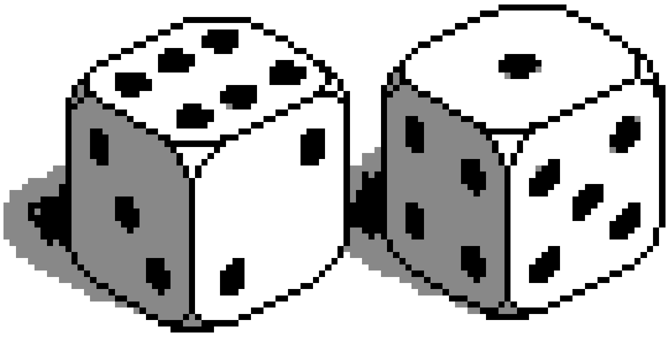
Monte Carlo Inference

Course Notes

Prepared by Steve Brooks and Ruth King



UNIVERSITY OF CAMBRIDGE
Statistical Laboratory

First published 1996.
Second edition published 1998.
Third edition published 1999.
Fourth edition published 2001.
Fifth edition published 2002.
Sixth edition published 2003.

Copyright © Steve Brooks and Ruth King, 2003.

All rights reserved. No part of this work may be translated, reproduced, stored or transmitted, in any form or by any means, electronic, mechanical, or by similar or dissimilar methodology now known or hereafter developed, without prior, written permission of the authors.

The moral right of the authors has been asserted.

Printed in the Statistical Laboratory, Cambridge University.

CONTENTS

Chapter 1

INTRODUCTION

1.1 Scope of the Course

This course focusses on the use of stochastic simulation techniques for statistical inference. Most of the techniques under this heading are also often referred to as "computationally intensive methods" referring to the extent to which these methods rely on substantial computing power. An initial classification of Monte Carlo methods comprises three distinct areas:

1. Graphical data exploration.
2. Stochastic modelling.
3. Statistical inference.

There is obviously some overlap in these three but, in this course, we will focus mainly on the third area. That is, we shall aim at understanding computational techniques which require innovative algorithms to apply standard inferences.

However, for classical and particularly Bayesian inference, areas 2 and 3 have become so closely linked that it makes sense to consider the two together. Hence in Chapter 4 we will consider modern approaches to Bayesian inference and their associated computational techniques and in Chapter 5 we will consider modern approaches to classical inference via numerical optimisation techniques. On the other hand, some techniques used for computationally intensive calculations such as the bootstrap (Chapter 3) are intrinsically non-parametric and, as such, the numerical techniques can be treated totally in isolation of modelling considerations.

There are two main aims to this course. The first is to gain some understanding and knowledge of the techniques and tools which are available (given sufficient computing power). The second is to illustrate the interplay between probability and statistics in providing clever methods for modern statistical inference. These techniques are not merely a means to an end, but have intrinsic value in their own right as exercises in probability and statistics.

The course is divided into four distinct chapters. We begin with an introduction to random variable generation techniques beginning with the simulation of Uniform random numbers and using these to simulate from more complex distributions. In Chapter 3 we take a look at nonparametric inference as well as several techniques for improving the performance of Monte Carlo simulation algorithms. In Chapter 4 we take a look at Bayesian inference and introduce the Markov chain Monte Carlo algorithm for computing complex integrals. Finally, in Chapter 5 we look at the familiar classical problem of obtaining maximum likelihood estimates via Monte Carlo simulation and examine the links between some of these techniques and those used in the Bayesian context.

This is not a course on computing itself and we will not get into the details of computer programming. Nor will we deal with specialised computer packages often used in statistical computing. All of the examples will be implemented using simple S-Plus functions - far from the most efficient way of implementing the different techniques. However, the emphasis of the course is to illustrate the various methods and their application on relatively simple examples.

1.2 Course Texts

There is no single textbook which covers all aspects of the course, but the following list of texts (which is not exhaustive as new books appear at almost monthly intervals) may be useful.

1. Efron, B. and R.J. Tibshirani, *An introduction to the bootstrap*, New York: Chapman and Hall (1993).
2. Gamerman, D. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*, London: Chapman and Hall (1997).
3. Gentle, J.E., *Random number generation and Monte Carlo methods*, New York: Springer (1998).
4. Lange, K. *Numerical Analysis for Statisticians*, New York: Springer (1998).
5. Manly, B.F.J. *Randomization, bootstrap and Monte Carlo methods in biology*, London: Chapman and Hall (1998).
6. Ripley, B.D., *Stochastic Simulation*, London: Wiley (1987).
7. Robert, C.P. and G. Casella, *Monte Carlo statistical methods*, New York: Springer (1999).

