acceptance rejection pseudocode that uses draws from the unnormalized density of an exponential of rate 1:

$$g_Y(x) = \exp(-x)\mathbb{I}(x \geq 0)$$

in order to draw from the beta distribution with parameters 2 and 2 which has unnormalized density

$$g_X(x) = x(1 - x)\mathbb{I}(x \in [0, 1]).$$

9.7: Suppose the goal is to use draws from unnormalized density

$$g_A(s) = \exp(-0.9x)\mathbb{I}(x \geq 0)$$

to draw from unnormalized density

$$g_B(s) = x\exp(-x)\mathbb{I}(x \geq 0)$$

It is not true that $g_A(s) \geq g_B(s)$ for all $s \geq 0$, but it is true that for

$$m = \max_{s:g_A(s)>0} \frac{g_B(s)}{g_A(s)},$$

it holds that

$$mg_A(s) \geq g_B(s).$$

a) Find m and round up to four sig figs.

b) Write pseudocode for AR for using draws from A to obtain draws from B.

9.8: Create an acceptance rejection algorithm that uses draws from T an exponential of rate 1 to draw from S with density $f_S(s) = s\mathbb{I}(s \in [0,1])$.

9.9: A *double exponential* random variable of rate λ has density

$$f_T(t) = \frac{\lambda}{2}\exp(-\lambda|t|).$$

Another way to think about it as an exponential of rate λ that has a fifty-fifty chance of being positive or negative. Pseudocode for drawing from this distribution is as follows.

AR_dblexp_norm	
1)	Repeat
2)	Draw T double exponential of rate 1
3)	Draw U a standard uniform
4)	Until $U < \exp(-T^2/2 + \|T\| - 1/2)$
5)	Return T

Write pseudocode for an acceptance rejection algorithm that uses a double exponential of rate 1 to make draws using AR from a standard normal with unnormalized density $\exp(-x^2/2)$.

9.10: Write pseudocode for an AR algorithm that uses draws from a standard Cauchy random variable with unnormalized density

$$f_X(s) = \frac{1}{1+s^2}$$

to sample from a double exponential random variable of rate λ, with density

$$f_T(s) = (1/2)\lambda\exp(-\lambda|s|).$$

Part II

High dimensional Monte Carlo

Chapter 10

High dimensional models with known normalizing constant

Question of the day

An experimenter believes 4 phenotypes are equally likely in the population. After testing 100 random subjects, she finds

phenotype	I	II	III	IV
number	20	30	18	32

Find the p-value for this data with respect to the statistic

$$S_2 = \sum_{i=1}^{4} (X_i - 25)^2.$$

Summary

Say that $(X_1, \ldots, X_k)$ has a **multinomial distribution** with parameters n and $(p_1, \ldots, p_k)$ a probability vector, if for all i, $X_i \sim \text{Bin}(n, p_i)$ and $X_1 + \cdots + X_k = n$. To sample from this distribution,

1: For j from 1 to n, draw D_j from $(1, \ldots, k)$ using probability vector $(p_1, \ldots, p_k)$.

2: For i from 1 to k, let $X_i = \sum_{j=1}^{n} \mathbb{I}(D_j = i)$.

Until now sampling has occurred in one dimension, but most of the actual applications of Monte Carlo are to higher dimensional examples. Generating

$$X_1, \ldots, X_n$$

that are independent reduces to n applications of the one dimensional case. But sampling from

$$(X_1, \ldots, X_n)$$

where the X_i are not independent can be challenging.

10.1 Multinomial distribution

Consider the experiment in the question of the day. In this study, each test subject was assigned to one of four phenotypes. In this case, the result is a four dimensional vector $(X_1, \ldots, X_4)$ such that $X_1 + X_2 + X_3 + X_4 = 100$.

This last equation is an example of a *linear constraint* because it can be written in the form

$$AX = b$$

for some constant matrix A, vector of random variables X, and constant vector b.

When we have a table of data that satisfies a linear constraint, it is called a *contingency table*. In our case, the contingency table only has one row, so is fairly simple to generate from. (Tables with more than one row will be dealt with later on in the course.)

The values of $X_1, \ldots, X_4$ are not independent as knowing any three of them allows us to immediately calculate the final value. However, the original test subjects *are* independent. Let $D_1, \ldots, D_n$ be iid random variables over $\{1, 2, 3, 4\}$ where $\mathbb{P}(D_i = j) = p_j$.

Definition 20
Say that $(p_1, \ldots, p_k)$ is a **probability vector** for a random variable X if $X \in \{1, \ldots, k\}$ and $\mathbb{P}(X = i) = p_i$ for all $i \in \{1, \ldots, k\}$.

In the Question of the Day, the probability vector for the outcomes of each trial is $(1/4, 1/4, 1/4, 1/4)$. Counting the number of times each phenotype appears gives a *multinomial distribution.*

Definition 21
Suppose that $D_1, \ldots, D_n$ are iid with probability vector $p_1, \ldots, p_k$. For $i \in \{1, \ldots, k\}$, let

$$X_i = \sum_{j=1}^{n} \mathbb{I}(D_j = i).$$

Then say $(X_1, \ldots, X_k)$ has a **multinomial distribution** with parameters $n, p_1, \ldots, p_k$. Write $(X_1, \ldots, X_k) \sim \mathsf{Multinom}(n, p_1, \ldots, p_k)$.

The $(X_1, \ldots, X_k)$ are not independent, but the D_i are. So the simplest Monte Carlo method for sampling from this distribution is to just use the definition as given. That is, first generate the D_i and then use this to calculate the X_i.

Each X_i is the sum of n terms, so takes time $\Theta(n)$ to calculate. There are k different X_i, so the total running time will be $\Theta(nk)$.

This is not a great running time, but there is a way to do better. Instead of doing each sum separately, update the appropriate X_i value when each D_j is drawn.

Multinomial	*Input:* $n, p_1, \ldots, p_k$
1)	Let $(X_1, \ldots, X_k) \leftarrow (0, 0, \ldots, 0)$
2)	For i from 1 to n
3)	Draw $D_i \leftarrow (p_1, \ldots, p_k)$
4)	$X_{D_i} \leftarrow X_{D_i} + 1$

This takes time $\Theta(n)$. This is considered an exponential time algorithm, since n can be exponentially large in the size of the input of $n, p_1, \ldots, p_k$. Because of the decimal system, n can be represented using $\Theta(\ln(n))$ symbols. That means that n itself is exponentially large in the number of symbols used to represent n.

To see this more clearly, consider a simpler example. Suppose `list(n)` is a function that give a positive integer `n` as input, lists out the integers 1 through n. Then `list(9)` returns

1
2
3
4
5
6
7
8
9

even though the length of the input was a single character `9`. Similarly, the use of two characters gives input value that could be from `10` up to `99`, making the size of the output range from 10 up through 99. In other words, for the input consisting of d digits, the output size ranges from 10^{d-1} up to $10^d - 1$. That is, the size of the output is exponentially large in the number of digits of the input.

10.2 Conditional Marginal Distributions

To develop a faster algorithm, consider how the conditional marginal distributions of the multinomial behave. For any vector of random variables $(X_1, \ldots, X_n)$, call the distribution of any particular X_i a *marginal* distribution.

With marginal distributions, to sample from $(X_1, \ldots, X_n)$, first sample

$$X_1$$

from its marginal distribution. Next sample from

$$[X_2|X_1].$$

That is, sample the next coordinate X_2 conditioned on the value of X_1 that was just sampled. Then sample

$$[X_3|X_1, X_2],$$

and so on until we have the complete vector.

CondMarg

1) Draw X_1 from its marginal distribution.
2) For i from 2 to n
3) Draw X_i conditioned on the values of $X_1, \ldots, X_{i-1}$
4) Return $(X_1, \ldots, X_n)$

Whenever you know the conditional marginal distributions, this is the algorithm to use in sampling from vectors!

To see how this idea works with the multinomial distribution, consider the Question of the Day example. There are 100 subjects, with a $1/4$ chance of being phenotype I. So $X_1 \sim \mathsf{Bin}(100, 1/4)$.

Suppose 21 out of these 100 are phenotype I. Then that leaves $100-21 = 79$ experiments. Each of these experiments *cannot* be phenotype I. Conditioned on that fact, these each have a $1/3$ chance of being phenotype II, III, or IV. So $[X_2|X_1 = 21] \sim \mathsf{Bin}(100-79, 1/3)$. Once X_2 is chosen, then $[X_3|X_2, X_1]$ is drawn, and finally $[X_4|X_3, X_2, X_1]$ is whatever is left over.

In general, start with the marginal distribution of X_1. The number of D_j that equal 1 has a binomial distribution with parameters n and p_1. Moreover, once X_1 is found, the remaining $(X_2, \ldots, X_k)$ form a new multinomial distribution. The number of test subjects remaining will be $n - X_1$. The probability vector is renormalized to add up to 1.

$$\frac{1}{p_2 + \cdots + p_k}(p_2, \ldots, p_k).$$

Fact 19
Suppose

$$(X_1, \ldots, X_n) \sim \mathsf{Multinom}(n, p_1, \ldots, p_k).$$

Then $X_1 \sim \mathsf{Bin}(n, p_1)$, and for all $i \in \{2, \ldots, k\}$,

$$[X_i|X_1, \ldots, X_{i-1}] \sim \mathsf{Multinom}(n - (X_1 + \cdots + X_{i-1}), p_i/s_i, \ldots, p_k/s_i).$$

Here $s_i = p_i + p_{i+1} + \cdots + p_k$.

This fact gives the following algorithm.

FastMultinomial *Input:* $n, p_1, \ldots, p_k$

1) Let $(X_1, \ldots, X_k) \leftarrow (0, 0, \ldots, 0)$
2) Draw X_1 as binomial with parameters n and p_1
3) For i from 2 to k
4) Draw X_i as binomial with parameters $n - \sum_{j<i} X_j$ and $p_i/(1 - \sum_{j<i} p_j)$
5) Return $(X_1, \ldots, X_k)$

If the binomial distribution is sampled from in $\Theta(\ln(n))$ time (which can be accomplished using acceptance/rejection) then this becomes a $\Theta(k \ln(n))$ algorithm, which is the best possible in general.

In both these algorithms, the conditional marginal distribution of X_i given $X_1, \ldots, X_{i-1}$ is known.

Example 21
Given this fast method for generating multinomials, it can be used to answer the question of the day.

Answer First code the method for generating the multinomials.

```
rmultinomial <- function(n = 100, p = c(0.25, 0.25, 0.25,
    0.25)) {
  x <- rep(0, length(p))
  pnorm <- 1
  for (i in 1:length(p)) {
    x[i] <- rbinom(1, n, p[i]/pnorm)
    n <- n - x[i]
    pnorm <- pnorm - p[i]
  }
  return(x)
}
```

Next code a method for calculating the test statistic.

```
test.stat <- function(data, prob) {
  return(sum((data - sum(data)*prob)^2))
}
```

Then run the Monte Carlo experiment.

```
p <- c(1/4, 1/4, 1/4, 1/4)
sdata <- test.stat(data = c(20, 30, 18, 32), prob = p)
r <- replicate(10^6, (test.stat(rmultinomial(), p) >= 148))
mean(r)
sd(r)/sqrt(length(r))
```

When I ran it, I found a p-value of 0.1147 ± 0.0004.

10.3 Permutations

Definition 22
A vector $(x_1, \ldots, x_n)$ is a **permutation** of $(1, 2, \ldots, n)$ if each $x_i \in \{1, \ldots, n\}$ and for all $i \neq j$, $x_i \neq x_j$. Call the set of such vectors $\mathcal{S}_n$.

For instance, $(2, 3, 1, 4)$, $(1, 2, 3, 4)$, and $(4, 3, 2, 1)$ are all permutations of $(1, 2, 3, 4)$.

Consider the problem of sampling uniformly from the permutations of $(1, \ldots, n)$. The same conditional marginal distribution approach used for **FastMultinomial** works for this problem as well.

Fact 20
If $(X_1, \ldots, X_n) \sim \text{Unif}(\mathcal{S}_n)$, then

$$[X_i | X_1, \ldots, X_{i-1}] \sim \text{Unif}(\{1, \ldots, n\} \setminus \{X_1, \ldots, X_{i-1}\}.$$

For example, if the permutation vector with four elements starts with a 2 and a 4, so it looks like $(2, 4, -, -)$, then the third element is uniform over $\{1, 2, 3, 4\} \setminus \{2, 4\} = \{1, 3\}$.

By being careful, it is possible to build an algorithm that uses no extra memory and draws exactly $n - 1$ uniforms using $\Theta(n \ln(n))$ random bits.

Permutation	*Input:* n
1)	Let $(X_1, \ldots, X_n) \leftarrow (1, \ldots, n)$
2)	For i from 1 to n
3)	Draw $I \leftarrow \text{Unif}(\{i, \ldots, n\})$
4)	Swap the values of X_i and X_I
5)	Return $(X_1, \ldots, X_n)$

Since there are $n!$ different permutations, generating uniformly over this set requires on the order of $\ln(n!) = n \ln(n) + o(n \ln(n))$ bits.

A brief note about running times

In the above analysis of the running time, the number of bits needed to generate one uniform from 1 up to n, $\Theta(\ln(n))$ was included. This method of counting random bits uses a model called a *probabilistic Turing machine*.

Another common way to measure running time is to count the number of uniform random numbers over $[0, 1]$ are needed. Each one of these can be transformed into a uniform over $\{1, \ldots, n\}$ using the inverse transform method:

$$\lfloor nU \rfloor + 1.$$

Using this model, the running time would be considered $\Theta(n)$.

10.4 Densities and normalizing constant

Since the multinomial and permutation distributions are over finite state spaces, their densities are with respect to counting measure. For multinomials,

$$f_{(X_1,\ldots,X_k)}(x_1,\ldots,x_k) = \binom{n}{x_1\ x_2\ \cdots\ x_n} p_1^{x_1}\cdots p_k^{x_k}\mathbb{I}((x_1,\ldots,x_n)\in\Omega),$$

where Ω is the set of nonnegative integers vectors whose components add to n. Here

$$\binom{n}{x_1\ x_2\ \cdots\ x_n} = \frac{n!}{x_1!x_2!\cdots x_n!}.$$

This is a completely normalized distribution.

Conditioning on the value of X_1, the conditional density formula gives:

$$\begin{aligned} &f_{(X_2,\ldots,X_n)|X_1=x_1}(x_2,\ldots,x_n) \\ &\qquad\propto \binom{n}{x_1\ x_2\ \cdots\ x_n} p_1^{x_1}\cdots p_k^{x_k}\mathbb{I}((x_1,\ldots,x_n)\in\Omega) \\ &\qquad= \frac{n!p_1^{x_1}}{x_1!(n-x_1)!}\binom{n-x_1}{x_2\ \cdots\ x_n} p_2^{x_2}\cdots p_k^{x_k}\mathbb{I}((x_2,\ldots,x_n)\in\Omega) \end{aligned}$$

The $n!p_1^{x_1}/(x_1!(n-x_1)!)$ part is constant when $X_1 = x_1$ is fixed, and the rest looks like another multinomial distribution. This is the basis of Fact 19.

For permutations

$$f_{(X_1,\ldots,X_k)}(x_1,\ldots,x_k) = (1/Z)\mathbb{I}((x_1,\ldots,x_n)\in\mathcal{S}_n))$$

For permutations, $Z = \#(\mathcal{S}_n) = n!$. Again, the normalizing constant is known in the sense that it can be easily computed. In general, the conditional marginal distribution can be used efficiently on any problem where the normalizing constant is efficient to compute.

Problems

10.1: Suppose the random variables (X_1, X_2, X_3) are multinomial with $n = 10$ and $(p_1, p_2, p_3) = (0.2, 0.5, 0.3)$. (So $(X_1, X_2, X_3) \sim \mathsf{Multinom}(10, 0.2, 0.5, 0.3)$.)

a) What is the distribution of X_1?

b) What is the distribution of (X_2, X_3) given that $X_1 = 4$?

10.2: Suppose (R_1, R_2, R_3, R_4) are multinomial with $n = 20$ and $(p_1, p_2, p_3, p_4) = (0.1, 0.1, 0.1, 0.7)$.

a) What is the distribution of R_1?

b) What is the distribution of (R_2, R_3, R_4) given that $R_1 = 1$)?

10.3: Suppose that (X_1, X_2, X_3) are uniform over the permutations of the numbers 1, 2, and 3.

a) What is the distribution of X_1?

b) Give $X_1 = 2$, what is the distribution of (X_2, X_3)?

10.4: Suppose $(R_1, \ldots, R_{52})$

- Using factorial notation, how many different values are there for $(R_1, \ldots R_{52})$.
- What is the distribution of R_1?

10.5: Suppose $U_1 \sim \text{Unif}([0, 1])$ and $[U_i | U_1, \ldots, U_{i-1}] \sim \text{Unif}([0, U_{i-1}]$. Write pseudocode that takes n as input and returns a random draw from $(U_1, \ldots, U_n)$.

10.6: Write pseudocode that combines the conditional marginal method with acceptance rejection in order to generate a permutation uniformly on $n \geq 4$ elements such that the number in the first position is larger than the number in the fourth position.

Chapter 11

Sampling uniformly a direction in space

Question of the day

How can one draw a direction uniformly from n dimensional space?

Summary

- A random vector is **uniform over directions** if its distribution is invariant under any constant rotation.
- Such a vector can be generated by drawing uniformly over a ball of any fixed radius in n dimensions.
- Another way to generate this type of vector is by taking iid normal random variables.
- Usually such random vectors are **normalized** by dividing by their Euclidean length.

There are many applications where the need is to pick a random direction in $\mathbb{R}^n$. For instance,

- Optimization routines occasionally pick a random direction to search in.
- Random walks in high dimensions start moving in a random direction.

11.1 Uniform direction

This is easy to do in one dimension, either move right or move left.

$$D \sim \mathsf{Unif}(\{-1, 1\}).$$

In higher dimensions, things get a bit trickier. First, it is necessary to say what is meant by uniform over direction. One way to describe it is to say that a vector is uniform over direction if rotating it by any fixed value does not change the distribution.

Definition 23
Say that $X \in \mathbb{R}^n$ has a distribution that is **uniform over directions in $\mathbb{R}^n$** if $\mathcal{R}(X) \sim X$ for all fixed rotations $\mathcal{R}$.

For instance, if you take a vector in $\mathbb{R}^3$ that is uniform over directions, and rotate it 90 degrees around the y-axis, it will still be uniform. This type of definition, where uniform is defined as being invariant under a type of transformation, is also called *Haar measure*.

There are many such distributions that are invariant under rotation. One way to generate from such directions is to draw uniformly from inside a ball of a fixed radius.

Fact 21
Let $B_n = \{(x_1, \ldots, x_n) : x_1^2 + \cdots x_n^2 \leq 1\}$. Suppose

$$(X_1, \ldots, X_n) \sim \mathsf{Unif}(B_n).$$

Then $(X_1, \ldots, X_n)$ is uniform over directions in $\mathbb{R}^n$.

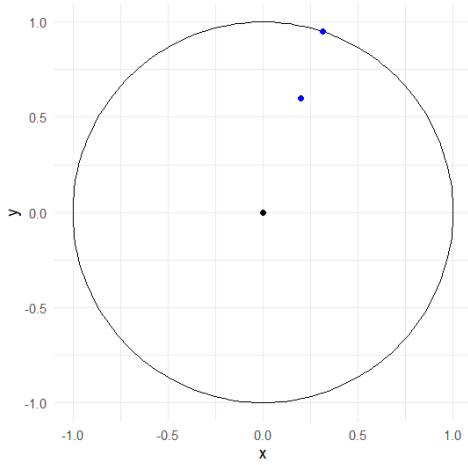
It is common to take a uniform direction and *normalize* it by dividing by its Euclidean length to make it of length 1.

Fact 22
If $(X_1, \ldots, X_n)$ is uniform over directions in $\mathbb{R}^n$, then so is

$$\frac{(X_1, \ldots, X_n)}{\|(X_1, \ldots, X_n)\|}.$$

Two dimensions

In two dimensions, a point drawn from inside the unit circle and then normalized to have length 1 will lie on the surface of the circle.



Because this point is uniform under rotation, the angle of the point on the edge of the unit circle will also be uniform from 0 up to τ. (Here τ is the *full circle constant*, and is $\tau = 2\pi$ where π is the *half circle constant*.) This gives the following fact.

Fact 23
If $\Theta \sim \mathsf{Unif}([0, \tau))$, then

$$(X, Y) = (\cos(\Theta), \sin(\Theta))$$

is a uniform direction in 2 dimensional space.

The corresponding algorithm then is

2D Circle

1) Draw $U_1 \leftarrow \mathsf{Unif}([0, 1])$
2) Return $(\cos(U_1\tau), \sin(U_1\tau))$

The surface of the unit circle is an example of a *manifold* of one dimension. A manifold is a curve or surface or hypersurface in space that looks flat when magnified about a point on the manifold. Because the manifold is one dimensional, this algorithm is the best possible. One uniform gives one draw from a one dimensional manifold.

Three dimensions

Three dimensions is slightly trickier than two dimensions, but there is a very useful fact about the surface of a sphere that makes sampling using the conditional distribution method possible.

Fact 24
For (X_1, X_2, X_3) uniform over the surface of a 3-sphere, for each i, $X_i \sim \mathsf{Unif}([-1, 1])$.

The algorithm draws X_3 first, which restricts the draw to the circle on the surface of the hypersphere where the z value equals X_3. The points (X_1, X_2) are then drawn uniformly from this circle.

3D surface of sphere

1) Draw $U_1, U_2 \leftarrow \mathsf{Unif}([0, 1])$
2) Let $X_3 \leftarrow 2U_1 - 1$, $R \leftarrow \sqrt{1 - X_3^2}$, $\Theta \leftarrow \tau U_2$
3) Return $(R\cos(\Theta), R\sin(\Theta), X_3)$

Again this is as good as can be hoped for: the algorithm uses 2 uniforms to get a draw that was uniform over a two dimensional manifold.

11.2 Higher dimensions

While the problem is easy in 3 or fewer dimensions, in 4 or higher dimensions things get stickier. Instead of using special properties, at this point it is best to jump to a more general idea.

AR could be used.

4D surface of hypersphere	
1)	Repeat
2)	Draw $(U_1, U_2, U_3, U_4) \leftarrow \mathsf{Unif}([-1,1]^4)$
3)	Until $U_1^2 + U_2^2 + U_3^2 + U_4^2 \leq 1$
4)	Return $(U_1, U_2, U_3, U_4)/\left\|(U_1, U_2, U_3, U_4)\right\|$

This is fine in four dimensions where the chance of acceptance is about 30.84%, but the chance of acceptance goes down exponentially fast in the dimension. So a better approach is needed for high dimensions.

11.3 Normal random variables

The key to a better approach is to use standard normal random variables.

Consider $Z_1, \ldots, Z_n$ iid standard normal random variables. Then their joint density is just the product of their individual densities, so

$$\begin{aligned} f_{Z_1,\ldots,Z_n}(z_1, \ldots, z_n) &= \prod_{i=1}^{n} \tau^{-1/2} \exp(-z_i^2/2) \\ &= \tau^{-n/2} \exp\left(-(1/2)\sum_{i=1}^{n} z_i^2\right) \\ &= \tau^{-n/2} \exp\left(-(1/2)\left\|(z_1, \ldots, z_n)\right\|^2\right) \end{aligned}$$

That means that the density does not depend on the z_i values except through their L_2 norm, their Euclidean distance from the origin. That gives the rotational symmetry, or uniformity over directions, that is the goal.

Fact 25
Let $Z_1, \ldots, Z_n$ be iid standard normal random variables. Then

$$\frac{1}{\left\|(Z_1, \ldots, Z_n)\right\|}(Z_1, \ldots, Z_n)$$

is uniform over directions in $\mathbb{R}^n$.

Using this method, a uniform direction on an $n-1$ dimensional manifold can be generated using only n draws from iid standard normals. So not much is lost!

Example 22
Use the normal method to estimate what proportion of the surface area of the Earth is within 10 degrees latitude of the equator.

Answer. We draw uniformly from the surface of a sphere, and then record how many of those points are within 10 degrees latitude of the equator. The first few lines draw uniformly from the sphere, and `w` is the indicator function (using lots of basic trigonometry) that our point lies within 10 degrees of the equator.

```
mc <- function() {
  z <- rnorm(3)  # will need three normals for each draw
  n <- sqrt(sum(z^2))
  norm_z <- z / n
  w <- abs(norm_z[3] / sqrt(sum(norm_z[1:2]^2))) <
           tan(10 * pi / 180)
  return(w)
}
```

Box-Mueller

This idea can be used in reverse to draw normal random variables. This is called the *Box-Mueller* method for drawing normal random variables. To do this, it is necessary to understand the distribution of the distance the point (Z_1, Z_2) is from the origin when Z_1 and Z_2 are iid standard normals.

Fact 26
Let (R, Θ) be the polar coordinate version of the rectangular coordinate point (Z_1, Z_2). For Z_1 and Z_2 are iid standard normals, $\Theta \sim \text{Unif}([0, \tau))$ and $R \sim r\exp(-r^2/2)\mathbb{I}(r \geq 0)$.

Proof. Remember that when converting from rectangular coordinates to polar, $dx\, dy = r\, dr\, d\theta$. The densities can be written with factors $\mathbb{I}(\theta \in [0, \tau))$ and $\mathbb{I}(r \geq 0)$ because angles are always given in $[0, \tau)$ and distance from the origin as a positive number. Hence,

$$\begin{aligned} f_{Z_1,Z_2}(z_1, z_2)\, dz_1\, dz_2 &= \tau^{-1}\exp(-(z_1^2 + z_2^2)/2)r\, dr\, d\theta \\ &= \left[\tau^{-1}\mathbb{I}(\theta \in [0, \tau])\, d\theta\right]\left[r\exp(-r^2/2)\, dr\right]. \end{aligned}$$

Note $\tau^{-1}\mathbb{I}(\theta \in [0, \tau])$ is the density of a uniform over $[0, \tau]$ and $r\exp(-r^2/2)$ is the Rayleigh density. □

To generate samples from the Rayleigh density, first generate an exponential of rate $1/2$ using $T = -2\ln(U)$ and then take the square root.

Fact 27
Let $T \sim \text{Exp}(1/2)$. Then $\sqrt{T}$ has the Rayleigh distribution with density

$$r \exp(-r^2/2)\mathbb{I}(r \geq 0).$$

Proof. Let $T \sim \text{Exp}(1/2)$ and $a > 0$. Then

$$\begin{aligned}\mathbb{P}(\sqrt{T} \leq a) &= \mathbb{P}(T \leq a^2) \\ &= \int_0^{a^2} (1/2)\exp(-(1/2)s)\, ds \\ &= 1 - \exp(-a^2/2).\end{aligned}$$

Differentiating then gives the desired result. □

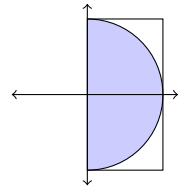
This gives rise to the Box-Mueller method for generating normal random variables.

Box-Mueller	
1)	Draw $U_1, U_2 \leftarrow \text{Unif}([0,1])$
2)	$\Theta \leftarrow \tau U_1$, $R \leftarrow \sqrt{-2\ln(U_2)}$
3)	Return $(R\cos(\Theta), R\sin(\Theta))$

This is the best possible: two draws of uniforms over $[0,1]$ will give two draws from standard normal random variables.

11.4 Ratio of Uniforms

Older computers did not have easy access to functions like sine and cosine. Even when they did, these functions could take a long time to evaluate. To solve this problem, a method was developed called *ratio of uniforms* which used our 2D acceptance rejection in a circle in order to draw uniformly from $[-\tau/2, \tau/2]$. Consider the following picture.



Use acceptance rejection with draws from the rectangle to get a draw inside the circle. This requires on average $2/[(1/2)(\tau/2)] \leq 1.274$ draws to get an accepted value (X_1, X_2). From trigonometry $X_2/X_1 = \tan(\Theta)$, so has a standard Cauchy distribution. All without using trigonometric functions!

Ratio-of-Uniforms-Cauchy
1) Repeat
2) Draw (U_1, U_2) iid Unif$([0,1])$
3) Until $U_1^2 + (2U_2 - 1)^2 \leq 1$
4) Return (U_2/U_1)

Definition 24
A method is of type **ratio-of-uniforms** if its output is U_2/U_1 where (U_1, U_2) is drawn uniformly over a set.

Kinderman and Monahan introduced this method in 1977, calling it the ratio-of-uniforms method. Because the Cauchy has the rise and fall behavior of so many important distributions (such as Binomial and Gamma) it is useful in sampling from them. Often the constant time algorithms for sampling from distributions such as Binomial have a ratio-of-uniforms form.

Problems

11.1: Let $V_1, \ldots, V_5$ be chosen uniformly from the surface of a 5 dimensional sphere. Using 10^6 samples, estimate

$$\mathbb{E}[||V||_1] = \mathbb{E}\left[\sum_{i=1}^{5} |V_i|\right]$$

and report your estimate in $a \pm b$ form.

11.2: Using 10^6 draws, estimate

$$\mathbb{P}(W_1 + \cdots + W_6 \geq 0.5)$$

where $(W_1, \ldots, W_6)$ is uniformly drawn from the surface of a six dimensional sphere.

11.3: Use the normal method to estimate what proportion of the Earth's surface is between 40 and 50 degrees latitude north of the equator. Take 10^6 samples and estimate your answer in the form $a \pm b$.

11.4: Use the normal method to estimate what proportion of the Earth's surface is between the latitudes of 10 degrees north and 10 degrees south of the equator. Take 10^6 samples and estimate your answer in the form $a \pm b$.

11.5: Write code to draw (X, Y) uniformly from the unit circle (the boundary of a ball of radius 1) in $\mathbb{R}^2$. Using 10^6 draws, estimate $\mathbb{P}(Y \geq 0.7)$, reporting your answer as $a \pm b$.

11.6: Write code to draw (X, Y) uniformly from the unit circle (the boundary of a ball of radius 1) in $\mathbb{R}^2$. Using 10^6 draws, estimate $\mathbb{P}(X + 0.3Y \geq 0.4)$, reporting your answer as $a \pm b$.

11.7: Consider (X, Y) uniform over the unit disc $\mathsf{Unif}(\{(x, y) : x^2 + y^2 \leq 1\})$. Consider the distance from the origin $R = \sqrt{X^2 + Y^2}$

a) Find, using the properties of uniforms, $\mathbb{P}(R \leq 0.3)$.

b) Find for $r \in [0, 1]$, $\mathbb{P}(R \leq r)$.

c) Write code to sample from R using the inverse transform method.

d) Draw (X, Y) uniformly from the unit disc by first drawing R using the last part, and θ uniformly from 0 to τ, then converting from polar coordinates to Cartesian coordinates.

e) Using 10^6 draws, estimate $\mathbb{P}(Y \geq 0.5)$, reporting your answer as $a \pm b$.

11.8: For (X, Y, Z) uniform from the volume inside the unit sphere $\{(x, y, z) : x^2 + y^2 + z^2 \leq 1\}$, what is the density of $R = \sqrt{X^2 + Y^2 + Z^2}$.

Chapter 12

High dimensional models with unknown normalizing constant

Question of the day

A model of soil quality lists each plot as either *good* (label with a 1) or *bad* (label with a 0)

0	1	1	1
1	1	0	1
0	1	0	1
1	1	1	0

For a vector x, let $h(x)$ be the number of adjacent plots (left-right or up-down) with the same label. The probabilistic model uses a parameter $\beta \geq 0$, and sets

$$\mathbb{P}(X = x) = \frac{\exp(\beta h(x))}{Z(\beta)},$$

Here $Z(\beta)$ is the normalizing constant for the density, which is a function of the parameter β.

For $\beta = 1$, in the 4 by 4 lattice, estimate the probability that $h(X) \leq 16$.

Summary

The **Acceptance/Rejection** method works just as well in high dimensions as in one dimension. It can be used to get samples from distributions where the normalizing constant is unknown. Unfortunately, the run time of basic AR tends to increase exponentially with the dimension for simple examples.

Definition 25
The **state space** (often written Ω) is the set of all possible outcomes in a model.

In the question of the day, the graph structure is that of a 4 by 4 lattice. The lattice points can be numbered from 1 to 16:

4	8	12	16
3	7	11	15
2	6	10	14
1	5	9	13

Each of these sixteen values can be either 0 or 1. The mathematical notation for the set of n-tuples of length 16 whose components are either 0 or 1 is $\{0,1\}^{16}$. In other words, when you raise a set to an integer power n, the result is the set of n tuples whose components live in the set.

Definition 26
Say that $v \in A^n$ if v is an n-tuple where each component is an element of A. If $A \subseteq \mathbb{R}$, then we say v is an n **dimensional vector.**

Counting measure is an example of what is called a product measure. One consequence of this is that the size of a set that is raised to a power is the size of the set raised to that power. That means that for a state space like $\{0,1\}^{16}$, $\#(\{0,1\}^{16}) = \#(\{0,1\})^{16} = 2^{16} = 65536$.

This fact can be written as follows.

Fact 28
For A a finite set, $\#(A^n) = \#(A)^n$.

Given the ability to generate from one dimension, and if the goal is to generate from

$$X_1, X_2, \ldots, X_n$$

that are iid (independent, identically distributed) random variables, then just generate the points X_i one at a time.

In most models where Monte Carlo methods are used, the goal is to generate a high dimensional distribution

$$(X_1, \ldots, X_n)$$

where the X_i are *not* independent.

Such models can be described by an unnormalized joint density.

$$f_{(X_1,\ldots,X_n)}(x_1,\ldots,x_n)$$

with respect to measure μ. Being an unnormalized joint density means that for A an n-dimensional subset of $\mathbb{R}^n$,

$$\mathbb{P}((X_1,\ldots,X_n)\in A)=\frac{\int_{(x_1,\ldots,x_n)\in A} f_{(X_1,\ldots,X_n)}(x_1,\ldots,x_n)\,d\mu}{\int_{(x_1,\ldots,x_n)\in\mathbb{R}^n} f_{(X_1,\ldots,X_n)}(x_1,\ldots,x_n)\,d\mu}$$

For many of these models, it is prohibitively difficult to calculate the denominator. So what do we do?

12.1 AR for the Ising model

The distribution of the question of the day has a name: it is called the *Ising model*.

Definition 27
For a graph $G=(V,E)$, say that $X\in\{0,1\}^V$ is distributed according to the **Ising model** with parameter β (write $X\sim\mathsf{Ising}(G,\beta)$) if X has unnormalized density

$$g_X(x)=\exp(\beta h(x))$$

where $h(x)=\sum_{\{i,j\}\in E}\mathbb{I}(x(i)=x(j))$.

The simplest approach is to use acceptance rejection. In the question of the day, the maximum that $h(x)$ can take on is 24. So for $\beta\geq 0$,

$$\begin{aligned} h(x)&\leq 24\\ \beta h(x)&\leq 24\beta\\ \exp(\beta(h(x))&\leq\exp(24\beta). \end{aligned}$$

Hence for unnormalized target density $g_X(x)=\exp(\beta h(x))$, we can use unnormalized proposal density $g_Y(x)=\exp(24\beta)$. This makes

$$\frac{\exp(\beta h(x))}{\exp(24\beta)}=\exp(\beta(h(x)-24)).$$

The density $\exp(24\beta)$ is constant over the state space $\{0,1\}^{16}$. That makes $Y\sim\mathsf{Unif}(\{0,1\}^{16})$. The nice thing about sampling uniformly over a product state space is that you can sample independently and uniformly over each piece.

Fact 29
The variables $\{U_1,\ldots,U_n\}$ are independent with $U_i\sim\mathsf{Unif}(A_i)$ if and only if $(U_1,\ldots,U_n)\sim\mathsf{Unif}(A_1\times\cdots\times A_n)$.

That means to sample $Y \sim \text{Unif}(\{0,1\}^{16})$, just draw $Y_1, \ldots, Y_{16}$ independently uniform over $\{0,1\}$. If the proposed state Y happens to be all 0's, or all 1's, then $h(X) = 24$, and the probability of accepting is 1. That means there is at least a $2/2^{16}$ probability of accepting, which means that the expected number of draws before accepting is at most

$$\frac{1}{2/2^{16}} = 2^{15} = 32768.$$

This is not a problem for a fast computer, but this was also a tiny 4 by 4 lattice. Since the chance of acceptance goes down exponentially in the square of the side length of the lattice, it goes down quickly indeed. A 5 by 5 lattice has 25 nodes, and so the bound given on the number of times a draw needs to be made is $2^{25}/2 = 16\,777\,216$ or more than 16 million!

Returning to the 4 by 4 lattice, in pseudocode, this AR algorithm looks like the following

AR for Ising *Input :* $\beta, G = (V, E)$

1) Repeat
2) Draw $Y_1, \ldots, Y_{\#(V)}$ iid $\text{Unif}(\{0,1\})$, $U \leftarrow \text{Unif}(\{0,1\})$
3) Let $h \leftarrow \sum_{\{i,j\} \in E} \mathbb{I}(Y(i) = Y(j))$
4) Until $U \leq \exp(-\beta(24 - h(Y)))$
5) Return Y

Implementing in R

To implement this method in R, start with the **rep** function. Here **rep** stands for repeat. This function repeats a value to create a vector of the desired length. For instance,

```
rep(0,16)
```

has output

```
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Next, the vector needs to be put that into a matrix. Fortunately, R has a **matrix** data type. We can place this into a matrix that is 4 by 4 by setting the parameter nrow to 4. This tells us there are 4 rows, and since there are 16 entries in the matrix, gives 4 columns for free without specifying anything. The result is a 4 by 4 matrix. At this point, the matrix in R looks like this:

```
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
[4,]    0    0    0    0
```

Now can we calculate $h(x)$ for a matrix in this form? Well, first we want to add up the vertical cases. That is, we want to count the times that the entries in rows 1 through 3 match the entries in rows 2 through 4.

Next, count the times that the entries in columns 1 through 3 match the entries in columns 2 through 4. The **nrow** function gives the number of rows in a matrix, while **ncol** gives the number of columns. The following code calculates h for a matrix `x`.

```
h <- function(x) {
  h <- sum(x[1:(nrow(x)-1), ] == x[2:nrow(x), ])
  h <- h + sum(x[, 1:(ncol(x)-1)] == x[, 2:ncol(x)])
  return(h)
}
```

Given the function to calculate $h(x)$, it can be determined if a particular proposed state should be accepted or rejected. For an a by b lattice, each of the a rows has $b-1$ horizontal edges, and each of the b columns has $a-1$ vertical edges. This makes the total number of edges in such a lattice

$$a(b-1)+b(a-1) = 2ab - a - b.$$

That means that the AR algorithm will look something like this.

```
ar.ising <- function(beta, k) {
  repeat {
    x <- matrix(as.integer(runif(k[1] * k[2]) > 0.5),
      nrow = k[1])
    u <- runif(1)
    if (u < exp(-beta * (2 * k[1] * k[2] - k[1] - k[2] - h(x)))
      return(x)
  }
}
```

This code can be modified to find out how *many* steps we took to get a single draw.