1.3 Acknowledgments

Similar courses to this are now given at most universities and we would like to acknowledge the fact that some of the material in this course is based upon material written by other people, most notably Stuart Coles at Bristol and Gareth Roberts at Lancaster.

Chapter 2

SIMULATION

2.1 Introduction

In this chapter, we will look at different techniques for simulating from distributions and stochastic processes. Unlike in subsequent chapters, limited use will be made of applications, but suffice it to say that the applications of these simulation techniques are as varied as the subject of statistics itself. In any situation where we have a statistical model, simulating from that model generates realisations which can be analysed as a means of understanding the properties of that model. In subsequent chapters, we will also see how simulation can be used as a basic ingredient for a variety of approaches to statistical inference. However, within this chapter we shall deal only with simulation techniques, leaving the applications for later. We begin by describing what we mean by simulation.

To simulate: *"feign, pretend, to have or feel, pretend to be, act like, resemble, mimic"*.

In its technical sense, simulation involves using a *model* to produce results, rather than experiment with the real system under study (which may not yet exist). For example, to simulate the spread of an epidemic, we set up a mathematical model for the spread (probability of catching the disease etc,...) and then use the model to predict the spread over time.

If the model has a random or "chance" element, we call this a *stochastic simulation*, otherwise it is *deterministic*. Here, we are concerned with the former. Examples of models which may be investigated by simulation include epidemics, stock control and queueing problems. For example, what is the best queueing strategy in a bank, if we want to minimise queueing times? – One long queue, or several shorter queues, with one at each till.

To run a stochastic simulation, we need some *source of randomness*. For example, in epidemics does the individual catch the disease or not? In queues, how long does each individual spend at the cashiers window? How often do new people join the queue(s)?

So, what possible sources of randomness are there?

2.1.1 Simple Physical Experiments

Examples of simple physical experiments include tossing coins, throwing dice etc,... If we wanted to generate a digit from 0 to 9 with equal probability for each, we might

(i) Throw a regular 10-sided cylinder, or

(ii) Use coins. – Toss a coin four times. We count a head as a 0 and a tail as 1, the result is four digits *abcd*, eg; 0110 (written in order), which we can convert from binary to get the corresponding number in decimal notation. In this case, we get

$$(0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 6.$$

If the resulting number is greater than 9, we ignore it and start again.

The resulting number is a realisation of a uniformly distributed random digit over the range $[0, 9]$. However, this method is rather wasteful, since digits are rejected $6/16 = 3/8$ of the time.

Exercise: Suppose we have two 6-sided dice, how could these be used to generate random digits in that range?

Unfortunately, these experimental methods are impractical for all but the simplest simulations.

2.1.2 The Table Look-Up Method

Random Number Tables (see page 5) are available, which are long sequences of the digits $0 - 9$. A sequence of random digits can be obtained by reading the table by rows, by columns, or any other rule.

For example, suppose that we wish to choose between three alternatives, each with equal probability. How can we use the tables?

STEP 1. START AT A "RANDOM" PLACE ON THE TABLE.
STEP 2. LOOK AT THE NEXT DIGIT.

IF 0. THEN GO ON TO THE NEXT DIGIT. (PROBABILITY 1/10).

IF 1–3, THEN TAKE THE 1ST ALTERNATIVE. (PROBABILITY 3/10).

IF 4–6, THEN TAKE THE 2ND ALTERNATIVE. (PROBABILITY 3/10).

IF 7–9, THEN TAKE THE 3RD ALTERNATIVE. (PROBABILITY 3/10).

2.1.3 Use of Computers

Most useful simulations will need to be performed using computers because;

(a) The processes being studied are usually complicated (otherwise there would be no need to simulate!).

(b) Many trials of the process may be needed, since the simulations are stochastic and outcomes will "vary". We will want to know the extent and nature of this variation.

Simple physical experiments and tables are largely inadequate for lengthy computer simulations, so we would like to be able to generate random sequences within the computer program. The problem with this is that computer programs, running on correctly functioning computers, should give deterministic results. Thus the best that we can do is to *imitate* the behaviour of a random sequence by a deterministic one.

54734	90733	38578	46580	30208	86763	93892	79629	37883	45842
08643	53397	07688	75238	57518	90149	33608	98778	84978	18119
98367	26705	26926	31821	78484	38958	26659	48993	53792	61056
89687	36805	67466	29267	25703	95680	17921	65947	77860	73107
24459	22334	77828	57664	73201	33119	52002	45551	74607	97448
72706	78091	74682	01805	97922	06348	19275	19109	63153	60825
48227	75998	54685	73141	21945	22121	95653	82804	67143	12104
21160	62366	52232	46862	18928	80366	56537	63682	28510	32359
53073	18901	10179	70193	30624	51270	68290	23781	00329	59029
81603	98358	49901	60723	85470	99404	12294	25720	15005	17741
11907	10750	04180	66978	05864	72761	45798	41923	85041	65173
26475	51224	02057	91980	12379	22828	36880	39999	29479	49723
76905	82745	93962	76673	69173	98201	53719	43861	56460	01182
01592	68057	11804	32403	77852	40008	75291	69753	03053	47902
68109	88399	84166	73100	83489	64863	33680	43345	56240	66367
40525	23036	77227	31730	14377	78434	97931	56211	50818	11861
41497	71101	17147	35211	94210	14808	95085	83420	76381	52966
78297	66135	48913	85138	94369	20929	96617	72576	96669	64416
69979	13346	93983	72188	79921	99574	33969	44921	03242	82572
61047	42132	95155	53509	04321	48942	50344	96749	21040	24013
99821	41132	15988	34831	01296	50283	60474	41846	49912	15130
25428	83667	76150	95545	32404	50040	82507	51930	59726	69254
30817	80015	76101	17960	92030	19177	58204	49680	82427	62882
53868	90005	73729	04312	13235	50808	51129	81237	00580	58747
44053	35939	19080	18652	55923	63279	06403	05305	77964	24703
35639	04555	39110	92811	64256	57899	67431	11795	91794	39710
66766	34076	73741	76665	99350	76346	12209	59880	77489	09792
12675	39268	42902	84035	49612	66647	24914	42940	15273	33545
49928	77487	37552	87515	58961	71280	88857	65785	94707	37880
33819	31751	51338	10561	27940	62697	72819	48688	79995	01782
73522	08223	41452	90938	35263	41885	77797	80714	54109	15348
95744	60370	08656	67121	79485	27999	23190	68623	71820	88588
37641	05226	35572	39106	09503	01162	58063	93137	91891	26044
52246	63443	14358	50595	99913	66538	86022	75271	33704	76207
16431	84167	76388	68010	82657	11397	96100	91408	65848	20295
18015	65292	50710	59022	93793	08804	98475	08917	75559	08620
87622	01840	90075	00923	16868	79336	36973	53074	42114	94708
36137	43582	32790	10823	55204	97098	98299	93139	77640	96709
71710	05377	66729	03180	12331	76756	25917	26070	35496	15246
29252	51928	91280	22580	32754	58671	19604	00113	69046	81223

Table. 2.1: Random Number Table

So, what do we mean by randomness? Tables of random numbers in which one page consisted entirely of 3's would not generally be thought of as being random, though the probability that this occurs is as likely as that of any other sequence of digits on a page of that size.

> "A vague notion embodying the idea of a sequence in which each term is unpredictable to the uninitiated, and whose digits pass a certain number of tests traditional with mathematicians and depending somewhat on the use to which the sequence is to be put."

Lehmer, 1951

The sequence of numbers generated by the computer are designed to embody, as far as possible, the features of "randomness" (or unpredictability) mentioned above. We call such numbers *pseudo-random* numbers.

2.1.4 A Simple Simulation Experiment

The generation of pseudo-random numbers is the foundation-stone of much of the work that we shall discuss in this course. In particular we shall show how we can generate observations from any distribution, no matter how complex, given a sufficient supply of pseudo-random numbers.

Before we discuss how such numbers can be generated on the computer, we first introduce a simple simulation experiment which, intrinsically, has nothing to do with computers at all. However, it highlights the general principle of *designing* simulation experiments to achieve optimal accuracy in the sense of minimising statistical variability.

Perhaps the most famous simulation experiment is that of Buffon's needle, designed to calculate (fairly inefficiently) an estimate of π. Buffon's original experiment is as follows. Imagine a grid of parallel lines spaced a distance d apart on a flat surface, and that we drop a needle of length l (with $l \leq d$) onto the surface. We repeat this experiment n times and count R, the number of times the needle touches one of the lines. Then, for large n,

$$\frac{2ln}{dR} \approx \pi.$$

See Exercise 2.1.

If we let p denote the probability that the needle crosses one of the lines, then $p = 2l/\pi d$. Now, let $\phi = 1/\pi$, then we might estimate π, by finding some estimate $\hat{\phi}$ of ϕ. Suppose that we estimate this probability by $\hat{p}$ by experiment, then R, the number of "successes" follows a binomial distribution, with $R \sim Bin(n,p)$. Thus $Var(\hat{p}) = p(1-p)/n$. Now, since $\hat{\phi} = \hat{p}d/2l$, $Var(\hat{\phi}) = (d\pi/2l - 1)/\pi^2$, which is minimised (subject to $l \leq d$) by setting $l = d$. Hence, the optimal needle length is the exact same length as the distance between successive lines.