```
ar.ising.report <- function(beta, k) {
  t <- 0
  repeat {
    t <- t + 1
    x <- matrix(as.integer(runif(k[1] * k[2]) > 0.5),
      nrow = k[1])
    u <- runif(1)
    if (u < exp(-beta *
                (2 * k[1] * k[2] - k[1] - k[2] - h(x))))
      return(t)
  }
}
```

To answer the Question of the Day, the code could be used as follows:

```
results <- replicate(10^2, h(ar.ising(1, c(4,4))) <= 16)
mean(results)
sd(results)/sqrt(length(results))
```

which returned 0.10 ± 0.03 when I ran it.

Problems

12.1: Consider a 3 by 3 lattice.

a) How many nodes are there in the lattice?

b) How many edges are there in the lattice?

12.2: Consider a 5 by 4 lattice.

a) How many nodes are there in the lattice?

b) How many edges are there in the lattice?

12.3: Using 200 exact draws from the Ising model on a 3 by 3 lattice with $\beta = 0.8$, find the average of the h function value. Write your answer as $a \pm b$.

12.4: Using 200 exact draws from the Ising model on a 3 by 3 lattice with $\beta = 1.0$, find the average of the h function value. Write your answer as $a \pm b$.

12.5: Using 200 draws from the Ising model on a 3 by 3 lattice with $\beta = 1.2$, find the average of the h function value. Write your answer as $a \pm b$.

12.6: Using 200 exact draws from the Ising model on a 3 by 3 lattice with $\beta = 0$, find the average of the h function value. Write your answer as $a \pm b$.

12.7: For state space $[0, 1]^{10}$, consider $X = (X_1, \ldots, X_{10})$ with unnormalized density

$$g_X(x_1, x_2, \ldots, x_{10}) = x_1 + 2x_2 + \cdots + 10x_{10}.$$

In R, this density can be computed using:

```
g <- function(x) {
  return(sum(1:10 * x))
}
```

Using uniform draws over the state space in AR, write a function in R to draw from this distribution.

12.8: For state space $[0,2]^3$, consider $X = (X_1, X_2, X_3)$ with unnormalized density

$$h_X(x_1, x_2, x_3) = x_1 x_2 x_3 \mathbb{I}((x_1, x_2, x_3) \in [0,2]^3).$$

Using uniform draws over the state space in AR, write a function in R to draw from this distribution.

Chapter 13

Introduction to Markov chains

Question of the Day

Why does randomly swapping pairs of cards in a deck lead to an approximately uniform permutation of the cards?

Summary

- A **Markov chain with update function** ϕ is a stochastic process $\{X_t\}$ where the next state is a function of the current state and some randomness. That is, for $U_0, U_1, \ldots$ a stream of random variables, $X_{t+1} = \phi(X_t, U_t)$.
- A distribution π is **stationary** for the Markov chain if $X_t \sim \pi$ implies $\phi(X_t, U_t) \sim \pi$.
- A distribution ν is **limiting** for the Markov chain if for all x_0, the distribution of X_t given $X_0 = x_0$ converges to ν.
- A Markov chain is **connected** if for any two states i and j of the chain, there exists a time t such that $\mathbb{P}(X_t = j \mid X_0 = i) > 0$.
- A connected Markov chain is **aperiodic** if for any two states i and j of the chain, there exists a time t such that for all $t' \geq t$, $\mathbb{P}(X_{t'} = j \mid X_0 = i) > 0$.
- The **Ergodic Theorem** (aka the Fundamental Theorem of Markov chains) says that if a Markov chain is both aperiodic and connected, then it has a unique stationary distribution that is also a limiting distribution.

13.1 What is a Markov chain?

Using acceptance rejection for the Ising model and other similar distributions works well when the number of dimensions n is low, but usually takes time that grows exponentially in n.

In order to do better, the idea of *Markov chain Monte Carlo* was created during World War II at about the same time that Monte Carlo methods in general started being used.

The idea is to take advantage of a special type of stochastic process called a *Markov chain*. These had been invented by Markov decades earlier to assist in cracking codes, now they would be used in the opposite direction to create random objects.

The idea is that a Markov chain starts at state X_0. Then the next state is found by first drawing some randomness U_0, and then setting

$$X_1 = \phi(X_0, U),$$

where ϕ is a deterministic function called the *update function*. If this process is repeated, then the result is a Markov chain.

Definition 28
Given random variable $X_0 \in \Omega$, an iid stream $U_0, U_1, \ldots$ of random choices in Ω_R and update function $\phi : \Omega \times \Omega_R \to \Omega$, for $t \in \{1, 2, \ldots\}$ the stream formed by

$$X_t = \phi(X_{t-1}, U_{t-1})$$

is a **Markov chain with update function ϕ**. The set Ω is called that **state space** of the chain.

Example 23
Suppose $X_0 = 0$ and $D_0, D_1, D_2, \ldots$ are iid $\mathsf{Unif}(\{-1, 1\})$. Then for $\phi(x, d) = x + d$, this Markov chain either moves the state 1 to the right or to the left at each step. It is called *symmetric random walk on the integers*.

A deck of cards can be thought of as encoding a permutation.

Notation 3
Let $\mathcal{S}_n$ be the set of permutations of n elements.

The $\mathcal{S}$ stands for *symmetric group*.

Example 24
For a permutation $x \in \mathcal{S}_n$, suppose $\phi_{\text{tr}}(x, (i, j))$ is the permutation that swaps the values in components i and j. So

$$\phi_{\text{tr}}(x, (i, j))(k) = \begin{cases} x(j) & k = i \\ x(i) & k = j \\ x(k) & k \neq i, k \neq j \end{cases}$$

Then if $U_t = (I_t, J_t) \sim \mathsf{Unif}(\{1, \ldots, n\}^2)$, then update function ϕ_{tr} gives the *transposition chain* on permutations.

For instance, $\phi((5,3,1,2,4),3,5) = (5,3,4,2,1)$. In terms of a deck of cards, the transposition chain picks two cards at random (so there is a $(1/n)^2$ chance of picking the same card twice) and swaps them in the deck.

If one continues transposing random pairs of cards, after a while, the deck will become mixed up. More precisely, the state X_t will have a distribution that is close to uniform.

On the other hand, if a deck of cards is already mixed up, that is, already in a uniformly chosen permutation, then randomly transposing two cards leaves the distribution (but not the state) unchanged: it is still uniform over $\mathcal{S}_n$. This type of distribution is called *stationary*.

Definition 29
A distribution π is **stationary** for a Markov chain if

$$X_t \sim \pi \Rightarrow X_{t+1} \sim \pi.$$

A step of a Markov chain can be thought of in two ways.

1. A step takes a state X_{t-1} and changes the state to X_t.
2. If X_{t-1} has distribution π_{t-1}, then after one step of the Markov chain, X_t has distribution $\mathcal{M}\pi_{t-1} = \pi_t$.

In other words, a Markov chain step is an *operator* on probability distributions that transforms the distribution to a new distribution.

Definition 30
For a Markov chain step $\mathcal{M}$, say that $\mathcal{M}\pi_{t-1} = \pi_t$ if

$$[X_t \mid X_{t-1} \sim \pi_{t-1}] \sim \pi_t.$$

Then a stationary distribution is a distribution π such that $\mathcal{M}\pi = \pi$.

Example 25
Let $X_t \sim \mathsf{Unif}(\{-1,0,1\})$, $D_t \sim \mathsf{Unif}(\{-1,1\})$, and the Markov chain step is $X_{t+1} = X_t + D_t$. Then the distribution of X_{t+1} has density:

$$f_{X_{t+1}}(i) = \frac{1}{6}\mathbb{I}(i=-2) + \frac{1}{6}\mathbb{I}(i=-1) + \frac{1}{3}\mathbb{I}(i=0) + \frac{1}{6}\mathbb{I}(i=1) + \frac{1}{6}\mathbb{I}(i=2).$$

There is an interesting phenomenon in mathematics that if you have a *fixed point* like a *stationary distribution* with respect to an operator, and you apply this operator over and over again, you often end up converging to the fixed point.

Example 26
A numerical example of this phenomenon is the mapping

$$x \mapsto x + (2 - x^2)/(2x).$$

If you plug $\sqrt{2}$ into this mapping, you get $\sqrt{2} + (2-2)/(2\sqrt{2}) = \sqrt{2}$, so this mapping has $\sqrt{2}$ as a fixed point. Now suppose you start at 2 and apply this mapping over and over. You get

x	$x + (2 - x^2)/(2x)$
2	1.5
1.5	1.416666
1.416666	1.414215

This matches the $\sqrt{2}$ to the first 6 digits already! Repeating this procedure gives a more and more accurate result.

In the same way, when you take many steps in a Markov chain, the resulting distribution will converge to the stationary distribution under a few simple conditions. In order to make this precise, it is necessary to define a notion of convergence of distributions. First define a distance between distributions. For a distribution π, call a set A *measurable* if $\pi(A)$ gives the probability that $X \sim \pi$ falls in A.

Definition 31
The **total variation distance** between probability distributions π_1 and π_2 is

$$\text{dist}_{\text{TV}}(\pi_1, \pi_2) = \sup_{\text{measurable } A} |\pi_1(A) - \pi_2(A)|.$$

Here sup standard for the *supremum*, which is the least upper bound for a set of values. Because $\pi_1(A)$ and $\pi_2(A)$ are probabilities in $[0, 1]$, there difference is in $[-1, 1]$, and their absolute value is in $[0, 1]$. Therefore, the total variation distance is between 0 (when the distributions are effectively the same) and 1 (when the two distributions put all their respective probability on disjoint sets.)

Definition 32
A sequence of probability distributions $\{\pi_t\}$ **converges** to π (write $\pi_t \to \pi$) if

$$\lim_{t \to \infty} \text{dist}_{\text{TV}}(\pi_t, \pi) = 0.$$

Then π is a limiting distribution of the Markov chain if no matter where the chain starts, as time goes on the distribution of the current state is approaching π.

Definition 33
A Markov chain $\{X_t\}$ over state space Ω has **limiting distribution** π if

$$(\forall x_0 \in \Omega)([X_t \mid X_0 = x_0] \to \pi).$$

In order for a finite state Markov chain to have the stationary distribution (fixed point) equal to the limiting distribution, the chain must have two properties.

1. It must be possible to get from any state i to any other state j by taking steps in the Markov chain.
2. It should not be possible to partition the state space into regions $A_1, \ldots, A_k$ where $k \geq 2$ such that for all $i \in \{1, \ldots, k-1\}$, $\mathbb{P}(X_t \in A_{i+1} \mid X_t \in A_i) = 1$ and $\mathbb{P}(X_t \in A_1 \mid X_t \in A_k) = 1$.

These two properties are formally defined as follows.

Definition 34
A Markov chain over a finite state space Ω is **connected** (aka **irreducible**) if

$$(\forall i, j \in \Omega)(\exists t)(\mathbb{P}(X_t = j \mid X_0 = i) > 0).$$

Definition 35
A Markov chain over a finite state space Ω is **aperiodic** if

$$(\forall i, j \in \Omega)(\exists t)(\forall t' \geq t)(\mathbb{P}(X_{t'} = j \mid X_0 = i) > 0).$$

The big theorem about Markov chains is that a connected, aperiodic chain has a limiting distribution equal to a unique stationary distribution.

Theorem 7
Ergodic Theorem for finite state Markov chains
If a finite state Markov chain is connected and aperiodic, then it has a unique stationary distribution that is also the limiting distribution for the chain.

The proof can be found in *Stochastic Processes*, at https://d71b37a6-f420-4d99-8d4filesusr.com/ugd/c2b9b6_98b69ae5caa54f68aeaabf198918e918.pdf.

The easiest way to ensure that a connected chain is aperiodic is to make sure there is a state i such that $\mathbb{P}(X_t = i \mid X_{t-1} = i) > 0$.

Fact 30
Suppose $\{X_t\}$ is a connected Markov chain with state i such that $\mathbb{P}(X_t = i \mid X_{t-1} = i) > 0$. Then the chain is also aperiodic.

Proof. Let $x, y \in \Omega$. Then there is a time r such that $\mathbb{P}(X_r = i \mid X_0 = x) > 0$ and s such that $\mathbb{P}(X_s = y \mid X_0 = i) > 0$. Let $t = r + s$, and $t' \geq t$. Then

$$\mathbb{P}(X_{t'} = y \mid X_0 = x) \geq \mathbb{P}(X_r = i \mid X_0 = x)\mathbb{P}(X_{t'-s} = i \mid X_r = i)\mathbb{P}(X_{t'} = y \mid X_{t'-s} = i).$$

Since all three factors on the right are positive, the left hand side is also positive.

□

Finally, it is possible to find the limiting distribution of the transposition chain on permutations.

Fact 31
The transposition chain for permutations has the uniform distribution as its limiting distribution.

Proof. To use the Ergodic Theorem, the chain must be shown to be connected, aperiodic, and have the uniform distribution as the stationary distribution.

- The fact that the chain is connected follows from any comparison sorting algorithm that only switches two items as a time. (For instance, **BubbleSort** does the trick.)
- Since there is a $1/n^2 > 0$ chance that $I = J$ and the state remains the same, it is also aperiodic.
- Suppose $X_{t-1} \sim \mathsf{Unif}(\mathcal{S}_n)$. Let $x \in \mathcal{S}_n$. Then

$$\begin{aligned}\mathbb{P}(X_t = x) &= \sum_y \mathbb{P}(X_{t-1} = y)\mathbb{P}(X_t = x \mid X_{t-1} = y) \\ &= \frac{1}{n!}\sum_y \mathbb{P}(X_t = x \mid X_{t-1} = y).\end{aligned}$$

Suppose that y differs from x by a single transposition that swaps the items at positions a and b. There are n choose 2 such states y for every x.

If the current state is y, the chance of picking the transposition that swap to x is $2/n^2$, since either $(a, b) = (I, J)$ or $(a, b) = (J, I)$.

The last case to consider is when $y = x$. In that case

$$\mathbb{P}(X_t = x \mid X_{t-1} = x) = \mathbb{P}(I = J) = \frac{1}{n}.$$

Summing up gives

$$\frac{1}{n!}\sum_y \mathbb{P}(X_t = x \mid X_{t-1} = y) = \frac{1}{n!}\left[\binom{n}{2}\frac{2}{n^2} + \frac{1}{n}\right] = \frac{1}{n!}\left[\frac{2n(n+1)}{2n^2} + \frac{1}{n}\right] = \frac{1}{n!}.$$

□

13.2 Implementing the transposition chain in R

To implement this update function in R, simply supply the chain with the current state and the tuple of positions to be swapped. The `rev` command reverses the order of a vector.

```
tr_chain <- function(x, swap) {
  w <- x
  w[swap] <- x[rev(swap)]
  return(w)
}
```

Check that this is working:

```
tr_chain(c(5, 1, 4, 3, 2), c(2, 4))
```

```
## [1] 5 3 4 1 2
```

Starting from any permutation, taking a number of random swaps will leave the state close to stationarity.

```
x <- c(1, 2, 3, 4, 5)
n <- length(x)
for (i in 1:1000)
  x <- tr_chain(x, floor(runif(2) * n) + 1)
print(x)
```

```
## [1] 5 1 4 3 2
```

This simple chain on permutations does not accomplish anything that cannot be done with simpler methods, but more sophisticated Markov chains will allow approximate sampling from distributions where no other way exists.

Problems

13.1: Consider the update function on state space $\{0, 1, \ldots, n-1\}$ so that either adds 1 or subtracts 1 mod n with probability 0.3, and with probability 0.4 stays where it is. Let

$$f(u) = \mathbb{I}(u > 0.7) - \mathbb{I}(u < 0.3),$$

and

$$\phi(x, u) = x + f(u) - n\mathbb{I}(x + f(u) = n) + n\mathbb{I}(x + f(u) = -1).$$

Suppose $n = 10$.

a) Write an R function that takes a state x, standard uniform u, and n and returns $\phi(x, u)$ over $\{0, \ldots, n-1\}$.

b) Given $X_0 = 0$, find X_{100} 1000 times, and report an estimate for $\mathbb{E}[X_{100}]$ as $a \pm b$.

13.2: For the chain of the previous problem with state space $\Omega = \{0, 1, \ldots, 9\}$,

a) If $X_0 \sim \mathsf{Unif}(\Omega)$, find

$$\mathbb{P}(X_1 = 5).$$

b) More generally, show that for all $i \in \Omega$),

$$\mathbb{P}(X_1 = i) = 1/10.$$

c) What does that tell us about the stationary distribution of the chain?

13.3: Suppose $\{X_i\}$ is a Markov chain has a stationary distribution over $\{0, 1, 2\}^{10}$ that is uniform over states with $\sum_{i=1}^{10} X_i \leq 7$. Steps in the chain connect any states, and the chain is aperiodic. What can be said about the limiting distribution of the Markov chain, and how do you know this?

13.4: A Markov chain on three states $\{a, b, c\}$ is connected and aperiodic, and has a stationary distribution with $\pi(a) = \pi(b) = 0.2, \pi(c) = 0.6$. What can you say about the distribution of $\lim_{t\to\infty} X_t$ and why?

Chapter 14

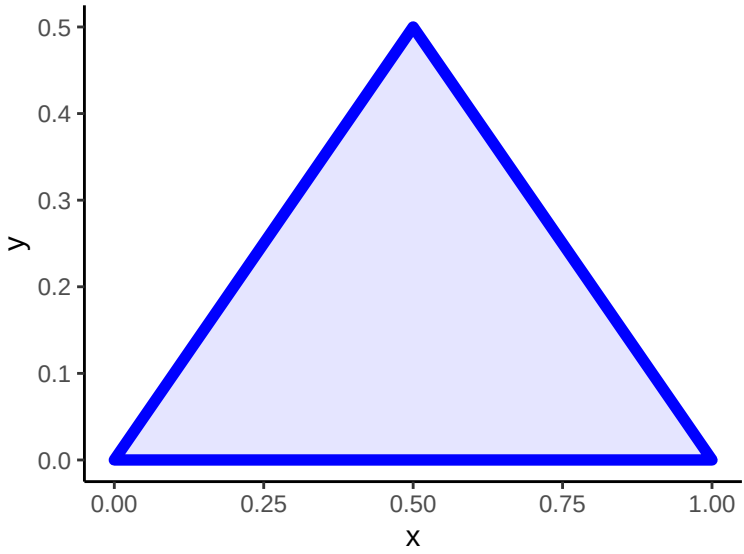
Gibbs samplers

Question of the Day

Design a Markov chain whose state space is

$$\Omega = \{(x, y) : 0 \leq x \leq 1, 0 \leq y \leq x\mathbb{I}(x \leq 0.5) + (1 - x)\mathbb{I}(x > 0.5)\}.$$

This region looks like this:



Summary

- **Gibbs sampling** is a protocol for building a Markov chain with a particular stationary distribution π. It works by selecting one or more dimensions of the current state, erasing their values, and then filling the missing values randomly using π conditioned on the remaining values of the state.

Earlier AR was used to sample from problems like the Question of the Day, now a Markov chain will be constructed.

14.1 The idea of the Gibbs sampler

Let $f(x) = x\mathbb{I}(x \leq 0.5) + (1-x)\mathbb{I}(x > 0.5)$. Then a state of the chain has two components, 1 and 2. Let (X_1, X_2) be the current state of the chain. Then note that given X_1, the distribution of X_2 is uniform from 0 up to $f(X_1)$. Given X_2, the distribution of X_1 is slightly trickier, but can quickly be solved to get

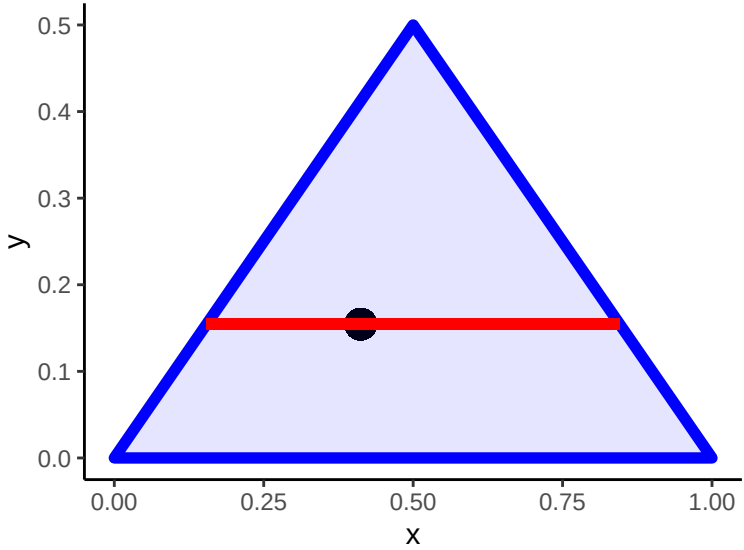
$$[X_1 \mid X_2] \sim \mathsf{Unif}\left([X_2, 1 - X_2]\right).$$

So the Gibbs sampler works as follows. Suppose the current state is (X_1, X_2). First, draw X_1 conditioned on X_2, and then draw X_2 conditioned on X_1.

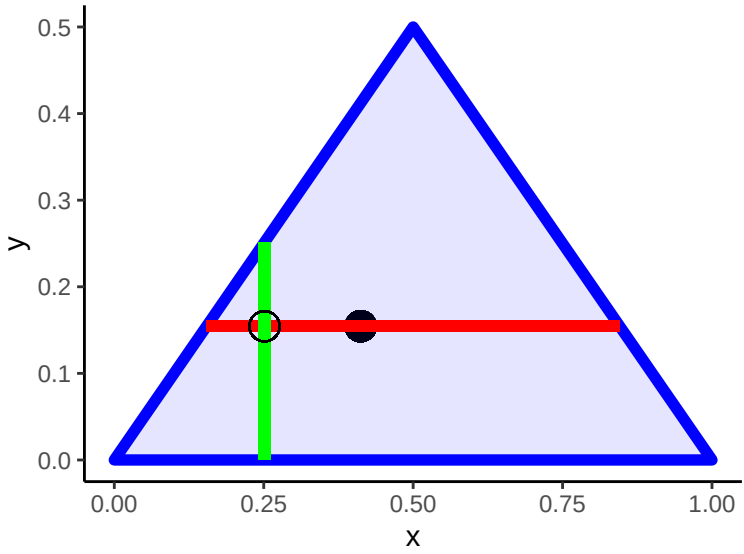
Algorithm 1
QotD_Gibbs (X_1, X_2)

1. Draw X_1 uniformly from $[X_2, 1 - X_2]$.
2. Draw X_2 uniformly from $[0, f(X_1)]$.

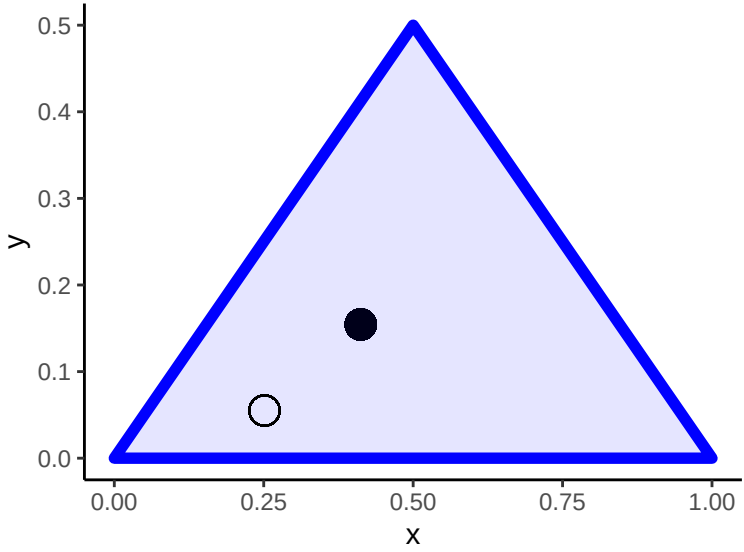
For instance, suppose the current state is $(0.4213, 0.1544)$ (kept to four sig figs for convenience.) Then first draw a new value for (0.4213) that is uniformly from 0.1544 up to 0.8456. Pictorially, this means drawing a new value for X_1 that is uniform on the red dotted line.



Perhaps the new value chosen is 0.2513. Then the value of y is chosen to lie between 0 and 0.2513. (See picture below.)



Suppose value 0.0550 was chosen on the green line. This one step of the Markov chain is complete, and the new state is $(0.2513, 0.0550)$.



Statisticians find it useful to have a notation for removing one component from an n-tuple. The notation x_{-i} means $x_1, \ldots, x_{i-1}$ together with $x_{i+1}, \ldots, x_n$. That is, x_{-i} is the vector $x_1, \ldots, x_n$ with the ith component removed. For example if $x = (3, 5, 1, 2, 2)$, then $x_{-2} = (3, 1, 2, 2)$.

Note: this notation works in R as well!

```
c(3, 5, 1, 2, 2)[-2]
```

```
## [1] 3 1 2 2
```

By using this notation, the Gibbs sampler can be described as follows.

Algorithm 2
Deterministic_scan_Gibbs_sampler $(x_1, \ldots, x_n)$

1: For i from 1 to n do

2: Replace x_i with a draw from π conditioned on x_{-i}.

3: Return x

Note that this entire algorithm is one step in the Markov chain. It is called *deterministic scan* because all the components are run through in order from 1 through n.

This can sometimes lead to problems if the order of the dimensions matters. A *random permutation scan* permutes the components first uniformly at random before changing them.

Algorithm 3
Random_permutation_scan_Gibbs_sampler $(x_1, \ldots, x_n)$

1: Let s be a uniform permutation on n elements

2: For i from 1 to n do

3: Replace $x_{s(i)}$ with a draw from π conditioned on $x_{-s(i)}$.

4: Return x

You do not want to use this method when n is small, as there will be a $1/n$ chance of using the same component at the end of one step as at the beginning of the next step.

A third alternative is to just randomly pick a component to change.

Algorithm 4
Random_scan_Gibbs_Sampler$(x_1, \ldots, x_n)$

1: Choose i uniformly from 1 to n

2: Replace x_i with a draw from π conditioned on x_{-i}.

3: Return x

This has a $1/n$ chance of picking the same component to update twice in a row. Not a big deal when n is large, but is never used for this reason when $n = 2$.

One step in any version of the Gibbs sampler is stationary with respect to π.

To work for all three types of Gibbs sampler, it is necessary to show that updating one component is a stationary step. Let $X = (X_1, \ldots, X_n)$. Then marginally, X_{-i} comes

from π restricted to components $\{1, \ldots, i-1, i+1, \ldots, n\}$. Then by definition, drawing $[X_i \mid X_{-i}]$ from π restricted to components $\{1, \ldots, i-1, i+1, \ldots, n\}$ is a draw from π.

14.2 The Ergodic Theorem for continuous state spaces

With a continuous (or infinite) state space, it is necessary to have slightly stronger conditions for the limiting distribution to equal to the stationary distribution.

Definition 36
For a distribution ϕ over the state space, a chain is **ϕ-connected** (aka ϕ-irreducible) if for every measurable set A with $\phi(A) > 0$, and every state x, there is a positive integer n such that the probability of moving from x to A in n moves is positive.

The equivalent of aperiodicity is similar.

Definition 37
For a ϕ-connected chain, the chain is **ϕ-aperiodic** if for every measurable set A with $\phi(A) > 0$, and every state x, there is a positive integer N such that for all $n \geq N$, the probability of moving from x to A in n moves is positive.

Then the Ergodic Theorem is slightly weaker than the finite state case in that there is not a guarantee of a stationary distribution, but if such a stationary distribution π exists, and the chain is both π-connected and π-aperiodic, then the limiting distribution will be π as well.

Theorem 8
The Fundamental Theorem of Markov chains
If there is a stationary distribution π for a chain that is π-connected and π-aperiodic, then for a set Ω' with $\pi(\Omega') = 1$, for all $x_0 \in \Omega'$,

$$[X_t | X_0 = x_0] \to \pi$$

in distribution.

Note: This works for both countably infinite state spaces and for continuous spaces.

14.3 Discrete Gibbs sampler

The Gibbs sampler in the Question of the Day was over a continuous state space, but it also works for discrete spaces as well.

Example 27
Construct a Gibbs sampler over $\Omega = \{1,2,3\}^{10}$ where the target density is

$$f(x_1,\ldots,x_{10}) = \left[C\sum_{i=1}^{10} x_i\right]\mathbb{I}(x \in \Omega).$$

Answer Consider a concrete example. If the current state is $(3,2,1,3,3,2,1,2,2,1)$, and the goal is to replace component 3, the remaining vector is $x_{-3} = (3,2,\_,3,3,2,1,2,2,1)$. Note $\sum_{j\neq i} x_{-3} = 19$. Then

$$\begin{aligned}\mathbb{P}(x_3 = i \mid x_{-3} = ((3,2,\_,3,3,2,1,2,2,1)) &= \frac{\mathbb{P}(x = (3,2,i,3,3,2,1,2,2,1))}{\mathbb{P}(x_{-3} = ((3,2,\_,3,3,2,1,2,2,1)))} \\ &= \frac{C(i+19)}{C(1+19)+C(2+19)+C(3+19)},\end{aligned}$$

since 1, 2, or 3 are the only possible values for x_3. Note that the C cancels out: this should always happen in your Gibbs step. The result is that

$$\mathbb{P}(x_3 = i \mid x_{-3} = ((3,2,\_,3,3,2,1,2,2,1)) = \frac{i+19}{6+3\cdot 19}.$$

So the Deterministic Scan Gibbs sampler is as follows.
dsgs_example(x)

1. For i from 1 to 10
2. Draw x_i from density

$$f(i) = \frac{i+s}{6+3s}\mathbb{I}(i \in \{1,2,3\}),$$

where $s = \sum_{j\neq i} x_i$

3. Return x

14.4 How many steps to run?

Recall that by ensuring stationarity, all that is guaranteed is that the chain will eventually (given an infinite amount of time) approach the target distribution. Unfortunately, there is really no way to know for sure how long that time is.

Therefore, when actually running a Markov chain, it is customary to devote some of the steps of the Markov chain to *burn in*, and some steps to gathering statistics of interest. For instance, one might run for 10,000 steps of burn in, hopefully reaching a point where the current state is stationary, and then run for 10,000 steps to take data.

Early practitioners often used burn in that was one tenth of the total number of steps,

but a better number to use is one half like in the example above. The reason for this is that if your burn in is insufficient, your sample will be biased at the beginning, and it is easier to reduce the bias by doing more burn in than taking more statistical steps.

On the other hand, by limiting the burn in to one half of the steps, the most that you are losing in the running time if your burn in is too long is a factor of 2, which is not too bad.

14.5 Gibbs sampling in R

So let's consider using the chain for the Question of the Day to estimate the probability that $X_1 \geq 0.7$. (Of course, since this is a toy example, this value can be found exactly to be 0.18.)

Consider a Gibbs update for the question of the day chain that takes as input the current state and two uniforms. The first uniform can be used to update `x[1]`, and the second uniform can be used to update `x[2]`.

Recall that $[X_1 \mid X_2] \sim \text{Unif}([X_2, 1 - X_2])$. This interval has width $1 - 2X_2$. Similarly, $[X_2 \mid X_1] \sim \text{Unif}([0, X_1\mathbb{I}(X_1 \leq 0.5) + (1 - X_1)\mathbb{I}(X_1 > 0.5)])$, so is easy to scale.

```
step_gibbs_qotd <- function(x, u_1, u_2) {
  x[1] <- u_1 * (1 - 2 * x[2]) + x[2]
  x[2] <- u_2 * (x[1] * (x[1] <= 0.5) +
                   (1 - x[1]) * (x[1] > 0.5))
  return(x)
}
```

Okay, now to undertake a burn in of 1,000 steps followed by a taking of 1,000 steps of data.

```
steps <- 1000

gibbs_qotd_data <- function(steps) {
  burnin    <- steps
  datasteps <- steps
  x <- c(0, 0)
  res <- rep(0, datasteps)
  u_1 <- runif(burnin)
  u_2 <- runif(burnin)
  for (i in 1:burnin)
    x <- step_gibbs_qotd(x, u_1[i], u_2[i])
  u_3 <- runif(datasteps)
  u_4 <- runif(datasteps)
  for (i in 1:datasteps) {
    res[i] <- (x[1] > 0.7)
    x <- step_gibbs_qotd(x, u_3[i], u_4[i])
```

```
  }
  return(res)
}
```

```
res <- gibbs_qotd_data(100)
mean(res)
```

```
## [1] 0.13
```

To understand the standard error, repeat the entire process multiple times and record the result.

```
res <- replicate(5, mean(gibbs_qotd_data(1000)))
tibble(
  est_mean = mean(res),
  est_sd   = sd(res) / sqrt(length(res))
)
```

```
## # A tibble: 1 x 2
##   est_mean  est_sd
##      <dbl>   <dbl>
## 1    0.172 0.00582
```

So the estimate is 0.172 ± 0.006.
Running for more steps increasing the tightness.

```
res2 <- replicate(5, mean(gibbs_qotd_data(10^4)))
tibble(
  est_mean = mean(res2),
  est_sd   = sd(res2) / sqrt(length(res2))
)
```

```
## # A tibble: 1 x 2
##   est_mean  est_sd
##      <dbl>   <dbl>
## 1    0.178 0.00106
```

Note that the gain from running ten times as many steps resulted in a much tighter bound on error than expected by the general $1/\sqrt{n}$ rule.

Utilizing AR in Gibbs steps

When sampling from a complex distribution, one or more of the $[X_i \mid X_{-i}]$ steps can be difficult to handle. In this case, using AR to obtain the one dimensional distribution could be helpful.

As an example consider creating a Markov chain whose goal is to sample uniformly from the area underneath

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

For $(X, Y) \sim \text{Unif}(\Omega)$, $[Y|X] \sim \text{Unif}([0, X^2(1 - X)])$, but $[X|Y]$ has a more complicated relationship. The cubic equation can be used to solve the inequality for X given Y, but who remembers the cubic equation?

So to generate from X, just use AR together with draws that are uniform from $[0, 1]$. Unfortunately, because this requires a random number of draws, the ability to write the Markov chain as a simple update function is lost: the random draws need to be made inside the chain.

```
step_gibbs_Omega <- function(p) {
  p[2] <- p[1]^2 * (1 - p[1]) * runif(1)
  a <- FALSE
  while(!a) {
    x <- runif(1)
    a <- (p[2] <= x^2 * (1 - x))
  }
  p[1] <- x
  return(p)
}
```

To estimate $\mathbb{P}(X > 0.5)$, run the following

```
gibbs_Omega_data <- function(steps) {
  burnin    <- steps
  datasteps <- steps
  x <- c(0, 0)
  res <- rep(0, datasteps)
  for (i in 1:burnin)
    x <- step_gibbs_Omega(x)
  for (i in 1:datasteps) {
    res[i] <- (x[1] > 0.5)
    x <- step_gibbs_Omega(x)
  }
  return(res)
}
```

```
res3 <- replicate(5, mean(gibbs_Omega_data(100)))
tibble(
  est_mean = mean(res3),
  est_sd   = sd(res3) / sqrt(length(res3))
)
```

```
## # A tibble: 1 x 2
##   est_mean est_sd
##      <dbl>  <dbl>
## 1     0.64 0.0158
```

So the estimate is 0.640 ± 0.016.
Running for more steps increasing the tightness.

```
res4 <- replicate(5, mean(gibbs_Omega_data(10^3)))
tibble(
  est_mean = mean(res4),
  est_sd   = sd(res4) / sqrt(length(res4))
)
```

```
## # A tibble: 1 x 2
##   est_mean  est_sd
##      <dbl>   <dbl>
## 1    0.684 0.00589
```

So with one thousand burn in and data gathering steps, the result was $\boxed{0.684 \pm 0.006}$.

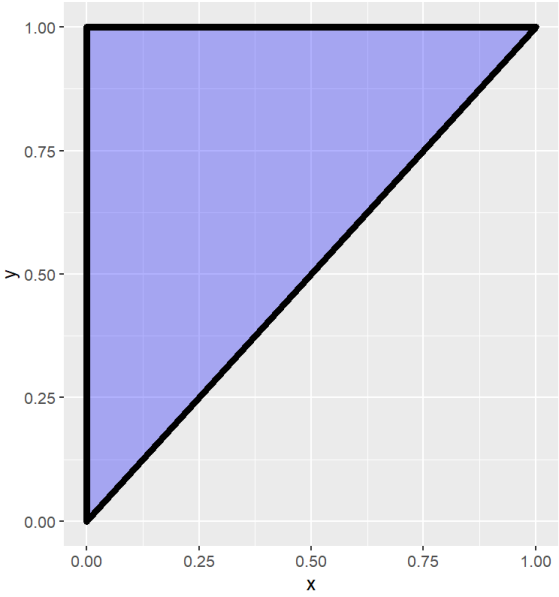
Problems

14.1: Consider the transposition chain. Suppose the goal is to use this chain to estimate $\mathbb{P}(x(1) \leq x(n))$, where $x \sim \mathsf{Unif}(\mathcal{S}_n)$. For $n = 10$, and $t \in \{10, 50, 100\}$, try running a Markov chain for t burn in steps and t data collecting steps to estimate this probability.

Repeat your Markov chain runs 10 times and report your estimate as $a \pm b$.

14.2: Using the transposition chain above with 1000 burnin steps and 1000 data gathering steps to estimate the probability that $x(1) \leq x(3)$ and $x(1) \leq x(7)$ when $n = 10$. Replicate your chain run 5 times, and report your answer as $a \pm b$.

14.3: Suppose that (X, Y) is uniform over the triangle in $\mathbb{R}^2$ with vertices $(0, 0)$, $(0, 1)$, and $(1, 1)$.



a) What is the distribution of X given Y?

b) What is the distribution of Y given X?

14.4: Suppose that (A, B) is uniform over the area inside the unit circle:

$$\{(a, b) : a^2 + b^2 \leq 1\}.$$

a) What is $[A|B]$?

b) What is $[B|A]$?

14.5: Implement a random scan Gibbs sampler for a Markov chain which is uniform over

$$\Omega_{10} = \left\{(x_1, \ldots, x_{10}) \in \{0, 1, 2\}^{10} : \sum_{i=1}^{10} x_i \leq 7\right\}$$

as an update function in R that takes as input the current state x, a dimension i in $\{1, \ldots, 10\}$, and a standard uniform u and returns the next state of the chain.

14.6: Implement a random scan Gibbs sampler for a Markov chain which is uniform over

$$\Omega_5 = \left\{(x_1, \ldots, x_5) \in \{-1, 0, 1\}^5 : -3 \leq \sum_{i=1}^{5} x_i \leq 3\right\}$$

as an update function in R that takes as input the current state x, a dimension $i \in \{1, \ldots, 5\}$, and a standard uniform u and returns the next state of the chain.

14.7: Continuing the earlier problem, using 1000 burn in and 1000 data gathering steps, estimate $\sum_{i=1}^{10} X_i$ for $(X_1, \ldots, X_{10}) \sim \mathsf{Unif}(\Omega_{10})$. Repeat your Markov chain 5 times and report your estimate as $a \pm b$.

14.8: Continuing the earlier problem, using 1000 burn in and 1000 data gathering steps, estimate $\sum_{i=1}^{10} X_i$ for $(X_1, \ldots, X_{10}) \sim \mathsf{Unif}(\Omega_5)$. Repeat your Markov chain 5 times and report your estimate as $a \pm b$.

14.9: Create a random scan Gibbs sampler that has stationary distribution uniform over the six dimensional unit hypersphere, that is

$$\{(x_1, \ldots, x_6) : x_1^2 + \cdots + x_6^2 \le 1\}.$$

Implement your sampler as an R function that inputs the current state 'x', a dimension 'i', a standard uniform 'u', and returns the next state in the Markov chain.

14.10: Create a random scan Gibbs sampler that has stationary distribution uniform over the volume:

$$V = \{(x_1, \ldots, x_6) \in [0,1]^6 : x_1 + \cdots + x_6 \le 3\}.$$

Implement your sampler as an R function that inputs the current state 'x', a dimension 'i', a standard uniform 'u', and returns the next state in the Markov chain.

14.11: Create a deterministic scan Gibbs sampler that has stationary distribution uniform over the six dimensional unit hypersphere, that is

$$\{(x_1, \ldots, x_6) : x_1^2 + \cdots + x_6^2 \le 1\}.$$

Implement your sampler step as an R function that inputs the current state 'x', a vector of six iid standard uniforms 'u', and returns the next state in the Markov chain.

14.12: Create a deterministic scan Gibbs sampler that has stationary distribution uniform over the volume:

$$V = \{(x_1, \ldots, x_6) \in [0,1]^6 : x_1 + \cdots + x_6 \le 3\}.$$

Implement your sampler step as an R function that inputs the current state 'x', a vector of six iid standard uniforms 'u', and returns the next state in the Markov chain.

Chapter 15

Gibbs samplers for spatial point processes

Question of the Day

Design a Gibbs Sampling Markov chain for the Ising model on a graph $G = (V, E)$. The state space is

$$\Omega = \{0, 1\}^V$$

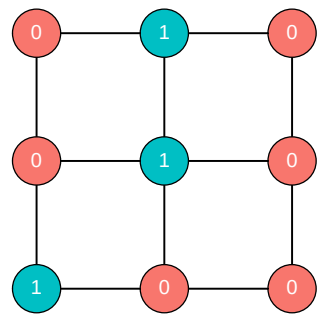
and the unnormalized density is (given parameter β)

$$f(x) = \exp\left(\beta h(x)\right).$$

Here $h(x)$ counts the number of edges that have the same label on the endpoints. It can be expressed mathematically as

$$h(x) = \sum_{\{i,j\} \in E} \mathbb{I}(x(i) = x(j)).$$

For instance, on a 3 by 3 square lattice, one might have the following random draw.



Because there are 5 edges whose endpoints have the same label, here $h(x) = 5$ and this particular configuration has weight $\exp(5\beta)$.

Summary

- A **Gibbs sampling chain for spatial processes** typically updates one component at a time. Usually the conditional distribution for a node only depends upon the immediate neighbors.

Libraries

This chapter will use several libraries, such as the `tidyverse`, `tidygraph`, and `igraph` libraries for making graphs. The package `ggraph` will be needed for displaying graphs.

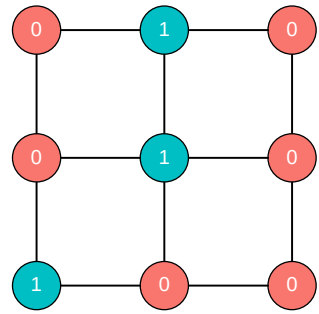
```
library(tidyverse)
library(igraph)
library(tidygraph)
library(ggraph)
```

15.1 Visualizing the Gibbs sample

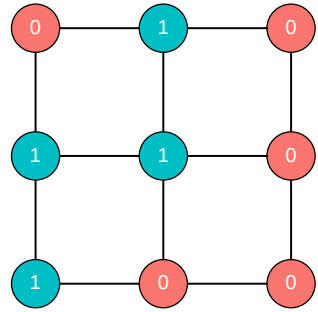
The Gibbs sampler can be thought of as a Markov chain where at each step one (or more) of the components are erased, and the value at that component is redrawn conditioned on the values of the rest of the components.

Suppose in the 3 by 3 lattice example above that the node in the middle row on the left hand column (call this node 4) is decided to be removed. Then there are two possible values for that node. The node can be set to 0 or it can be set to 1.

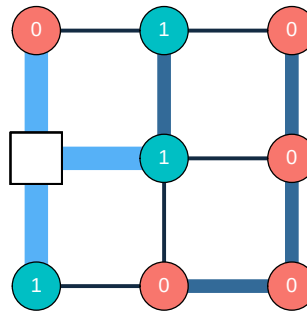
Let $x_{4\to 0}$ denote the state with node 4 valued at 0.



Let $x_{4\to 1}$ denote the state with node 4 valued at 1.



Then $h(y) = 5$, and $h(w) = 6$. Note that the thin edges marked in dark blue contribute to both $h(y)$ and $h(w)$ by 3. The thicker edges marked in light blue might contribute to h depending on the value of node 4.



Note that the four dark blue, medium width edges will always contribute 4 to h no matter what the value of node 4 is. If node 4 is set to 0, one of the thick light blue edges will add 1 to h, otherwise, two of the thick light blue edges will add to h.

Mathematically, let n_0 count the number of neighbors of node 4 that are labeled 0, and n_1 count the number of neighbors of node 4 that are labeled 1. That is for $n_0 = 1$ and $n_1 = 2$ counting the number of neighbors of 4 with labels 0 and 1 respectively,

$$h(x_{4\to 0}) = 4 + n_0$$
$$h(x_{4\to 1}) = 4 + n_1.$$

Let Y be the next state of the chain. Then the Gibbs sampler says to choose Y from the distribution given by density f conditioned on $Y \in \{x_{4\to 0}, x_{4\to 1}\}$. Then using the

conditional probability formula:

$$\begin{aligned}\mathbb{P}(Y = x_{4\to 0} \mid Y \in \{x_{4\to 0}, x_{4\to 1}\}) &= \frac{\mathbb{P}(Y = x_{4\to 0})}{\mathbb{P}(Y \in \{x_{4\to 0}, x_{4\to 1}\})} \\ &= \frac{\exp(\beta(h(x_{4\to 0})))}{\exp(\beta(h(x_{4\to 0}))) + \exp(\beta(h(x_{4\to 1})))} \\ &= \frac{\exp(\beta(4+n_0)}{\exp(\beta(4+n_0) + \beta(4+n_1)} \\ &= \frac{\exp(4\beta)\exp(\beta n_0)}{\exp(4\beta)\exp(\beta n_0) + \exp(4\beta)\exp(\beta n_1)}.\end{aligned}$$

Now something great happens! Both the numerator and the denominator have a factor of $\exp(4\beta)$ in them that disappears.

$$\mathbb{P}(Y = x_{4\to 0} \mid Y \in \{x_{4\to 0}, x_{4\to 1}\}) = \frac{\exp(\beta n_0)}{\exp(\beta n_0) + \exp(\beta n_1)}.$$

Of course, that factor would have canceled regardless of the labels on the edges not adjacent to node 4. In other words, the probability of moving to a 0 or a 1 for the Ising model only depends on the *neighbors* of the node that is being changed, the rest of the state does not matter. This reliance on the local properties of the graph helps make this Markov chain step much faster. This gives the following algorithm.

Step_Gibbs_Ising (x, i, u)

1. Let n_0 be the number of neighbors of i labeled 0, and n_1 be the number of neighbors of i labeled 1.
2. Let $x(i) \leftarrow \mathbb{I}(u > \exp(\beta n_0)/[\exp(\beta n_0) + \exp(\beta n_1)])$.

If the input i is chosen uniformly from the nodes, and input u is a standard uniform, then this step is stationary with respect to the density f.

15.2 Simulating Spatial Processes in R

If the entire n dimensional state is needed for the update, the update will take $\Theta(n)$ time.

For graph $G = (V, E)$ and $i \in V$, let $\mathcal{N}(i) = \{j : \{i, j\} \in E\}$ be the **neighbors of** i. Let $\deg(i) = \#(\mathcal{N}(i))$ be the **degree of** i.

The goal is to create an update for node i that only needs time of the order of the number of neighbors of i. That is, the goal is an update that can be done in $\Theta(\deg(i))$ time. So only pass to the state what is needed to update the current state, the values of the neighbors of that state. That will be input as `x_n`.