We can use the S-Plus package to simulate Buffon's experiment, via the following S-Plus function.

```
buffon <- function(n) {
     # Function to simulate Buffon's needle experiment.
     # Here, we set d=l=1
     r <- NULL
     r[1] <- 0
     x <- runif(n,0,1)   # x is a vector of n Uniform(0,1)'s
     theta <- runif(n,0,pi)  # theta is a vector of n Uniform(0,pi)'s
     t <- sin(theta)
```

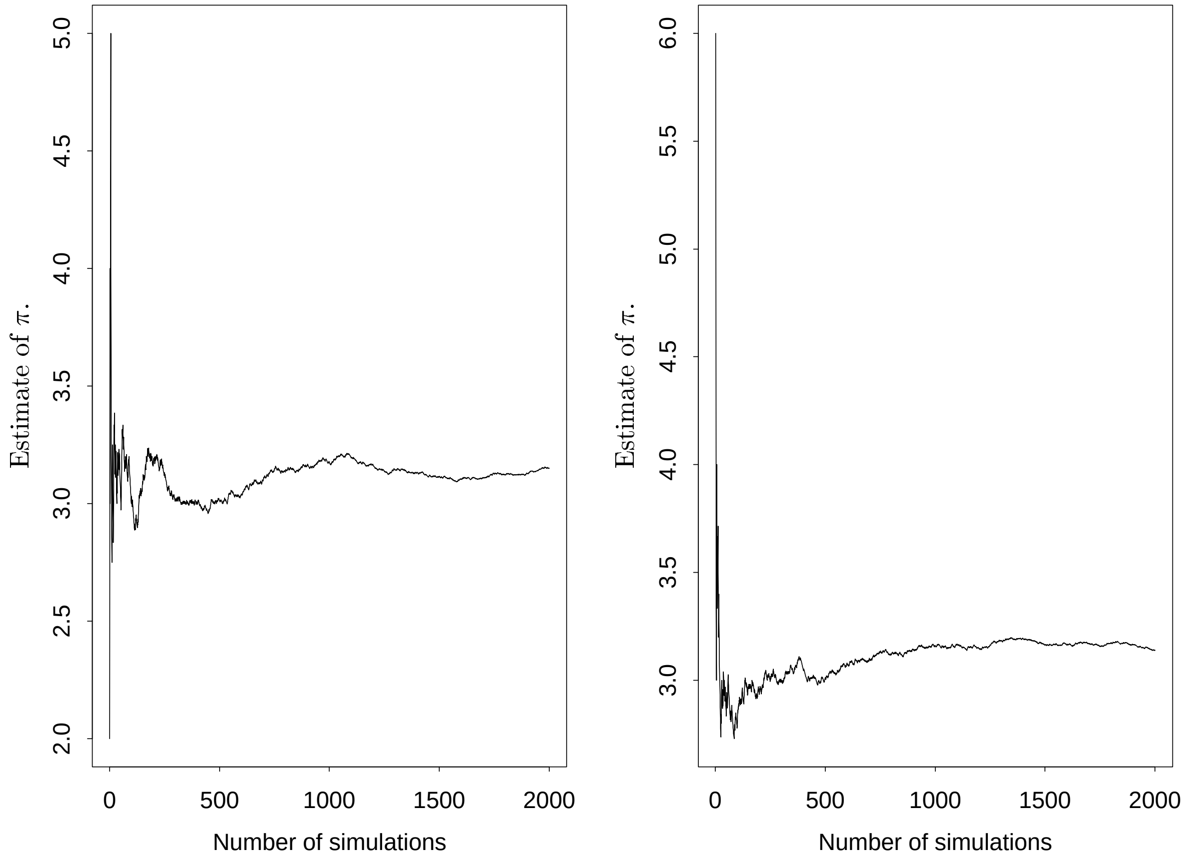


Figure. 2.1: Two sequences of realisations from Buffon's needle experiment.

```
    for(i in 1:n) {     # set up a loop over i-values

        if(t[i] > x[i] )       # needle touches line
           r[i+1] <- r[i] + 1  # increase R

        else r[i+1] <- r[i]    # needle misses line
    }

    r                   # lists the r-values throughout the simulation

    plot(1:n, 2*(1:n/r[2:(n+1)]),type="l",
         xlab="Number of simulations", ylab="Estimate of Pi")
}
```

2.2 Pseudo-Random Numbers

All of the simulation methods that we will discuss derive their operational substance from an infinite supply of random numbers, i.e., an independent sequence $\{U_i\}$ of random variables uniformly distributed on $[0,1]$.