```
step_local_gibbs <- function(x_n, i, u, beta) {
  n_0 = sum(x_n == 0)
  n_1 = sum(x_n == 1)
  return(as.integer(u < (exp(beta * n_1) /
                         (exp(beta * n_1) + exp(beta * n_0)))))
}
```

The data type that will be used for graphs will be the *tidygraph*, which is a data type that comes from the package of the same name.

The following evaluates h for a graph g with configuration x in the tidygraph framework. It works by pulling out the source of the edges (the `from` variable) and sink of the edges (the `to` variable) and comparing their `x` value.

```
h <- function(x, g) {
  from <- g |> activate(edges) |> pull(from)
  to <- g |> activate(edges) |> pull(to)
  return(sum(x[from] == x[to]))
}
```

Now create a function to burn in and then collect data from a Markov chain that keeps at each time step the value of h.

```
gibbs_local <- function(steps, g, beta) {
  burnin    <- steps
  datasteps <- steps
  n <- g |> activate(nodes) |> as_tibble() |> nrow()
  x <- rep(0, n)
  u1 <- runif(burnin)
  i1 <- floor(n * runif(burnin)) + 1
  res <- rep(0, datasteps + 1)
  for (i in 1:burnin) {
    x_n <- x[neighbors(g, i1[i])]
    x[i1[i]] <- step_local_gibbs(x_n, i1[i], u1[i], beta)
  }
  res[1] <- h(x, g)
  u2 <- runif(datasteps)
  i2 <- floor(n * runif(datasteps)) + 1
  for (i in 1:datasteps) {
    x_n <- x[neighbors(g, i2[i])]
    h_current <- sum(x_n == x[i2[i]])
    x[i2[i]] <- step_local_gibbs(x_n, i2[i], u2[i], beta)
    h_new <- sum(x_n == x[i2[i]])
```

```
    res[i + 1] <- res[i] - h_current + h_new
  }
  return(res)
}
```

The following makes a graph that is a 4 by 4 square lattice.

```
g <- create_lattice(c(4, 4))
```

Now run the chain on graph g for 1000 burnin and 1000 data collecting steps with $\beta = 0.5$. The goal is to estimate the probability that the value of $h(x)$ for a draw from the Ising model is at most 14. The whole MCMC run is repeated 5 times to get a sense of the variance of the procedure.

```
res_local <-
  replicate(5,
            mean(gibbs_local(1000, g, 0.5) <= 14))
mean(res_local)
```

```
## [1] 0.4291708
```

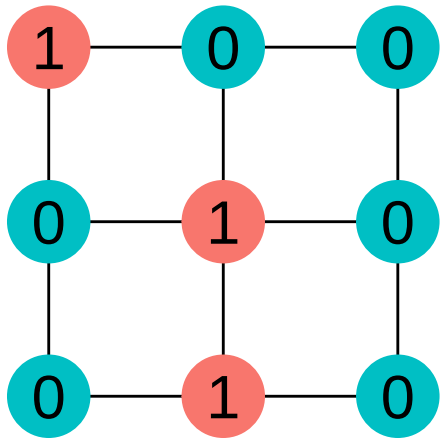
```
sd(res_local) / sqrt(length(res_local))
```

```
## [1] 0.02786228
```

The result is an estimate of $\boxed{0.43 \pm 0.03}$.

Problems

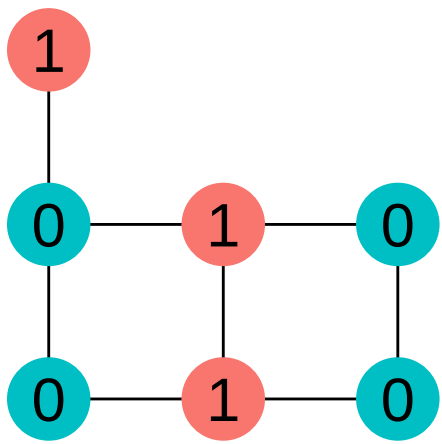
15.1: Consider an Ising model with state



Label the node that is in the center of the top row node 2. Suppose a random scan Gibbs chain selects node 2 to be updated. For $\beta = 1.2$, find exactly the probability that node 2 will be updated by the Gibbs step to have label 0.

15.2: Continuing the last problem with the same graph and state, suppose the node in the upper right corner is updated in a random scan Gibbs chain. What is the chance that the update changes the value of the node to 1?

15.3: Consider the following graph. If a random scan Gibbs chain for the Ising model is run on this graph, what is the maximum number of node labels that need to be examined to take one step in the chain?



15.4: Consider the graph with nodes $\{a, b, c, d, e\}$ and edges

$$\{\{a, b\}, \{b, c\}, \{c, d\}, \{d, e\}, \{e, a\}\}.$$

What is the maximum number of nodes that need to be examined to take one random scan Gibbs chain for the Ising moel?

15.5: Consider the following unnormalized density for $x = (x_1, \ldots, x_{10})$.

$$g(x) = (x_1x_2 + x_2x_3 + x_3x_4 + \cdots + x_9x_{10})\mathbb{I}(x \in [0, 4]^{10}).$$

Note that x_{10} only appears in the density with x_9, so suppose a Markov chain is run where g, x_9, and x_10 are known at the current state of the chain. Suppose a step is taken in the Gibbs chain that replaces the tenth component.

For instance, if $g(x) = 64.2$, $x[9] = 3.3$, and $x[10] = 1.2$, then setting $y_i = x_i$ for $i < 10$, and $y_{10} = A$ where A is a random variable, then consider $f_A(s)$, the density of A. Given these values, the sum of the first eight terms in g would be the sum of all the terms minus the last term, so

$$\left[\sum_{i=1}^{8} x_ix_{i+1}\right] = 64.2 - (3.3)(1.2) = 60.24.$$

so the unnormalized density of A is

$$g_A(y_{10}) = (60.24 + 3.3y_{10})\mathbb{I}(y_{10} \in [0, 4])$$

Give a function h such that for $U \sim \text{Unif}([0, 1])$, $h(U) \sim A$.

15.6: Using the unnormalized density from the previous problem, suppose that $X \sim g$, that $\sum X_i = 62$, $X_1 = 0.8$ and $X_3 = 2.9$. What is the density of X_2?

15.7: Continuing the last problem, you need to know g and some of the x_i values to calculate the probabilities for the changed dimension in the Gibbs chain. What is the largest number of x_i values that you need to know?

15.8: Suppose you use a random scan Gibbs chain to sample from unnormalized density

$$g(x) = \sum_{i=1}^{n-1} x_ix_{i+1}$$

What is the order of the time needed to take one step in the chain?

Chapter 16

Random walks on finite groups

Question of the day

Consider a standard deck of 52 cards. There are a multitude of ways to shuffle such a deck, include the commonly used pass and riffle shuffling. But even using those methods, each person does them in a slightly different fashion. Why does pretty much every way of shuffling cards work?

Summary

- In mathematics a **group** is a set equipped with a group operation such that any member of the group can be transformed by another member of the group to get a third member of the group. This operation has to be associative, and for every group member, there has to be another group member that undoes the transformation of the first.
- A **random walk on a finite group** is a Markov chain that at each step picks a random member of the group to apply that transformation to the current state to get the next state.
- Random walks on a finite group always have a uniform stationary distribution.
- A random walk is symmetric if the probability of moving from state a to b is the same as moving from state b to a.
- Symmetric random walks conditioned to lie inside a subset of the group also have a uniform stationary distribution.

Libraries

This chapter will utilize the **tidygraph** and **ggraph** packages to illustrate some Markov chains.

```
library(tidygraph)
library(ggraph)
```

Mathematically, shuffling a deck of cards is running a Markov chain whose stationary distribution is the uniform distribution over all 52! permutations of the cards.

There are many different ways of shuffling cards. Riffle shuffling where the deck is split in half and the two sides are randomly interlaced. Pass shuffling where a clump of cards is taken from the bottom or middle of the deck and put on top in reverse order. Fifty-two card pickup.

Often people will use more than one method in shuffling the deck during their turn. And the interesting thing is, they all work! That is, pretty much *any* way of shuffling cards ends up with a uniform permutation. The goal today is to answer why.

16.1 The symmetric group

A *group* is a set with certain properties. Each member of the group can be thought of as an element of the set. But each element of the group can also be thought of as a particular transformation of the set.

The permutations of $(1, 2, \ldots, n)$ are also called the *symmetric group*. Elements of the set have this dual nature, in that they can be viewed both of a particular element of the set and a way of transforming other elements!

For instance, the permutation

$$(1, 2, 6, 4, 5, 3)$$

can be thought of as a particular permutation of cards labeled 1 through 6.

It can also be thought of as a transformation: if I apply this permutation to another permutation, swap the third and the sixth positions. So for instance,

$$(5, 4, 1, 3, 6, 2) \bullet (1, 2, 6, 4, 5, 3) = (5, 4, 2, 3, 6, 1).$$

The symbol $\bullet$ means to apply the transformation given by the permutation on the right to the state on the left. If you have time and enough energy, you can prove that this type of transformation has several important properties.

- First, the operation is *associative*, that is, for permutations τ_1, τ_2 and τ_3,

$$(\tau_1 \bullet \tau_2) \bullet \tau 3 = \tau_1 \bullet (\tau_2 \bullet \tau_3).$$

- Second, there is an *identity permutation* i such that for any other permutation τ it holds that $\tau \bullet i = \tau$. The identity permutation just leaves cards in place, so it is $(1, 2, \ldots, n)$.

- Third, for every permutation, there is another permutation that undoes the effects of the first. That is, for every τ there is a τ^{-1} such that $\tau \bullet \tau^{-1} = i$. Note that $(1, 2, 6, 4, 5, 3)$ is its own inverse, as applying the transformation twice swaps the third and sixth cards back into their original positions!

Let's make an official definition.

A set G together with a binary operation $\bullet$ is a **group** if the following statements hold.

1) *Closure*: $(\forall a, b \in G)(a \bullet b \in G)$.
2) *Associativity*: $(\forall a, b, c \in G)((a \bullet b) \bullet c = a \bullet (b \bullet c))$.
3) *Identity*: $(\exists i \in G)(\forall a \in G)(a \bullet i = a)$.
4) *Inverses*: Say that a is an **inverse** of b if $a \bullet b = b \bullet a = i$. Then $(\forall a \in G)(\exists b \in G)(a \bullet b = b \bullet a = i)$. (Write $b = a^{-1}$.)

This type of mathematical object arises in many different places.

Examples of groups and the group operation include:

- $G = \mathcal{S}_n$ and $[\tau_1 \bullet \tau_2](i) = \tau_1(\tau_2(i))$.
- $G = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ and $i \bullet j = i + j$.
- $G = \mathcal{Q}_0 \setminus \{\}$ and $a \bullet b = (a)(b)$.
- G is the set of n by m matrices with invertible entries, and $A \bullet B = A + B$.
- G is the set of 2 by 2 invertible matrices with real entries, and $A \bullet B = AB$.

Once you have the rules for groups, it is possible to prove fun facts like $(g^{-1})^{-1} = g$.

For g in a group G, $(g^{-1})^{-1} = g$.

Let $g \in G$. Then $g \bullet g^{-1} = g^{-1} \bullet g = i$, so g fits the definition of the inverse of g^{-1}. That is, $g = (g^{-1})^{-1}$.

Another useful fact is that inversion flips the order of the $\bullet$ operator.

For g_1, g_2 elements of a group g, $(g_1 \bullet g_2)^{-1} = g_2^{-1} \bullet g_1^{-1}$.

Again the proof just comes from verifying that $g_2^{-1} \bullet g_1^{-1}$ has the desired properties of an inverse using the associativity and identity rules:

$$\begin{aligned}(g_1 \bullet g_2) \bullet (g_2^{-1} \bullet g_1^{-1}) &= g_1 \bullet (g_2 \bullet g_2^{-1}) \bullet g_1^{-1}) \\ &= g_1 \bullet i \bullet g_1^{-1} \\ &= g_1 \bullet g_1^{-1} \\ &= i.\end{aligned}$$

Showing that $(g_2^{-1} \bullet g_1^{-1}) \bullet (g_1 \bullet g_2) = i$ is similar

Recall our permutation Markov chain where a permutation is chosen uniformly from the set of transposition permutations.

Say that $\tau \in \mathcal{S}_n$ is a **transposition permutation** if $\#\{i : \tau(i) \neq i\} \leq 2$.

Then the transposition Markov chain is as follows.

Algorithm 5
Permutation_random_walk(x)

1) Choose τ uniformly from the set of transpositions permutations
2) Return $x \bullet \tau$

This idea can be generalized to any group, any subset of that group, and any probability distribution over that group.

Let M be a random variable over a group G, and $X_0 \in G$. Let $M_0, M_1, \ldots$ be iid M, and for $t \in \{1, 2, 3, \ldots\}$, set $X_{t+1} = X_t \bullet M_t$. Then the stochastic process $\{X_t\}$ is a **random walk on a group**.

This includes things like shuffling cards (any of the techniques) or mixing up a Rubik's cube.

Time for a universal theorem: no matter what the random walk, the uniform distribution over a finite group is uniform!

A random walk over a finite group has the uniform distribution as a stationary distribution.

Let G be a group, $X_t \sim \mathsf{Unif}(G)$, and let $g \in G$. Then

$$\begin{aligned}
\mathbb{P}(X_{t+1} = g) &= \sum_{j \in G} \mathbb{P}(X_{t+1} = g, X_t = j) \\
&= \sum_{j \in G} \mathbb{P}(X_t = j)\mathbb{P}(X_{t+1} = g \mid X_t = j) \\
&= \frac{1}{\#(G)} \sum_{j \in G} \mathbb{P}(g = j \bullet M_t)
\end{aligned}$$

Note that $g = j \bullet M_t \Leftrightarrow j^{-1} \bullet g = M_t$)

At this point, note that for fixed group element g, $j \mapsto j^{-1}g$ is a 1-1 and onto map.

- It is onto because for $k \in G$, $(g \bullet k^{-1})^{-1} \mapsto (k^{-1})^{-1} \bullet g^{-1} \bullet g = k$.
- It is 1-1 because if $j_1^{-1} \bullet g = j_2^{-1} \bullet g$, then operating by $g^{-1} on the right gives $j_1^{-1} = j_2^{-1}$, and then operating by j_1 on both sides on the right and j_2 by both sides on the left gives $j_1 = j_2$.

Therefore,

$$\begin{aligned}\mathbb{P}(X_{t+1} = g) &= \frac{1}{\#(G)} \sum_{j \in G} \mathbb{P}(M_t = j^{-1} \bullet g) \\ &= \frac{1}{\#(G)} \sum_{k \in G} \mathbb{P}(M_t = k) \\ &= \frac{1}{\#(G)},\end{aligned}$$

since the probability $M_t = k$ summed over all $k \in G$ has to add up to 1.

This is why people do not have to worry when mixing up a Rubik's cube or shuffling a deck of cards. Any way of doing so will give the uniform distribution!

16.2 Uniform stationary distributions

Many Markov chains are *symmetric*, meaning that the probability of moving from one state to another is the same as the chance of moving from the second state back to the first state.

Definition 38
A finite state Markov chain $\{X_t\}$ is *symmetric* if

$$(\forall x, y \in \Omega)(\mathbb{P}(X_t = y \mid X_{t-1} = x) = \mathbb{P}(X_t = x \mid X_{t-1} = y)).$$

A nice thing about symmetric Markov chains is that the uniform distribution is stationary.

Fact 32
Symmetric Markov chains have the uniform distribution over the state space as a stationary distribution.

Proof. Suppose that $\mathcal{M}$ is a symmetric Markov chain, and X_{t-1} is uniform over Ω. Then let $x \in \Omega$ and consider $\mathbb{P}(X_t = x$. For this event to occur, X_{t-1} had to be some state y, and summing over the probabilities gives back the target probability.

$$\begin{aligned}\mathbb{P}(X_t = x) &= \sum_y \mathbb{P}(X_t = x, X_{t-1} = y) \\ &= \sum_y \mathbb{P}(X_t = x | X_{t-1} = y)\mathbb{P}(X_{t-1} = y) \\ &= \sum_y \mathbb{P}(X_t = y | X_{t-1} = x)(1/\#(\Omega)) \\ &= 1/\#(\Omega).\end{aligned}$$

The rest of the sum evaluates to 1, since if $X_{t-1} = x$, summing over the probabilities of moving to every other state in the Markov chain yields 1, since the state must move somewhere! This completes the proof. □

Getting back to groups, suppose that the Markov chain does not wander over the entire group G, but over a subset of the group S. If the random walk attempts to take the state outside S *partial reflection* occurs, and the state stays exactly where it was.

Let M be a random variable over a group G, and $M_0, M_1, \ldots$ be an iid stream of draws from M. Fix $S \subset G$. Then for $X_t \in S$, let

$$X_{t+1} = \begin{cases} X_t \bullet M & \text{if } X_t \bullet M \in S \\ X_t & \text{if } X_t \bullet M \notin S \end{cases}$$

Say that the stochastic process $\{X_t\}$ forms a **random walk on a group with partially reflecting boundaries**.

Suppose G is all integers, $S = \{1, \ldots, n\}$, and $M \sim \text{Unif}(\{-1, 1\})$. Then at each step of the Markov chain, the state is either added to by 1 or subtracted from by 1. But if that move would cause it to leave $\{1, \ldots, n\}$, then it stays where it currently is. This is a simple symmetric random walk on $\{1, \ldots, n\}$ with partially reflecting boundaries.

Unlike the random walk over the entire group, the stationary distribution might not be uniform. However, if the distribution of M is *symmetric* it will be.

Say that a random variable M over a group G is *symmetric* if $\mathbb{P}(M = m) = \mathbb{P}(M = m^{-1})$ for all $m \in G$.

Suppose that a random walk on $S \subset G$ with partially reflecting boundaries uses symmetric draws M. Then the chain is symmetric, and the uniform distribution over S is stationary for the chain.

Note for all x and y in S,

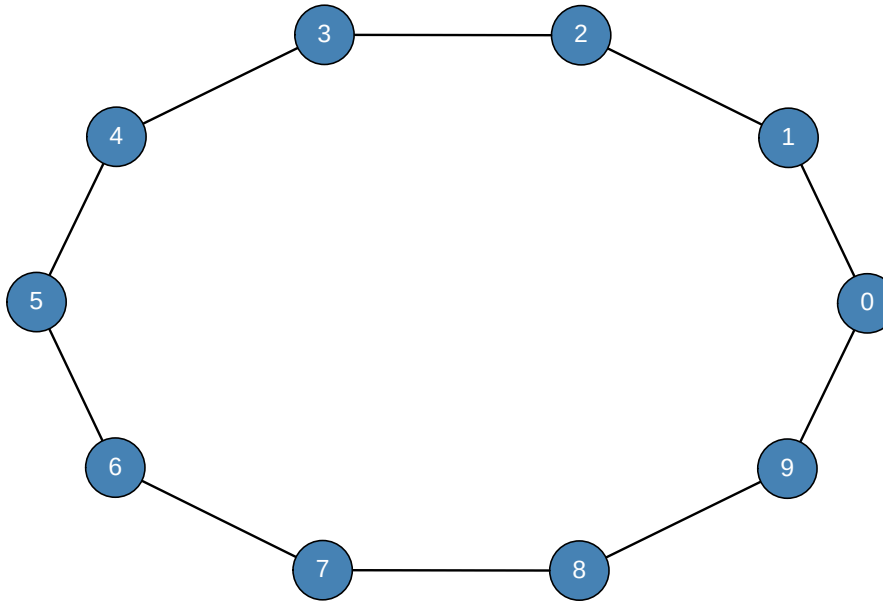
$$p(x, y) = \mathbb{P}(M_t = yx^{-1}) = \mathbb{P}(M_t = (yx^{-1})^{-1}) = \mathbb{P}(M_t = xy^{-1}) = p(y, x).$$

Continuing our simply symmetric random walk with partially reflecting boundaries, the uniform distribution over $\{1, \ldots, n\}$ is stationary.

16.3 Sampling uniformly using random walks in R

Consider the problem of sampling uniformly from the group of integers mod n. For instance, when $n = 10$, the graph looks like this:

```
create_ring(10) %>%
  ggraph(layout = "kk") +
    geom_edge_link() +
    geom_node_point(size = 12, shape = 21, color = "black",
                    fill = "steelblue") +
    geom_node_text(aes(label = 0:9), color = "white",
                   vjust = 0.4) +
    theme_graph() +
    coord_cartesian(clip = 'off')
```



The step $M \in \{-1, 1\}$, and the group uses addition with modular arithmetic, which means if the state reaches n, then it starts over at 0. The step looks as follows

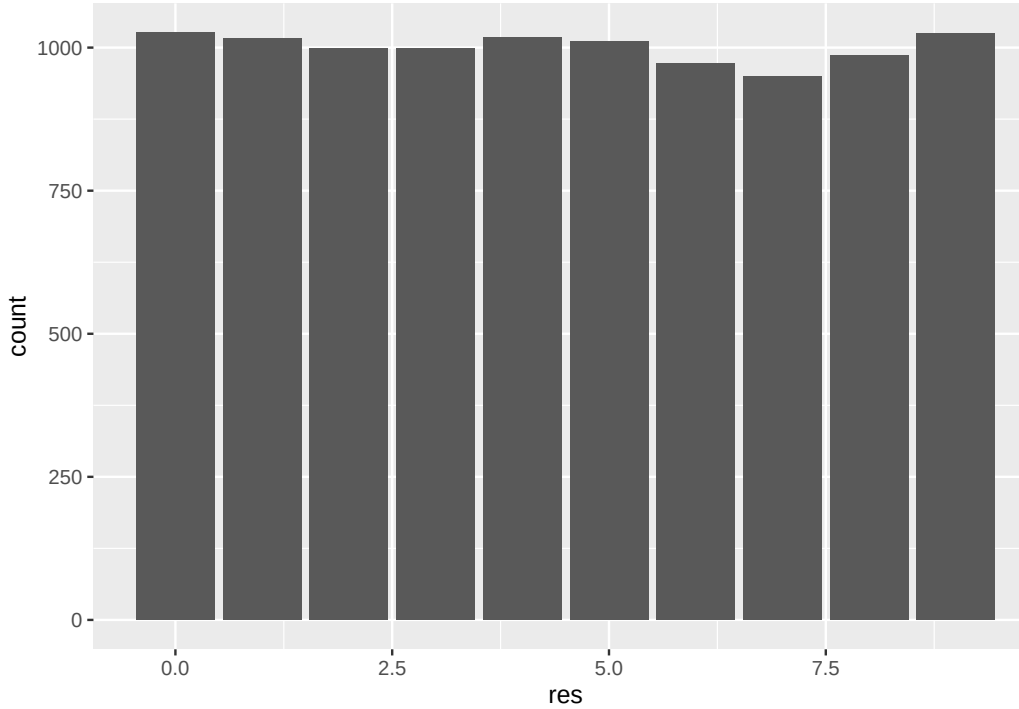
```
step_ring <- function(x, m, n) return((x + m) %% n)
```

Note that any distribution on M will result in a uniform stationary distribution! For instance, suppose $\mathbb{P}(M = -1) = 0.7$ and $\mathbb{P}(M = 1) = 0.3$.

```
ring_walk <- function(steps, n) {
  burnin <- steps
  datasteps <- steps
  m1 <- as.integer(2 * (runif(burnin) < 0.3) - 1)
  x <- 0
  for (i in 1:burnin)
    x <- step_ring(x, m1[i], n)
  m2 <- as.integer(2 * (runif(datasteps) < 0.3) - 1)
  x <- c(x, rep(0, datasteps))  # reserve memory space
  for (i in 1:datasteps)
    x[i + 1] <- step_ring(x[i], m2[i], n)
  return(x)
}
```

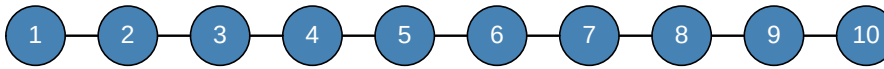
Now test it out:

```
res <- ring_walk(10000, 10)
ggplot() +
  geom_bar(aes(res))
```



Now suppose the goal is to sample from $\{1, \ldots, n\}$.

```
create_path(10) %>%
  ggraph(layout = "linear") +
    geom_edge_link() +
    geom_node_point(size = 12, shape = 21, color = "black",
                    fill = "steelblue") +
    geom_node_text(aes(label = 1:10), color = "white",
                   vjust = 0.4) +
    theme_graph() +
    coord_cartesian(clip = 'off')
```



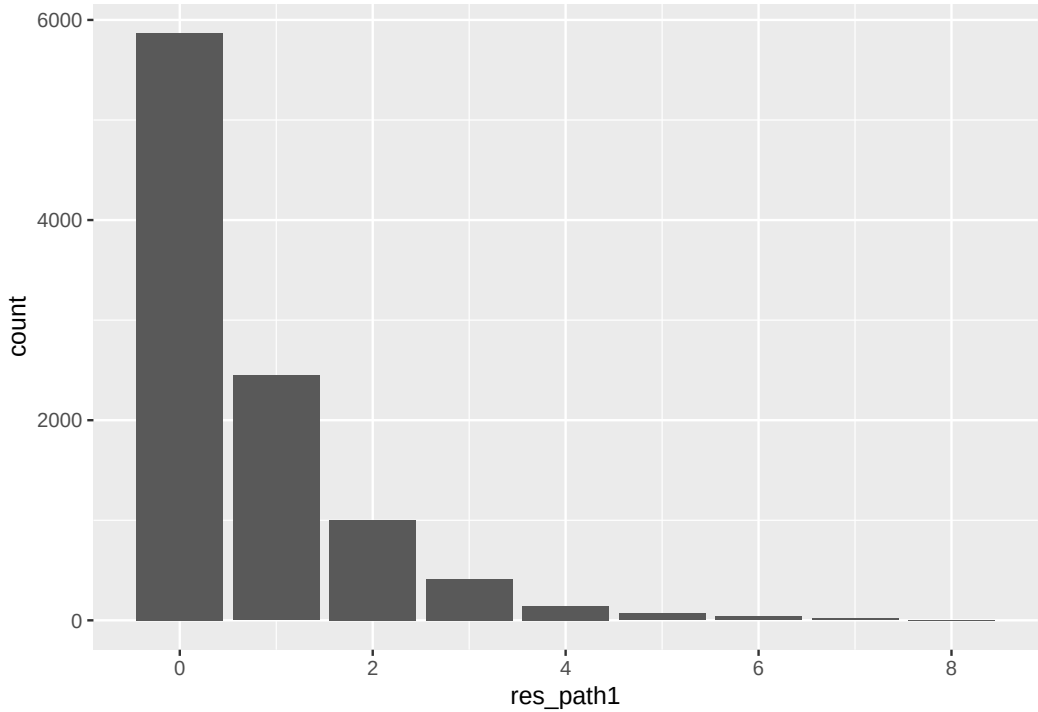
Now when we try to reduce the node value by 1 from node 1, the state stays the same. Similarly, an attempt to increase the node value from 10 by 1 leaves the state at 10.

```
step_path <- function(x, m, n) return(max(0, min(x + m, 10)))
```

```
path_walk1 <- function(steps, n) {
  burnin <- steps
  datasteps <- steps
  m1 <- as.integer(2 * (runif(burnin) < 0.3) - 1)
  x <- 0
  for (i in 1:burnin)
    x <- step_path(x, m1[i], n)
  m2 <- as.integer(2 * (runif(datasteps) < 0.3) - 1)
  x <- c(x, rep(0, datasteps))  # reserve memory space
  for (i in 1:datasteps)
    x[i + 1] <- step_path(x[i], m2[i], n)
  return(x)
}
```

Now test it out:

```
res_path1 <- path_walk1(10000, 10)
ggplot() +
  geom_bar(aes(res_path1))
```

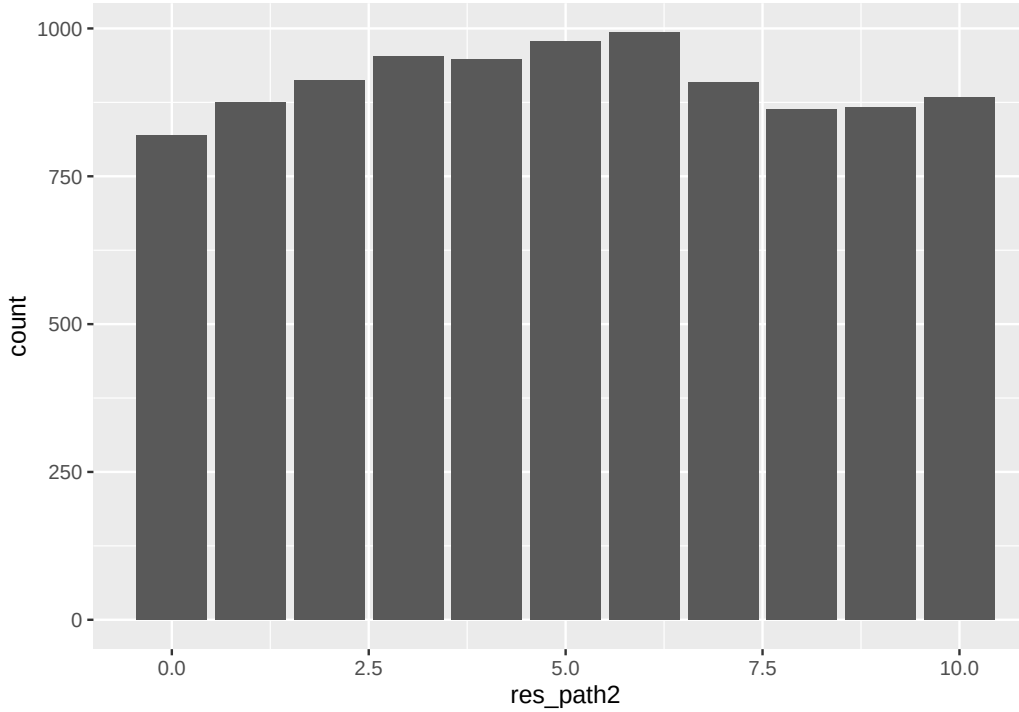


Because the chain moves left must more than to the right, the state tends to clump near 1. Making the moves symmetric returns the stationary (and hence limiting) distribution to uniform.

```
path_walk2 <- function(steps, n) {
  burnin <- steps
  datasteps <- steps
  m1 <- as.integer(2 * (runif(burnin) < 0.5) - 1)
  x <- 0
  for (i in 1:burnin)
    x <- step_path(x, m1[i], n)
  m2 <- as.integer(2 * (runif(datasteps) < 0.5) - 1)
  x <- c(x, rep(0, datasteps))   # reserve memory space
  for (i in 1:datasteps)
    x[i + 1] <- step_path(x[i], m2[i], n)
  return(x)
}
```

Now test it out:

```
res_path2 <- path_walk2(10000, 10)
ggplot() +
  geom_bar(aes(res_path2))
```



Problems

16.1: Consider taking a asymmetric random walk on the integers mod 5 (so the state space is $\{0, 1, 2, 3, 4\}$), using M where $\mathbb{P}(M = -1) = 0.6$ and $\mathbb{P}(M = 1) = 0.4$. So with probability 0.4 add 1 to the current state, and if it reaches 5 replace it with a 0. Else (with probability 0.6) add -1 to the current state, and if it reaches -1 replace it with a 4. This chain is aperiodic and connected.

This chain has a limiting (normalized) distribution. What is it?

16.2: Consider taking a random walk on the integers mod 100. At each step, we add either 1, 2, 3, or 4, each with probability $1/4$. This chain is aperiodic and connected. This chain has a limiting distribution. What is it, and how do you know?

16.3: Again consider the random walk over the integers mod 5 with move

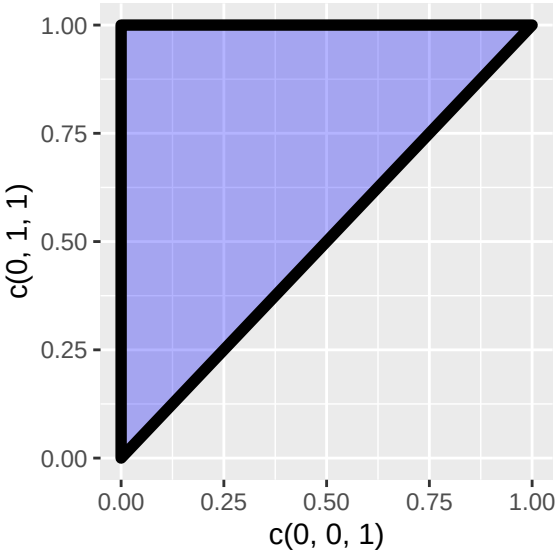
$$\mathbb{P}(M = -1) = 0.6, \ \mathbb{P}(M = 1) = 0.4.$$

Implement the chain. Using 10^4 burnin steps and 10^4 data gathering steps, estimate the mean of the limiting distribution.

16.4: Consider taking a random walk on the integers mod 100. At each step, we add either 1, 2, 3, or 4, each with probability $1/4$. This chain is aperiodic and connected. This chain has a limiting distribution.

Implement the chain. Using 10^5 burnin steps and 10^5 data gathering steps, estimate the mean of the limiting distribution.

16.5: Suppose that (X, Y) is uniform over the triangle in $\mathbb{R}^2$ with vertices $(0, 0)$, $(0, 1)$, and $(1, 1)$.



Write code for one step in a random walk with partially reflecting boundaries over this region that takes as input the current state and a vector with two components, and returns the next state.

16.6: Suppose that (X, Y) is uniform over the interior of an ellipse given by

$$A = \{(x, y) : x^2 + 2y^2 \leq 1\}.$$

Write code for one step in a random walk with partially reflecting boundaries over this region that takes as input the current state and a vector with two components, and returns the next state.

16.7: Returning to the step for the triangle problem from earlier, use 10 replications of your chain for 10^4 burnin and data gathering steps to estimate $\mathbb{E}[X]$.

16.8: Returning to the step for the ellipse problem from earlier, use 10 replications of your chain for 10^4 burnin and data gathering steps to estimate $\mathbb{E}[X]$.

16.9: Suppose that a Markov chain with state space $[0, 10]$ uses a random walk with partially reflecting boundaries. The move is a beta with parameters 3 and 3 minus 0.5. Note that the density of this move is

$$f(s) = (1/2 + s)^2(1/2 - s)^2\mathbb{I}(s \in [-0.5, 0.5]),$$

which is symmetric around o (since $f(s) = f(-s)$.)

This Markov chain has a limiting distribution equal to the stationary distribution. What is this distribution?

16.10: Suppose that a Markov chain with state space $[-5, 5] \times [-5, 5]$ uses a random walk with partially reflecting boundaries. The move is two dimensional (Z_1, Z_2) where the Z_i are iid standard normal random variables.

This Markov chain has a limiting distribution equal to the stationary distribution. What is this distribution?

Chapter 17

Auxiliary random variables Markov chains

Question of the Day

Design an auxiliary variable Markov chain to approximately sample from unnormalized density

$$g(x) = x^2(1-x)\mathbb{I}(x \in [0,1])$$

with respect to Lebesgue measure.

Summary

- Adding an *auxiliary random variable* can convert a density problem to a uniform problem.
- Using Gibbs sampling with auxiliary random variables can result in a chain that mixes much faster.
- If the density is a product, then an auxiliary random variable can be added for each term in the product. This gives the *product slice sampler*.

Libraries

The libraries used in this chapter include

- tidygraph for manipulation of graphs,
- igraph for storing graphs,
- and ggraph for displaying graphs.

```
library(tidygraph)
library(igraph)
library(ggraph)
```

Random walks with partially reflecting boundaries can be used to draw uniformly from complicated sets. When it is needed to sample from a density (unnormalized) over a space, *auxiliary variables* can be used to make the problem uniform.

Recall that if random variable X has density f with respect to some measure, making $[Y \mid X] \sim \mathsf{Unif}([0, f(X)])$ creates a joint distribution

$$(X, Y) \sim \mathsf{Unif}(\{(x, y) : 0 \leq y \leq f(x)\}).$$

Conversely, if $(X, Y) \sim \mathsf{Unif}(\{(x, y) : 0 \leq y \leq f(x)\})$, then $X \sim f$.

To turn this into a Markov chain, utilize Gibbs sampling. Drawing Y given X is easy. More difficult is to draw X given Y. The value of X is chosen uniformly from the set of values x such that $f(x) \geq Y$. This is called the *slice sampler*.

Definition 39
A **slice sampler** Markov chain to draw X from density f works as follows. First, draw Y uniformly from $[0, f(X)]$. Second, draw X uniformly from $\{x : f(x) \geq Y\}$.

17.1 Slice sampler for product form densities

In the Question of the Day, this means drawing X such that $X^2(1 - X) \geq Y$. This can be solved with a cubic equation, but it is easy to find examples that are even more difficult to solve.

An alternate solution is to employ an auxiliary random variable for every factor in the product of the density.

In the Question of the Day,

$$f(x) = x^2(1 - x)\mathbb{I}(x \in [0, 1]) = x \cdot x \cdot (1 - x)\mathbb{I}(x \in [0, 1]).$$

There will be a Y_1 for the first x factor, a Y_2 for the second x factor, and Y_3 for the $1 - x$ factor. Then the draw for X must satisfy, $Y_1 \leq X$, $Y_2 \leq X$, $Y_3 \leq 1 - X$, so $X \leq 1 - Y_3$.

This yields the following algorithm.

Algorithm 6
step_qotd_slice_sampler(X)

1) Draw $Y_1 \sim \mathsf{Unif}([0, X])$.
2) Draw $Y_2 \sim \mathsf{Unif}([0, X])$.
3) Draw $Y_3 \sim \mathsf{Unif}([0, 1 - X])$.
4) Draw $X \sim \mathsf{Unif}([\max(Y_1, Y_2), 1 - Y_3])$.

More generally, the product slice sampler can be defined as follows.

The **product slice sampler** for unnormalized density

$$f(x) = \prod_{i=1}^{m} f_i(x)$$

operates as follows. For each i from 1 to m, draw $Y_i \sim \text{Unif}([0, f_i(X)])$ uniformly. Then draw X uniformly from

$$\bigcap_{i=1}^{m} \{x : f_i(x) \geq Y_i\}.$$

Swendsen and Wang (1986) (Swendsen and Wang 1986) were the first to use a product slice sampler, for the Ising model. The idea was generalized by Edwards and Sokal (1998) (Edwards and Sokal 1988).

Swendsen-Wang algorithm

Recall that the Ising model over graph $G = (V, E)$ has an unnormalized density that can be written as

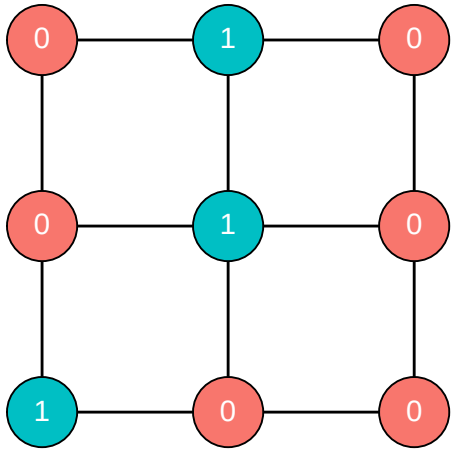
$$\exp\left(\beta \sum_{\{i,j\} \in E} \mathbb{I}(x(i) = x(j))\right) = \prod_{\{i,j\} \in E} \exp(\beta \mathbb{I}(x(i) = x(j))).$$

Therefore, the product slice sampler applied to the Ising model draws an auxiliary random variable uniformly from 0 to $\exp(\beta \mathbb{I}(x(i) = x(j)))$ for every edge E in the graph.

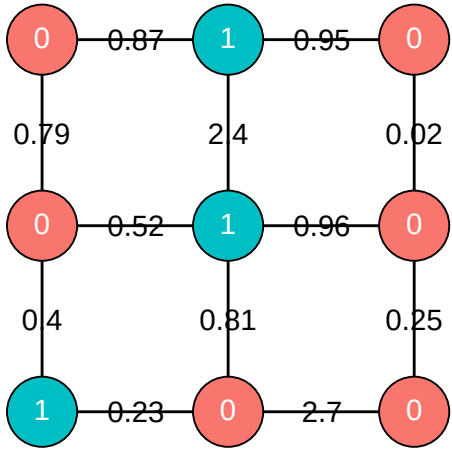
So if $x(i) \neq x(j)$, then the auxiliary variable $Y_{\{i,j\}}$ is uniform over $[0, 1]$, and if $x(i) = x(j)$, then it is uniform over $[0, \exp(\beta)]$. Suppose the value of $Y_{\{i,j\}} = 0.92\ldots$. This is compatible with either $x(i) = x(j)$ or $x(i) \neq x(j)$, and so the choice of X is indifferent to these two possibilities.

On the other hand, if $\beta = 1$ and $Y_{\{i,j\}} = 1.42\ldots$, then the only compatible configurations have to have $x(i) = x(j)$. Across the graph, the edges with $Y_{\{i,j\}} > 1$ have to have the X value equal at both endpoints. This divides the graph into *clusters*, sets of vertices that are connected by a path using edges with $Y_e > 1$. (These clusters of the graph are also known as *connected components*.)

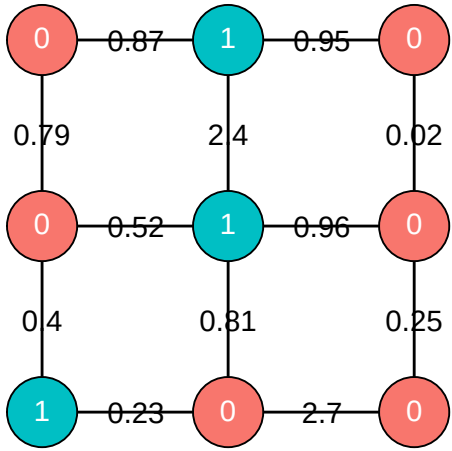
For example, suppose this is our Ising state.



Then after drawing the auxiliary random variables:

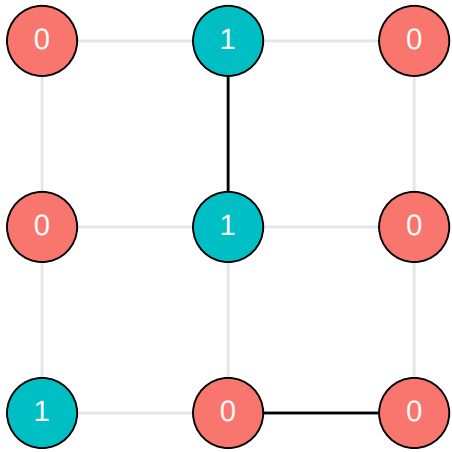


Now only keep those edges labeled with a value above 1:



Note two things.

- The chance of keeping an edge is $(\exp(\beta) - 1)/(\exp(\beta) - 0)$, or just $1 - \exp(-\beta)$. So instead of uniforms, Bernoullis with parameter $1 - \exp(-\beta)$ could have been used to keep edges.
- The kept edges must connect nodes with the same label.



The components function can be used to find the components for the graph when only keeping these two edges.

```
clusters <-
  qotd_ising |>
    activate(edges) |>
    filter(sw > 1) |>
    components()
clusters
```

```
## $membership
## [1] 1 2 3 4 2 5 6 7 7
##
## $csize
## [1] 1 2 1 1 1 1 2
##
## $no
## [1] 7
```

This says that there are 7 components, and the `membership` indicates the name of the cluster. For instance, both nodes 8 and 9 are part of cluster 7.

To continue taking the step in the Swendsen-Wang chain, the clusters must be assigned labels uniformly from $\{0, 1\}$.

```
colors <- as.integer(runif(clusters[["no"]]) > 0.5)
```

Then the colors variable is:

```
colors
```

```
## [1] 0 0 1 1 1 1 1
```

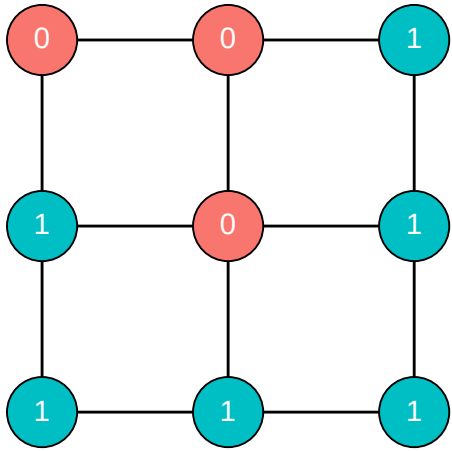
In this random draw, the first and second cluster will be assigned label 0, while the remaining clusters will be assigned label 1.

```
labels <- colors[clusters[["membership"]]]
labels
```

```
## [1] 0 0 1 1 0 1 1 1 1
```

Since the second cluster consisted of nodes 2 and 5, both these nodes were labeled 0.

```
qotd_ising3 <-
  qotd_ising |>
  activate(nodes) |>
  mutate(ising = labels)
qotd_ising3 |> ggraph_lab_ising()
```



17.2 Product slice samplers in R

Begin with the product slice update for $f(x) = x^2(1-x)\mathbb{I}(x \in [0,1])$. Remember the algorithm starts with a state X, and then draws

$$[(Y_1, Y_2, Y_3) \mid X] \sim (\mathsf{Unif}([0, X]), \mathsf{Unif}([0, X]), \mathsf{Unif}([0, 1-X])),$$

where the components are independent.

Then $[X \mid Y] \sim \mathsf{Unif}(\max(Y_1, Y_2), 1 - Y_3)$. This can be implements as follows.

```
step_qotd_pss <- function(x, u) {
  y <- c(x * u[1], x * u[2], (1 - x) * u[3])
  x <- ((1 - y[3]) - max(y[1], y[2])) * u[4] + max(y[1], y[2])
  return(x)
}
```

Test it out by estimating $\mathbb{P}(X \leq 0.3)$.

```
chain_qotd_pss <- function(steps) {
  burnin    <- steps
  datasteps <- steps
  x <- 0.5
  res <- rep(0, datasteps)
  m1 <- runif(burnin)
  m2 <- runif(burnin)
  m3 <- runif(burnin)
  m4 <- runif(burnin)
```

```
  for (i in 1:burnin) {
    x <- step_qotd_pss(x, c(m1[i], m2[i], m3[i], m4[i]))
  }
  m5 <- runif(datasteps)
  m6 <- runif(datasteps)
  m7 <- runif(datasteps)
  m8 <- runif(datasteps)
  for (i in 1:datasteps) {
    res[i] <- (x <= 0.3)
    x <- step_qotd_pss(x, c(m5[i], m6[i], m7[i], m8[i]))
  }
  return(res)
}
```

```
res <- replicate(10, mean(chain_qotd_pss(10^4)))
tibble(
  est_mean = mean(res),
  est_err  = sd(res) / sqrt(length(res))
) |> kable()
```

est_mean	est_err
0.08303	0.0008636

A longer run gives

```
res <- replicate(10, mean(chain_qotd_pss(10^5)))
tibble(
  est_mean = mean(res),
  est_err  = sd(res) / sqrt(length(res))
) |> kable()
```

est_mean	est_err
0.083743	0.000429

So the estimate is $\boxed{0.08375 \pm 0.00044}$.

Swendsen-Wang in R

A step in the Swendsen-Wang algorithm on a graph $G = (V, E)$ with $n = \#(V)$ nodes and $m = \#(E)$ edges requires a set of n iid Bern$(1/2)$ random variables, and m iid Bern$(\exp(-\beta))$. Given these inputs, the algorithm proceeds using the `components` function to determine the connected components of the graph, and recolors appropriately.

This method of writing the step does waste some of our random choices, since the number of clusters could be smaller than the number of nodes. Taking advantage of this would require code that does not follow our update function way of running Markov chain steps, so for expediency this will be ignored.

```
step_swendsen_wang <- function(x, g, node_bern, edge_bern) {
  return(node_bern[
    (g |>
      activate(edges) |>
      filter(edge_bern == 1) |>
      components())[["membership"]]])
}
```

Each step in Swendsen-Wang rewrites every node, and takes $\Theta(m)$ time to run. So there is no harm from a computational complexity point of view in recalculating h directly after each step.

```
h_graph <- function(x, g) {
  from <- g |> activate(edges) |> pull(from)
  to <- g |> activate(edges) |> pull(to)
  return(sum(x[from] == x[to]))
}
```

Similarly, there is no need to generate the random variables all at once.

```
chain_swendsen_wang <- function(steps, g, beta) {
  burnin    <- steps
  datasteps <- steps
  n <- g |> activate(nodes) |> as_tibble() |> nrow()
  m <- g |> activate(edges) |> as_tibble() |> nrow()
  x <- rep(0, n)
  for (i in 1:burnin)
    x <- step_swendsen_wang(
           x, g, as.integer(runif(n) < 0.5),
           as.integer(runif(m) < (1 - exp(-beta))))
  res <- rep(0, datasteps + 1)
  for (i in 1:datasteps) {
    x <- step_swendsen_wang(
           x, g, as.integer(runif(n) < 0.5),
           as.integer(runif(m) < (1 - exp(-beta))))
    res[i] <- h_graph(x, g)
      }
  return(res)
}
```

```
g <- create_lattice(c(4, 4))
res_local <- replicate(5,
```

```
  (chain_swendsen_wang(100,
     create_lattice(c(4, 4)), 0.5) <= 14) |>
  mean()
)
tibble(
  est_mean = mean(res_local),
  est_err  = sd(res_local) / sqrt(length(res_local))
) |> kable()
```

est_mean	est_err
0.1643564	0.0155291

Therefore, the resulting estimate is $\boxed{0.479 \pm 0.025}$.

Problems

17.1: Consider the density proportional to

$$g_X(x) = \sin(x)\cos(x)^2 \mathbb{I}(x \in [0, \tau/4]).$$

a) For a product slice sample, what are the distributions of Y_1, Y_2, and Y_3 given X?

b) For a product slice sample, given (Y_1, Y_2, Y_3), what is the distribution of X?

17.2: Consider the density proportional to

$$g_W(w) = wexp(-2w)\mathbb{I}(w \geq 0).$$

a) For a product slice sample, what are the distributions of Y_1 and Y_2 given W?

b) For a product slice sample, given (Y_1, Y_2), what is the distribution of W?

17.3: Going back to

$$g_X(x) = \sin(x)\cos(x)^2 \mathbb{I}(x \in [0, \tau/4]),$$

implement an update function that takes as input the current state `x` and a vector of four numbers `u` and returns the next state after updating Y_1, Y_2, Y_3, X in that order.

17.4: Consider the unnormalized density $f(x) = x^{1/2}\exp(-x)\mathbb{I}(x \in [0, 2])$.

a) Write R code for a product slice sampler that takes as input the current state and three standard uniform random variables, and returns the next state in the Markov chain.

b) Use your code, with 10^4 steps in the Markov chain and 10 replications to estimate $\mathbb{E}[X]$ for $X \sim f$.

17.5: Use 1000 steps of burnin and data gathering Markov chain Monte Carlo to estimate the mean of a draw from the density $\sin(x)\cos(x)^2\mathbb{I}(x \in [0, \tau/4])$. Replicate 10 times and report the estimate as $a \pm b$.

17.6: Use 1000 steps of burnin and data gathering Markov chain Monte Carlo to estimate the mean of a draw from the density $x\exp(-2x)\mathbb{I}(x \geq 0)$. Replicate 10 times and report the estimate as $a \pm b$.

Chapter 18

The Metropolis method

Question of the Day

Consider a simple symmetric random walk Markov chain on $\{1, 2, \ldots\}$ that adds 1 or -1 to the current state with equal probability unless that would move the state below 1, in which case stay where you are. Modify this Markov chain so that the stationary distribution is

$$\mathbb{P}(X = i) \propto \frac{1}{i^2}$$

using the Metropolis method.

Summary

- The **Metropolis-Rosenbluth-Rosenbluth-Teller-Teller** ($\mathsf{MR}^2\mathsf{T}^2$) algorithm is a combination of random walk with partially reflecting boundaries and an auxiliary random variable.
- $\mathsf{MR}^2\mathsf{T}^2$ takes as input a random walk on a group together with a target density p. It creates a new chain as follows. At each step from current state X, propose a group move M so that $X \bullet M$ would be the next step in the random walk. Then draw a standard uniform U. Accept the move in the Metropolis chain if $U \leq p(X \bullet M)/p(X)$, otherwise reject the move and stay at the current state.

18.1 MR² T²

The middle of the 20th century was dominated by the effects of World War II. One of the scientific breakthroughs that the war brought was the creation of the first electronic computers. Early examples could do hundreds or thousands of mind numbing calculations, which altered the development of algorithms forever.

Markov chain Monte Carlo (MCMC) arose out of that era, as researchers realized that these computers were perfectly suited to repeating random trials over and over again. The

first major MCMC algorithm was a combination of a random walk with partially reflecting boundaries and an auxiliary random variable. Developed by Nicholas Metropolis, Arianna Rosenbluth, Marshall Rosenbluth, Augusta Teller, and Edward Teller (Metropolis et al. 1953). For this reason, it will be referred to as MR^{2}T^2 here.

Suppose the goal is to build a Markov chain over a state space Ω that is a group, where the output X has density p. Then given $X \sim p$, let $[W \mid X] \sim \text{Unif}([0, p(X)])$, and consider the augmented space $\Omega_{(X,W)} = \{(x, w) : x \in \Omega, 0 \leq w \leq p(x)\}$. Then as seen earlier, $(X, W) \sim \text{Unif}(\Omega_{(X,W)})$.

Now consider the following algorithm. Starting at state X, draw W uniformly from $[0, p(X)]$. Then draw M according to a symmetric function on the group. Now consider the point $(X \bullet M, W)$. If this point is in $\Omega_{(X,W)}$, then move to $(X \bullet M, W)$ and return $X \bullet M$ as the next state. Otherwise, stay at (X, W), and return X as the next state.

Now consider the chance that $(X \bullet M, W)$ is in $\Omega_{(X,W)}$. Recall that the simulation can use $W = p(X)U$, where U is a standard uniform number. Note that $(X \bullet M, W) \in \Omega_{(X,W)}$ if

$$(W \leq p(X \bullet M)) = (p(X)U \leq p(X \bullet M)) = \left(U \leq \frac{p(X \bullet M)}{p(X)}\right)$$

This leads to the following definition of MR^{2}T^2.

Definition 40
Given a group Ω and M a random variable over the group, let $M_0, M_1, \ldots$ be iid M, $U_0, U_1, U_2, \ldots$ be iid $\text{Unif}([0, 1])$. Then the MR^{2}T^2 Markov chain uses

$$X_{t+1} = \begin{cases} X_t \bullet M_t & \text{if } U_t \leq p(X \bullet M)/p(X) \\ X_t & \text{if } U_t > p(X \bullet M)/p(X) \end{cases}$$

Of course, for these chains to be useful, it is necessary that they be ϕ-aperiodic. To be connected, it is necessary to find paths between regions using the proposed moves. Because these chains tend to have a positive probability of staying in place for all states in the chain, they are usually automatically aperiodic.

Solving the Question of the Day

There are an infinite number of choices for M that solve the question of the day. A simple one is $M \sim \text{Unif}(\{-1, 1\})$, and the group operation is just addition.

MR^{2}T^2 accepts the proposed move when a standard uniform is smaller than the density of the proposed move divided by the density of the current state.

The density is $p(i) = [1/i^2]\mathbb{I}(i \in \{1, 2, \ldots\})$. So if X_t is the current state and $X_t + M_t$ the proposed move, then acceptance occurs for $U \sim \text{Unif}([0, 1])$ when two things happen. First, $X_t + M_t$ must be at least 1, and second:

$$U \leq \frac{[1/(X_t + M_t)^2]}{[1/(X_t)^2]}.$$

Note that if $X_t + M_t = 0$, reject, but $1/(X_t + M_t)$ is undefined. A clever way to write this in one line using indicator functions is:

$$U \leq \frac{[1/((X_t + M_t)^2 + \mathbb{I}(X_t + M_t = 0))]\mathbb{I}(X_t + M_t \geq 1)}{[1/(X_t)^2]}$$
$$= \frac{X_t^2 \mathbb{I}(X_t + M_t \geq 1)}{(X_t + M_t)^2 + \mathbb{I}(X_t + M_t = 0)}.$$

That way the denominator is never 0, but if $X_t + M_t = 0$ the whole fraction still evaluates to 0!

Then the update step looks as follows.

step_mrrtt_qotd (x, m, u)

1) Let $y \leftarrow x + m$.
2) If $u \leq (x^2/[y + \mathbb{I}(y = 0)]^2)\mathbb{I}(y \geq 1)$, return y,
3) Else return(x).

18.2 Acceptance rate

The one choice that the designer of an MR^{2}T^2 chain has is the choice of the random move M. If this move is too ambitious, then the chain will be relatively unlikely to accept the move. If this move is too small, then the chain will tend to sit in one area and not move around the state space.

Definition 41
Let $\mathcal{M}$ be an MR^{2}T^2 chain with stationary distribution π. Then for $X_t \sim \pi$, $\mathbb{P}(X_{t+1} = X_t)$ is the **acceptance rate** of the chain.

A rule of thumb is to keep the acceptance rate around $1/4$, but really anything in the $[1/4, 1/2]$ range is good. See (Roberts and Rosenthal 2001) for a justification. An acceptance rate of $1/4$ indicates a chain that is trying to move to new areas, but not trying to jump so far that the chain is staying at the same state too much.

18.3 MR^{2}T^2 chains in R

Begin with the question of the day. The MR^{2}T^2 chain step is as follows.

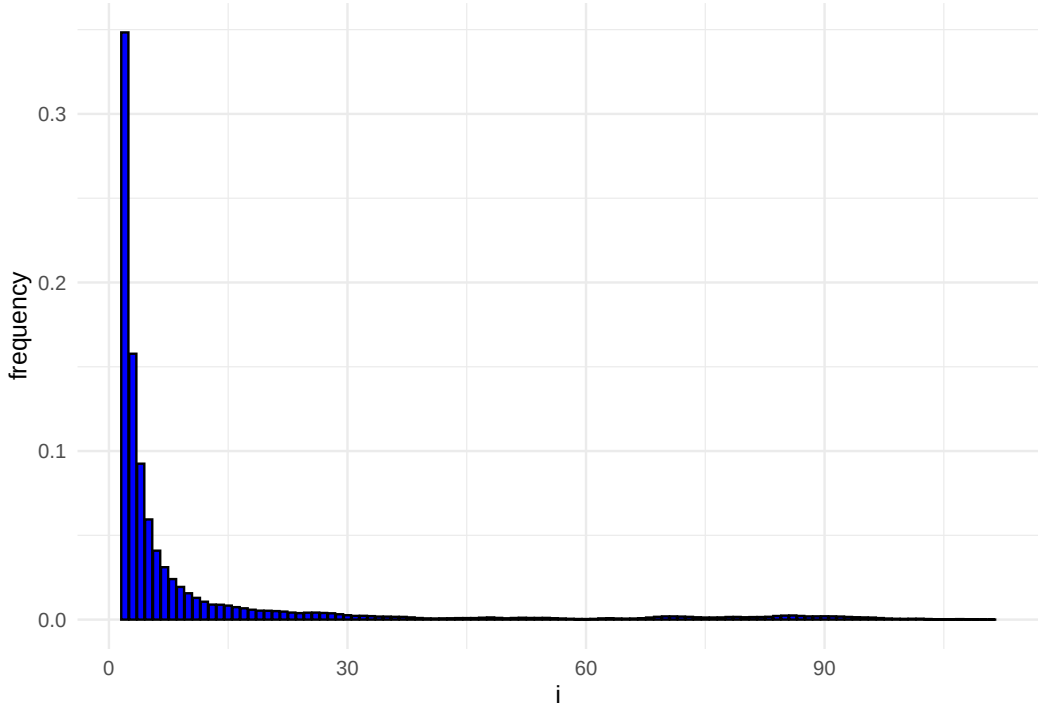
```
step_qotd_mrrtt <- function(x, m, u) {
  y <- x + m
  if (u <= (y > 1) * x^2 / (y + (y < 1))^2)
    return(y)
  else
    return(x)
}
```

Now burn in the chain and collect data.

```
chain_qotd_mrrtt <- function(steps) {
  burnin    <- steps
  datasteps <- steps
  x <- 1
  res <- rep(0, datasteps)
  m1 <- 2 * (runif(burnin) < 0.5) - 1
  u1 <- runif(burnin)
  for (i in 1:burnin) {
    x <- step_qotd_mrrtt(x, m1[i], u1[i])
  }
  m2 <- 2 * (runif(burnin) < 0.5) - 1
  u2 <- runif(burnin)
  for (i in 1:datasteps) {
    res[i] <- x
    x <- step_qotd_mrrtt(x, m2[i], u2[i])
  }
  return(res)
}
```

Finally, consider the output using a histogram.

```
tibble(
  x = chain_qotd_mrrtt(10^5)
  ) |>
  ggplot(aes(x)) +
  geom_bar(aes(y = (..count..)/sum(..count..)),
           fill = "blue", color = "black") +
  theme_minimal() + ylab("frequency") + xlab("i")
```



To find the acceptance rate, keep that as an extra data point.

```
chain_qotd_mrrtt_accrate <- function(steps) {
  burnin    <- steps
  datasteps <- steps
  x <- 1
  res <- rep(0, datasteps)
  m1 <- 2 * (runif(burnin) < 0.5) - 1
  u1 <- runif(burnin)
  for (i in 1:burnin) {
    x <- step_qotd_mrrtt(x, m1[i], u1[i])
  }
  m2 <- 2 * (runif(datasteps) < 0.5) - 1
  u2 <- runif(datasteps)
  for (i in 1:datasteps) {
    old_x <- x
    x <- step_qotd_mrrtt(x, m2[i], u2[i])
    res[i] <- old_x != x
  }
  return(res)
}
```

```
res <- replicate(10, mean(chain_qotd_mrrtt_accrate(10^4)))
tibble(
  est_mean = mean(res),
  est_err  = sd(res) / sqrt(length(res))
) |> kable()
```

est_mean	est_err
0.61529	0.0135131

An acceptance rate of about 61% is a touch too high, but in fact with this particular state space and density acceptance always occurs when the state moves to the left. Therefore, the acceptance rate will always be at least 50%, so this is not too bad.

A continuous example

This method works just as well on continuous problems. Here only one dimension will be used to keep things simple, but most often in practice MR^{2}T^2 is used on high dimensional problems.

Consider the probability of sampling from a standard normal distribution, A proposal move will involve adding a uniform over $[-s, s]$. Here s is the *scale* of the move. Other than that, things are pretty much the same as in the discrete case.

```
step_normal_mrrtt <- function(x, m, u) {
  y <- x + m
  if (u <= dnorm(y) / dnorm(x))
    return(y)
  else
    return(x)
}
```

Now run the chain as usual. A tibble will be used to return both the chain state and the acceptance rate data.

```
chain_normal_mrrtt <- function(steps, s = 0.1) {
  burnin    <- steps
  datasteps <- steps
  x <- 0
  m1 <- 2 * s * runif(burnin) - s
  u1 <- runif(burnin)
  for (i in 1:burnin) {
    x <- step_normal_mrrtt(x, m1[i], u1[i])
  }
  m2 <- 2 * s * runif(datasteps) - s
  u2 <- runif(datasteps)
  res_acc   <- rep(0, datasteps)
```

```
  res_state <- rep(0, datasteps)
  for (i in 1:datasteps) {
    old_x <- x
    x <- step_normal_mrrtt(x, m2[i], u2[i])
    res_acc[i] <- old_x != x
    res_state[i] <- x
  }
  return(tibble(res_acc, res_state))
}
```

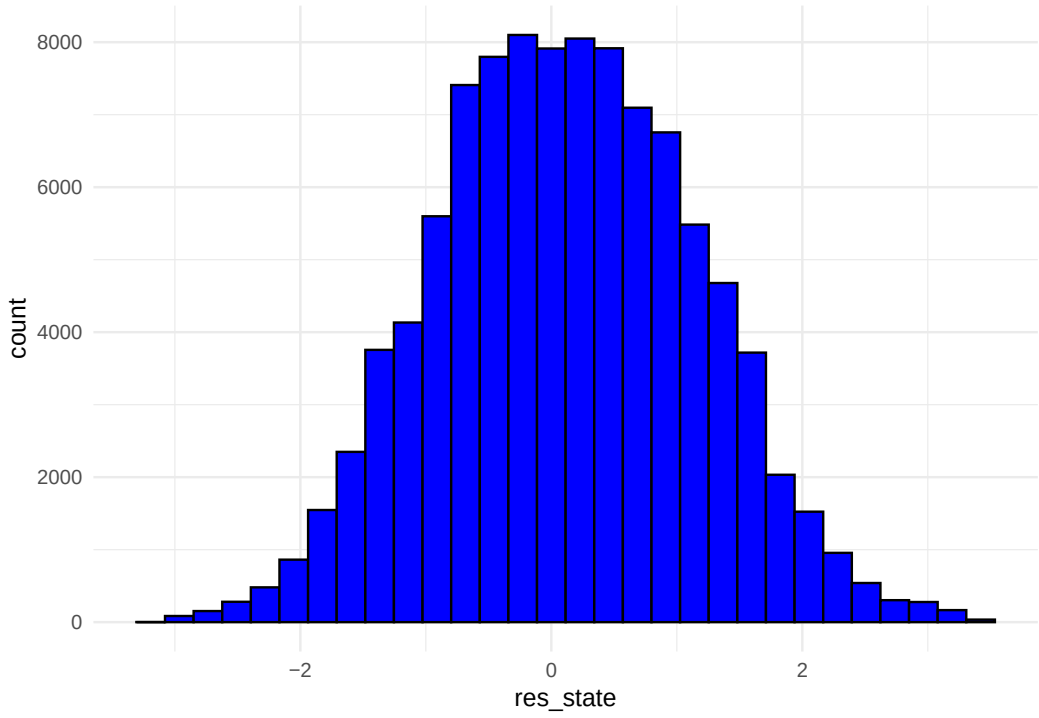
Gather the data:

```
res <- chain_normal_mrrtt(10^5, s = 0.1)
```

First look at the histogram of the output:

```
res |>
  ggplot() +
    geom_histogram(aes(res_state), fill = "blue", color = "blac
    theme_minimal()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binw
```



Next look at the summary statistics of the acceptance rate:

```
res |>
  summarize(
    est_rate = mean(res_acc),
    est_err  = sd(res_acc) / sqrt(length(res_acc))
  )
```

```
## # A tibble: 1 x 2
##   est_rate  est_err
##      <dbl>    <dbl>
## 1    0.980 0.000446
```

So the chain accepted around 98% of the time. Far too high! This can be lowered by increasing s, the scale parameter for the move.

```
res2 <- chain_normal_mrrtt(10^5, s = 0.5)
res2 |>
  summarize(
    est_rate = mean(res_acc),
    ese_err  = sd(res_acc) / sqrt(length(res_acc))
  )
```

```
## # A tibble: 1 x 2
##   est_rate  ese_err
##      <dbl>    <dbl>
## 1    0.901 0.000943
```

Down to 90%, still pretty high.

```
res3 <- chain_normal_mrrtt(10^5, s = 2)
res3 |>
  summarize(
    est_rate = mean(res_acc),
    est_err  = sd(res_acc) / sqrt(length(res_acc))
  )
```

```
## # A tibble: 1 x 2
##   est_rate est_err
##      <dbl>   <dbl>
## 1    0.632 0.00153
```

Closer to our target.

```
res4 <- chain_normal_mrrtt(10^5, s = 10)
res4 |>
  summarize(
    est_rate = mean(res_acc),
    est_err  = sd(res_acc) / sqrt(length(res_acc))
  )
```

```
## # A tibble: 1 x 2
##   est_rate est_err
##      <dbl>   <dbl>
## 1    0.160 0.00116
```

Oops, overshot.

```
res4 <- chain_normal_mrrtt(10^5, s = 5)
res4 |>
  summarize(
    est_rate = mean(res_acc),
    est_err  = sd(res_acc) / sqrt(length(res_acc))
  )
```

```
## # A tibble: 1 x 2
##   est_rate est_err
##      <dbl>   <dbl>
## 1    0.312 0.00147
```

There is a good acceptance rate! The chain is now moving about 31% of the time, which means that it is exploring new areas quickly. It should be noted that just because the acceptance rate is near $1/4$ does not *prove* that the chain is mixing quickly, but it is a good start.

Problems

18.1: Consider the unnormalized density

$$g_X(s) = \exp(-s^{3/2})\mathbb{I}(s \geq 0).$$

a) Write code for a Metropolis-Hastings step with proposal move $Y = X + M$ where $M \sim \mathsf{Unif}([-2, 1])$.

b) Use your code with 10 replications of 1000 burnin and data gathering steps to estimate $\mathbb{E}[X]$. Report your result as $a \pm b$.

18.2: Consider the unnormalized density $f(i) = [1/\sqrt{i}]\mathbb{I}(i \in \{1, 2, \ldots\}$.

a. Write R code for a single update in a $\mathsf{MR}^2\mathsf{T}^2$ Markov chain that has f as the stationary distribution.

b. Use your code, with 10^4 steps in the Markov chain and 10 replications to estimate $\mathbb{E}[X]$ for $X \sim f$.

Chapter 19

The Metropolis-Hastings method

Question of the Day

Design a Markov chain on $\{1, 2, \ldots\}$ using $\mathbb{P}(M = -1) = 0.6$ and $\mathbb{P}(M = 1) = 0.4$ for proposals whose stationary distribution is

$$\mathbb{P}(X = i) = \frac{C}{i^2}\mathbb{I}(i \in \{1, 2, \ldots\}).$$

Summary

- Given a target density p and a proposal density $q(x, \cdot)$ from the current state x such that for all pairs of states (a, b), $(q(a, b) > 0) = (q(b, a) > 0)$, the **Metropolis-Hastings** algorithm accepts the proposal with probability

$$\min\left(1, \frac{p(y)q(y, x)}{p(x)q(x, y)}\right).$$

- The Metropolis-Hastings chain has stationary distribution density p.

The $\mathsf{MR}^2\mathsf{T}^2$ chain can be used to turn a symmetric random walk with partially reflecting boundaries into a chain whose stationary distribution is the target distribution. The mechanism is to accept or reject the proposed state with a certain probability formed from using an auxiliary random variable.

The next step is to try to turn a *nonsymmetric* random walk into a chain whose stationary distribution is the target distribution. The way this will be done is using the idea of *reversibility* of a Markov chain.

19.1 Reversible Markov chains

To understand what reversible means, consider the simple symmetric random walk with partially reflecting boundaries on $\{1, 2, 3, 4, 5\}$, where the move is $M \sim \mathsf{Unif}(\{-1, 1\})$.

Suppose $X_t \sim \pi$, and then consider the distribution of the pair (X_t, X_{t+1}). Each of these has a density. For instance:

$$\begin{aligned}\mathbb{P}((X_t, X_{t+1}) = (3,4)) &= (1/5)(1/2)\\ \mathbb{P}((X_t, X_{t+1}) = (1,1)) &= (1/5)(1/2)\\ \mathbb{P}((X_t, X_{t+1}) = (3,5)) &= (1/5)(0).\end{aligned}$$

The probability of each pair (x, w) is the probability that $X_t = w$ times the probability that $X_{t+1} = w$ given that $X_t = w$. Now consider the reverse of the three pairs given above:

$$\begin{aligned}\mathbb{P}((X_t, X_{t+1}) = (4,3)) &= (1/5)(1/2)\\ \mathbb{P}((X_t, X_{t+1}) = (1,1)) &= (1/5)(1/2)\\ \mathbb{P}((X_t, X_{t+1}) = (5,3)) &= (1/5)(0).\end{aligned}$$

The probabilities are the same! This means that if I start the Markov chain $X_0 \sim \pi$ and run the Markov chain forward in time $(X_0, X_1, \ldots, X_n)$, the distribution of these states will be exactly the same as that of $(X_n, X_{n-1}, \ldots, X_0)$ where $X_n \sim X_n$. Running video of this chain looks the same whether the video is run forward or in reverse. Such a Markov chain is called *reversible with respect to* p.

Definition 42
Suppose that for $X_t \sim \nu$, (X_t, X_{t+1}) has the same distribution as (X_{t+1}, X_t). Then the chain is **reversible with respect to** ν.

For (X_t, X_{t+1}) to have the same distribution as (X_{t+1}, X_t), X_t and X_{t+1} must have the same distribution. This means that reversible with respect to ν implies that ν is a stationary distribution for the chain.

Fact 33
If a Markov chain is reversible with respect to ν, then ν is a stationary distribution for the chain.

Is it easy to tell when a Markov chain is reversible? For discrete state space Markov chains, it is simple. Let $p(x) = \mathbb{P}(X = x)$ if $X \sim \nu$, and let $m(x, y) = \mathbb{P}(X_{t+1} = x \mid X_t = x)$ in the chain. Then the chain is reversible if and only if for all $x, y \in \Omega$, it holds that $p(x)m(x, y) = p(y)m(y, x)$.

Fact 34
A Markov chain over a discrete space is reversible with respect to ν if and only if

$$(\forall x, y, \Omega)(p(x)m(x, y) = p(y)m(y, x)),$$

where $p(x)$ is $\mathbb{P}(X = x)$ for $X \sim \nu$, and $m(x, y) = \mathbb{P}(X_{t+1} = y \mid X_t = x)$.

Suppose for all x, y that $p(x)m(x,y) = p(y)m(y,x)$. Let $x, y \in \Omega$.

$$\begin{aligned}\mathbb{P}(X_t = x, X_{t+1} = y) &= \mathbb{P}(X_t = x)\mathbb{P}(X_{t+1} = y \mid X_t = x)\\ &= p(x)m(x,y)\\ &= p(y)m(y,x)\\ &= \mathbb{P}(X_t = y)\mathbb{P}(X_{t+1} = x \mid X_t = y)\\ &= \mathbb{P}(X_{t+1} = x, X_t = y).\end{aligned}$$

Since $\mathbb{P}((X_t, X_{t+1}) = (x,y)) = \mathbb{P}((X_{t+1}, X_t) = (x,y))$ for all x and y, they must have the same distribution, and the chain is reversible.

Now assume the chain is reversible. Then let $(x,y) \in \Omega$. Then

$$\mathbb{P}((X_t, X_{t+1}) = (x,y)) = \mathbb{P}((X_{t+1}, X_t) = (x,y)),$$

and as seen earlier the left hand side is $p(x)m(x,y)$ and the right hand side $p(y)m(y,x)$. So the two expressions are equal.

19.2 Metropolis-Hastings

The MR^{2}T^2 method combined with Hastings modification is commonly known as *Metropolis-Hastings*, or MH. The idea is to not just use the target density in the acceptance ratio, but also the density of the proposal state.

Definition 43
For a target density p and set of densities q_x such that

$$(\forall x, y \in \Omega)(q(x,y) > 0 \leftrightarrow q(y,x) > 0),$$

the **Metropolis-Hastings chain** is as follows.

For state $X_t \in \Omega$, propose Y such that $\mathbb{P}(Y = y \mid X_t = x) = q(x,y)$. Let U be a standard uniform independent of X_t and Y. If

$$U \le \frac{p(Y)q(Y,X_t)}{p(X_t)q(X_t,Y)}$$

set $X_{t+1} = Y$, otherwise $X_{t+1} = X_t$.

Fact 35
The Metropolis-Hasting Markov chain is reversible with respect to density p.

Let x and y be distinct elements of Ω. In the case $x = y$, then $p(x)m(x,y) = p(y)m(y,x) = p(x)m(x,x)$ by definition.

So suppose $x \neq y$. Then to move the state from $X_t = x$ to $X_{t+1} = y$, it is necessary that both $Y = y$, and $U \le p(y)q(y,x)/[p(x)q(x,y)]$. This occurs with probability

$$m(x,y) = q(x,y)\min\left\{1, \frac{p(y)q(y,x)}{p(x)q(x,y)}\right\}.$$

Similarly,

$$m(y,x) = q(y,x)\min\left\{1, \frac{p(x)q(x,y)}{p(y)q(y,x)}\right\}.$$

Suppose $[p(y)q(y,x)]/[p(x)q(x,y)] > 1$. Then

$$p(x)m(x,y) = p(x)q(x,y),$$

and

$$m(y,x) = p(y)q(y,x)\cdot\frac{p(x)q(x,y)}{p(y)q(y,x)} = p(x)q(x,y) = p(x)m(x,y).$$

The case where $[p(y)q(y,x)]/[p(x)q(x,y)] \leq 1$ is similar.

From our earlier fact, this means that p is the density of a stationary distribution for the chain.

Solving the Question of the Day

Let us apply this procedure to the question of the day. For $\Omega = \{1, 2, \ldots\}$, the MH acceptance ratio is

$$\begin{aligned}\frac{p(y)q(y,x)}{p(x)q(x,y)} &= \frac{(C/y^2)\mathbb{I}(y\in\Omega)[0.6\mathbb{I}(x=y-1)+0.4\mathbb{I}(x=y+1)]}{(C/x^2)\mathbb{I}(x\in\Omega)[0.6\mathbb{I}(y=x-1)+0.4\mathbb{I}(y=x+1)]}\\ &= \frac{x^2\mathbb{I}(y\in\Omega)[0.6\mathbb{I}(x=y-1)+0.4\mathbb{I}(x=y+1)]}{y^2\mathbb{I}(x\in\Omega)[0.6\mathbb{I}(x=y+1)+0.4\mathbb{I}(x=y-1)]}\\ &= \frac{x^2}{y^2}\mathbb{I}(x,y\in\Omega)[(6/4)\mathbb{I}(x=y-1)+(4/6)\mathbb{I}(x=y+1)].\end{aligned}$$

So one step in the Markov chain can be setup as follows.

Algorithm 7
step_qotd_mh(x, m, u)

1. Let $y \leftarrow x + m$
2. If $u \leq (x^2/(y+\mathbb{I}(y=0))^2)((3/2)\mathbb{I}(y=x+1)+(4/6)\mathbb{I}(y=x-1))(\mathbb{I}(y\geq 1)$ return y.
3. Else return x.

19.3 Metropolis Hastings in general state spaces

When dealing with general state spaces, the acceptance ratio becomes what is known as a *Radon-Nikodym derivative*, which is a generalization of the concept of a derivative.

Definition 44
For two measures ν_1 and ν_2 such that for all measurable A, $\nu_1(A) > 0$ implies $\nu_2(A) > 0$, the **Radon-Nikodym derivative** is a function $f = d\nu_1/\nu_2$ such that for all measurable B,

$$\int_B \frac{d\nu_1}{d\nu_2}\, d\nu_2 = \int_B d\nu_1 = \nu_1(B).$$

To convert a general Markov chain to a reversible chain with stationary distribution π, consider the current state x and the proposed state y. Then the density of (X_t, X_{t+1}) when $X_t \sim \pi$ at (x, y) divided by the density of (X_t, X_{t+1}) at (y, x) is the acceptance ratio in Metropolis-Hastings.

Definition 45
Suppose there is a function $q : \Omega^2 \to [0, \infty)$ such that for every $x \in \Omega$, the random variable Y_x has density $f_{Y_x}(y) = q(x, y)$ and p is the density of the target distribution. Then $\{X_i\}$ is the **Metropolis Hastings** Markov chain if it works as follows. First, draw Y_x given $X_t = x$. Second, draw U a standard uniform independent of all other random variables. Third, if

$$U \le \frac{p(Y_x)q(Y_x, x)}{p(x)q(x, Y_x)},$$

then set $X_{t+1} = Y_x$. Otherwise, $X_{t+1} = X_t$.

19.4 Simulation of Metropolis-Hastings in R

First load in the `tidyverse` and `knitr` packages.