Definition 2.1: *A sequence of pseudo-random numbers $\{U_i\}$ is a deterministic sequence of numbers in $[0,1]$ having the same relevant statistical properties as a similar sequence of random numbers.*

Thus, our first aim is to generate a sequence a numbers deterministically, which "embody" the features of randomness as far as possible.

In general, we use a recurrence relation of the form

$$X_n = f_n(X_{n-1},...,X_0) \quad n = 1,2,3,...$$

or, more usually, $X_n = f_n(X_{n-1},...,X_{n-r})$, for some r. This is less computationally expensive than taking $r = n$ and, in fact, $r = 1$ works quite well! We call X_0 the *seed* of the sequence. The seed is very important, since it is this that determines the entire sequence.

Example

Suppose that we require a set of random integer observations, generated from a Uniform distribution in the range $[0, M-1]$, i.e., $\mathbb{P}(X = i) = 1/M \ \forall i = 0,...,M-1$. If we let $M = 100$, then we *might* define $f(X)$ to be the nearest integer to $(100 \times$ fractional part of $\log_e)$. Then, if $X_0 = 82$, $\log_e(X_0) = 4.406$, so that $X_1 = 41$. Similarly, $X_2 = 71$ and $X_3 = 26$, but then we find that $X_i = 26 \ \forall i \geq 3$.
Similarly, if we set $X_0 = 39$, then $X_{24} = X_{48} = ... = 39$, and we only have 24 different values!

There are various problems with "ad-hoc" functions:

1. They *always* lead to periodic sequences. Once a number is repeated, a cycle starts. Note that the period is always less than M. Why?

2. They rarely "cover" the underlying parameter space well. For example, only four of the cycle of 24 above are greater than 75, and none lie between 77 and 94.

3. The choice of starting value may be crucial.

Therefore, the function f must be chosen very carefully, so as to ensure that these undesirable properties are minimised.

Additionally, and in practice, it seems reasonable (and sufficient) to insist that the joint distributions of a subsequence of pseudo-random numbers $(U_{i+1},...,U_{i+k})$ are not far from uniformity in $[0,1]^k$ for values of k which are small, say $k \leq 6$, i.e., for any continuous function f defined on $[0,1]^k$,

$$\frac{1}{n}\sum_{i=1}^{n} f(U_{i+1},...,U_{i+k}) \approx \mathbb{E}[f(v_1,...,v_k)]$$

where the v_i really are independent uniform random variables on $[0,1]$.

However, sequences produced by pseudo-random generators will inevitably have some structure, and there is a very great industry devoted to identifying and assessing the consequences of that structure. For example, significance tests of independence or uniformity can be applied, to the output $\{U_1,...,U_n\}$ of pseudo-random number generators. Such tests include: tests for independence, tests for uniformity and tests for k-tuples. For the latter, the chi-squared test is applied, to test for uniformity of k-tuples $\{(U_{ki}, ..., U_{ki+k-1})\}$, by dividing $[0,1]^k$ into a small number of regions. Note that this test cannot be applied to $\{(U_i, ..., U_{i+k-1})\}$, since these k-tuples are not independent. There are two commonly used types of pseudo-random number generators, and we shall now discuss these in turn.

2.2.1 Congruential Generators

These are defined by

$$X_i = (aX_{i-1} + c) \bmod M \tag{2.1}$$

for a multiplier a, shift c, and modulus M, where a, c and M are all integers, and we take a and c to be in the range $[0, M-1]$. The pseudo-random sequence $\{U_i\}$ is then determined by (2.1) and constrained to between 0 and 1, by setting

$$U_i = \frac{X_i}{M}.$$

Generators with $c = 0$ are called multiplicative. The future of X_i is determined by its current value. Since the $M+1$ values $X_0, ..., X_M$ cannot be distinct, at least one value must occur twice, X_i and X_{i+k}, say. Then $X_i, ..., X_{i+k-1}$ will be repeated as $X_{i+k}, ..., X_{i+2k-1}$, and so the sequence $\{X_i\}$ is periodic with period $k \leq M$. The full period M can always be achieved with $a = c = 1$, though the resulting sequence is not particularly random-looking. In general, the period depends upon the choice of a and c and perhaps the seed. For $c > 0$, the maximum possible cycle length M is obtained if and only if

1. c and M have no common factors (except c).
2. $(a-1)$ is a multiple of every prime number that divides M.
3. $(a-1)$ is a multiple of 4 if M is a multiple of 4.