```
library(tidyverse)
library(knitr)
```

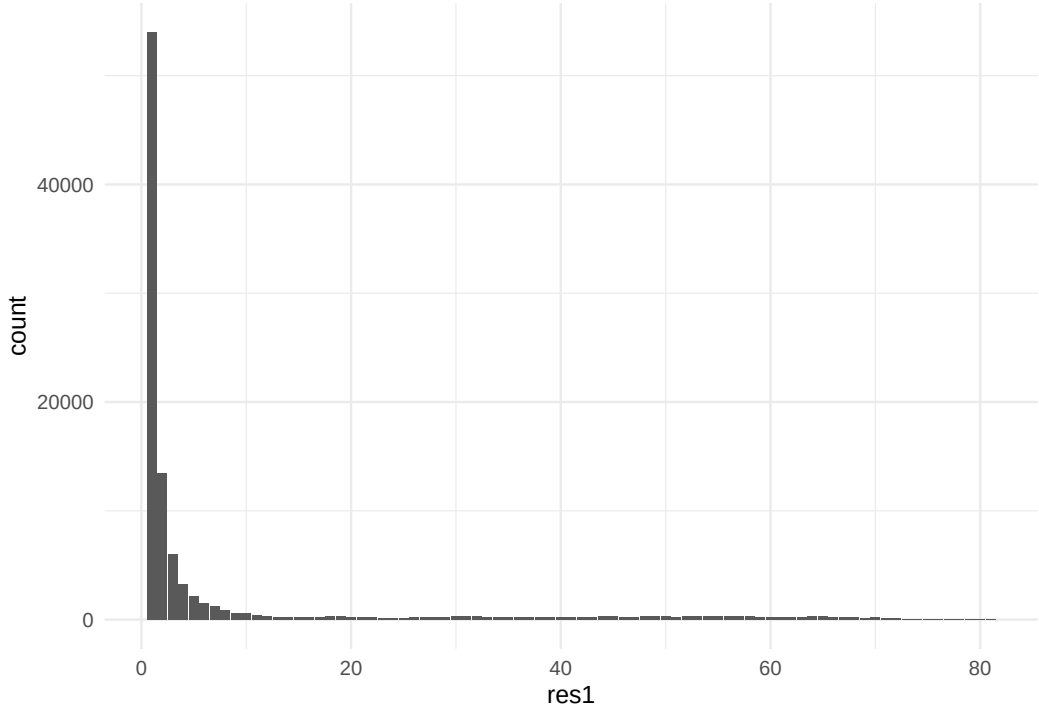
One way to set up the step is to define a separate function for the probability of $(X_t, X_{t+1}) = (a, b)$. The input `pr` (stands for probability right) is $\mathbb{P}(M = 1)$.

```
step_qotd_mh1 <- function(x, m, u, pr) {
  y <- x + m
  r <- function(a, b, pr)
    return(1 / (a + (a == 0))^2 * (a != 0) *
           (((1 - pr) * (b <= a) + pr * (b > a))))
  if (u < r(y, x, pr) / r(x, y, pr))
    return(y)
  else
    return(x)
}
```

Now create the chain and take steps.

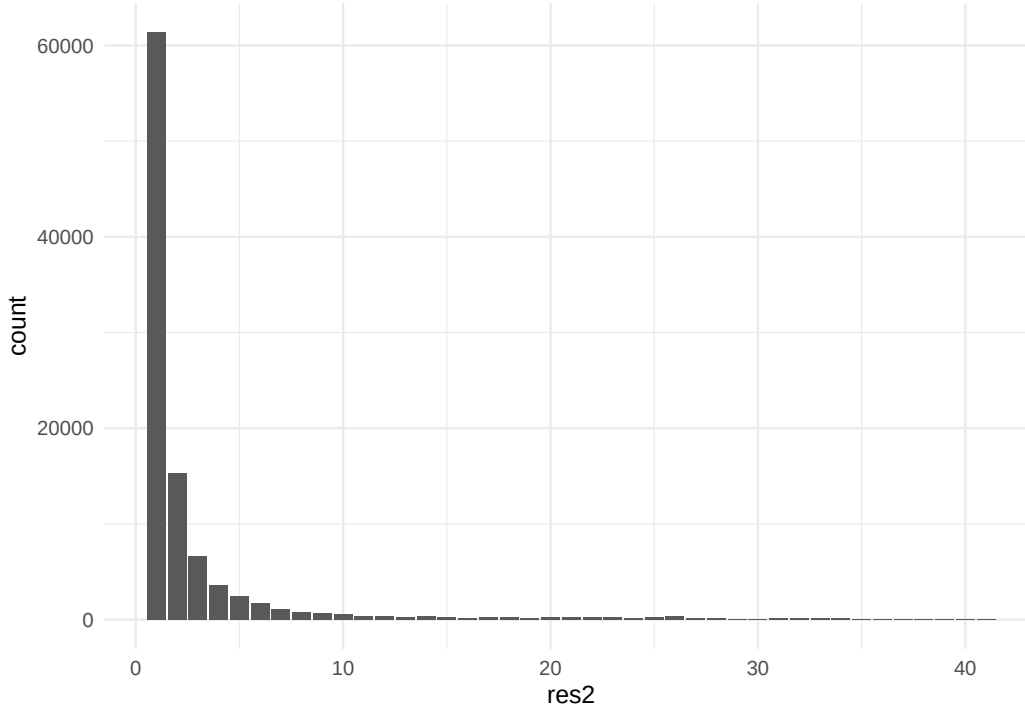
```
chain_qotd_mh1 <- function(steps, prob_right) {
  burnin    <- steps
  datasteps <- steps
  x <- 1
  m1 <- 2 * (runif(burnin) < prob_right) - 1
  u1 <- runif(burnin)
  for (i in 1:burnin) {
    x <- step_qotd_mh1(x, m1[i], u1[i], prob_right)
  }
  res <- rep(0, datasteps)
  m2 <- 2 * (runif(datasteps) < prob_right) - 1
  u2 <- runif(datasteps)
  for (i in 1:datasteps) {
    x <- step_qotd_mh1(x, m2[i], u2[i], prob_right)
    res[i] <- x
  }
  return(res)
}
```

```
res1 <- chain_qotd_mh1(10^5, 0.2)
ggplot() +
  geom_bar(aes(res1)) +
  theme_minimal()
```



Note that changing the proposal move does not change the output!

```
res2 <- chain_qotd_mh1(10^5, 0.1)
ggplot() +
  geom_bar(aes(res2)) +
  theme_minimal()
```



However, the acceptance rates might change.

```
tibble(
  "0.2 right acc rate " = mean(diff(res1) != 0),
  "0.1 right acc rate" = mean(diff(res2) != 0)
) |> kable()
```

0.2 right acc rate	0.1 right acc rate
0.2744627	0.201122

Problems

19.1: Consider the unnormalized density $f(i) = [1/\sqrt{i}]\mathbb{I}(i \in \{1, 2, \ldots\}$.

a) Write R code for a single update in a $\mathsf{MR}^2\mathsf{T}^2$ Markov chain with proposed move

$$\mathbb{P}(M = 1) = 0.8, \ \mathbb{P}(M = -1) = 0.2$$

that has f as the stationary distribution.

b) Use your code, with 10^4 steps in the Markov chain and 10 replications to estimate $\mathbb{E}[X]$ for $X \sim f$.

Part III

Monte Carlo for Finance

Chapter 20

Pricing derivatives

Question of the Day

Say that stock price has a 40% chance of rising 10% and a 60% of falling 5% each day. A particular *Asian option* is bought at time $t = 0$. At time $t = 5$, it has value equal to $\max((S_1 + S_2 + S_3 + S_4 + S_5)/5 - 110, 0)$. What price should an investor pay for this option?

Summary

- A **derivative** is a financial security whose value is a function of some other asset.
- A **martingale** is a stochastic process $\{S_t\}$ such that $\mathbb{E}[S_t \mid S_r] = S_r$ for all $r \leq t$.
- A probability distribution $\mathbb{P}^*$ is a **(**risk neutral equivalent martingale measure) if under $\mathbb{P}^*$, $\{S_t\}$ is a martingale.
- The **Fundamental Theorem of Asset Pricing** says that if $\mathbb{P}^*$ is a risk neutral equivalent martingale measure for $\{S_t\}$, then for any derivative, $f(S_t)$ is also a risk neutral equivalent martingale measure.
- This allows us to use Monte Carlo to price derivatives: first create a risk neutral equivalent martingale measure, then simulate $f(S_t)$ under that measure and average to estimate the price of the derivative.

A share of stock, a truck, a house, a card collection, these are all things that have some value. That makes these things *assets*. If the asset is easy to trade among different people, the asset becomes a *financial instrument*. These financial instruments are often abstract entities, such as the contractual ability to buy another asset at a certain price.

Securities are financial instruments that are *fungible*, meaning that owning one asset is equivalent to owning another copy of the asset. For instance, a share of stock is fungible

because it does not matter which share of a class of stock that you each. Each share within that class of stock has the same voting rights and so on.

In the question of the day, each share of stock that can be purchased is a security, as they are fungible. For securities, the total value will just be the price per share times the number of shares bought.

That brings us to a *derivative*.

Definition 46
A **derivative** is a financial security whose value is a function of some other asset.

The *Asian option* mentioned in the Question of the Day is such a derivative. Its value at time 5 is a function of the price of the stock over times from 0 to 5:

$$A_5 = \max\left(\frac{S_1 + S_2 + \cdots + S_5}{5} - 110, 0\right).$$

It is called an *option* because the purchaser of the derivative has the option to either exercise the option at time 5, in which case the owner receives $(S_1 + \cdots + S_5)/5 - 110$. Or the owner might choose to not exercise the option, and receives 0 dollars. That means the value of the option at time 5 is the larger of these two values (assuming the owner makes the decision to maximize the value!)

Various types of options, European, American, Asian, are named after geographical locations. The Asian type of option, also known as an *average option* bases its value on the average value of a security over some time period.

20.1 European options

To understand how to price the option, consider a simpler example. Suppose the current stock price is 100 dollars, and as before there is a 40% chance of the stock going up 10%, and a 60% chance that it goes down 10%. Moreover, there is a *European option* that allows the owner the option to buy the stock for \$105 after 1 day. What should someone be willing to pay for this option?

Well, if the stock goes up, it is worth \$110, so if bought for \$105, the owner could immediately sell it for a gain of \$5. However, if the stock went down to \$95, then the owner would just tear up the option. So the overall value of the option as time 1 is

$$D_1 = \max(S_1 - 105, 0) = (S_1 - 105)^+,$$

where S_1 is the price of the stock after one day and the superscript $+$ just means take the larger of the value and 0.

A common mistake is to say that the value of the option at time 0 (D_0) should be the expected value of D_1. But that ignores the risk (or lack of risk) inherent in the option.

A better way of thinking about this is the idea of an *Arrow Security*.

Definition 47
Suppose exactly one of the events $\{a_1, \ldots, a_n\}$ must be true. Then an **arrow security** for outcome i has value 1 if a_i is true, and 0 otherwise.

An arrow security is the financial equivalent of the indicator function. For the stock example, suppose $a_u = (S_1 = 110)$ and $a_d = (S_1 = 95)$. Then in the model either a_u or a_d (but not both) must be true.

Moreover, the option payoff can be *replicated* by buying 5 shares of A_u, since if the stock goes up, the value of the option is 5. The stock can be replicated by buying 110 shares of A_u and 95 shares of A_d. Finally, if someone owns one share of A_u and one share of A_d, after one day they are guaranteed to have one dollar.

Note that all of this is complicated by the existence of a risk-free interest rate*, which is a rate of interest given by a money market account. For simplicity, this will be ignored in this chapter, but for those with more advanced finance knowledge, rest assured that the ideas in this section can be extended easily to incorporate interest rates.*

So that gives the following equations:

$$\begin{aligned} D_0 &= 5\,\text{price}(A_u) \\ S_0 &= 110\,\text{price}(A_u) + 95\,\text{price}(A_d) \\ 1 &= \text{price}(A_u) + \text{price}(A_d). \end{aligned}$$

Because the A_i assets return either 0 or 1 dollars, the price should not be less than 0 or greater than 1. Exactly one is true, and they add up to 1. Therefore, the prices of the A_i assets form probabilities!

Let $p^*(u) = \text{price}(A_u)$ and $p^*(d) = \text{price}(A_d)$. Then

$$S_0 = 110p^*(u) + 90p^*(d) = \mathbb{E}_*(S_1).$$

The right hand side is the expected value of S_1 using the star probabilities. Similarly,

$$D_0 = 5p^*(u) + 0p^*(d) = \mathbb{E}_*(D_1)$$

is the expected value of D_1 under the star probabilities.

Solving

$$\begin{aligned} 110p^*(u) + 95p^*(d) &= 100 \\ p^*(u) + p^*(d) &= 1 \end{aligned}$$

gives

$$p^*(u) = \frac{1}{3},\ p^*(d) = \frac{2}{3}.$$

Note that this is different from $p(u) = 0.4$ and $p(d) = 0.6$, the probabilities in the model. In general *the star probabilities are different from the model probabilities.* That is why it is important to have the star notation, it indicates that the model probabilities are *not* being used.

20.2 Arbitrage and martingales

It was assumed in this calculation that no one would be willing to pay more than \$1 or less than \$0 for an A_i asset, because otherwise they would be guaranteed to make or lose money. This principle that prices should be set so that there is no way to make money with probability 1 is called the *no arbitrage principle*.

Definition 48
A set of prices has **arbitrage** if there is a way to buy and sell assets such that the probability of making a profit greater than 0 equals 1.

When a set of prices is arbitrage free, the star probabilities can be used to make $\mathbb{E}_*(S_1 \mid S_0) = S_0$. This extends to any time greater than 0, making the stochastic process a *martingale*.

Definition 49
Assuming the appropriate conditional expectations exists, if

$$(\forall t)(\mathbb{E}[S_t \mid S_0, \ldots, S_{t-1}] = S_{t-1}),$$

call $\{S_t\}$ a **martingale**.

Induction can then be used to show the following.

Fact 36
For a martingale where S_0 is constant, $\mathbb{E}[S_t] = S_0$ for all $t \geq 0$.

Definition 50
If using probability measure $\mathbb{P}^*$, $\{S_t\}$ is a martingale, call $\mathbb{P}^*$ a **risk neutral equivalent martingale measure**.

20.3 The Fundamental Theorem of Asset Pricing

The Fundamental Theorem of Asset Pricing states that in a price system with no arbitrage, the star probabilities can be used to price any derivative of an asset.

Theorem 9
The Fundamental Theorem of Asset Pricing
Given a set of prices with no arbitrage, suppose that $\{S_t\}$ is the price of an asset at time t, and $\mathbb{P}^*$ is the risk neutral equivalent martingale measure. Then for any computable function f, $\{D_t = f(S_0, S_1, \ldots, S_t)\}$ is also a martingale under $\mathbb{P}^*$.

How is this theorem used in practice?

1) For the base asset $\{S_t\}$, find the risk neutral equivalent martingale measure $\mathbb{P}^*$.

2) Use $D_0 = \mathbb{E}[D_t \mid D_0]$ to find the value of D_0.

Sometimes this first step can be done analytically. For instance, for $D_1 = (S_1 - 105)^+$,

$$D_0 = \mathbb{E}_*[D_1] = (1/3)5 + (2/3)0 = 5/3.$$

That is, the European option should be priced at about $\boxed{\$1.67}$. (As is common in this context, the price of the option was rounded up to the nearest penny.)

20.4 Pricing Asian options in R

The Asian option in the Question of the Day is more complex. Because the price involves S_1, S_2, S_3, S_4, S_5, there are $2^5 = 32$ possible outcomes to consider. Simple modifications to the model could make this even harder to calculate analytically.

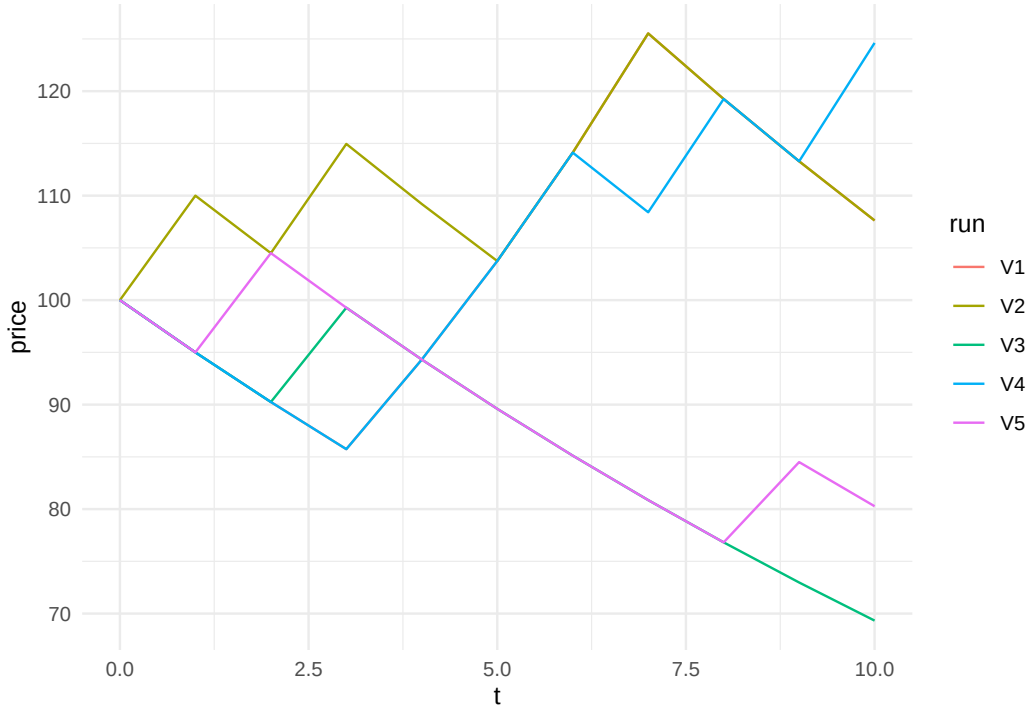
Therefore, Monte Carlo is often used to price these types of options. To price the option, first, the set of stock prices must be simulated. The model where a stock either goes up or down by the same amount with the same prices is called the *binomial model*.

Let `u` hold one factor that the stock can change by, `d` the other factor it can change by (typically $u > d$), `s_0` the starting price, `p` the two dimensional vector $(p(u), p(d))$, and `t` the number of steps to take in the model. The cumulative product function (`cumprod`) can be used to take the products of the ups and downs.

```
rbinomialmodel <- function(t, u, d, p, s_0 = 1) {
  changes <- (runif(t) < p[1])
  return( s_0 *
            c(1, cumprod(rep(u, t)^changes) *
                 cumprod(rep(d, t)^(1 - changes))) )
}
```

Several runs of the stock price under the true model then looks like this.

```
stock_price <- replicate(5, rbinomialmodel(10, 1.1, 0.95, 0.4,
  as_tibble() |>
  mutate(t = 0:10) |>
  pivot_longer(V1:V5, names_to = "run", values_to = "price")
stock_price |>
  ggplot() +
    geom_line(aes(x = t, y = price, color = run)) +
    theme_minimal()
```



The value of the Asian option is the average of all values but the first minus the *strike price* 110, unless this is smaller than 0, in which case it is 0.

```
asian_value <- function(s, K) {
  max(mean(s[-1]) - K, 0)
}
```

For the data above:

```
stock_price |>
  group_by(run) |>
  summarize(asian_value(price, 110)) |>
  kable()
```

run	asian_value(price, 110)
V1	0.000000
V2	2.220569
V3	0.000000
V4	0.000000
V5	0.000000

To estimate the price, run this many times **using the p^* values**.

```
trials <- 100000
res <- replicate(trials,
                 asian_value(rbinomialmodel(5, 1.1, 0.95, 1 / 3
tibble(
  est_mean = mean(res),
  est_err  = sd(res) / sqrt(length(res))
) |> kable()
```

est_mean	est_err
1.24623	0.01141

Using this many trials, the estimated mean was $\boxed{1.234 \pm 0.012}$.

Chapter 21

Pricing derivatives with control variates

Question of the day

Say that stock price has a 40% chance of rising 10% and a 60% of falling 5% each day. A particular *Asian option* is bought at time $t = 0$. At time $t = 5$, it has value equal to $\max((S_1 + S_2 + S_3 + S_4 + S_5)/5 - 110, 0)$. Use a control variate to estimate what price should an investor pay for this option while keeping the variance as small as possible.

Summary

- Suppose that random variable X has density f_X and the goal is to estimate the mean of $h(X)$. Suppose instead it is possible to draw Y from density f_Y. Then

$$\mathbb{E}[h(X)] = \mathbb{E}\left[h(Y) \cdot \frac{f_X(Y)}{f_Y(Y)}\right]$$

  is **importance sampling**, and this can be used for financial Monte Carlo in the same way as for other Monte Carlo methods.

- A **control variate** is a random variable T with known mean that is correlated with the random variable W being used for our simulations. Then use

$$f(W, T) = W - \frac{\mathrm{Cov}(W, T)}{\mathbb{V}(T)}(T - \mathbb{E}[T])$$

  as an unbiased estimate of the target value. Note

$$\mathbb{V}(f(W, T)) = (1 - \mathrm{Cor}(W, T)^2)\mathbb{V}(W),$$

  so this estimate can have a much smaller variance than the original W.

Suppose a system of prices does not have arbitrage. For p^* the *risk neutral equivalent martingale measure* probabilities, the Fundamental Theorem of Asset Pricing says that for all $t \geq 0$ it holds that $\mathbb{E}_*[f(S_0, \ldots, S_t)] = f(S_0)$ for all computable functions f.

This statement can be used with $\mathbb{E}_*[S_t] = S_0$ to find the probabilities p^*, but then finding $\mathbb{E}(f(S_0, \ldots, S_t))$ involves an integration whose calculation typically grows exponentially fast in t.

Definition 51
A stock price S_t follows a **binomial model** if there are positive constants u and d, and a probability p such that at each step, the stock has a p chance of $S_{t+1} = uS_t$, and a $1-p$ chance of $S_{t+1} = dS_t$.

In the Question of the Day, $u = 1.1$ and $d = 0.95$. Knowing that

$$100 = p^*(u) \cdot 110 + p^*(d) \cdot 95$$

and $p^*(u) + p^*(d) = 1$ gives the star probabilities as $p^*(u) = 1/3$ and $p^*(d) = 2/3$. This gives the probabilities, now the question is how to find the mean of the derivative price.

Fact 37
For the binomial model,

$$p^*(u) = \frac{1-d}{u-d},\ p^*(d) = \frac{u-1}{u-d},$$

as long as $d < 1 < u$.

If both d and u are greater (or both smaller) than 1, then the stock is guaranteed to make (lose) money which means arbitrage exists!

Proof. Let $p^* = p^*(u)$ be the probability that the stock goes up, then $1-p^*$ is the probability it goes down. Hence,

$$S_0 = uS_0p^* + dS_0(1-p^*),$$

and canceling S_0 and solving gives the result. □

Once the risk neutral measure is found, Monte Carlo can be used to provide an estimate for the mean by drawing samples from the random variable. For $W_1, \ldots, W_n$ iid W with finite standard deviation,

$$\mathrm{SD}(\bar{W}) = \mathrm{SD}(W)/\sqrt{n},$$

so if $\mathrm{SD}(W)$ is large, it can take a long time to get the standard deviation below a certain level.

21.1 Changing the random variable

In the case of the Asian option in the Question of Day, most of the time the option will not be exercised. With the binomial model with $u = 1.1$ and $d = 0.95$, the Asian option will be exercised only about 22% of the time.

If p^* were higher, then the option would be exercised more often. For instance, if $p = 0.5$, then the option is exercised about 58% of the time. One could create a new random variable Y with this simulation instead of X

However, in general $f(Y)$ will be different from $f(X)$, and $\mathbb{E}(f(Y)) \neq \mathbb{E}(f(X)$. How to convert?

The answer lies in a form of importance sampling. The Y simulation is coming from a different density than the X simulation that preceded it. To compensate for this fact, multiply by the proper density of the X density versus the Y density.

Theorem 10
Importance sampling with random variables
Suppose X and Y have densities f_X and f_Y with respect to measure ν, and $(f_X(s) > 0) \to (f_Y(s) > 0)$. Then for a computable function h,

$$\mathbb{E}\left[h(X)\right] = \mathbb{E}\left[h(Y)\frac{f_X(Y)}{f_Y(Y)}\right]$$

Proof. From the way expectations of functions of a random variable are defined:

$$\begin{aligned}
\mathbb{E}\left[h(Y)\frac{f_X(Y)}{f_Y(Y)}\right] &= \int_{f_Y(y)>0} h(y)\frac{f_X(y)}{f_Y(y)} \cdot f_Y(y)\, d\nu \\
&= \int_{f_Y(y)>0} h(y) f_X(y)\, d\nu \\
&= \int_{f_X(y)>0} h(y) f_X(y)\, d\nu \\
&= \mathbb{E}[h(X)].
\end{aligned}$$

□

Importance sampling for the binomial model

In the binomial model, the underlying random choices are for each time step did the stock multiply by u or did it multiply by d. Bernoulli random variables can be used at each step to determine if the stock used u as the factor.

For instance, if $X = (X_1, X_2, X_3) = (1, 1, 0)$, that means that $S_1 = uS_0$, $S_2 = uS_1$, and $S_3 = dS_2$. The density of X at $(1, 1, 0)$ is $p_X \cdot p_X \cdot (1 - p_X)$. That makes the ratio of the density of X using p_X and Y using p_Y at $(1, 1, 0)$

$$\frac{p_X \cdot p_X \cdot (1 - p_X)}{p_y \cdot p_y \cdot (1 - p_Y)},$$

or $(p_X/p_Y)^2[(1 - p_X)/(1 - p_Y)]$[1].

More generally, if the stock goes up $\sum Y_i$ times, then the ratio is

$$\frac{f_X(Y)}{f_Y(Y)} = \left(\frac{p_X}{p_Y}\right)^{\sum Y_i}\left(\frac{1 - p_X}{1 - p_Y}\right)^{n - \sum Y_i}.$$

21.2 Control Variates

Another way to reduce variance is to work with a random variable that is known to be positively (or negatively) correlated with the target variable.

Suppose that a is our target for estimation, and $\hat{a}$ is an estimate for a. Now suppose that $\hat{b}$ has expected value b. Then for any real constant c,

$$\hat{a}_c = \hat{a} + c(\hat{b} - b)$$

will have mean also equal to $\hat{a}$. The variance of the new estimate will be (by the sum of variance rule)

$$\mathbb{V}(\hat{a}_c) = \mathbb{V}(\hat{a}) + c^2\mathbb{V}(\hat{b}) + 2c\,\mathrm{Cov}(\hat{a}, \hat{b}).$$

This is quadratic in c. To make this as small as possible, choose

$$c = -\frac{\mathrm{Cov}(\hat{a}, \hat{b})}{\mathbb{V}(\hat{b})}.$$

Under these conditions:

$$\mathbb{V}(\hat{a}_c) = (1 - \mathrm{Cor}(\hat{a}, \hat{b})^2)\mathbb{V}(\hat{a}),$$

so the more correlated the control variate is with the original variable, the better.

This can be summarized as follows.

Definition 52
Suppose W has mean a and T with known mean is correlated with W. Then T is a **control variate**,

$$f(W, T) = W - \frac{\mathrm{Cov}(W, T)}{\mathbb{V}(T)}(T - \mathbb{E}(T)),$$

also has mean a, and

$$\mathbb{V}(f(W, T)) = (1 - \mathrm{Cor}(W, T)^2)\mathbb{V}(W).$$

A control variate for Asian options

For instance, in the Asian option, the final value of the stock can be used as a control variate. When this final value is higher, the value of the Asian option tends to be higher as well. Since it is evaluated under the risk neutral measure, it has expected value equal to S_0. The variance of this random variable can be computed analytically, or simply estimated.

21.3 Reducing variance in R

Using importance sampling

To utilize these ideas in R, it is necessary to keep track of how many times the stock went up or down in a particular run.

```
raopt <- function(t, u, d, p_x, p_y, s_0, K) {
  y     <- (runif(t) < p_y)
  n_up  <- sum(y)
  ratio <- (p_x / p_y)^(n_up) *
             ((1 - p_x) / (1 - p_y))^(t - n_up)
  s     <- s_0 * cumprod(u^y) * cumprod(d^(1 - y))
  return(max(mean(s) - K, 0) * ratio)
}
```

Now replicate to get the answer using the risk neutral martingale measure (so $p_X = p_Y = 1/3$.)

```
trials <- 1000
res <- replicate(trials,
                 raopt(5, 1.1, 0.95, 1 / 3, 1 / 3, s_0 = 100,
                       K = 110))
tibble(
  est_mean = mean(res),
  est_err  = sd(res) / sqrt(length(res))
) |> kable()
```

est_mean	est_err
1.167809	0.111179

Now change p_Y to be $1/2$.

```
trials <- 1000
res <- replicate(trials,
                 raopt(5, 1.1, 0.95, 1 / 3, 1 / 2, s_0 = 100,
                       K = 110))
tibble(
  est_mean = mean(res),
  est_err  = sd(res) / sqrt(length(res))
) |> kable()
```

est_mean	est_err
1.226008	0.0560491

Note that the estimated error is close to half what it was in the first simulation! Can we do better?

```
trials <- 1000
res <- replicate(trials,
                 raopt(5, 1.1, 0.95, 1 / 3, 0.7, s_0 = 100,
                       K = 110))
```

```
tibble(
  est_mean = mean(res),
  est_err  = sd(res) / sqrt(length(res))
) |> kable()
```

est_mean	est_err
1.208531	0.0440722

Can this be pushed further?

```
trials <- 1000
res <- replicate(trials,
                 raopt(5, 1.1, 0.95, 1 / 3, 0.8, s_0 = 100,
                       K = 110))
tibble(
  est_mean = mean(res),
  est_err  = sd(res) / sqrt(length(res))
) |> kable()
```

est_mean	est_err
1.264674	0.0603751

So at a certain point, this change does not improve things.

Then to get the best accuracy, use our choice of $p_Y = 0.7$ with many samples.

```
trials <- 100000
res <- replicate(trials,
                 raopt(5, 1.1, 0.95, 1 / 3, 0.7, s_0 = 100,
                       K = 110))
tibble(
  est_mean = mean(res),
  est_err  = sd(res) / sqrt(length(res))
) |> kable()
```

est_mean	est_err
1.253841	0.004555

Control Variates in R

Now consider the control variate problem. First, consider returning both the Asian option price and the control variate value.

```
raopt_cv <-  function(t, u, d, p_x, p_y, s_0, K) {
  y     <- (runif(t) < p_y)
  n_up  <- sum(y)
  ratio <- (p_x / p_y)^(n_up) *
```

```
              ((1 - p_x) / (1 - p_y))^(t - n_up)
  s      <- s_0 * cumprod(u^y) * cumprod(d^(1 - y))
  return(c(max(mean(s) - K, 0) * ratio, last(s)))
}
```

Use 10000 samples to estimate the covariance between the two, and the variance of the control variate.

```
res_cv <- replicate(10000, raopt_cv(5, 1.1, 0.95, 1 / 3,
                                    1 / 3, 100, 110))
cov(res_cv[1, ], res_cv[2, ])
```

```
## [1] 42.37929
```

```
var(res_cv[2, ])
```

```
## [1] 264.5681
```

So the value of c to use is $-42.37/264.56$.

```
c <- -cov(res_cv[1, ], res_cv[2, ]) / var(res_cv[2, ])
```

```
raopt_cv2 <- function(t, u, d, p_x, p_y, s_0, K, c = 0) {
  y      <- (runif(t) < p_y)
  n_up   <- sum(y)
  ratio <- (p_x / p_y)^(n_up) *
             ((1 - p_x) / (1 - p_y))^(t - n_up)
  s      <- s_0 * cumprod(u^y) * cumprod(d^(1 - y))
  return(max(mean(s) - K, 0) * ratio + c * (last(s) - s_0))
}
```

Note that once c has been estimated, the data to price the derivative must be replicated independently! If the same data is used, then c and the estimated value of the control variate are *not* independent, and all our calculations fall apart.

```
trials <- 100000
res_cv1 <- replicate(trials,
                 raopt_cv2(5, 1.1, 0.95, 1 / 3, 1 / 3,
                           s_0 = 100, K = 110, c))
res_cv2 <- replicate(trials,
                 raopt_cv2(5, 1.1, 0.95, 1 / 3, 1 / 3,
                           s_0 = 100, K = 110, 0))
```

```
tibble(
  est_mean    = mean(res_cv2),
  est_err     = sd(res_cv2) / sqrt(length(res_cv2)),
  est_mean_cv = mean(res_cv1),
  est_err_cv  = sd(res_cv1) / sqrt(length(res_cv1))
) |> kable()
```

est_mean	est_err	est_mean_cv	est_err_cv
1.257183	0.0115098	1.254005	0.0086079

Combining

A natural question to ask is: can the importance sampling and control variate methods be combined?

```
res_cv_is <-
  replicate(10000,
            raopt_cv(5, 1.1, 0.95, 1 / 3, 0.7, 100, 110))
c_is <- -cov(res_cv_is[1, ], res_cv_is[2, ]) /
             var(res_cv_is[2, ])
```

```
trials <- 100000
res_cv3 <- replicate(trials,
                     raopt_cv2(5, 1.1, 0.95, 1 / 3, 0.7, s_0 = 100,
tibble(
  est_mean_cv_is = mean(res_cv3),
  est_err_cv_is  = sd(res_cv3) / sqrt(length(res_cv3))
) |> kable()
```

est_mean_cv_is	est_err_cv_is
1.274732	0.0045371

This is pretty much the benefit that came from importance sampling alone. The reason for this is that the importance sampling made the dependence on the final stock price much smaller since higher stock values were given smaller weights in the ratio. So really importance sampling alone gives the best bang for the buck here.

Problems

21.1: Consider the problem of estimating

$$I = \int_{-1}^{1} \frac{1}{1+x^2}\,dx.$$

a) Note that $I = \mathbb{E}((\tau/2)\mathbb{I}(|T| \le 1))$, where T is a standard Cauchy random variable. Find $\mathbb{V}(\mathbb{I}(|T| \le 1))$ in terms of I.

b) Suppose $U \sim \text{Unif}([-1,1])$. Find a function $\ell(U)$ such that the mean of $\ell(U)$ equals I.

c) What is the variance of $\ell(U)$?

d) Assuming it is equally easy to generate a Cauchy or a Uniform, which would you use to estimate I with Monte Carlo?

Solution

a) Since $(\tau/2)\mathbb{I}(|T| \le 1) \in \{0,1\}$, the variance is just the product of $(\tau/2)^2$ times the probability the expression inside the indicator is true times the probability it is not true. (For $B \sim \text{Bern}(p)$, $\mathbb{V}(B) = p(1-p)$.) So this variance is $I(1-I)$, or about $\boxed{(\tau/4)^2(1/2)(1-1/2) \approx 2.467}$

b) Using the importance sampling theorem,

$$\mathbb{I}(U \in [-1,1])\frac{1/(1+U^2)}{1/2} = \boxed{\frac{2}{1+U^2}}$$

is the function.

c) To find the variance, the second moment is needed:

$$\begin{aligned}\mathbb{E}[\ell(U)^2] &= \int_{\mathbb{R}} \left(\frac{2}{1+u^2}\right)^2 \frac{1}{2}\mathbb{I}(u \in [-1,1])\, du \\ &= 1 + \frac{\tau}{4} = \boxed{2.570\ldots}.\end{aligned}$$

d) Use the $\boxed{\text{Cauchy}}$ approach, since the variance is slightly smaller.

21.2: Consider the problem of estimating $\mathbb{E}(U^2)$.

a) What is the variance of U^2?

b) Suppose you can generate $B \sim \text{Beta}(1,2)$. Find a function of B such that the mean is $\mathbb{E}(U^2)$.

c) Find the variance of your function of B.

d) Assuming it is equally easy to generate a uniform or a beta random variable, which should you use to estimate $\mathbb{E}(U^2)$?

Chapter 22

Brownian motion

Question of the day

A stock price is modeled using Geometric Brownian motion with $S_0, 100, \mu = 0.05, \sigma = 0.1$). If an Asian option pays

$$((S_1 + S_2 + S_3)/3 - 110)^+$$

at time 3, what should the price of the option be at time 0?

Summary

- **Standard Brownian motion** is a stochastic process that is 0 at time 0, has independent normal increments, and is continuous with probability 1.
- **Geometric Brownian motion** is a function of a standard Brownian motion $\{W_t\}$ that is
$$S_t = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right)$$
given parameters (S_0, μ, σ).

The binomial model is a discrete time model: at each time step the stock moves from value S_{t-1} to either $S_t = S_{t-1}u$ or $S_t = S_{t-1}d$. Suppose instead a continuous time model is needed.

In this situation, the common way to model a stock price is as a function of *Brownian motion*. Brownian motion can be viewed as the limit of simple symmetric random walk on the integers. It is defined by four properties.

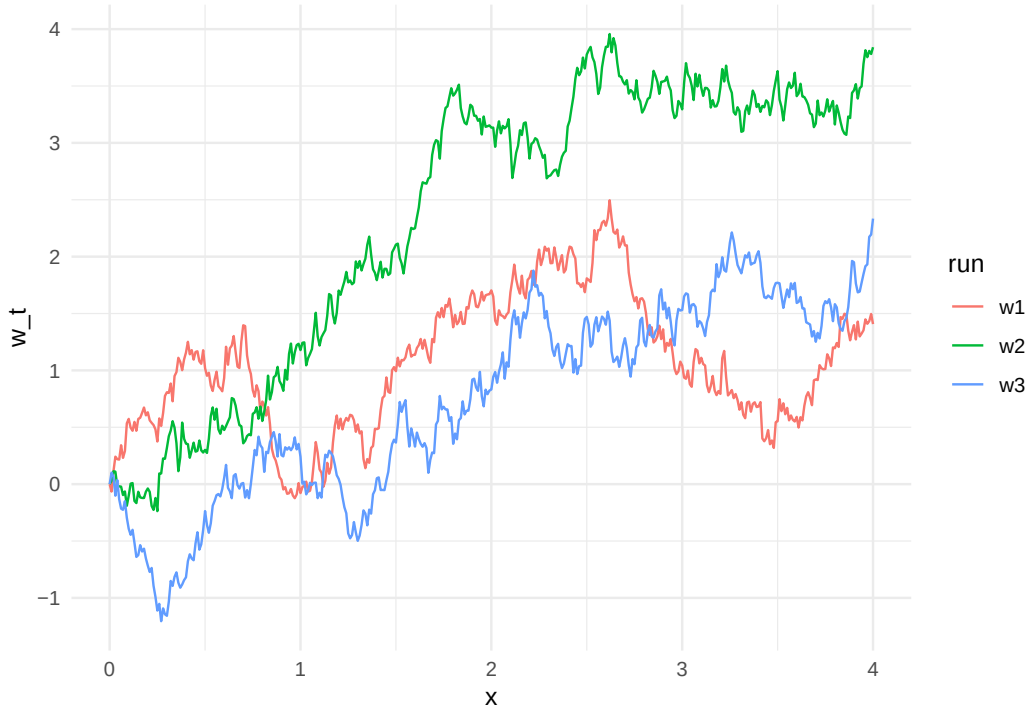
Definition 53
The stochastic process $\{W_t\}_{t\geq 0}$ is a **Weiner Process** or **standard Brownian Motion** if it has the following four properties.

1) **Standardized** $W_0 = 0$.

2) **Independent increments** For $0 < s_1 < t_1 < s_2 < t_2$,, $W_{t_1} - W_{s_1}$ is independent of $W_{t_2} - W_{s_2}$.

3) **Normal increments** For $s_1 < t_1$, $W_{t_1} - W_{s_1} \sim \mathsf{N}(0, t_1 - s_1)$.

4) **Continuous** W_t is continuous with probability 1.



The easiest method of simulating Brownian motion at a fixed set of times is *forward simulation.*

Suppose you wish to simulate at times $t_1 < t_2 < \ldots, t_k$. Start with W_0 equal 0. Order your target times from smallest to largest. For your smallest time t_1, $W_{t_1} - W_0 \sim \mathsf{N}(0, t_1)$. So set $W_{t_1} = W_0 + Z_1\sqrt{t_1}$ where Z_1 is a standard normal. Then set $W_{t_2} = W_{t_1} + Z_2\sqrt{t_2 - t_1}$, and so on.

22.1 Geometric Brownian Motion

The mathematical model of Brownian motion was originally proposed by Thiele in 1880 and independently by Bachelier in 1900. In these early models, Brownian motion was directly taken as a model of the stock price.

Unfortunately, this missed an important idea: the small motions of stocks tend to be *proportional* to the price, not additive. This can be handled with *Geometric Brownian Motion.*

Definition 54
Let $\{W_t\}$ be standard Brownian motion. For constants S_0, μ, and σ, the process

$$S_t = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right)$$

is **geometric Brownian motion** (GBM) with initial value S_0, growth rate μ, and variance (aka volatility) σ^2.

Let h be a small positive number close to 0, and then

$$\begin{aligned}\frac{S_{t+h}}{S_t} &= \frac{S_0 \exp((\mu - \sigma^2/2)(t+h) + \sigma W_{t+h})}{S_0 \exp((\mu - \sigma^2/2)(t) + \sigma W_t)} \\ &= \exp(h\mu - h\sigma^2/2)\exp(\sigma(W_{t+h} - W_t)).\end{aligned}$$

Then

$$\mathbb{E}\left[\frac{S_{t+h}}{S_t}\right] = \exp(h\mu - h\sigma^2/2)\mathbb{E}[\exp(\sigma(W_{t+h} - W_t))].$$

Since for $N \sim \mathsf{N}(0, v)$, $\mathbb{E}(\exp(\sigma N)) = \exp(\sigma^2 v/2)$,,

$$\mathbb{E}\left[\frac{S_{t+h}}{S_t}\right] = \exp(h\mu - h\sigma^2/2)\exp(h\sigma^2/2) = \exp(h\mu) \approx 1 + \mu h.$$

So μ represents the average growth rate of the stock over a small time interval from t to $t + h$.

Risk neutral equivalent martingale measure

When $\mu = 0$,

$$\mathbb{E}\left[\frac{S_{t+h}}{S_t}\right] = 1,$$

and the stock price forms a martingale. This is equivalent to the *risk neutral equivalent martingale measure* for Geometric Brownian motion, and can then be used to price derivatives of stocks whose price follows a geometric Brownian motion.

22.2 Solving the Question of the Day in R

Consider simulating Brownian motion at time $(1.2, 2.1, 2.4)$. The `diff` command can be used (after adding a time at 0) to get the lengths of the intervals between times.