The proof of this comes from number theory and is not in the remit of this course (see for example Ripley 1987).

For multiplicative generators, the maximal period is $M-1$, because if 0 ever occurs then it is repeated indefinitely. Usually M is chosen to make the modulus operation efficient, and then a and c, chosen to make the period as long as possible.

The NAG Fortran Library $U[0,1]$ generator G05CAF has $M = 2^{59}$, $a = 13^{13}$, $c = 0$ and good lattice properties. However, if NAG is not available, the following is commonly recommended

$$X_i = (69069X_{i-1} + 1) \bmod 2^{32}, \qquad U_i = 2^{-32}X_i.$$

Remark: it is always helpful to have two or more generators available and to run important simulations using each, to reduce the likelihood that anomalous results are due to quirks of the generator.

2.2.2 Shift-Register Generators

Cryptographers are more concerned with pseudo-random sequences of bits (0's and 1's) and the use of special hardware. Thus they have tended to prefer pseudo-random generators based upon shift-registers, which record the last d bits, $b_{i-1}, ..., b_{i-d}$ of the sequence. Then

$$b_i = f(b_{i-1}, ..., b_{i-d})$$

for some function $f : [0,1]^d \longrightarrow [0,1]$. The usual choice is of the form

$$b_i = (a_1 b_{i-1} + ... + a_d b_{i-d}) \bmod 2 \tag{2.2}$$

for binary constants $a_1, ..., a_d$. This is particularly easy to implement on a hardware circuit by use of a "shift-register", hence the name.

There are many ways to obtain pseudo-random numbers from a sequence of bits, the simplest is

$$U_i = \frac{b_{iL}}{2} + \frac{b_{iL+1}}{4} + ... + \frac{b_{iL+M}}{2^{M+1}}$$

for integers L and M, with $0 < M < L$.

Tausworthe suggested an alternative form of (2.2) given by

$$b_i = (b_{i-p} + b_{i-(p-q)}) \bmod 2$$

then

$$U_i = \sum_{s=1}^{L} 2^{-s} b_{it+s} \qquad t \in \mathbb{Z}.$$

Such generators are called Tausworthe generators. Many PC's use these kinds of generators, as does the BBC microcomputer, which takes $p = 33$, $q = 13$ and $t = L = 32$, for which the period is maximal and equals $2^{32} - 1$.

2.3 Basic Random Variable Generation

Of more interest, perhaps, is the problem of the generation of independent random variables $X_1, X_2, ...$ with a given distribution function, F, or probability density function, f. The aim here is to generate independent random variables $X_1, X_2, ...$ with a given cumulative distribution function (cdf) F, or probability density function (pdf) f. Assume that we have access to $\{U_i\}$, an infinite supply of independent $U[0,1]$ (pseudo) random variates. We want to transform the U_i into X_i, with distribution f. Usually there are many choices of algorithms for generating the X_i from U_i for specific distributions.

2.3.1 Generating from Specific Distributions

Normal Distribution

To sample from the normal distribution, take

$$X = \sum_{i=1}^{12} U_i - 6.$$

This has mean 0, variance 1 and is approximately normally distributed, by the central limit theorem (CLT). The approximation is fairly good, but by no means exact.

An alternative (exact) method is the Box-Muller method, which is defined by the following algorithm.

1 Generate U_1, and set $\theta = 2\pi U_1$.

2 Generate U_2, and set $E = -\log U_2$ and $R = \sqrt{2E}$.

3 Set $X = R\cos\theta$ and $Y = R\sin\theta$.

Then X and Y are independent $N(0,1)$ random variates.
Proof: See Exercise 2.6, and the Ratio of Uniforms Method, Section 2.3.6.

Exponential Distribution

If we take a uniform random variable U, and $\lambda \in \mathbb{R}$, then

$$X = -\lambda^{-1} \log U$$

has an exponential(λ) distribution.
Check.

$$\begin{aligned} \mathbb{P}(X \leq x) &= \mathbb{P}(U \geq \exp(-\lambda x)) \\ &= 1 - \exp(-\lambda x). \end{aligned}$$

This is known as the method of Inversion.

There are hundreds of specific algorithms for generating particular distributions. Most procedures are based upon a small number of general principles, we shall now introduce a few of these.

2.3.2 Discrete Distributions

Given a discrete distribution, $p_X = p_X(x) = \mathbb{P}(X = x)$, how would we simulate an *iid* sequence of random variables $X_1, X_2, ...$ each with this distribution?

We use our generator giving $U_1, U_2, ...$ which we regard as being observations chosen randomly on the interval $[0, 1]$. Then we can use the *table look-up method.*

Suppose that X takes discrete values on $0, 1, ..., n$. We know that

$$\sum_{x=0}^{n} p_X(x) = 1.$$

Therefore, suppose we divide the line $[0, 1]$ into segments of length $p_X(0), p_X(1), ..., p_X(n)$, where $p_X(x) \geq 0$, i.e., we consider subintervals of the form $(\mathbb{P}(X \leq j-1), \mathbb{P}(X \leq j)]$. If we let $P_X(j) = \mathbb{P}(X \leq j)$, then the length of the jth subinterval is given by

$$p_X(j) = \mathbb{P}(X \leq j) - \mathbb{P}(X \leq j-1),$$

and we can sample X-observations as follows.

i) Select a Uniform random variable on $[0, 1]$, U.

ii) Set $X = 0$ if $U \leq P_X(0)$.

iii) Set $X = j$ if $P_X(j-1) < U \leq P_X(j)$, $j = 1, ..., n$.

This algorithm works since if $U \sim U[0, 1]$, then $\mathbb{P}(a < U \leq b) = b - a$, so that $\mathbb{P}(P_X(j-1) < U \leq P_X(j))$ is simply $p_x(j)$. Thus, the algorithm returns the value $X = j$ with probability $p_X(j)$.

Example

Suppose that we wish to simulate a random variable, X, from a Poisson distribution, with $\lambda = 2$. We need to calculate the cumulative sums $P_X(x)$, and tabulate.

j	0	1	2	3	4
$P_X(j)$	0.1353	0.4060	0.6767	0.8571	0.9473

j	5	6	7	8	9
$P_X(j)$	0.9834	0.9955	0.9989	0.9998	1.000

Using this table, and the look-up algorithm, we can see what the following $U[0,1]$ random variables give rise to.

U	X
0.0318	0
0.4167	2
0.4908	2
0.2459	1
0.9673	5
0.8124	3

Such tables are finite and, if the random variable has an infinite range, then the range has to be truncated. In the above, using an accuracy of four decimal places, resulted in the range being truncated to $[0, 9]$.

Example

Sometimes the cumulative sums have a simple form and we do not need to set up an initial table. For example, suppose we wish to simulate a random variable X, from a discrete distribution with

$$p_X(x) = \begin{cases} \frac{1}{n} & x = 1, 2, ..., n \\ 0 & \text{else.} \end{cases}$$

Then, the cumulative sums are such that $P_X(j) = j/n \; j = 1, ..., n$, and the procedure is

i) Select a $U[0,1]$ random variable, U.

ii) Set $X = j$ if $(j-1)/n < U \leq j/n$, $j = 1, ..., n$.

However, we can re-write (ii) as

ii) Set $X = j$ if $j - 1 < nU \leq j$, or

Set $X = 1 + [nU]$, where $[a]$ denotes the integral part of a.

Thus, we have X explicitly in terms of U. Note that this is closely related to the *Inversion* method that we discuss in the next section.

So, if $n = 6$, the values of X that the six $U[0,1]$ random variables give rise to are

U	X
0.0318	1
0.4167	3
0.4908	3
0.2459	2
0.9673	6
0.8124	5

2.3.3 Inversion

This is the simplest of all procedures and is nothing more than a straightforward application of the probability integral transform: if $X \sim F$, then $F(X) \sim U[0,1]$. Figure 2.2 illustrates how this works.