```
t <- c(1.2, 2.1, 2.4)
diff(c(0, t))
```

```
## [1] 1.2 0.9 0.3
```

Taking the square root and multiplying by standard normals gives normals whose variances equal the width of the interval.

```
dw <- rnorm(3) * sqrt(diff(c(0, t)))
dw
```

```
## [1] -0.7606729  0.9069491 -0.9961706
```

Finally use the `cumsum` command to add up the first 1, the first 2, and the first 3 of this vector. That will be the Brownian motion.

```
cumsum(dw)
```

```
## [1] -0.7606729  0.1462762 -0.8498944
```

Next consider simulating geometric Brownian motion for a given set of times, (S_0, μ, σ) values. Begin by generating the Brownian motion using `cumsum` and forward simulation. Remember to multiply a normal by the square root of the length of the interval in order to get a difference that is normal with variance equal to the length of the interval.

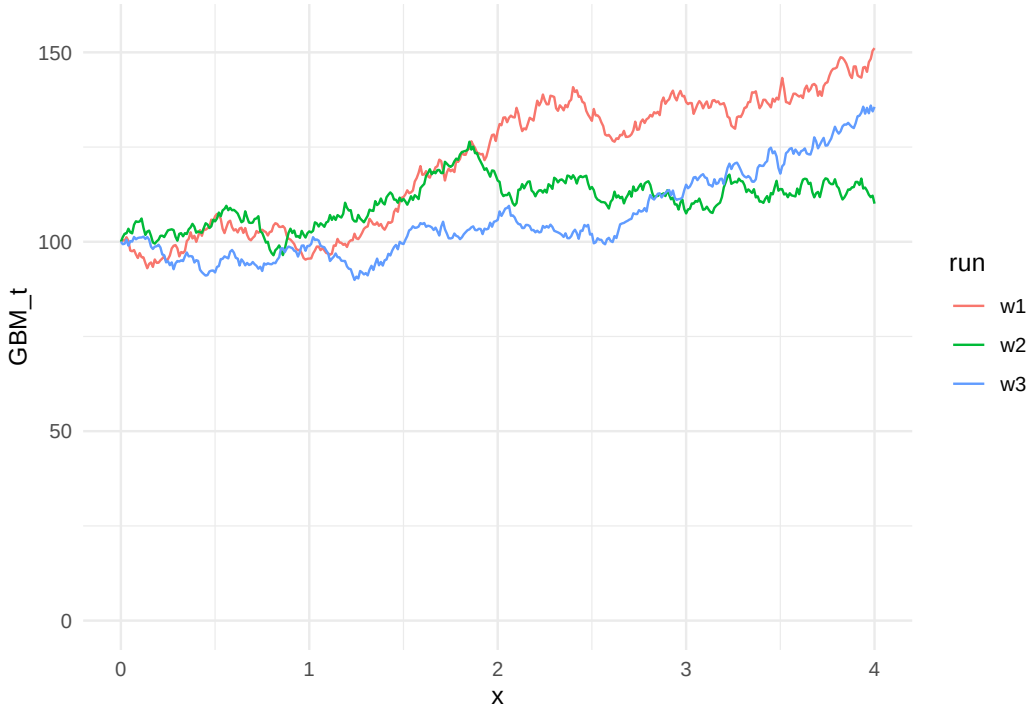
```
rgbm <- function(times, s_0, mu, sigma) {
  w <- cumsum( sqrt( diff( c(0, times ) ) ) *
               rnorm( length( times ) ) )
  return(s_0 * exp((mu - sigma^2 / 2) * times + sigma * w))
}
```

For example, to get the stock prices at times 1, 2, and 3 for a GBM stock starting at value \$100 with $\mu = 0.05$ and $\sigma = 1$:

```
rgbm(c(1, 2, 3), 100, 0.05, 0.1)
```

```
## [1] 97.86084 93.43556 95.73693
```

Here is a more detailed plot of GBM with these parameters.



The Fundamental Theorem of Asset Pricing says that under the risk neutral martingale equivalent measure, the price of any asset is a martingale. Using $\mu^* = 0$ for Geometric Brownian motion gives the values of the stock price used in finding the Asian option.

```
n <- 10^5
mu_star = 0
res <-
  replicate(n, max(mean(rgbm(c(1, 2, 3), 100, mu_star, 0.1) -
                    110), 0))
tibble(
  est_mean = mean(res),
  est_err  = sd(res) / sqrt(length(res))
) |> kable()
```

est_mean	est_err
1.665026	0.0146559

Adding a control variate

A control variate can be used to lower the variance of the result. A simple control variate is S_3.

```
asian_option_cv <- function(times, s_0, mu, sigma, K) {
  s <- rgbm(times, s_0, mu, sigma)
  return(c(max(mean(s) - K, 0), last(s)))
}
```

Get the data

```
n <- 10^5
res_cv <- replicate(n, asian_option_cv(c(1, 2, 3), 100, 0,
                                       0.1, 110))
```

Now see how correlated the Asian option is with the final stock price.

```
cor(res_cv[1, ], res_cv[2, ])
```

```
## [1] 0.6706318
```

Note that this control variate is highly correlated with the Asian option value. So it can be used to reduce the variance of our estimate.

```
c <- -cov(res_cv[1, ], res_cv[2, ]) / var(res_cv[2, ])
```

```
tibble(
  est_mean     = mean(res_cv[1, ]),
  est_err      = sd(res_cv[1, ]) / sqrt(length(res_cv[1, ])),
  est_mean_cv  = mean(res_cv[1, ] + c * (res_cv[2, ] - 100)),
  est_err_cv   = sd(res_cv[1, ] + c * (res_cv[2, ] - 100)) /
                 sqrt(length(res_cv[2, ]))
) |> kable()
```

est_mean	est_err	est_mean_cv	est_err_cv
1.697123	0.0150075	1.708456	0.0111324

Note that there was a substantial reduction in error.

An even better control variate is $(S_3 - 110)^+$, this is the value of a *European call option* at time 3 with strike price 110.

```
cor(res_cv[1, ], pmax(res_cv[2, ] - 110, 0))
```

```
## [1] 0.8752982
```

Finding the mean of this is a bit more tricky. If Z is a standard normal, then

$$(S_3 - 110)^+ \sim (100\exp(-(0.1^2/2)(3) + (0.1)(\sqrt{3})Z) - 110)^+,$$

so the mean can be estimated as follows.

```
  est_cv <- pmax((100 * exp(-(0.1^2 / 2) * 3 + (0.1) * sqrt(3)
                            rnorm(10^7)) - 110), 0)
tibble(
  est_mean = mean(est_cv),
  est_err  = sd(est_cv) / sqrt(length(est_cv))
) |> kable()
```

est_mean	est_err
3.316411	0.0025239

```
cv2 <- pmax(res_cv[2, ] - 110, 0)
c2 <- -cov(cv2, res_cv[1, ]) / var(cv2)
```

The resulting estimate is

```
n <- length(res_cv[1, ])
tibble(
  est_mean     = mean(res_cv[1, ]),
  est_err      = sd(res_cv[1, ]) / sqrt(n),
  est_mean_cv  = mean(res_cv[1, ] + c * (res_cv[2, ] - 100)),
  est_err_cv   = sd(res_cv[1, ] + c * (res_cv[2, ] - 100)) /
                   sqrt(n),
  est_mean_cv2 = mean(res_cv[1, ] + c2 * (cv2 - 3.316256)),
  est_err_cv2  = sd(res_cv[1, ] + c2 * (cv2 - 3.316256)) /
                   sqrt(n),
) |> kable()
```

est_mean	est_err	est_mean_cv	est_err_cv	est_mean_cv2	est_err_cv2
1.697123	0.0150075	1.708456	0.0111324	1.702204	0.0072574

As the correlation is higher between this new control variate and the Asian option value than the old control variate, the variance is even smaller.

Chapter 23

Simulating solutions to Stochastic Differential Equations

Question of the day

What is the best way to simulate a draw from the solution to a Stochastic Differential Equation?

Summary

- A **Stochastic Differential Equation** involves differentials of time and standard Brownian Motion.
- The **Euler-Maruyama method** can be used to sample from an SDE that has the form $dX_t = a(t, X_t)\, dt + b(t, X_t)\, dW_t$, by using

  $$\hat{X}_{t+h} = \hat{X}_t + h \cdot a(t, X_t) + \sqrt{h} Z_t$$

  where Z_t is a standard normal random variable.
- More sophisticated methods exist that converge faster to the true solution, or in some cases, draw exactly from the SDE.

First consider the notion of a *differential*. A differential represents a small change in a variable (or stochastic process.) So dy represents a small change in y, dt represents a small change in t, and dW_t represents a small change in the value of a standard Brownian motion $\{W_t\}$.

A *differential equation* (DE) is an equation that involves one or more differentials. For instance, suppose that

$$dy = t\, dt.$$

This equation says that the variables y and t are connected in some way, that a small change in the value of t will lead to a change in the value of y that is t times the small change in the value of t.

To put some numbers on it, if $t = 4$ and $dt = 0.0001$, then $dy \approx 0.0004$. This is not exact, because 0.0001 is not infinitely small, but it is a good first approximation.

Euler suggested and analyzed the following simple numerical method for approximation the solutions to a given DE. The input to the function is a value h that is being used as a stand in for dt. Today this method bears his name.

Definition 55
Consider the differential equation $dy = f(t, y)\ dt$ with boundary condition $y(0) = y_0$. Let $h > 0$ and n be a positive integer. Define

$$\hat{y}_0 = x_0$$

and for every $i \in (1, 2, \ldots, n)$ in order,

$$\hat{y}_{ih} = \hat{y}_{(i-1)h} + f((i-1)h, \hat{y}_{(i-1)h}) \cdot h.$$

Call the $\{\hat{y}_t\}$ the **Euler's method** approximation of the value of y.

The time here starts at o for convenience, but by shifting this method it is possible to start at an arbitrary time t_0.

23.1 The Euler-Maruyama method

A *stochastic differential equation*, or SDE, involves not only deterministic changes in a variable, but random changes as well. Suppose that $\{W_t\}$ is standard Brownian motion (also known as a Weiner process after Norbert Weiner). Roughly speaking, the change in $\{W_t\}$ over some small time interval is

$$dW_t = W_{t+dt} - W_t \sim \mathsf{N}(0, dt).$$

So the variance of the differential change is dt, and the mean of the change is o. It turns out that because differential change in time is small, *any* distribution with mean o and variance dt can be used. For instance,

$$dW_t = W_{t+dt} - W_t \sim \mathsf{Unif}(\{-\sqrt{dt}, \sqrt{dt}\}).$$

works. However, in the description here the normal distribution will be used because it converges exactly without the need to invoke the Central Limit Theorem.

An SDE is similar to a DE, exempt it is allowed to include these stochastic differentials dW_t. If a stock price was growing exponentially and deterministically, then it obeys the DE,

$$dS_t = S_t \mu\ dt.$$

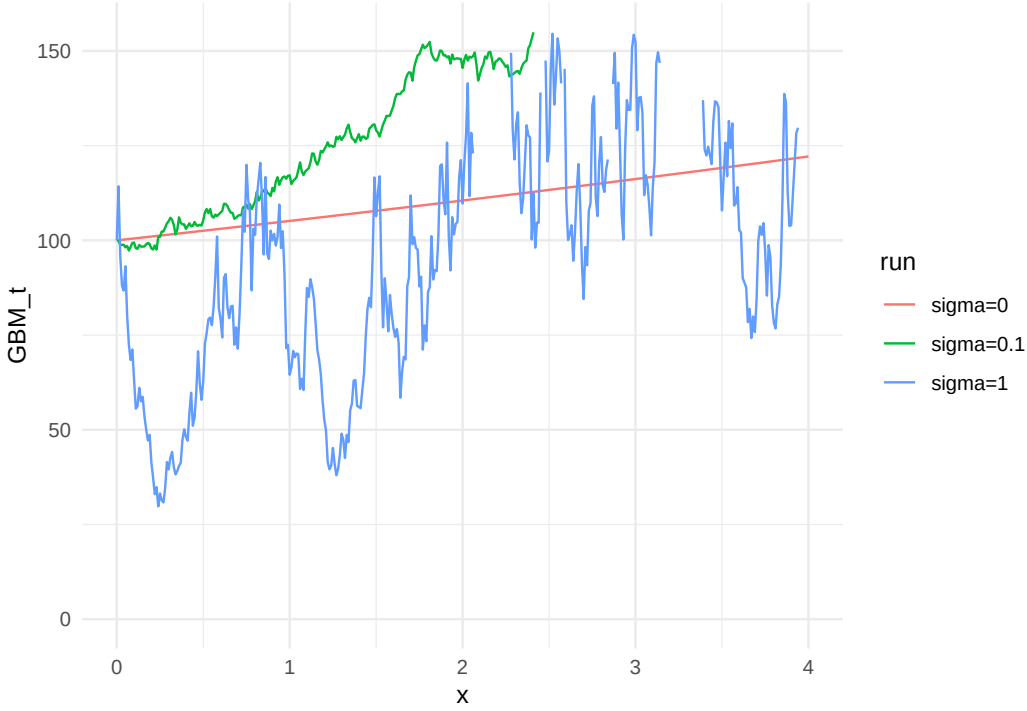
In this equation μ is called the *rate of growth*. If $S_t > 0$ and $\mu > 0$, then $dS_t > 0$ so the stock price is growing. If $S_t > 0$ and $\mu < 0$ then $dS_t < 0$ so the stock price is decreasing.

In an SDE, the differential of W_t is thrown in to spice things up and add a bit of randomness to the expression. In particular, the SDE for geometric Brownian motion is

$$dS_t = S_t(\mu\, dt + \sigma\, dW_t).$$

The value of the constant σ controls how much the randomness affects the equation. When $\sigma = 0$, it returns to the usual exponential growth or decline. But as σ increases, the shocks introduced by the W_t can overwhelm the output.

This makes the solution to the SDE itself a random variable! It is a special type of random variable which is a random function of time.



Note that for $\sigma = 1$ even though on average the stock price is growing, it is also converging to o! That's because if the log of the stock price looks like:

$$\ln(S_t) = \ln(S_0) + (\mu - \sigma^2/2)t + \sigma W_t.$$

So if $\sigma^2/2 > \mu$, then on average the log of the stock price is going towards negative infinity, which means the stock price is going to zero.

Geometric Brownian motion is a special case of a *diffusion*, and SDE with a dt and a W_t term.

Definition 56
If the stochastic process $\{X_t\}$ satisfies

$$dX_t = a(t, X_t)\, dt + b(t, X_t)\, dW_t,$$

it is a **diffusion** with **drift** $a(x, t)$ and **volatility** $b(x, t)$.

23.2 Euler-Maruyama

A handful of SDE examples such as the one for geometric Brownian motion can be solved exactly, but most cannot. For these, it is still possible to conduct simulations. An approximate simulation method utilizes an approach similar to the Euler method. It was introduced by Maruyama in 1955 (Maruyama 1955).

Definition 57
Consider the diffusion

$$dS_t = a(t, S_t)\, dt + b(t, S_t)\, dW_t,$$

with known start value S_0. Then the **Euler-Maruyama method** proceeds as follows. Let $h > 0$ and n be a positive integer.

Set $\hat{S}_0 = S_0$. For every i from 1 to n in order,

$$\hat{S}_{ih} = \hat{S}_{(i-1)h} + a\left((i-1)h, \hat{S}_{(i-1)h}\right) \cdot h + b\left((i-1)h, \hat{S}_{(i-1)h}\right) \cdot Z_i \cdot \sqrt{h},$$

where $\{Z_1, \ldots, Z_n\}$ are iid standard normal random variables.

Some notes.

- The Euler-Maruyama algorithm is a Monte Carlo algorithm, since the output depends on the draws of the Z_i.
- Running the algorithm multiple times can give an idea of the range of functions produced.

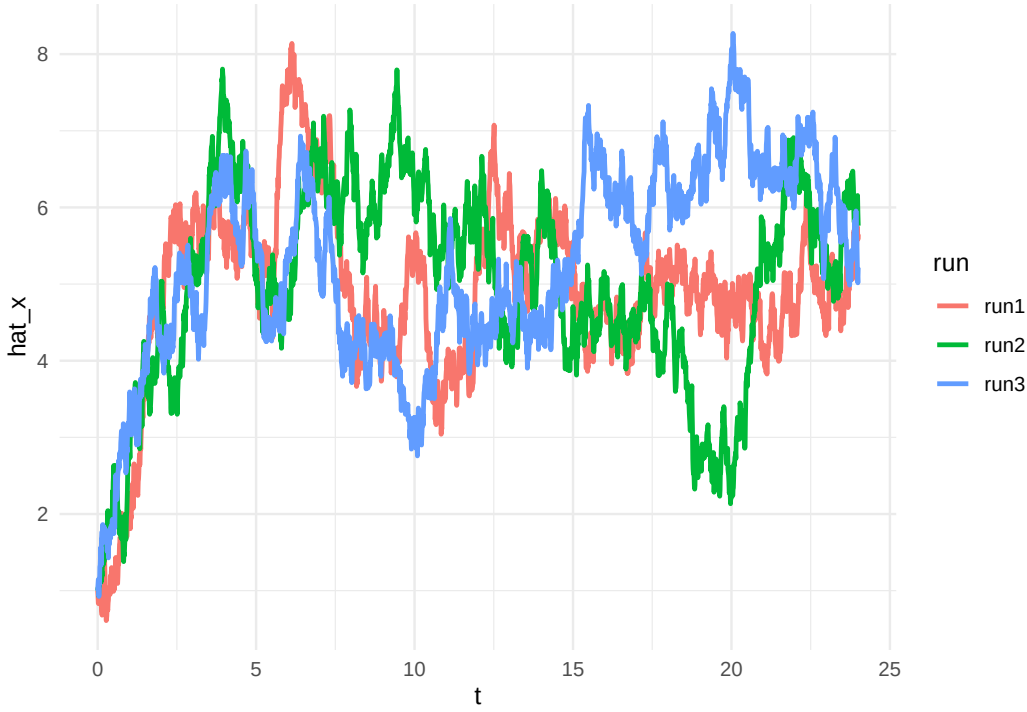
To this last point, consider a particular diffusion called the *Ornstein-Uhlenbeck process*, used to model interest rates.

A diffusion is an **Ornstein-Uhlenbeck process** if it has the form

$$dX_t = -\theta \cdot (X_t - \mu)\, dt + \sigma\, dW_t$$

for μ a constant and θ, σ positive constants.

Then three runs of Euler-Maruyama with $\theta = 0.5, \mu = 5, \sigma = 1.3$ is as follows.



SDE perfect simulation

There does exist a form of acceptance rejection for certain classes of diffusions. See (Beskos et al, 2006) for details.

Milstein's method

The Euler-Maruyama is considered a *half-order* method, because the convergence to the true answer goes as $\sqrt{h}$. There are methods that converge faster. For instance, *Milstein's method* converges as h, making it a *first-order method.*

23.3 Running Euler-Maruyama in R

To run an Euler-Maruyama algorithm in R, the first thing to do is to build an `a` and a `b` function for the drift and volatility functions respectively. For instance, for the Ornstein-Uhlenbeck process from earlier with $\theta = 0.5, \mu = 5, \sigma = 1.3$:

```
a <- function(t, s) return(-0.5 * ( s - 5 ))
b <- function(t, s) return(1.3)
```

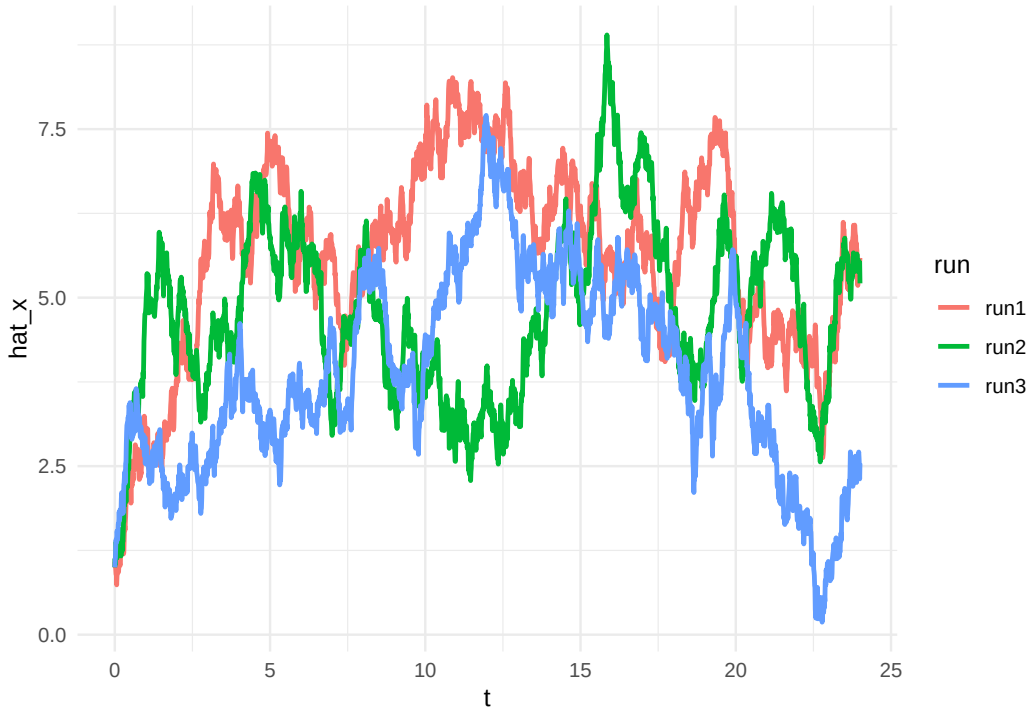
Now the Euler-Maruyama method can accept these functions as inputs.

```
euler_maruyama <- function(x_0, t_0, t_1, h, a, b) {
  t <- seq(t_0, t_1, by = h)
```

```
  n <- length(t)
  hat_x <- rep(x_0, n)
  z <- rnorm(n - 1)
  sqrt_h <- sqrt(h)
  for (i in 2:n) {
    hat_x[i] <- hat_x[i - 1] +
      a(t[i - 1], hat_x[i - 1]) * h +
      b(t[i - 1], hat_x[i - 1]) * z[i - 1] * sqrt_h
  }
  return(tibble(t, hat_x))
}
```

Now complete three simulations of the result for time from 0 to 24.

```
run1 <- euler_maruyama(1, 0, 24, 0.01, a, b) |> mutate(run = "r
run2 <- euler_maruyama(1, 0, 24, 0.01, a, b) |> mutate(run = "r
run3 <- euler_maruyama(1, 0, 24, 0.01, a, b) |> mutate(run = "r
r <- union_all(run1, run2) |> union_all(run3)
r |>
  ggplot() +
    geom_line(aes(t, hat_x, color = run), lwd = 1) +
    theme_minimal()
```

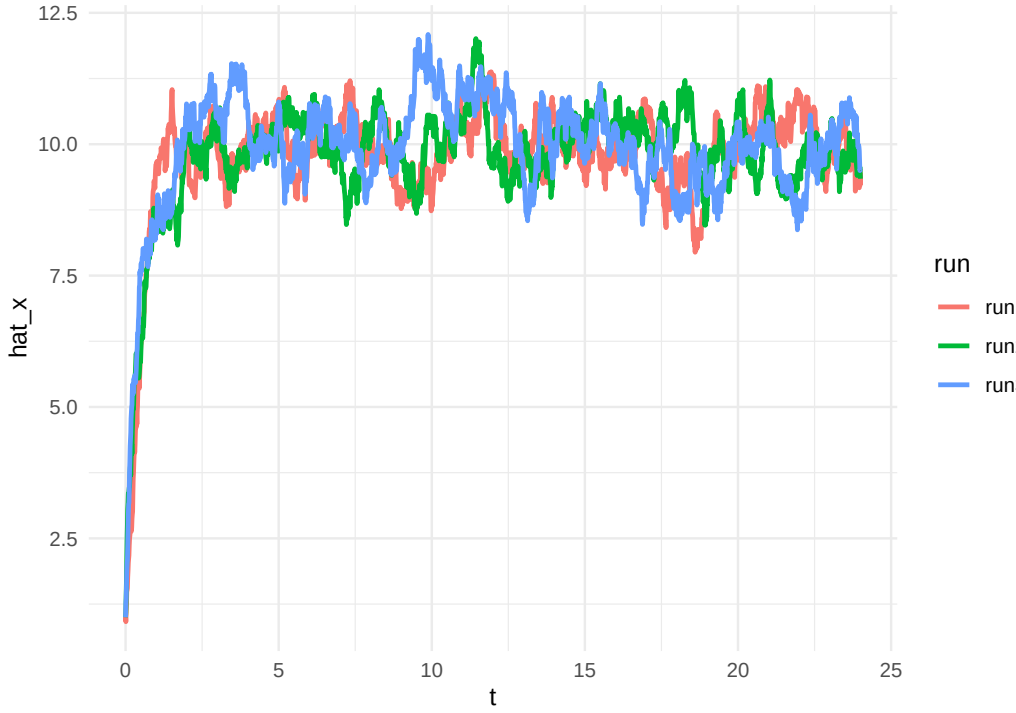


Note that they tend to move towards 5 as quickly as possible, since otherwise there is pressure from the `a` function to move towards 5. Consider giving it a mean of 10, and strengthening the move towards 10:

```
a2 <- function(t, s) return(-2 * ( s - 10 ))
```

Then the plots look something like:

```
run1_2 <- euler_maruyama(1, 0, 24, 0.01, a2, b) |> mutate(run =
run2_2 <- euler_maruyama(1, 0, 24, 0.01, a2, b) |> mutate(run =
run3_2 <- euler_maruyama(1, 0, 24, 0.01, a2, b) |> mutate(run =
r2 <- union_all(run1_2, run2_2) |> union_all(run3_2)
r2 |>
  ggplot() +
    geom_line(aes(t, hat_x, color = run), lwd = 1) +
    theme_minimal()
```



23.4 Questions

Consider using Euler-Maruyama method to estimate Geometric Brownian Motion with $a(t, s) = (\mu - \sigma^2/2))$ and $b(t, s) = \sigma$, with $S_0 = 100$, $\mu = 0.05$ and $\sigma = 0.1$.

a. Estimate S_t using Euler-Maruyama with $h = 0.1$ over $[0, 4]$. Plot the result.

b. Estimate S_t using Euler-Maruyama with $h = 0.01$ over $[0, 4]$. Plot the result.

First set up the functions

```
a2 <- function(t, s) {
  return(0.045)
}

b2 <- function(t, r) {
  return(0.1)
}
```

Next set up Euler-Maruyama:

```
euler_maruyama <- function(x_0, t_0, t_1, h, a, b) {
  t <- seq(t_0, t_1, by = h)
  n <- length(t)
  hat_x <- rep(x_0, n)
  z <- rnorm(n - 1)
  sqrt_h <- sqrt(h)
  for (i in 2:n) {
    hat_x[i] <- hat_x[i - 1] + a(t[i - 1], hat_x[i - 1]) * h +
      b(t[i - 1], hat_x[i - 1]) * z[i - 1] * sqrt_h
  }
  return(tibble(t, hat_x))
}
```

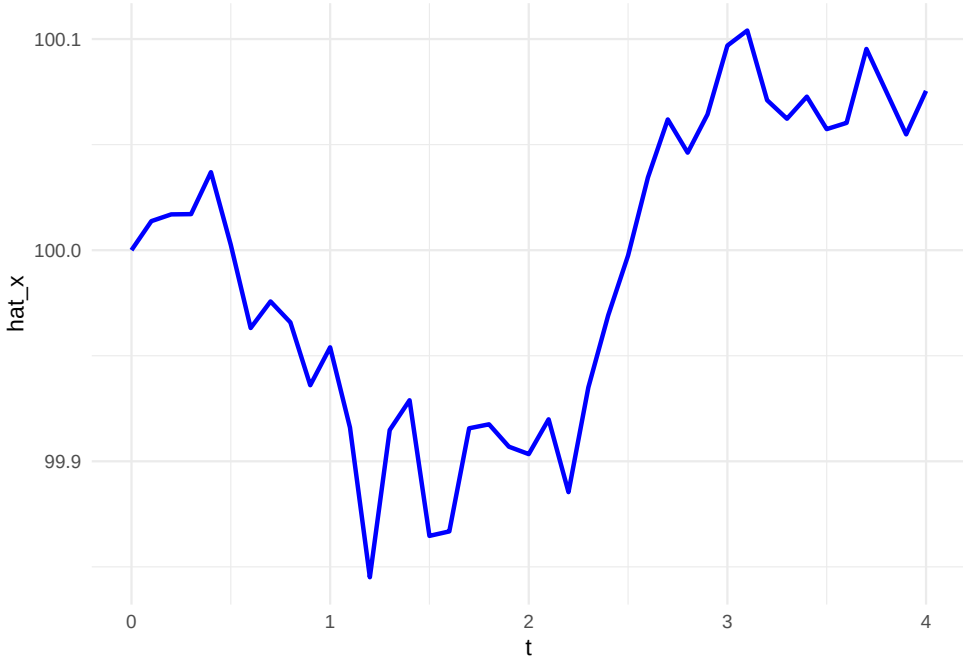
Now the runs can be plotted.

a. Make our run:

```
run <- euler_maruyama(100, 0, 4, 0.1, a2, b2)
```

Graph the result:

```
run |>
  ggplot() +
    geom_line(aes(t, hat_x), lwd = 1, color = "blue") +
    theme_minimal()
```

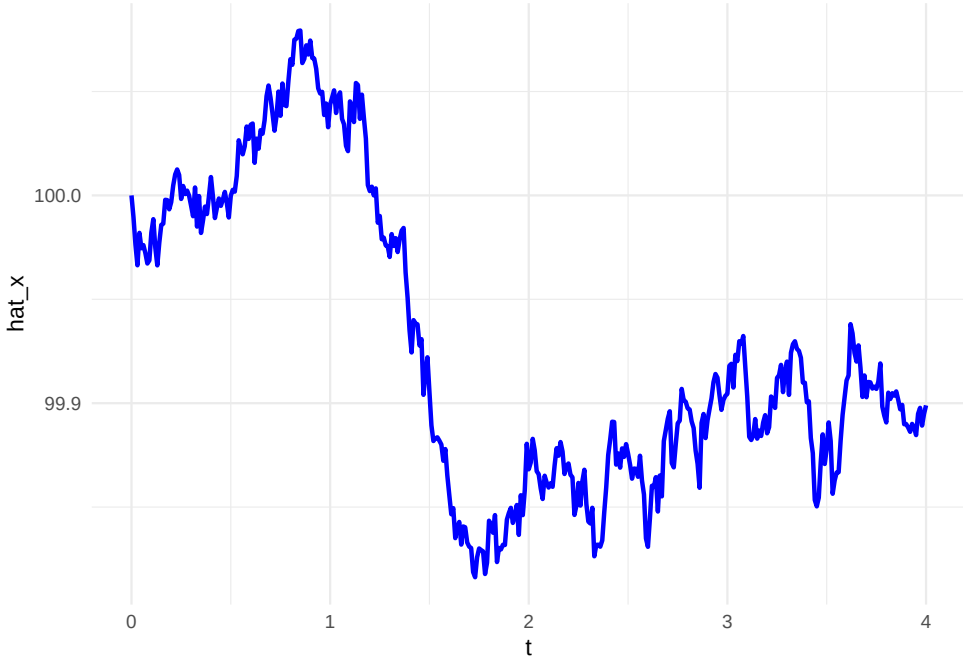


b. Make our run:

```
run_01 <- euler_maruyama(100, 0, 4, 0.01, a2, b2)
```

Graph the result:

```
run_01 |>
  ggplot() +
    geom_line(aes(t, hat_x), lwd = 1, color = "blue") +
    theme_minimal()
```



The *Vasicek model* of interest rates $\{r_t\}$ is a stochastic differential equation:

$$dr_t = a(b - r_t)\,dt + \sigma\,dW_t$$

For $r_0 = 0.4$, $a = 0.1$, $b = 0.05$, and $\sigma = 0.05$, simulate the Vasicek model three times using the Euler-Maruyama method over times $[0, 3]$ and plot the results on the same graph.

References

Beskos, Alexandros, Omiros Papaspiliopoulos, Gareth O Roberts, and Paul Fearnhead. 2006. "Exact and Computationally Efficient Likelihood-Based Estimation for Discretely Observed Diffusion Processes (with Discussion)." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68 (3): 333–82.

Maruyama, Gisiro. 1955. "Continuous Markov Processes and Stochastic Equations." *Rendiconti Del Circolo Matematico Di Palermo* 4 (1): 48.

Chapter 24

Multidimensional Brownian Motion

Question of the Day

Suppose that there are two stocks that have value S_1 and W_1 after one year. Both stocks start with price \$100 at time 0. They follow geometric Brownian motion with means 0.07 and 0.08, and volatilities 0.1 and 0.15 respectively. Moreover, the first stock pays a continuous dividend of 0.03 and 0.04. The instantaneous correlation of the stocks is 50%. After one year, a *European spread call* with strike price \$1 pays

$$(S_1 - W_1 - 1)^+.$$

The risk free interest rate is 0.02.

How much should this call option be worth at time 0?

Summary

- **Continuous dividends** reduce the share price by $-dS_t\ dt$ in GBM.
- The **risk free rate of return** r is the amount of return given by investments that are 100% safe. It appears in the risk neutral equivalent martingale measure as an $r\ dt$ term.
- If Z_1, Z_2 are iid standard normals, then $(Z_1, \rho Z_1 + \operatorname{sqrt}(1-\rho^2)Z_2)$ has marginals that are standard normals and correlation ρ.
- If Z_1, Z_2 are iid standard normals and $B\mathsf{Bern}(\rho)$, then $(Z_1, BZ_1 + (1-B)Z_2)$ have marginals that are standard normals and correlation ρ.
- If $Z^T = (Z_1, \ldots, Z_n)$ are iid standard normals then for an $n \times n$ matrix A, AZ has marginals that are normally distributed, and correlation matrix $\Sigma = AA^T$.
- Given a positive definite matrix Σ, finding A so that $AA^T = \Sigma$ can be accomplished with the **Cholesky decomposition** algorithm.

So far, the derivatives under consideration have has prices that depended on only one asset. In practice, there are derivatives whose price depends on two or more assets. An example is a *spread call*. When this call is exercised, it returns the difference in value between a pair of stocks minus the strike price. Of course, the bearer would only exercise the option if this was a positive amount. That makes the value of the call at its exercise time t would be

$$\max(S_t - W_t - K, 0) = (S_t - P_t - K)^+$$

if the stock prices at time t were S_t and P_t, with strike price K.

The Question of the Day includes stock prices that are correlated, as well as paying continuous dividends and now there is a risk free rate for those who are investing their money in 100% risk free assets. Each of these effects must be incorporated into our model.

24.1 GBM with Continuous dividends

In the Question of the Day, the stocks are following continuous Brownian motion, but they are also converting part of their value into money. When a company pays out part of the company value to owners of a share of stock, this is called a *dividend*. Payment of dividends to the bearer of a share of stock reduces the share value.

For instance, if a company had total worth M, and paid out 3% of its worth as dividends to shareholders, the company would be worth $(1 - 0.03)M$ after that. Assuming a simple situation where all shares of stock were equal, the price of the shares would fall from S_t to $(1 - 0.03)M$.

To keep things easy, the dividends in the question of the day are being paid out continuously. Therefore, the GBM for S_t is

$$dS_t = S_t \left[(\mu - d)\, dt + \sigma\, dW_t\right],$$

where d is the rate of the continuous dividends.

The risk free neutral equivalent martingale measure ignores the average grown in the stock μ, but not the continuous dividends.

$$dS_t^* = S_t[-d\, dt + \sigma\, dW_t],$$

24.2 Risk free measure

When money is placed in 100% safe investments, it will still continue to grow, typically because the investments are immune to inflation, and having the money allows a bank to make investments of its own. The rate of return on money placed in risk free investments is called the *risk free rate of return*, and is usually denoted r. Investments that are risk free are referred to as the *money market*.

Money in a risk free investment grows as follows:

$$dM_t = rM_t\, dt.$$

If r is positive, then the money grows exponentially. If r is negative then run for the hills. When $r < 0$ you are in a deflationary economy where saving costs you money over time.

In the version of GBM with risk free return, the risk free rate of return matches that of the money market. That is,

$$dS_t^* = S_t \left[(r - d)\, dt + \sigma\, dW_t \right],$$

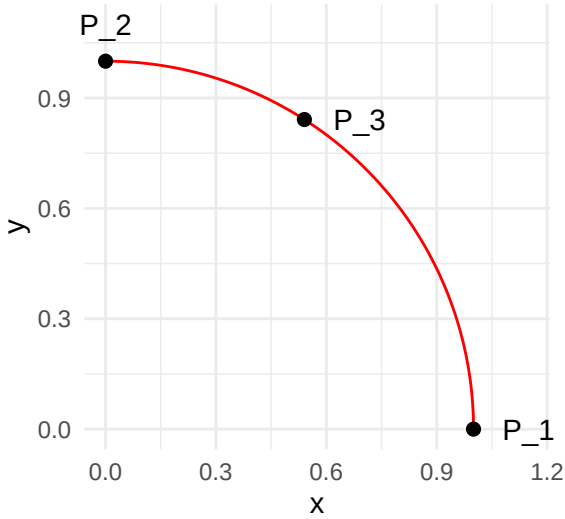
24.3 Correlated random variables

Since $W_t \sim \mathsf{N}(0, t)$, simulation of W_t for one of the stocks in the Question of the Day is easy. However, this stock has *instantaneous correlation* with another stock. When dealing with only two normal random variables, making them correlated is straightforward.

To make two vectors in $\mathbb{R}^2$ with angle θ between them, start with two points $P_1 = (1, 0)$ and $P_2 = (0, 1)$. Then the points P_1 and

$$P_3 = P_1 \cos(\theta) + \sin(\theta) P_2$$

have angle θ between them.



A similar rule holds for standard normal random variables and correlation, where $\cos(\theta) = \rho$ is the correlation, and $\sin(\theta) = \sqrt{1 - \rho^2}$.

Definition 58
Let Z_1 and Z_2 by iid $\mathsf{N}(0, 1)$. Then

$$(Z_1, \rho Z_1 + \mathrm{sqrt}(1 - \rho^2) Z_2)$$

are bivariate normal random variables with correlation ρ.

Proof. Let $(Y_1, Y_2) = (Z_1, \rho Z_1 + \mathrm{sqrt}(1 - \rho^2) Z_2)$. Then Y_i is standard normal by the rules for scaling and adding normal random variables. Also:

$$\mathbb{E}[Y_1 Y_2] = \mathbb{E}[\rho Z_1^2 + \mathrm{sqrt}(1 - \rho^2) Z_1 Z_2] = \rho,$$

which completes the proof.

□

For the Question of the Day, one stock price can be found by using $W_1 = Z_1$ and the other can use $B_1 = \rho Z_1 + \text{sqrt}(1-\rho^2)Z_2$ to simulate the correlated stock prices.

Instaneous correlation

This can be taken down to the level of two Brownian motions if the entire path is needed.

Definition 59
Say that Weiner Processes W_t and B_t have **instantaneous correlation** ρ if

$$dW_t \cdot dB_t = \rho\, dt.$$

Then the change in two standard Brownian motions can be built as follows. For a change in time h,

1. Draw Z_1, Z_2 iid $\mathsf{N}(0,1)$.
2. Let $\epsilon_1 \leftarrow Z_1 \,\text{sqrt}(h)$.
3. Let $\epsilon_2 \leftarrow [\rho Z_1 + \text{sqrt}(1-\rho^2)Z_2] \cdot \text{sqrt}(h)$.

Note that by the properties of scaling and adding random variables,

$$\epsilon_1 \sim \mathsf{N}(0,h),\ \epsilon_2 \sim \mathsf{N}(0,[\rho^2 + (1-\rho^2)]h) = \mathsf{N}(0,h)$$

and

$$\mathbb{E}[\epsilon_1\epsilon_2] = \text{sqrt}(h)\rho\,\text{sqrt}(h)\mathbb{E}[Z_1^2] = \rho h.$$

Correlated normal random variables in higher dimensions

In higher dimensions, there is the *correlation matrix* for any vector $(X_1, \ldots, X_n)$ of random variables.

Definition 60
Given random variables $X_1, \ldots, X_n$, the **correlation matrix** is an $n \times n$ matrix Σ where

$$\Sigma_{ij} = \text{Cor}(X_i, X_j).$$

Because of the properties of correlation, the matrix Σ will always be real, symmetric, and positive definite. This is useful for the following reason

Fact 38
For any real symmetric positive definite matrix Σ, there exists a real lower triangular matrix L such that

$$\Sigma = LL^T.$$

The *Cholesky decomposition* algorithm can then be used to find L in $O(n^3)$ time.

So the steps to generate a multivariate normal with mean o and correlation matrix Σ are as follows.

1. Find the Cholesky decomposition $\Sigma = LL^T$.
2. Generate $Z_1, \ldots, Z_n$ standard normals.
3. Return LZ.

Note that for the two dimensional case:

$$\Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$$

and

$$L = \begin{pmatrix} 1 & 0 \\ \rho & \sqrt{1-\rho^2} \end{pmatrix}.$$

24.4 Correlated Brownian motion in R

For the European spread in the Question of the Day, the estimation can be done as follows.

```
spread_option <-
  function(K = 1, T = 1, N = 1, S1_0 = 100, S2_0 = 100,
           r = 0.02, mu1 = 0.07, mu2 = 0.08, sigma1 = 0.1,
           sigma2 = 0.15, d1 = 0.03, d2 = 0.04, rho = 0.50) {
    z1 <- rnorm(N)
    z2 <- rnorm(N)
    h <- T / N
    sqrt_h <- sqrt(h)
    epsilon1 <- z1 * sqrt_h
    epsilon2 <- (rho * z1 + sqrt(1 - rho^2) * z2) * sqrt_h
    x1 <- (r - d1 - (sigma1^2 / 2)) * T +
      sigma1 * sum(epsilon1)
    x2 <- (r - d2 - (sigma2^2 / 2)) * T +
      sigma2 * sum(epsilon2)
    S1_T <- S1_0 * exp(x1)
    S2_T <- S2_0 * exp(x2)
    return(max(c(0, S1_T - S2_T - K)) * exp(-r * T))
  }
```

Now run the simulation.

```
trials <- 10^3
res <- replicate(trials, spread_option())
mean(res)
```

```
## [1] 4.862715
```

```
sd(res) / sqrt(length(res))
```

```
## [1] 0.2265141
```

For higher volatilities, the spread is likely to be greater, making the option more valuable.

```
trials <- 10^3
res <- replicate(trials,
                 spread_option(1, 1, 1, 100, 100, 0.06, 0.7,
                               0.8, 0.2, 0.3, 0.03, 0.04,
                               0.5 ))
mean(res)
```

```
## [1] 10.57536
```

```
sd(res) / sqrt(length(res))
```

```
## [1] 0.44422
```

Suppose that there are two stocks that have value S_1 and W_1 after one year. Both stocks start with price \$100 at time 0. They follow geometric Brownian motion with means 0.07 and 0.08, and volatilities 0.1 and 0.15 respectively. Moreover, the first stock pays a continuous dividend of 0.03 and 0.04. The instantaneous correlation of the stocks is 50%. After one year, a *European spread call* with strike price \$1 pays

$$(S_1 - W_1 - 1)^+.$$

The risk free interest rate is 0.02.

How much should this call option be worth at time 0?

Chapter A

Worked problems

1.1: Given U a uniform over $[0, 1]$, that is, a standard uniform, create a random variable Y that is a function of U such that

$$\mathbb{E}[Y] = \int_0^1 x^3 \, dx.$$

Solution Since the density of a uniform over $[0, 1]$ is $f_U(u) = \mathbb{I}(u \in [0, 1])$,

$$\int_0^1 x^3 \, dx = \int_{\mathbb{R}} x^3 \mathbb{I}(x \in [0, 1]) \, dx = \mathbb{E}[Y],$$

where $\boxed{Y = U^3}$.

1.3: Consider the integral

$$I = \int_{s=0}^2 s^2 \, ds.$$

a) Find a change of variables for s so that the limit of the integral runs from 0 to 1.

b) Find a function h such that for $U \sim \mathsf{Unif}([0, 1])$, $\mathbb{E}[h(U)] = I$.

Solution

a) If $w = s/2$, then when $s = 2$ we have $w = 1$ and $s = 0$ gives $w = 0$. Also, $dw = ds/2$ so $ds = 2 \, dw$ and

$$\int_{s=0}^2 s^2 \, ds = \int_{w=0}^1 (2w)^2 (2 \, dw) = \boxed{\int_{w=0}^1 8w^2 \, dw}.$$

b) Using our usual approach, this means

$$\mathbb{E}[8U^2] = I,$$

so $\boxed{h(u) = 8u^2}$.

1.5: Given an integral

$$I = \int_0^1 \exp(-\sqrt{x})\, dx,$$

what function $h(u)$ has $\mathbb{E}[h(U)] = I$ for U a standard uniform?

Solution Because the density of U is $f_U(u) = \mathbb{I}(u \in [0,1]$, the function is just $h(u) = \boxed{\exp(-\sqrt{u})}$.

1.7: Write R code to estimate

$$I = \int_0^1 \exp(-\sqrt{x})\, dx$$

with seed 123456 using 10^6 samples. Hint: look up (or use your favorite search engine) to find out the functions in R to take the square root function and apply the exponential function to numbers.

Solution This could be done with

```
exp(-sqrt(runif(10^5))) |> mean()
```

1.9: If $U \sim \text{Unif}([0,1])$, what is $\mathbb{E}[\sqrt{U}]$?

Solution This is

$$\begin{aligned}\mathbb{E}[\sqrt{U}] &= \int_{\mathbb{R}} \sqrt{u}\mathbb{I}(u \in [0,1])\, du \\ &= \int_{u\in[0,1]} u^{1/2}\, du \\ &= u^{3/2}/(3/2)|_0^1 \\ &= 2/3,\end{aligned}$$

or $\boxed{0.6666\ldots}$.

2.1: Two useful symbols in LaTeX are `_` for subscripts and `^` for superscripts. For instance, `x^3` becomes x^3, and `x_7` becomes x_7. Use LaTeX and braces { and } to make the following.

a. x^2.

b. x^{-3}.

c. x_4.

d. x_{a+b}.

Solution a. This can be done with `x^2`.

b. This can be done with `x^{-3}`.

c. This can be done with `x_4`.

d. This can be done with `x_{a + b}`.

2.3: The `\frac` command in LaTeX creates fractions. For instance, `\frac{3}{4}` produces

$$\frac{3}{4}$$

Write LaTeX code to create

$$\frac{x_1 + x_2 + x_3}{3} = 7.3.$$

Solution This could be done with `\[ \frac{x_1 + x_2 + x_3}{3} \]`.

2.5: The `\int` command makes integrals in LaTeX. For instance, `\int_0^5 x^2 dx = 125 /3` could be used to make

$$\int_0^5 x^2\,dx = 125/3.$$

Write LaTeX code to make the following output

$$\int_{-1}^{1} 1 - x^2\,dx.$$

Solution This could be done with `\int_{-1}^1 1 - x^2 dx.`.

3.1: Find the following.

a) $\#(\{a, b, c, d\})$.
b) $\#(\{2, 4, 6, \ldots, 100\})$.
c) $\ell([3, 6])$.
d) $\ell((-16, 16))$.

Solution

a) $\boxed{4}$
b) $\boxed{50}$
c) $\boxed{3}$
d) $\boxed{32}$

3.3: For W with density $12s^2(1 - s)\mathbb{I}(s \in [0, 1])$, what is $\mathbb{P}(W \geq 1/2)$?

Solution This is

$$\begin{aligned}\mathbb{P}(W \geq 1/2) &= \int_{1/2}^{\infty} 12s^2(1 - s)\mathbb{I}(s \in [0, 1])\,ds \\ &= \int_{1/2}^{1} 12s^2(1 - s)\,ds \\ &= \frac{11}{16} = \boxed{0.6875}.\end{aligned}$$

3.5: Given X has unnormalized density $g(s) = \exp(-3s)\mathbb{I}(s \geq 0)$, find the normalized density of X.

Solution To find the normalizing constant, we integrate g from $-\infty$ to ∞:

$$\int_{-\infty}^{\infty} \exp(-3s)\, ind(s \geq 0)\, ds = \int_{0}^{\infty} \exp(-3s)\, ds$$
$$= \frac{\exp(-3s)}{-3}\Big|_0^{\infty} = 1/3.$$

We *divide* the unnormalized density by $1/3$ to get

$$f_X(s) = 3\exp(-3s)\mathbb{I}(s \geq 0).$$

4.1: Suppose we want to estimate

$$I = \int_{x\in[0,2]} 3x^2\, dx.$$

a) Using $U_2 \sim \mathsf{Unif}([0,2])$, find h_2 such that $\mathbb{E}[h_2(U_2) = I]$.

b) Transform the integral using $s = x/2$ into an integral over $[0,1]$, and then find h_1 such that $\mathbb{E}[h_1(U_1)] = I$ where $U_1 \sim \mathsf{Unif}([0,1])$.

Solution

a) U_2 has density $(1/2)\mathbb{I}(s \in [0,2])$, so h_2 should be

$$\boxed{h_2(x) = 6x^2}.$$

b) Using $s = x/2$ we have $ds = (1/2)\, dx$, or $2\, ds = dx$. Hence

$$I = \int_{s\in[0,1]} 3(2s)^2(2)\, ds = \int_{s\in[0,1]} 24s^2\, ds.$$

Hence $\boxed{h_1(s) = 24s^2.}$.

(One check on your answers: if $U_1 \sim \mathsf{Unif}([0,1])$, then $2U_1 \sim U_2$. So h_2 should be $2^2 = 4$ times h_1.)

4.3: Consider the integral

$$I = \int_{\mathbb{R}} \frac{1}{1+x^4}\, dx.$$

a) Say $Y \sim$ Cauchy, so $f_Y(s) = [(\tau/2)(1+s^2)]^{-1}$. Find a function $h(Y)$ such that $\mathbb{E}[h(Y)] = I$.

b) Use Wolfram Alpha to find the maximum value of $h(Y)$ for $Y \in \mathbb{R}$.

Solution

a) The importance sampling framework gives:

$$h(Y) = \frac{g(Y)}{f_Y(Y)} = \frac{(1+Y^4)^{-1}}{[\pi(1+Y^2)]^{-1}} = \boxed{\frac{(\tau/2)(1+Y^2)}{1+Y^4}}.$$

b) The maximum value of $h(Y)$ is

$$\frac{\tau}{4}(1+\sqrt{2}) \approx \boxed{3.792}.$$

4.5: Suppose $\mathrm{SD}(X) = 3.2$ and $X_1, \ldots, X_{10}$ are iid as X. What is the standard deviation of

$$\frac{X_1 + \cdots + X_{10}}{10}?$$

Solution The standard deviation of the sample average goes down as the square root of the number of samples. Therefore it is

$$\frac{\mathrm{SD}(X)}{\sqrt{10}} = \frac{3.2}{\sqrt{10}} \approx \boxed{1.011}.$$

4.7: Estimate $\int_0^\infty \exp(-x^{3/2})\,dx$ using $T \sim \mathsf{Exp}(1)$ and importance sampling with 1000 samples.

Solution The following does the steps, calculate $h(T)$ for $T \sim \mathsf{Exp}(1)$, then replicate the draws 1000 times, finally create a tibble with the mean and standard deviation.

```
# First, make the function to generate the random exponential
    and run it through the function
w <- function() {
  u <- rexp(1)
  return(exp(u - u^(3/2)))
  }

# Next, replicate the results
res <- replicate(1000, w())

# Make the tibble to store the statistics of the results
tibble(
  a = mean(res),
  b = sd(res) / sqrt(length(res))
)
```

The answer from my run is approximately $\boxed{0.900 \pm 0.011}$.

4.9: Write R code to estimate $\int_0^1 \exp(\sqrt{x})\,dx$ using 1000 draws. Estimate the error and report your answer in the form $a \pm b$.

Solution

```
# First generate from the random variable whose mean equals
    the answer we are looking for:
res <- exp(sqrt(runif(1000)))

#Next find the mean:
mean(res)
## The output is 1.981654

# Finally find the error:
sd(res) / sqrt(length(res))
## The output is  0.01445422
```

Therefore the result is $\boxed{2.021 \pm 0.014}$.

5.1: Let $Y \sim \text{Exp}(2)$.

a. Set up the integral for the fourth moment of Y.

b. Solve the integral using WolframAlpha.

Solution a. Given the density of Y is $2\exp(-2y)\mathbb{I}(y \geq 0)$, the integral is

$$\boxed{\int_0^\infty 2y^4 \exp(-2y)\, dy}.$$

b. Using WolframAlpha call `integrate 2*y^4*exp(-2*y) from 0 to infinity` the answer is $3/2$, or $\boxed{1.500}$ to 4 sig figs.

5.3: Suppose X has mean 4.1 and standard deviation 1.2, and $X_1, \ldots, X_{10}$ are iid X. Let $Y = (X_1 + \cdots + X_{10})/10$

a) Find $\mathbb{E}[Y]$.

b) Find $\text{SD}[Y]$.

Solution

a) The mean of a sample average is just the same as the original random variable, so $\boxed{4.100}$.

b) The standard deviation of a sample average is the original standard deviation divided by the square root of the number of terms in the average. So this is

$$\frac{1.2}{\sqrt{10}} = \boxed{0.3794\ldots}.$$

5.5: Let $U \sim \text{Unif}([0,1])$ (so it has density $f_U(u) = \mathbb{I}(u \in [0,1])$. Find the ith moment of U.

Solution This is

$$\mathbb{E}[U^i] = \int_{\mathbb{R}} u^i \mathbb{I}(u \in [0,1])\, du = \int_0^1 u^i\, du = \frac{u^{i+1}}{i+1}\bigg|_0^1 = \boxed{\frac{1}{i+1}}.$$

6.1: Write a Monte Carlo algorithm in R to estimate $\mathbb{P}(U_1 + \cdots + U_6 \geq 5)$ where the U_i are iid Unif$([0,1])$.

Solution This can be accomplished with

```
mc <- function() {
  return(sum(runif(6)) >= 5)
}
results <- replicate(10^6,mc())
print(mean(results))
```

6.3: Suppose we model $U_1, \ldots, U_n$ given parameter θ as iid Unif$([0,\theta])$. Given $U_1, \ldots, U_n$, our test statistic is

$$T = \min\{U_1, \ldots, U_n\}$$

We are trying to test if $\theta = 10$, and we consider high values of T to be evidence against this hypothesis.

a) If $n = 8$, what is the p-value for a test statistic of 4? (Calculate this value exactly as a probability.)

b) Use R to test your last answer to the last part by writing a function to generate variates from T, and then use Monte Carlo to estimate the p-value.

Solution

a) The goal here is to find the probability that the minimum of 8 standard uniforms is greater than 4. This is:

$$\begin{aligned}\mathbb{P}(\min\{U_1, \ldots, U_8\} > 4) &= \mathbb{P}(U_1 > 4, \ldots, U_8 > 4) \\ &= \mathbb{P}(U_1 > 4)^8 \\ &= ((10-4)/10)^8 \\ &= \boxed{0.01679\ldots}\end{aligned}$$

b) The following code:

```
results <- replicate(10^6,(min(runif(8))*10) > 4)
print(mean(results))
print(sd(results)/sqrt(length(results)))
```

does the trick. When I ran it, it returned 0.01683, of course every time you run it you will get a slightly different answer.

7.1: Suppose $\mathbb{P}(A = -1) = 0.6$ and $\mathbb{P}(A = 1) = 0.4$. Find a function $h(u)$ such that $h(U) \sim A$ when $U \sim$ Unif$([0,1])$.

Solution Using the ITM gives

$$\boxed{h(u) = (-1)\mathbb{I}(U < 0.6) + \mathbb{I}(U \geq 0.6)}.$$

7.3: For $U \sim \text{Unif}([-1,1])$, find the density of U^2.

Solution First let's find the cdf of U^2. Since $U^2 \geq 0$, for $a < 0$, $\mathbb{P}(U^2 \leq a) = 0$. For $a \geq 0$,

$$\begin{aligned}\mathbb{P}(U^2 \leq a) &= \mathbb{P}(|U| \leq \sqrt{a}) \\ &= \mathbb{P}(-\sqrt{a} \leq U \leq \sqrt{a}) \\ &= (1/2)2\sqrt{a}\mathbb{I}(a \leq 1) + \mathbb{I}(a > 1).\end{aligned}$$

So the cdf is $\sqrt{a}\mathbb{I}(a \in [0,1]) + \mathbb{I}(a > 1)$. Differentiating then gives

$$\boxed{f_{U^2}(a) = (1/2)a^{-1/2}\mathbb{I}(a \in [0,1]).}$$

Note that

7.5: Find a function g such that $g(U)$ has density

$$f(s) = (1/s^2)\mathbb{I}(s \geq 1).$$

assuming $U \sim \text{Unif}([0,1])$.

Solution First we find the cdf:

$$\text{cdf}(a) = \int_{-\infty}^{a} (1/s^2)\mathbb{I}(s \geq 1)\, ds.$$

For $a < 1$ this is o, for $a \geq 1$,

$$\begin{aligned}\text{cdf}(a) &= \int_1^a (1/s^2)\, ds \\ &= -1/s|_1^a \\ &= 1 - 1/a.\end{aligned}$$

Setting $1 - 1/X = U$ gives $X = 1/(1-U)$. So we could use

$$\boxed{g(u) = 1/(1-u)},$$

but we could also use $g(u) = 1/u$ since $U \sim 1 - U$.

8.1: Suppose that the function `mc` draws numbers uniform from $\{-5, -4, \ldots, 4, 5\}$. Write an AR code in R that makes a function `mc` that draws uniformly from $\{-1, 0, 1\}$.

Solution Such code looks like

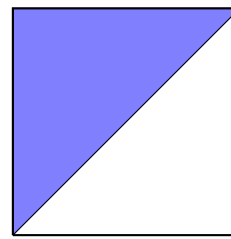
```
mc2 <- function() {
  repeat {
    X <- mc()
    if ((X >= -1)*(X <= 1) == 1) return(X)
  }
}
```

8.3: The function `rexp(1,2)` draws a single exponential random variable of rate 2. Use AR to build a function `mcexp` in R that draws from an exponential random variable of rate 2 conditioned to lie in $[0.5, 1.5]$.

Solution The following code uses the `while` function rather than `repeat`. Both are equivalent, but the structure is slightly different. In a `while` loop, the condition is first evaluated, and then the statement following the `while` is run, and the process starts over.

```
exp_cond <- function() {
  accept <- FALSE
  while (!accept) {
    x <- rexp(1, 2)
    accept <- (x > 0.5) & (x < 1.5)
  }
  return(x)
}
```

8.5: Write pseudocode to draw $(X, Y) \sim \text{Unif}(\Omega)$, where Ω is the trangular region connecting $(0, 0), (0, 1), (1, 1)$.



Solution The following code does this.

```
upper_triangle <- function() {
  accept <- FALSE
  while (!accept) {
    point <- runif(2)
    accept <- point[2] > point[1]
  }
  return(point)
}
```

9.1: The following code uses the inverse transform method to draw a sample uniformly from

$$f_X(x) = \frac{1}{x^2}\mathbb{I}(x \geq 1)$$

```
mc <- function() {
  u <- runif(1)
  return(1/u)
}
```

Write R code that uses this function (together with AR) to draw a sample from unnormalized density

$$g_X(x) = \frac{1}{x^{2.5}}\mathbb{I}(x \geq 1)$$

Solution Note that $g_X(x) \leq f_X(x)$ for all $x \geq 1$. So we must accept a draw $A \sim f_X$ as coming from $g_X(x)$ with probability

$$\frac{g_X(A)}{f_X(A)} = \frac{(1/A^{2.5})\mathbb{I}(A \geq 1)}{1/A^2} = A^{-0.5}\mathbb{I}(A \geq 1).$$

```
mc2 <- function() {
  acceptflag <- FALSE
  while (!acceptflag) {
    x <- mc()
    u <- runif(1)
    acceptflag <- u < x^(-0.5)
  }
  return(x)
}
```

9.3: Suppose that I have the ability to draw Y from the following density

$$g_Y(i) = i\mathbb{I}(i \in \{1, 2, 3\}),$$

and U from Unif$([0, 1])$.

a) Create the most efficient AR algorithm for drawing from

$$g_X(i) = [(1 + i)/2]\mathbb{I}(i \in \{1, 2, 3\}).$$

b) What is the expected number of times through the repeat loops for your AR algorithm?

Solution

a) The acceptance probabilities are

$$\frac{g_X(1)}{g_Y(1)} = \frac{1}{1}, \frac{g_X(2)}{g_Y(2)} = \frac{3/2}{2}, \frac{g_X(3)}{g_Y(3)} = \frac{2}{3},$$

so all are are most 1. So the AR method just will test if a standard uniform is at most $g_X(Y)/g_Y(Y)$.

AR
1) Repeat
2) Draw $Y \leftarrow g_Y$
3) Draw $U \leftarrow \text{Unif}([0,1])$
4) Until $U \leq (1+Y)/(2Y)$
5) Return Y

b) Note that $g_X(i)$ is unnormalized. The normalizing constant will be

$$g_X(1) + g_X(2) + g_X(3) = 6.$$

Hence

$$\mathbb{P}(X = 1) = 1/6,\ \mathbb{P}(X = 2) = 2/6,\ \mathbb{P}(X = 3) = 3/6.$$

That makes the probability of accepting is

$$\mathbb{E}\left[\frac{1+Y}{2Y}\right] = \frac{1}{6}\cdot\frac{2}{2} + \frac{2}{6}\cdot\frac{3}{4} + \frac{3}{6}\cdot\frac{4}{6} = \frac{3}{4}.$$

So the expected number of times through the loop will be the inverse of this, or

$$\frac{4}{3} = \boxed{1.333\ldots}$$

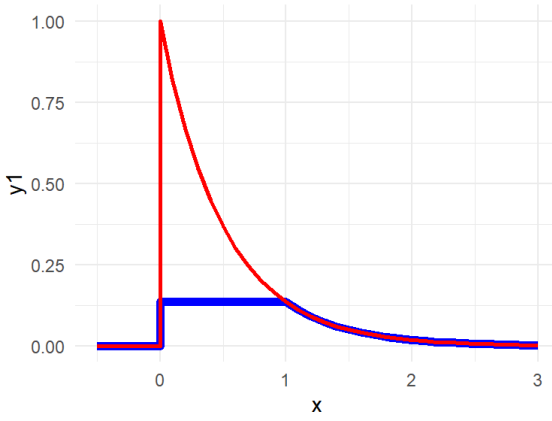
9.5: Consider the following unnormalized density:

$$f_W(w) = \exp(-2)\mathbb{I}(w \in [0,1]) + \exp(-2w)\mathbb{I}(w > 1).$$

Now consider the unnormalized density

$$f_T(t) = \exp(-2t)\mathbb{I}(t \geq 0).$$

Note that for all t, $f_T(t) \geq f_W(t)$. This looks as follows.



a) Consider a draw (X, Y) from underneath the density f_T. What is the probability that this falls underneath the density f_W?

b) Write pseudocode for an AR algorithm that draws from W using draws from T.

Solution

a) This is just the area under f_W divided by the area under f_T:

$$\begin{aligned} p &= \frac{\int_0^\infty \exp(-2)\mathbb{I}(5 \in [0,1]) + \exp(-2t)\mathbb{I}(t > 1)\, dt}{\int_0^\infty \exp(-2t)\, dt} \\ &= \frac{\exp(-2)(1-0) + \exp(-2t)/(-2)|_1^\infty}{\exp(-2t)/(-2)|_0^\infty} \\ &= \frac{\exp(-2) + \exp(-2)/2}{1/2} \\ &= 3\exp(-2) = \boxed{0.4060\ldots} \end{aligned}$$

b) The pseudocode is as follows:

AR_two_densities

1) Repeat
2) Draw $T \sim f_T$ and U a standard uniform independently
3) Until $U < f_W(T)/f_T(T)$
4) Return T

9.7: Suppose the goal is to use draws from unnormalized density

$$g_A(s) = \exp(-0.9x)\mathbb{I}(x \geq 0)$$

to draw from unnormalized density

$$g_B(s) = x\exp(-x)\mathbb{I}(x \geq 0)$$

It is not true that $g_A(s) \geq g_B(s)$ for all $s \geq 0$, but it is true that for

$$m = \max_{s: g_A(s) > 0} \frac{g_B(s)}{g_A(s)},$$

it holds that

$$m g_A(s) \geq g_B(s).$$

a) Find m and round up to four sig figs.

b) Write pseudocode for AR for using draws from A to obtain draws from B.

Solution

a) To find m, let

$$f(s) = \frac{g_B(s)}{g_A(s)} = s\exp(-s + 0.9s) = s\exp(-0.1s).$$

Then

$$f'(s) = \exp(-0.1s) - 0.1s\exp(-0.1s) = \exp(-0.1s)(1 - 0.1s).$$

The first factor is always positive, so the slope depends on the sign of $1 - 0.1s$. Since $1 - 0.1s$ decreases in s, there is a maximum when $1 - 0.1s = 0$ or $s = 10$. At $s = 10$, the ratio is $10\exp(-1)$, and so that is the maximum value. Finally, $10\exp(-1)$ rounded up to four sig figs is $\boxed{3.679}$.

b) The pseudocode is then

AR_A_B	
1)	Repeat
2)	Draw $A \sim g_A$ and U a standard uniform independently
3)	Until $U < g_B(A)/[3.679g_A(A)]$
4)	Return A

9.9: A *double exponential* random variable of rate λ has density

$$f_T(t) = \frac{\lambda}{2}\exp(-\lambda|t|).$$

Another way to think about it as an exponential of rate λ that has a fifty-fifty chance of being positive or negative. Pseudocode for drawing from this distribution is as follows.

AR_dblexp_norm	
1)	Repeat
2)	Draw T double exponential of rate 1
3)	Draw U a standard uniform
4)	Until $U < \exp(-T^2/2 + \|T\| - 1/2)$
5)	Return T

Write pseudocode for an acceptance rejection algorithm that uses a double exponential of rate 1 to make draws using AR from a standard normal with unnormalized density $\exp(-x^2/2)$.

Solution First, it is necessary to maximize

$$r(x) = \frac{\exp(-x^2/2)}{\exp(-|x|)}.$$

By symmetry, the maximum will be the same for $x > 0$ and $x < 0$. Note for $x > 0$

$$r'(x) = (1 - x)\exp(x - x^2/2).$$

Since the $\exp(x - x^2/2)$ part is positive, the slope is controlled by the $(1 - x)$ factor. This is positive for $x < 1$, hits zero at $x = 1$, and is negative thereafter. Hence the maximum value occurs at $x = 1$, and $r(1) = \exp(1/2)$.

This makes the pseudocode:

random_doubleexponential(λ)	
1)	Repeat
2)	Draw T exponential of rate λ
3)	Draw B a Bernoulli with parameter $1/2$
4)	Return $TB + (-T)(1 - B)$

10.1: Suppose the random variables (X_1, X_2, X_3) are multinomial with $n = 10$ and $(p_1, p_2, p_3) = (0.2, 0.5, 0.3)$. (So $(X_1, X_2, X_3) \sim \mathsf{Multinom}(10, 0.2, 0.5, 0.3)$.)

a) What is the distribution of X_1?

b) What is the distribution of (X_2, X_3) given that $X_1 = 4$?

Solution

a) The marginal distribution of X_1 will be $\boxed{\mathsf{Bin}(10, 0.2)}$.

b) Knowing $X_1 = 4$ means that (X_2, X_3) has $20 - 4 = 16$ experiments left. Renormalizing the probabilities gives $(0.5, 0.3)/(0.5 + 0.3)$, so

$$\boxed{(X_2, X_3) \sim \mathsf{Multinom}(6, 0.6250, 0.3750)}.$$

10.3: Suppose that (X_1, X_2, X_3) are uniform over the permutations of the numbers 1, 2, and 3.

a) What is the distribution of X_1?

b) Give $X_1 = 2$, what is the distribution of (X_2, X_3)?

Solution

a) In a uniform permutation X_1 is equally likely to be any of the elements, so $\boxed{X_1 \sim \mathsf{Unif}(\{1, 2, 3\})}$.

b) Given $X_1 = 2$, the remaining elements are uniform over the remaining elements. So

$$\boxed{(X_2, X_3) \text{ is uniform over } \{1, 3\}.}$$

10.5: Suppose $U_1 \sim \text{Unif}([0,1])$ and $[U_i|U_1, \ldots, U_{i-1}] \sim \text{Unif}([0, U_{i-1}]$. Write pseudocode that takes n as input and returns a random draw from $(U_1, \ldots, U_n)$.

Solution One way to do this is:

Conditional Sampling	*Input:* n
1)	Draw $U_1 \leftarrow \text{Unif}([0,1])$
2)	For i from 2 to n
3)	Draw $U_i \leftarrow \text{Unif}([0, U_{i-1}])$
4)	Return $(U_1, \ldots, U_n)$

11.1: Let $V_1, \ldots, V_5$ be chosen uniformly from the surface of a 5 dimensional sphere. Using 10^6 samples, estimate

$$\mathbb{E}[||V||_1] = \mathbb{E}\left[\sum_{i=1}^{5} |V_i|\right]$$

and report your estimate in $a \pm b$ form.

Solution This can be accomplished with

```
one_draw <- function(n = 5) {
  initial <- rnorm(n)
  z <- initial / sqrt(sum(initial^2))
  return(sum(abs(z)))
}
res <- replicate(10^6, one_draw())
mean(res)
sd(res) / sqrt(length(res))
```

In my run, this gave 1.87473 ± 0.00018.

11.3: Use the normal method to estimate what proportion of the Earth's surface is between 40 and 50 degrees latitude north of the equator. Take 10^6 samples and estimate your answer in the form $a \pm b$.

Solution Draw uniformly from the surface of a sphere, then figure out how many of those points are between 40 and 50 degrees latitude north of the equator.

```
earth_surface_lat <- function() {
  init <- rnorm(3)
  z <- init / sqrt(sum(init^2))
  lat <- acos(abs(z[3])) * sign(z[3]) * 90 / (pi / 2)
  return(lat)
}
```

```
res <- replicate(10^6, earth_surface_lat())
mean((res > 40) & (res < 50))
sd((res > 40) & (res < 50)) / sqrt(length(res))
```

In my run, the result was $\boxed{0.0614 \pm 0.0003}$

11.5: Write code to draw (X, Y) uniformly from the unit circle (the boundary of a ball of radius 1) in $\mathbb{R}^2$. Using 10^6 draws, estimate $\mathbb{P}(Y \geq 0.7)$, reporting your answer as $a \pm b$.

Solution Code for this is

```
unif_direction_dim2 <- function() {
  theta <- runif(1) * 2 * pi
  return(c(cos(theta), sin(theta)))
}
```

Now to get our results,

```
res <- replicate(10^6, unif_direction_dim2()[2] >= 0.7)
mean(res)
sd(res) / sqrt(length(res))
```

The estimate is then $\boxed{0.2536 \pm 0.0005}$.

11.7: Consider (X, Y) uniform over the unit disc $\text{Unif}(\{(x, y) : x^2 + y^2 \leq 1\})$. Consider the distance from the origin $R = \sqrt{X^2 + Y^2}$

a) Find, using the properties of uniforms, $\mathbb{P}(R \leq 0.3)$.

b) Find for $r \in [0, 1]$, $\mathbb{P}(R \leq r)$.

c) Write code to sample from R using the inverse transform method.

d) Draw (X, Y) uniformly from the unit disc by first drawing R using the last part, and θ uniformly from 0 to τ, then converting from polar coordinates to Cartesian coordinates.

e) Using 10^6 draws, estimate $\mathbb{P}(Y \geq 0.5)$, reporting your answer as $a \pm b$.

Solution

a) The area of the disc of radius 0.3 is $(1/2)\tau(0.3)^2$, and the area of the whole disc of radius 1 is $(1/2)\tau(1)^2$, so the probability is

$$\frac{(1/2)\tau(0.3)^2}{(1/2)\tau(1)^2} = (0.3)^2 = \boxed{0.09000}.$$

b) In general, the probability for $r \in [0, 1]$ that $R \leq r$ will be

$$\frac{(1/2)\tau r^2}{(1/2)\tau(1)^2} = \boxed{r^2}.$$

c) For cdf of R equal to r^2, setting $U = r^2$ gives $r = \sqrt{U}$. So this can be done with `rr <-function()return(sqrt(runif(1)))`.

d) Code to do this:

```
rdraw <- function() {
  r <- rr()
  theta <- 2 * pi * runif(1)
  return(c(r * cos(theta), r * sin(theta)))
}
res <- replicate(10^6, rdraw()[2])
mean(res >= 0.5)
sd(res >= 0.5) / sqrt(length(res))
```

My run gave answers 0.1950 ± 0.0004.

12.1: Consider a 3 by 3 lattice.

a) How many nodes are there in the lattice?

b) How many edges are there in the lattice?

Solution

a) There are $3 \cdot 3 = \boxed{9}$ nodes in the lattice.

b) There are $2 \cdot 3$ horizontal edges and $3 \cdot 2$ vertical edges, for a total of $\boxed{12}$ edges.

12.3: Using 200 exact draws from the Ising model on a 3 by 3 lattice with $\beta = 0.8$, find the average of the h function value. Write your answer as $a \pm b$.

Solution The following calculates $h(x)$ for a state x in a matrix variable.

```
h <- function(x) {
  ver <- sum(x[1:(nrow(x)-1), ] == x[2:nrow(x), ])
  hor <- sum(x[, 1:(ncol(x)-1)] == x[, 2:ncol(x)])
return(hor + ver)
}
```

The following uses AR to draw from the Ising model.

```
ar_ising <- function(beta, k) {
  repeat {
    x <- matrix(as.integer(runif(k[1] * k[2]) > 0.5),
    nrow = k[1])
    u <- runif(1)
    if (u < exp(-beta * (2 * k[1] * k[2] - k[1] - k[2] - h(x)
        )))
    return(x)
  }
}
```

Okay, now let's answer the question!

```
res <- replicate(200, h(ar_ising(0.8, c(3, 3))))
mean(res)
sd(res) / sqrt(length(res))
```

The output in my run was $\boxed{8.77 \pm 0.15}$.

12.5: Using 200 draws from the Ising model on a 3 by 3 lattice with $\beta = 1.2$, find the average of the h function value. Write your answer as $a \pm b$.

Solution Given the functions from the last problem, this can be done with:

```
res <- replicate(200, h(ar_ising(1.2, c(3, 3))))
mean(res)
sd(res) / sqrt(length(res))
```

The answer in my run was 10.36 ± 0.14.

12.7: For state space $[0, 1]^{10}$, consider $X = (X_1, \ldots, X_{10})$ with unnormalized density

$$g_X(x_1, x_2, \ldots, x_{10}) = x_1 + 2x_2 + \cdots + 10x_{10}.$$

In R, this density can be computed using:

```
g <- function(x) {
  return(sum(1:10 * x))
}
```

Using uniform draws over the state space in AR, write a function in R to draw from this distribution.

Solution The unnormalized density of U uniform over $[0, 1]^{10}$ is one times the indicator that it is in the state space. The greatest value that g can take on is $1 + 2 + \cdots + 10 = (10)(11)/2 = 55$. Hence the following code does the trick.

```
example_ar <- function() {
  a <- FALSE
  while (!a) {
    x <- runif(10)
    u <- runif(1)
    a <- u < g(x) / 55
  }
  return(x)
}
```

13.1: Consider the update function on state space $\{0, 1, \ldots, n-1\}$ so that either adds 1 or subtracts 1 mod n with probability 0.3, and with probability 0.4 stays where it is. Let

$$f(u) = \mathbb{I}(u > 0.7) - \mathbb{I}(u < 0.3),$$

and

$$\phi(x,u) = x + f(u) - n\mathbb{I}(x + f(u) = n) + n\mathbb{I}(x + f(u) = -1).$$

Suppose $n = 10$.

a) Write an R function that takes a state x, standard uniform u, and n and returns $\phi(x,u)$ over $\{0,\ldots,n-1\}$.

b) Given $X_0 = 0$, find X_{100} 1000 times, and report an estimate for $\mathbb{E}[X_{100}]$ as $a \pm b$.

Solution

a) This can be done as follows.

```
ring_step <- function(x, u, n = 10) {
  f_u <- (u > 0.7) - (u < 0.3)
  return(x + f_u - n * (x + f_u == n) +
                   n * (x + f_u == -1) )
}
```

b) First, implement one draw from X_{100}.

```
ring_draw_end <- function() {
  x <- 0
  u <- runif(100)
  for (i in 1:100)
    x <- ring_step(x, u[i])
  return(x)
}
```

Now to estimate $\mathbb{E}[X_{100}]$

```
res <- replicate(1000, ring_draw_end())
mean(res)
sd(res) / sqrt(length(res))
```

One run of this code returned $\boxed{4.54 \pm 0.09}$.

13.3: Suppose $\{X_i\}$ is a Markov chain has a stationary distribution over $\{0,1,2\}^{10}$ that is uniform over states with $\sum_{i=1}^{10} X_i \le 7$. Steps in the chain connect any states, and the chain is aperiodic. What can be said about the limiting distribution of the Markov chain, and how do you know this?

Solution The limiting distribution is also $\boxed{\text{uniform}}$ over states with $\sum_{i=1}^{10} X_i \le 7$. This follows from the $\boxed{\text{Ergodic Theorem for finite state Markov chains}}$.

14.1: Consider the transposition chain. Suppose the goal is to use this chain to estimate $\mathbb{P}(x(1) \le x(n))$, where $x \sim \text{Unif}(\mathcal{S}_n)$. For $n = 10$, and $t \in \{10, 50, 100\}$, try running a Markov chain for t burn in steps and t data collecting steps to estimate this probability.

Repeat your Markov chain runs 10 times and report your estimate as $a \pm b$.

Solution First the update function:

```
step_trans <- function(x, i, j) {
  x[c(i, j)] <- x[c(j, i)]
  return(x)
}
```

Now for the data gathering.

```
data_trans <- function(steps, n) {
  burnin    <- steps
  datasteps <- steps
  x <- 1:n
  res <- rep(0, datasteps)
  i_1 <- floor(runif(burnin) * n) + 1
  j_1 <- floor(runif(burnin) * n) + 1
  for (i in 1:burnin)
    x <- step_trans(x, i_1[i], j_1[i])
  i_2 <- floor(runif(datasteps) * n) + 1
  j_2 <- floor(runif(datasteps) * n) + 1
  for (i in 1:datasteps) {
    res[i] <- (x[1] <= x[n])
    x <- step_trans(x, i_2[i], j_2[i])
  }
  return(mean(res))
}
```

The analysis then proceeds as follows.

```
data10 <- replicate(10, data_trans(10, 10))
data50 <- replicate(10, data_trans(50, 10))
data100 <- replicate(10, data_trans(100, 10))
```

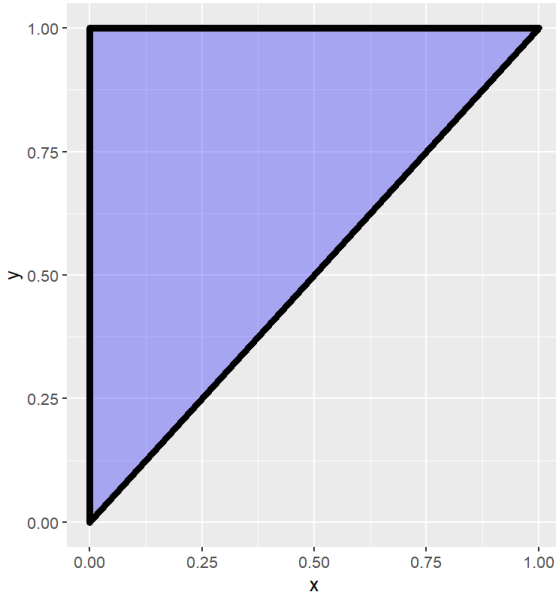
Now get results

```
n <- c(10, 50, 100),
est_mean <- c(mean(data10), mean(data50), mean(data100)),
est_err  <- c(sd(data10), sd(data50), sd(data100)) / sqrt(n)
est_mean
est_err
```

Therefore, the answer for this run is as follows:

$$\begin{aligned} n = 10 \qquad & a \pm b = \boxed{0.73 \pm 0.11} \\ n = 50 \qquad & a \pm b = \boxed{0.540 \pm 0.054} \\ n = 100 \qquad & a \pm b = \boxed{0.521 \pm 0.056} \end{aligned}$$

14.3: Suppose that (X, Y) is uniform over the triangle in $\mathbb{R}^2$ with vertices $(0, 0)$, $(0, 1)$, and $(1, 1)$.



a) What is the distribution of X given Y?

b) What is the distribution of Y given X?

Solution

a) Given Y, X runs from 0 up to Y. Hence

$$\boxed{[X \mid Y] \sim \mathsf{Unif}([0, Y])}.$$

b) Given X, Y runs from X up to 1. Hence

$$\boxed{[Y \mid X] \sim \mathsf{Unif}([X, 1])}.$$

14.5: Implement a random scan Gibbs sampler for a Markov chain which is uniform over

$$\Omega_{10} = \left\{(x_1, \ldots, x_{10}) \in \{0, 1, 2\}^{10} : \sum_{i=1}^{10} x_i \le 7\right\}$$

as an update function in R that takes as input the current state x, a dimension i in $\{1, \ldots, 10\}$, and a standard uniform u and returns the next state of the chain.

Solution Consider changing x_i. Then either 0, 1, or 2 are viable choices, as long as it does not make the sum too large. Hence

$$x_i \sim \mathsf{Unif}\left(\{0, 1, 2\} \cap \left(-\infty, 7 - \sum_{j \ne i} x_i\right]\right)$$

This can be implemented in R as follows.

```
step_sum <- function(x, i, u) {
  m <- min(2, 7 - (sum(x) - x[i]))
  x[i] <- floor((m + 1) * u)
  return(x)
}
```

14.7: Continuing the earlier problem, using 1000 burn in and 1000 data gathering steps, estimate $\sum_{i=1}^{10} X_i$ for $(X_1, \ldots, X_{10}) \sim \text{Unif}(\Omega_{10})$. Repeat your Markov chain 5 times and report your estimate as $a \pm b$.

Solution The following code accomplishes this:

```
sum_data <- function(steps) {

  # Initialization
  burnin    <- steps
  datasteps <- steps
  x <- rep(0, 10)
  res <- rep(0, datasteps)

  # Burnin steps
  u_1 <- runif(burnin)
  i_1 <- floor(runif(burnin) * 10) + 1
  for (i in 1:burnin)
    x <- step_sum(x, i_1[i], u_1[i])

  # Data gathering steps
  u_2 <- runif(datasteps)
  i_2 <- floor(runif(datasteps) * 10) + 1
  for (i in 1:datasteps) {
    res[i] <- sum(x)
    x <- step_sum(x, i_2[i], u_2[i])
  }
  return(mean(res))
}
```

Now to run and make estimates.

```
results <- replicate(5, sum_data(1000))
mean(results)
sd(results) / sqrt(length(results))
```

One estimate is $\boxed{6.159 \pm 0.016}$.

14.9: Create a random scan Gibbs sampler that has stationary distribution uniform over the six dimensional unit hypersphere, that is

$$\{(x_1, \ldots, x_6) : x_1^2 + \cdots + x_6^2 \leq 1\}.$$

Implement your sampler as an R function that inputs the current state 'x', a dimension 'i', a standard uniform 'u', and returns the next state in the Markov chain.

Solution Fix $i \in \{1, \ldots, 6\}$. Given the rest of the x_j values for $j \neq i$, x_j is uniform from postive 1 minus the sum of the squares of the rest of the components to the negative of that number.

This can be implemented in R as follows

```
step_rs_hypersphere <- function(x, i, u) {
  m <- sqrt(1 - (sum(x^2) - x[i]^2))
  x[i] <- 2 * m * u - m
  return(x)
}
```

14.11: Create a deterministic scan Gibbs sampler that has stationary distribution uniform over the six dimensional unit hypersphere, that is

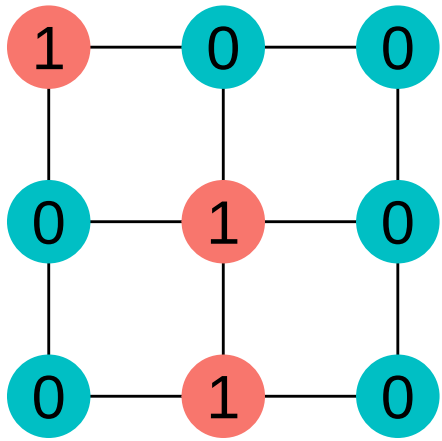
$$\{(x_1, \ldots, x_6) : x_1^2 + \cdots + x_6^2 \leq 1\}.$$

Implement your sampler step as an R function that inputs the current state 'x', a vector of six iid standard uniforms 'u', and returns the next state in the Markov chain.

Solution A deterministic scan Gibbs sampler updates all the components in order.

```
step_ds_hypersphere <- function(x, u) {
  m <- rep(0, 6)
  for (i in 1:6) {
    m[i] <- sqrt(1 - (sum(x^2) - x[i]^2))
    x[i] <- 2 * m[i] * u[i] - m[i]
  }
  return(x)
}
```

15.1: Consider an Ising model with state

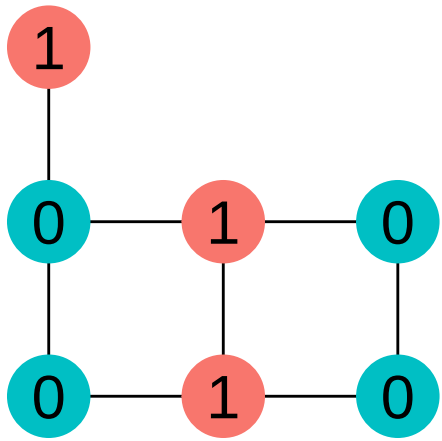


Label the node that is in the center of the top row node 2. Suppose a random scan Gibbs chain selects node 2 to be updated. For $\beta = 1.2$, find exactly the probability that node 2 will be updated by the Gibbs step to have label 0.

Solution There is one neighbor of node 2 labeled 0, and two neighbors labeled 1. Hence the probability node 2 will be 0 will be

$$\frac{\exp(1\beta)}{\exp(2\beta) + \exp(1\beta)} = \boxed{0.2314\ldots}.$$

15.3: Consider the following graph. If a random scan Gibbs chain for the Ising model is run on this graph, what is the maximum number of node labels that need to be examined to take one step in the chain?



Solution The maximum number of neighbors a node has is this graph is $\boxed{3}$, so that is how many neighbors need to be examined (at most) to take one Gibbs step in the chain.

15.5: Consider the following unnormalized density for $x = (x_1, \ldots, x_{10})$.

$$g(x) = (x_1x_2 + x_2x_3 + x_3x_4 + \cdots + x_9x_{10})\mathbb{I}(x \in [0,4]^{10}).$$

Note that x_{10} only appears in the density with x_9, so suppose a Markov chain is run where g, x_9, and x_10 are known at the current state of the chain. Suppose a step is taken in the Gibbs chain that replaces the tenth component.

For instance, if $g(x) = 64.2$, $x[9] = 3.3$, and $x[10] = 1.2$, then setting $y_i = x_i$ for $i < 10$, and $y_{10} = A$ where A is a random variable, then consider $f_A(s)$, the density of A. Given these values, the sum of the first eight terms in g would be the sum of all the terms minus the last term, so

$$\left[\sum_{i=1}^{8} x_ix_{i+1}\right] = 64.2 - (3.3)(1.2) = 60.24.$$

so the unnormalized density of A is

$$g_A(y_{10}) = (60.24 + 3.3y_{10})\mathbb{I}(y_{10} \in [0,4])$$

Give a function h such that for $U \sim \mathsf{Unif}([0,1])$, $h(U) \sim A$.

Solution This can be accomplished with the Inverse Transform Method.

First let's normalize the density:

$$\int_0^4 (60.24 + 3.3y_{10})\, dy_{10} = 267.36.$$

Second find the cdf. For $a \in [0, 4]$,

$$\text{cdf}_A(a) = \int_0^a (60.24 + 3.3y_{10})/267.36 \, dy_{10} = \frac{1.65a^2 + 60.24a}{267.36}.$$

Third, set this equal to u and solve:

$$\begin{aligned} u &= \frac{1.65a^2 + 60.24a}{267.36} \\ 267.36u &= 1.65a^2 + 60.24a \\ 1.65a^2 + 60.24a - 267.36u &= 0 \end{aligned}$$

Applying the quadratic formula and picking the positive value gives:

$$a = \frac{-60.24 + \sqrt{60.24^2 + 4(1.65)(267.36)}}{2(1.65)}$$

15.7: Continuing the last problem, you need to know g and some of the x_i values to calculate the probabilities for the changed dimension in the Gibbs chain. What is the largest number of x_i values that you need to know?

Solution Since each x_i interacts with at most x_{i-1} and x_{i+1}, you need to know the value of at most $\boxed{2}$ other components.

16.1: Consider taking a asymmetric random walk on the integers mod 5 ($\{0, 1, 2, 3, 4\}$), using M where $\mathbb{P}(M = -1) = 0.6$ and $\mathbb{P}(M = 1) = 0.4$. So with probability 0.4 add 1 to the current state, and if it reaches 5 replace it with a 0. Else (with probability 0.6) add -1 to the current state, and if it reaches -1 replace it with a 4. This chain is aperiodic and connected.

This chain has a limiting (normalized) distribution. What is it?

Solution The set of integers mod 5 form a group under addition, and because this is a random walk on a group, the stationary distribution is always uniform over the state space. So it is $\boxed{\text{Unif}(\{0, 1, 2, 3, 4\})}$.

16.3: Again consider the random walk over the integers mod 5 with move

$$\mathbb{P}(M = -1) = 0.6, \ \mathbb{P}(M = 1) = 0.4.$$

Implement the chain. Using 10^4 burnin steps and 10^4 data gathering steps, estimate the mean of the limiting distribution.

Solution

A single step in the chain can be implemented as

```
step_rwmod5 <- function(x, m) {(x + m) %% 5}
```

This the chain looks as follows.

```
chain_rwmod5 <- function(steps) {
  x <- 0
  m_burnin <- 2 * floor(runif(steps) < 0.4) - 1
  for (i in 1:steps) {
    x <- step_rwmod5(x, m_burnin[i])
  }
  m <- 2 * floor(runif(steps) < 0.4) - 1
  states <- rep(0, steps + 1)
  states[1] <- x
  for (i in 1:steps) {
    states[i + 1] <- step_rwmod5(states[i], m[i])
  }
  return(states)
}
```

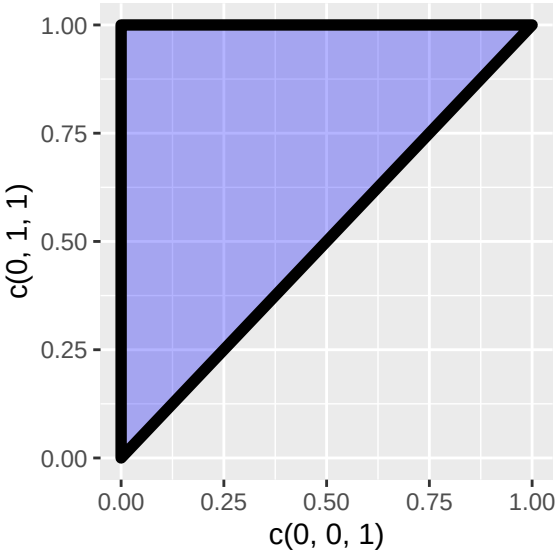
Then the mean can be found as follows:

```
chain_rwmod5(10^4) |> mean()
```

```
## [1] 1.9992
```

Which gives an estimate of the mean of about $\boxed{1.999}$

16.5: Suppose that (X, Y) is uniform over the triangle in $\mathbb{R}^2$ with vertices $(0, 0)$, $(0, 1)$, and $(1, 1)$.



Write code for one step in a random walk with partially reflecting boundaries over this region that takes as input the current state and a vector with two components, and returns the next state.

Solution

This can be done as:

```
step_triangle <- function(v, m) {
  y <- v + m
  if ((y[1] >= 0) & (y[2] <= 1) & (y[2] >= y[1]))
    return(y)
  else
    return(v)
}
```

16.7: Returning to the step for the triangle problem from earlier, use 10 replications of your chain for 10^4 burnin and data gathering steps to estimate $\mathbb{E}[X]$.

Solution

First set up the chain run.

```
chain_triangle_rwwprb <- function(steps) {
  burnin    <- steps
  datasteps <- steps
  x <- c(1, 1)
```

```
  res <- rep(0, datasteps)
  n_1 <- rnorm(burnin)
  n_2 <- rnorm(burnin)
  for (i in 1:burnin)
    x <- setp_triangle(x, c(n_1[i], n_2[i]))
  n_3 <- rnorm(datasteps)
  n_4 <- rnorm(datasteps)
  for (i in 1:datasteps) {
    res[i] <- x[1]
    x <- step_triangle(x, c(n_3[i], n_4[i]))
  }
  return(mean(res))
}
```

Now take some data.

```
res <- replicate(10, chain_triangle_rwwprb(10^5))
```

Summarize the results

```
mean(res)
```

```
## [1] 0.332991
```

```
sd(res) / sqrt(length(res))
```

```
## [1] 0.001053154
```

Therefore the estimate is $\boxed{0.3329 \pm 0.0011}$.

16.9: Suppose that a Markov chain with state space $[0, 10]$ uses a random walk with partially reflecting boundaries. The move is a beta with parameters 3 and 3 minus 0.5. Note that the density of this move is

$$f(s) = (1/2 + s)^2(1/2 - s)^2\mathbb{I}(s \in [-0.5, 0.5]),$$

which is symmetric around o (since $f(s) = f(-s)$.)

This Markov chain has a limiting distribution equal to the stationary distribution. What is this distribution?

Solution The stationary (and limiting distribution) for a random walk on a group with partially reflecting boundaries is always uniform. So the stationary distribution is $\boxed{\mathsf{Unif}([0, 10])}$.

17.1: Consider the density proportional to

$$g_X(x) = \sin(x)\cos(x)^2\mathbb{I}(x \in [0, \tau/4]).$$

a) For a product slice sample, what are the distributions of Y_1, Y_2, and Y_3 given X?

b) For a product slice sample, given (Y_1, Y_2, Y_3), what is the distribution of X?

Solution

a) There are three factors $\sin(x)$, $\cos(x)$, and $\cos(x)$ in g_X. Hence the distributions are

$$\boxed{\begin{aligned}[Y_1|X] &\sim \mathsf{Unif}([0, \sin(x)])\\ [Y_2|X] &\sim \mathsf{Unif}([0, \cos(x)])\\ [Y_3|X] &\sim \mathsf{Unif}([0, \cos(x)])\end{aligned}}$$

b) For X to be valid given (Y_1, Y_2, Y_3), it must hold that

$$\begin{aligned}Y_1 &\le \sin(X)\\ Y_2 &\le \cos(X)\\ Y_3 &\le \cos(X).\end{aligned}$$

Since $\sin(x)$ is an increasing function over the range of x, so is it inverse. Since $\cos(x)$ is a decreasing function over the range of x, its inverse is also decreasing. Hence

$$\begin{aligned}\mathrm{asin}(Y_1) &\le X\\ \mathrm{acos}(Y_2) &\ge X\\ \mathrm{acos}(Y_3) &\ge X.\end{aligned}$$

This means

$$\boxed{[X|Y_1, Y_2, Y_3] \sim \mathsf{Unif}([\mathrm{asin}(Y_1), \min(\mathrm{acos}(Y_2), \mathrm{acos}(Y_3))]).}$$

17.3: Going back to

$$g_X(x) = \sin(x)\cos(x)^2\mathbb{I}(x \in [0, \tau/4]),$$

implement an update function that takes as input the current state `x` and a vector of four numbers `u` and returns the next state after updating Y_1, Y_2, Y_3, X in that order.

Solution

This can be done with:

```
step_trig_pss <- function(x, u) {
  y <- u[1:3] * c(sin(x), cos(x), cos(x))
  x <- u[4] * (min(acos(y[2:3])) - asin(y[1])) + asin(y[1])
  return(x)
}
```

17.5: Use 1000 steps of burnin and data gathering Markov chain Monte Carlo to estimate the mean of a draw from the density $\sin(x)\cos(x)^2\mathbb{I}(x \in [0, \tau/4])$. Replicate 10 times and report the estimate as $a \pm b$.

Solution Here is the chain running code:

```
chain_trig_pss <- function(steps) {
  x <- 0
  m_burnin <- replicate(steps, runif(4))
  for (i in 1:steps) {
    x <- step_trig_pss(x, m_burnin[,i])
  }
  m <- replicate(steps, runif(4))
  states <- rep(0, steps + 1)
  states[1] <- x
  for (i in 1:steps) {
    states[i + 1] <- step_trig_pss(states[i], m[,i])
  }
  return(states)
}
```

Here then is the replication:

```
res <- replicate(10, chain_trig_pss(1000) |> mean())
mean(res)
```

```
## [1] 0.6651807
```

```
sd(res) / sqrt(length(res))
```

```
## [1] 0.005680378
```

So the final estimate is $\boxed{0.665 \pm 0.006}$.

18.1: Consider the unnormalized density

$$g_X(s) = \exp(-s^{3/2})\mathbb{I}(s \geq 0).$$

a) Write code for a Metropolis-Hastings step with proposal move $Y = X + M$ where $M \sim \text{Unif}([-2, 1])$.

b) Use your code with 10 replications of 1000 burnin and data gathering steps to estimate $\mathbb{E}[X]$. Report your result as $a \pm b$.

Solution

a) This step can be taken as follows. Note that it is important to take the absolute value of a to make sure that raising to the $3/2$ power.

```
step_hastings_cont <- function(x, m, u) {
  y <- x + m
  r <- function(a, b)
    (a >= 0) * exp(-abs(a)^(3 / 2)) * (1 / 3) *
    (b >= a - 2) * (b <= a + 1)
  if (u < r(y, x) / r(x, y))
    return(y)
  else
    return(x)
}
```

b) Create the chain.

```
chain_hastings_cont <- function(steps) {
  burnin    <- steps
  datasteps <- steps
  x  <- 0
  m1 <- 3 * runif(burnin) - 2
  u1 <- runif(burnin)
  for (i in 1:burnin) {
    x <- step_hastings_cont(x, m1[i], u1[i])
  }
  res <- rep(0, datasteps)
  m2 <- 3 * runif(datasteps) - 2
  u2 <- runif(datasteps)
  for (i in 1:datasteps) {
    x <- step_hastings_cont(x, m2[i], u2[i])
```

```
    res[i] <- x
  }
  return(mean(res))
}
```

Now collect data.

```
set.seed(572727427)
```

```
res <- replicate(10, chain_hastings_cont(1000))
```

Finally, analyze the data.

```
tibble(
  est_mean = mean(res),
  est_sd   = sd(res) / sqrt(length(res))
)
```

```
## # A tibble: 1 x 2
##   est_mean est_sd
##      <dbl>  <dbl>
## 1    0.658 0.0215
```

In my run, the answer is $\boxed{0.657 \pm 0.021}$.

Chapter B

Probability review

B.1 Elementary facts

Combinatorics The number of ways to arrange n objects in order is n factorial:

$$n! = n(n-1)(n-2)\cdots 1,$$

where $0! = 1$. The number of ways to choose r objects from n objects is:

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}.$$

For $n_1 + n_2 + \ldots n_r = n$, the number of ways to choose n_1 objects of type 1, n_2 objects of type 2, up to n_r objects of type r, is

$$\binom{n}{n_1, n_2, \ldots, n_r} = \frac{n!}{n_1! n_2! \cdots n_r!}.$$

Definitions These are the basic definitions for talking about probability.

The set of outcomes is called the *sample space* or *outcome space*, and is usually denoted Ω.

An *event* is a subset E of Ω such that $\mathbb{P}(E)$ is defined (an event is also sometimes called a *measurable* subset). When A is an event, the complement of A is also an event. Also if $A_1, A_2, \ldots$ is a sequence of events, then $\cup_{i=1}^{\infty} A_i$ is also an event. (Any set of events with these properties is called a σ-algebra or σ-field.)

$\mathbb{P}$ is a function that given an event A, outputs the probability that the outcome lies in A.

The events A and B are *disjoint* or *mutually exclusive* if $A \cap B = \emptyset$.

Measures A probability is a special type of measure that obeys the following four rules:

1: For event B, $0 \leq \mathbb{P}(B)$ (probabilities are nonnegative real numbers)

2: $\mathbb{P}(\emptyset) = 0$ (the probability nothing happens is zero).

3: For $B_1, B_2, \ldots$ disjoint events,

$$\mathbb{P}\left(\cup_{i=1}^{\infty} B_i\right) = \sum_{i=1}^{\infty} \mathbb{P}(B_i).$$

4: $\mathbb{P}(\Omega) = 1$ (the probability that something occurs is 1).

Simple facts Some basic facts follow from these rules.

Prop: $0 \leq \mathbb{P}(A) \leq 1$.
Prop: $\mathbb{P}(A^C) = 1 - \mathbb{P}(A)$.
Prop: $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(AB)$
Prop: $\mathbb{P}(\emptyset) = 0$.

A word about intersection For sets A and B, the intersection of A an B can be denoted $A \cap B$, AB, or A, B. All of these notations mean the same thing:

$$A \cap B := \{x : x \in A \text{ and } x \in B\}.$$

Conditional probabilities

If $\mathbb{P}(B) > 0$, the conditional probability of A given B is

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(AB)}{\mathbb{P}(B)}.$$

Bayes' Formula If $F_1, \ldots, F_n$ are disjoint and $\cup_{i=1}^{n} F_i = \Omega$, then

$$\mathbb{P}(F_i|A) = \frac{\mathbb{P}(A|F_i)\mathbb{P}(F_i)}{\mathbb{P}(A|F_1)\mathbb{P}(F_1) + \ldots \mathbb{P}(A|F_n)\mathbb{P}(F_n)}.$$

Random variables A *random variable* is a function of the outcome. The values the random variable can take on are called *states*, and lie in the *state space*. In other words, a random variable is a function from the sample space to the state space.

For a discrete random variable $X \in \{x_1, x_2, x_3, \ldots\}$, the expected value of X is

$$\mathbb{E}[X] = \sum_{i=1}^{\infty} x_i \mathbb{P}(X = x_i).$$

For a continuous random variable $X \in \mathbb{R}$ with density f_X, the expected value of X is

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} s f_X(s)\, ds.$$

For any two random variables X and Y,

$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y].$$

For two random variables X and Y are *uncorrelated* if and only if

$$\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y].$$

Independent random variables (see below) are always uncorrelated, but uncorrelated random variables are not always independent!

Independence

Two events A and B are *independent* if

$$\mathbb{P}(AB) = \mathbb{P}(A)\mathbb{P}(B) \Leftrightarrow \mathbb{P}(A \mid B) = \mathbb{P}(A).$$

Two random variables X and Y are independent if for any event $X \in A$ and $Y \in B$,

$$\mathbb{P}(X \in A, Y \in B) = \mathbb{P}(X \in A)\mathbb{P}(Y \in B).$$

B.2 A short guide to solving probability problems

Equally likely outcomes. If all outcomes are equally likely,

$$\mathbb{P}(E) = \frac{\text{number of outcomes in } E}{\text{total number of outcomes}}.$$

Trick #1: Use complements. It is often easier to find $\mathbb{P}(A^C)$ then $\mathbb{P}(A)$, remember

$$\mathbb{P}(A) = 1 - \mathbb{P}(A^C).$$

Trick #2: Use independence to turn intersections into products. If we want the probability of the intersection of $A_1, \ldots, A_n$, then we can break it apart only when the events are independent:

$$\mathbb{P}(A_1 \cdots A_n) = \mathbb{P}(A_1)\mathbb{P}(A_2) \cdots \mathbb{P}(A_n).$$

Trick #3: Use disjointness to turn unions into sums. If the events $A_1, \ldots, A_n$ are disjoint,

$$\mathbb{P}(A_1 \cup \cdots \cup A_n) = \mathbb{P}(A_1) + \mathbb{P}(A_2) + \ldots \mathbb{P}(A_n).$$

Trick #4: Use Principle of In/Ex to deal with any union. We can *always* break apart unions of events $A_1 \ldots A_n$ using the Principle of Inclusion/Exclusion, which we use most often when $n = 2$:

$$\mathbb{P}(A_1 \cup A_2) = \mathbb{P}(A_1) + \mathbb{P}(A_2) - \mathbb{P}(A_1 A_2).$$

Its easier to say the Principle of Inclusion/Exclusion in words than symbols: the probability of any event occurring is the sum of the probabilities that one event occurs minus the sum of the probabilities that 2 events occur plus the sum of the probabilities that 3 events occur etcetera until we reach the probability that all events occur.

Trick #5: Use De Morgan's Laws to covert unions and intersections. Convert back and forth between union and intersection using De Morgan's Laws:

$$(A_1 A_2 \cdots A_n)^C = A_1^C \cup A_2^C \cdots \cup A_n^C,$$
$$(A_1 \cup A_2 \cup \cdots \cup A_n)^C = A_1^C A_2^C \cdots A_n^C.$$

Trick #6: Use Bayes' Formula to reverse conditional probabilities. If you know $\mathbb{P}(A \mid F_i)$ for all i as well as $\mathbb{P}(F_i)$, and want $\mathbb{P}(F_i \mid A)$, then use Bayes' Formula.

Trick #7: Acceptance/Rejection Method 1 Suppose that we perform a trial which if successful, has outcomes $A_1, \ldots, A_n$. If we fail, then we try again until one of A_1 through A_n occur. Then

$$\begin{aligned}\mathbb{P}(A_i \text{ occurs on final trial}) &= \mathbb{P}(A_i \text{ on first trial} \mid \text{first trial a success}) \\ &= \frac{\mathbb{P}(A_i \text{ on first trial})}{\mathbb{P}(\text{first trial a success})}.\end{aligned}$$

Trick #8: Acceptance/Rejection Method 2 The other way to tackle acceptance rejection problem is using infinite series. Remember, when $|r| < 1$,

$$\sum_{i=0}^{\infty} r^i = \frac{1}{1-r}.$$

Common errors Some things to watch out for! Events use complements, unions, and intersections. A statement like $\mathbb{P}(A)^C$ doesn't make sense, since $\mathbb{P}(A)$ is a number. What was probably meant was $\mathbb{P}(A^C)$. Similarly, use +, - and times for numbers like probabilities, and never for sets. We haven't defined A + B, what was probably intended was $\mathbb{P}(A) + \mathbb{P}(B)$.

Steps to a problem: If you don't know how to get started on a problem, the following steps usually can get you going:

(1) Write down the sample space. Even if you can't write down the whole sample space, write down some of the outcomes. Make up symbols, like H for head or T for tails or W for win and L for a loss to make writing outcomes easier.

(2) Write down the events that you are given probabilities for, and the event that you are trying to find the probability of (the target event).

(3) See if you can express the target event in terms of union, intersection, or complements of the events that you are given (here is where the five tricks come into play).

Simple checks on an answer: Make sure that your final probabilities lie between 0 and 1. If you know that a set of probabilities must add to 1, then check by actually adding them. If you have a simple intuitive reason to believe that A is more likely than B, check that $\mathbb{P}(A) > \mathbb{P}(B)$.

B.3 A short guide to counting

Order matters When order matters, then there are $n!$ ways to order n objects.

Thinking about n choose k. There are several ways of thinking about $\binom{n}{k}$, all of which are equivalent.

1: It's the number of the ways to choose a subset of size k from a set of size n.

2: It's the number of ways to order a group of letters $A \dots AB \dots B$ where A appears k times and B appears $n-k$ times.

3: Given n spaces, it's the number of ways to mark k of those spaces in some way.

4: It's the number of ways of choosing k out of n trials to be successful.

Multichoosing Now $\binom{n}{n_1,\dots,n_r}$ is similar, in that it generalizes $\binom{n}{k}$. This is because $\binom{n}{k} = \binom{n}{n-k}$. The number n multichoose $n_1, n_2, \dots, n_r$ counts the following.

(1) It's the number of the ways to choose a partition of a set of size n where the first subset has size n_1, the second n_2, etcetera.

(2) It's the number of ways to order a group of letters $A_1 \dots A_1 A_2 \dots A_2 \dots A_r \dots A_r$ where A_i appears n_i times.

(3) Given n spaces, it's the number of ways to mark n_1 of those spaces with a 1, n_2 spaces with a 2, up to n_r spaces with n_r.

(4) Suppose each trial has r different outcomes. Then its the number of ways of choosing n_1 trials to have outcome 1, n_2 trials to have outcome 2, up to n_r trials having outcome r.

When all else fails. Almost any problem can be written as a problem with ordering. If you are uncomfortable with n choose r or can't figure out what should be ordered and what shouldn't then give everything in your problem a number and order everything.

For example, what's the probability of choosing a given five card hand from a set of 52 cards? One way: number of outcomes is 1, total number of outcomes is $\binom{52}{5}$, so

$$\mathbb{P}(\text{hand}) = \frac{1}{\binom{52}{5}}.$$

Another way: number all the cards $1, \dots, 52$ and order them in any one of 52! ways. Then any outcome where the five cards we are interested in appear first in the ordering of cards works. There are 5! ways to order these cards and (52 - 5)! ways to order the remaining 47 cards, so the total number of outcomes is 5!(47!), so

$$\mathbb{P}(\text{hand}) = \frac{5!47!}{52!},$$

which is the same answer as the other way.

Another example: given a random ordering of the letters MIIIISSSSPP, what's the probability that it spells MISSISSIPPI? Think about numbering every symbol, so we are ordering $x_1x_2x_3x_4x_5x_6x_7x_8x_9x_{10}x_{11}$, where $x_1 = M$, x_2 through x_5 equal I, etc. Then the total number of outcomes is 11!. The number of outcomes that are successful? Well x_1 has to be in first position, x_2, x_3, x_4 and x_5 have to occupy positions $2, 5, 8$, and 10 (which they can do in 4! ways, there are 4! ways to order the x_i that equal S and 2! ways to order the x_i that equal P. So

$$\mathbb{P}(\text{MISSISSIPPI}) = \frac{1!4!4!2!}{11!}.$$

B.4 How to find $\mathbb{E}[X]$

Step 1 Find the values that X can take on with positive probability (this is called the *positive support* of X). If X is discrete, this will be either a finite number of values $\{x_1, \ldots, x_n\}$ or a countable number of values $\{x_1, x_2, \ldots\}$. If X is continuous, it could be an interval or union of intervals, like $(0, \infty)$ or $(3, 4) \cup [10, 15)$.

Step 2 Use the right formula. If X is discrete, then $\mathbb{E}[X]$ is the sum over all values of x such that $\mathbb{P}(X = x) > 0$ of the outcome times the probability. So if $X \in \{x_0, x_1, \ldots\}$, then

$$\mathbb{E}[X] = \sum_{x:p(x)>0} xp(x) = \sum_{i=1}^{\infty} x_i \mathbb{P}(X = x_i).$$

If X is continuous with density f_X then

$$\mathbb{E}[X] = \int_{\mathbb{R}} x f_X(x)\, dx.$$

If $X \in \{0, 1, 2, 3, \ldots\}$, then the *Tail Sum Formula* gives an alternate way to find the expected value:

$$\mathbb{E}[X] = \sum_{i=0}^{\infty} \mathbb{P}(X > i).$$

If X is continuous and $\mathbb{P}(X \geq 0) = 1$, then the Tail Sum Formula is

$$\mathbb{E}[X] = \int_{x=0}^{\infty} \mathbb{P}(X > x)\, dx.$$

Conditional expectation To find $\mathbb{E}[A|B]$, treat B as a constant and calculate the probability in the exact same way as above. For all random variables A and B:

$$\mathbb{E}[\mathbb{E}[A|B]] = \mathbb{E}[A].$$

Note: If we wish to find $\mathbb{E}[g(X)]$ then use

$$\mathbb{E}[g(X)] = \sum_{x:\mathbb{P}(X=x)>0} g(x)\mathbb{P}(X=x) = \sum_{i=1}^{\infty} g(x_i)\mathbb{P}(X=x_i),$$

and

$$\mathbb{E}[g(X)] = \int_{\mathbb{R}} g(s)f_X(s)ds.$$

For uncorrelated random variables, $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$. Independent random variables are uncorrelated, but uncorrelated random variables might not be independent.

Some properties of expected value:

- For any two random variables (correlated or uncorrelated) $\mathbb{E}[X+Y] = \mathbb{E}[X]+\mathbb{E}[Y]$.

B.5 How to find $\mathbb{V}(X)$

Method 1: Use

$$\mathbb{V}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2.$$

Method 2: Use

$$\mathbb{V}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2].$$

Some properties

- For uncorrelated random variables, $\mathbb{V}(X+Y) = \mathbb{V}(X) + \mathbb{V}(Y)$.
- For random variable X and constant $\alpha \in \mathbb{R}$, $\mathbb{V}(\alpha X) = \alpha^2\mathbb{V}(X)$, $\mathrm{SD}(\alpha X) = \alpha\,\mathrm{SD}(X)$.

B.6 Distributions

The *distribution* of a random variable is a complete listing of $\mathbb{P}(X \in A)$ for all sets A of interest. The distribution also referred to as the law of X, and denoted $\mathcal{L}(X)$. When X and Y have the same distribution, this is denoted

$$X \sim Y, \text{ or } \mathcal{L}(X) = \mathcal{L}(Y).$$

The *distribution function* of a random variable X (also known as the cumulative distribution function) is

$$F(a) = \mathbb{P}(X \le a).$$

This is a function that is bounded, that is, it always lies between 0 and 1. It is also right continuous, that is if $a_1, a_2, a_3, \ldots$ decrease and their limit is a, then limit of $F(a_1), F(a_2), \ldots$ equals $F(a)$.

Because of a theorem from measure theory called the Carathéodory Extension Theorem, knowing F allows computation of $\mathbb{P}(X \in A)$ for any A of interest. In particular, if $A = (a, b]$, then $\mathbb{P}(X \in A) = F(b) - F(a)$. (Looks a bit like the fundamental theorem of

calculus, which is one reason why F is always capitalized when used for the distribution function.)

More precisely, if F_X is the distribution function of X and F_Y is the distribution function of Y, then

$$\mathcal{L}(X) = \mathcal{L}(Y) \iff F_X(a) = F_Y(a) \; \forall a.$$

If X is discrete then the graph of $F(a)$ will have jumps, if X is continuous then $F(a)$ will be continuous. Some more formulas that come in handy:

$$\begin{aligned}
\mathbb{P}(a < X \leq b) &= F(b) - F(a) \\
\mathbb{P}(a < X < b) &= F(b) - F(a) - \mathbb{P}(X = b) \\
\mathbb{P}(a \leq X < b) &= F(b) - F(a) - \mathbb{P}(X = b) + \mathbb{P}(X = a) \\
\mathbb{P}(a \leq X \leq b) &= F(b) - F(a) + \mathbb{P}(X = a).
\end{aligned}$$

Remember that for continuous random variables $\mathbb{P}(X = s) = 0$ for any s, so the right hand side of these formula just becomes $F(b) - F(a)$. Also for continuous X,

$$f(a) = \frac{dF(a)}{da}$$

and

$$F(a) = \int_{-\infty}^{a} f(a)da,$$

where $f(x)$ is the *probability density function* (sometimes just called the density) of X.

Finally, say that $X_1, X_2, \ldots$ are independent identically distributed, or iid, if they are independent and all have the same distribution.

B.7 Discrete distributions

A random variable is *discrete* if it only takes on a finite or countably infinite number of values. The distribution of a discrete random variable is also called discrete in this instance.

Uniform Written: $\mathsf{Unif}(\{1, \ldots, n\})$. The story: roll a fair die with n sides.

$$\begin{aligned}
\mathbb{P}(X = i) &= \frac{1}{n}\mathbb{I}(i \in \{1, \ldots, n\}) \\
\mathbb{E}[X] &= \frac{n+1}{2} \\
\mathbb{V}(X) &= \frac{(n-1)(n+1)}{12}
\end{aligned}$$

Bernoulli Written: $\mathsf{Bern}(p)$. The story: flip a coin that comes up heads with probability p, and count the number of heads on the single coin flip. Also, the number of successes in a single trial where the trial is a success with probability p.

$$\begin{aligned}\mathbb{P}(X = 1) = p,\ \ \mathbb{P}(X = 0) &= 1 - p\\ \mathbb{E}[X] &= p\\ \mathbb{V}(X) &= p(1-p).\end{aligned}$$

Binomial Written: $\mathsf{Bin}(n, p)$. The story: flip iid coins n times where the probability of heads is p and count the number of heads. Also, the number of successes in a single trial where the trial is a success with probability p. Also if $X_1, \ldots, X_n$ are iid $\mathsf{Bern}(p)$, then $X = X_1 + X_2 + \ldots X_n \sim \mathsf{Bin}(n, p)$.

$$\begin{aligned}\mathbb{P}(X = i) &= \binom{n}{i} p^i (1-p)^{n-i} \mathbb{I}(i \in \{\{0, \ldots, n\})\\ \mathbb{E}[X] &= np\\ \mathbb{V}(X) &= np(1-p).\end{aligned}$$

Geometric Written: $\mathsf{Geo}(p)$. The story: flip iid coins with probability p of heads and counting the number of flips needed for one head. Also, the number of trials needed for 1 success when the probability of success at each trial is p and each trial is independent.

$$\begin{aligned}\mathbb{P}(X = i) &= (1-p)^{i-1} p \mathbb{I}(\{0, 1, \ldots\})\\ \mathbb{E}[X] &= \frac{1}{p}\\ \mathbb{V}(X) &= \frac{1-p}{p^2}.\end{aligned}$$

Negative Binomial Written: $\mathsf{NegBin}(r, p)$. The story: flipping iid coins with probability p of heads and counting the number of flips needed for r heads to arrive. Also, the number of trials needed for r successes when the probability of success at each trial is p and each trial is independent.

Also $X = X_1 + X_2 + \ldots X_r$, where X_i are iid and distributed as $\mathsf{Geo}(p)$.

$$\begin{aligned}\mathbb{P}(X = i) &= \binom{i-1}{r-1} p^r (1-p)^{i-r} \mathbb{I}(\{0, 1, \ldots\})\\ \mathbb{E}[X] &= \frac{r}{p}\\ \mathbb{V}(X) &= r\frac{1-p}{p^2}.\end{aligned}$$

Hypergeometric Written: Hypergeo(N, m, n). The story: drawing n balls from an urn holding m green balls and $N - m$ blue balls and counting the number of green balls chosen.

$$\begin{aligned}\mathbb{P}(X = i) &= \frac{\binom{m}{i}\binom{N-m}{n-i}}{\binom{N}{n}}\mathbb{I}(\{0, 1, \ldots, n\}) \\ \mathbb{E}[X] &= \frac{nm}{N} \\ \mathbb{V}(X) &= \frac{N-n}{N-1}np(1-p).\end{aligned}$$

Zeta Written: Zeta(α). A.k.a. Zipf or power law. The story: things like city sizes and incomes have Zeta distributions.

$$\begin{aligned}\mathbb{P}(X = i) &= \frac{C}{i^{\alpha+1}}\mathbb{I}(\{1, 2, \ldots\}) \\ \mathbb{E}[X] &= \text{no closed form} \\ \mathbb{V}(X) &= \text{no closed form.}\end{aligned}$$

Special notes: Except for special values of α like 1, we do not have a closed form solution for the value of C, the normalizing constant. Choose C so that $\sum_{i=1}^{\infty}\mathbb{P}(X = i) = 1$. Similarly, there are no closed form solutions for $\mathbb{E}[X]$ or $\mathbb{V}(X)$. These must be evaluated numerically. When $\alpha < 1$, $\mathbb{E}[X]$ does not exist (or is considered infinite). Similarly, when $\alpha < 2$, $\text{Var}(X)$ does not exist (or can be considered infinite).

Poisson Written: Pois(μ). The story: given that the chance of an arrival in time t to $t + dt$ is $\lambda\, dt$, and $\mu = \lambda T$, then this is the number of arrivals in the interval $[0, T]$. $X_1, X_2, \ldots$, it is

$$\max_i X_1 + X_2 + \ldots + X_i < 1.$$

$$\begin{aligned}\mathbb{P}(X = i) &= e^{-\mu}\frac{\mu^i}{i!}\mathbb{I}(\{0, 1, \ldots\}) \\ \mathbb{E}[X] &= \mu \\ \mathbb{V}(X) &= \mu.\end{aligned}$$

B.8 Continuous Distributions

A random variable is *continuous* if $\mathbb{P}(X = a) = 0$ for all a. The distribution of a continuous random variable is also called continuous.

Uniform (continuous) Written: Unif(A). The story: a point is uniform over A if for all $B \subseteq A$, the chance the point falls in B is the Lebesgue measure of B divided by the Lebesgue measure of A. For general A:

$$f(x) = \frac{1}{\text{Lebesgue measure of } A}\mathbb{I}(x \in A)$$

When $A = [a, b]$, more specifically:

$$\begin{aligned} f(x) &= \frac{1}{b-a}\mathbb{I}(x \in (a, b]) \\ F(x) &= \frac{x-a}{b-a}\mathbb{I}(x \in [a, b]) + \mathbb{I}(x > b) \\ \mathbb{E}[X] &= \frac{b+a}{2} \\ \mathbb{V}(X) &= \frac{(b-a)^2}{12} \end{aligned}$$

Normal Written: $\mathsf{N}(\mu, \sigma^2)$. The story: when you sum variables with finite mean and standard deviation together, they are well approximated by a normal distribution.

$$\begin{aligned} f(x) &= \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \\ F(x) &= \Phi\left(\frac{x-\mu}{\sigma}\right) \\ \mathbb{E}[X] &= \mu \\ \mathbb{V}(X) &= \sigma^2 \end{aligned}$$

Addition of normals. Adding independent normal random variables gives back another normal random variable. If $X_i \sim \mathsf{N}(\mu_i, \sigma_i^2)$, and $X = X_1 + X_2 + \ldots + X_n$, then

$$X \sim \mathsf{N}\left(\sum_i \mu_i, \sum_i \sigma_i^2\right).$$

For X, Y independent $N(0, 1)$ random variables, the joint distribution of (X, Y) is rotationally invariant.

Normal random variables are symmetric around μ, and so $\Phi(x) = 1 - \Phi(-x)$.

Exponential Written: Exp(λ). What it is: when events occur continuously over time at rate λ, this is the time you have to wait for the first event to occur.

$$f(t) = \lambda e^{-\lambda t} \mathbb{I}(t \in (0, \infty))$$
$$F(t) = \begin{cases} 1 - e^{-\lambda t} & a \geq 0 \\ 0 & a < 0 \end{cases}$$
$$\mathbb{E}[X] = \frac{1}{\lambda}$$
$$\mathbb{V}(X) = \frac{1}{\lambda^2}$$

B.9 How to use the Central Limit Theorem (CLT)

The CLT says that if $X_1, X_2, \ldots$ are identically distributed random variables and $Z_n = X_1 + \ldots X_n$, then

$$\lim_{n \to \infty} \mathbb{P}\left(\frac{Z_n - \mathbb{E}[Z_n]}{\sqrt{\mathbb{V}(Z)}} \leq a \right) = \Phi(a).$$

We use it as an approximation tool for $Z = X_1 + \ldots X_n$:

$$\mathbb{P}\left(\frac{Z - \mathbb{E}[Z]}{\sqrt{\mathbb{V}(Z)}} \leq a \right) \approx \Phi(a).$$

Often we are interested in approximating the probability of things like $\mathbb{P}(Z \leq b)$ where $Z = X_1 + \ldots X_n$. This takes two steps.

Step 1 If Z is integral, apply the half integer correction. So instead of $\mathbb{P}(Z \leq i)$ we write $\mathbb{P}(Z \leq i + 1/2)$.

Step 2 Subtract off $\mathbb{E}[Z]$ and divide by the square root of $\text{Var}(Z)$. So

$$\mathbb{P}(Z \leq b + 0.5) = \mathbb{P}\left(\frac{Z - \mathbb{E}[Z]}{\sqrt{\mathbb{V}(Z)}} \leq \frac{b + 0.5 - \mathbb{E}[Z]}{\sqrt{\mathbb{V}(Z)}} \right).$$

Step 3 Apply the CLT and say

$$\mathbb{P}(Z \leq b) \approx \Phi\left(\frac{b + 0.5 - \mathbb{E}[Z]}{\sqrt{\mathbb{V}(Z)}} \right).$$

Bibliography

[1] J. von Neumann. Zur theorie der gesellschaftsspiele. *Math. Ann.*, 100:295–320, 1928. 9.1