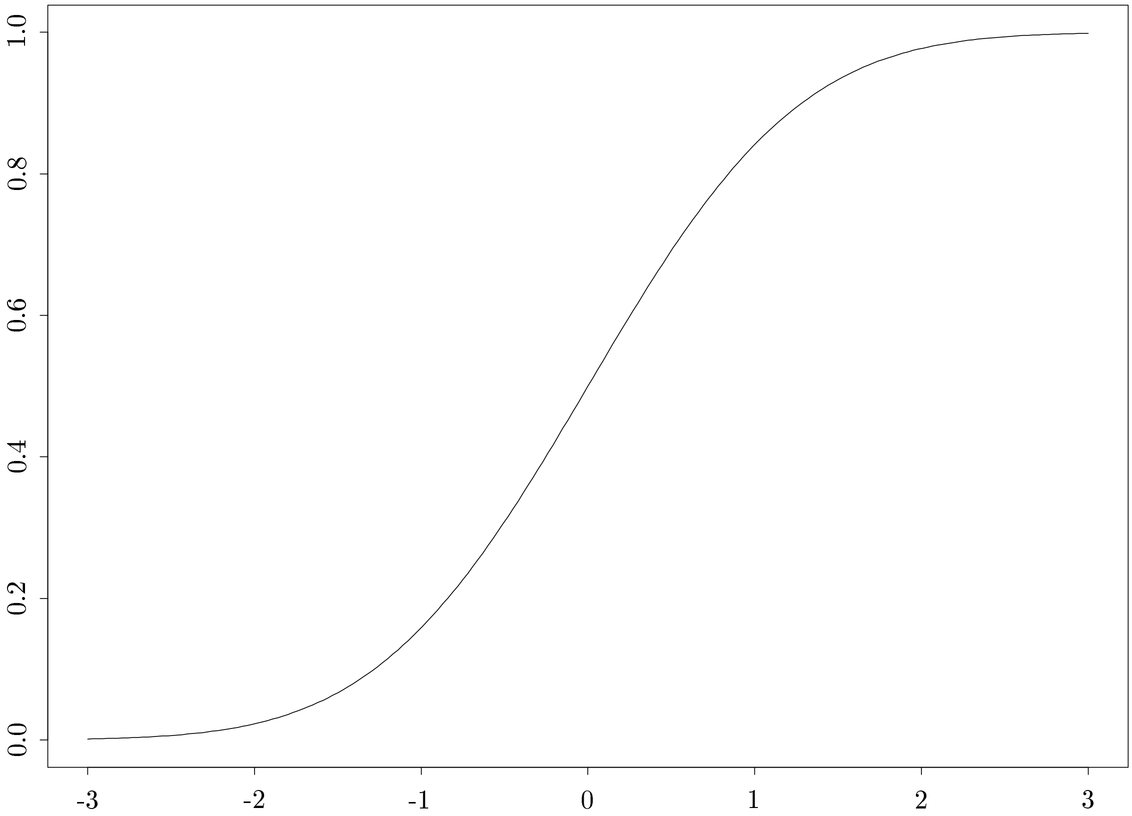


Figure. 2.2: Simulation by Inversion

More formally, suppose that we wish to simulate a continuous random variable X with cumulative distribution function

$$F(x) = \mathbb{P}(X \leq x).$$

Suppose also that the inverse function $F^{-1}(U)$ is well defined for $0 \leq U \leq 1$. Then we can use the following algorithm to sample from F.

STEP 1. GENERATE $U \sim U[0,1]$.
STEP 2. LET $X = F^{-1}(U)$.

Proof:

If $X = F^{-1}(U)$, then what is the distribution of X?

$$\mathbb{P}(X \leq x) = \mathbb{P}(F^{-1}(U) \leq x),$$

but since F is the cumulative distribution function of a continuous random variable, F is a strictly monotonic, increasing and continuous function of x. Hence,

$$\mathbb{P}(F^{-1}(U) \leq x) = \mathbb{P}(U \leq F(x)).$$

But, as U is a $U[0,1]$ random variable,

$$\mathbb{P}(U \leq F(x)) = F(x),$$

i.e.,

$$\mathbb{P}(X \leq x) = F(x),$$

where $X = F^{-1}(U)$. □

The method of inversion is easily implemented in S-Plus with the following code:

```
inversion <- function(n,inv.func){
     u <- runif(n)
     inv.func(u)
}
```

where `inv.func()` is the user-supplied inverse distribution function, F^{-1}.

Example

Let $X \sim \textit{exponential}(\lambda)$, then

$$f(x) = \lambda e^{-\lambda x} \text{ for } x \geq 0, \lambda > 0$$

and

$$F(x) = \int_0^x \lambda e^{-\lambda u} du = 1 - e^{-\lambda x}.$$

We need to find the function F^{-1}. Let

$$F(x) = u = 1 - e^{-\lambda x},$$

then

$$x = -\frac{1}{\lambda} \log_e(1-u).$$

So, we can set

$$X = F^{-1}(U) = -\frac{1}{\lambda} \log_e(1-U), \text{ where } U \sim U(0,1).$$

Thus, we define

```
inv.func <- function(u,lambda=1) {
       -(log(1-u))/lambda
}
```

to simulate from the exponential distribution (with $\lambda = 1$, by default), via the `inversion()` function in S-Plus.

The following six $U(0,1)$ random variables give rise to the $\textit{exponential}(2)$ random variables X given below. Repeating this 5000 times gives a sequence of observations with histogram given in Figure 2.3.

U	$X = -\frac{1}{2}\log_e(1-U)$
0.0318	0.016
0.4167	0.270
0.4908	0.337
0.2459	0.141
0.9673	1.710
0.8124	0.837

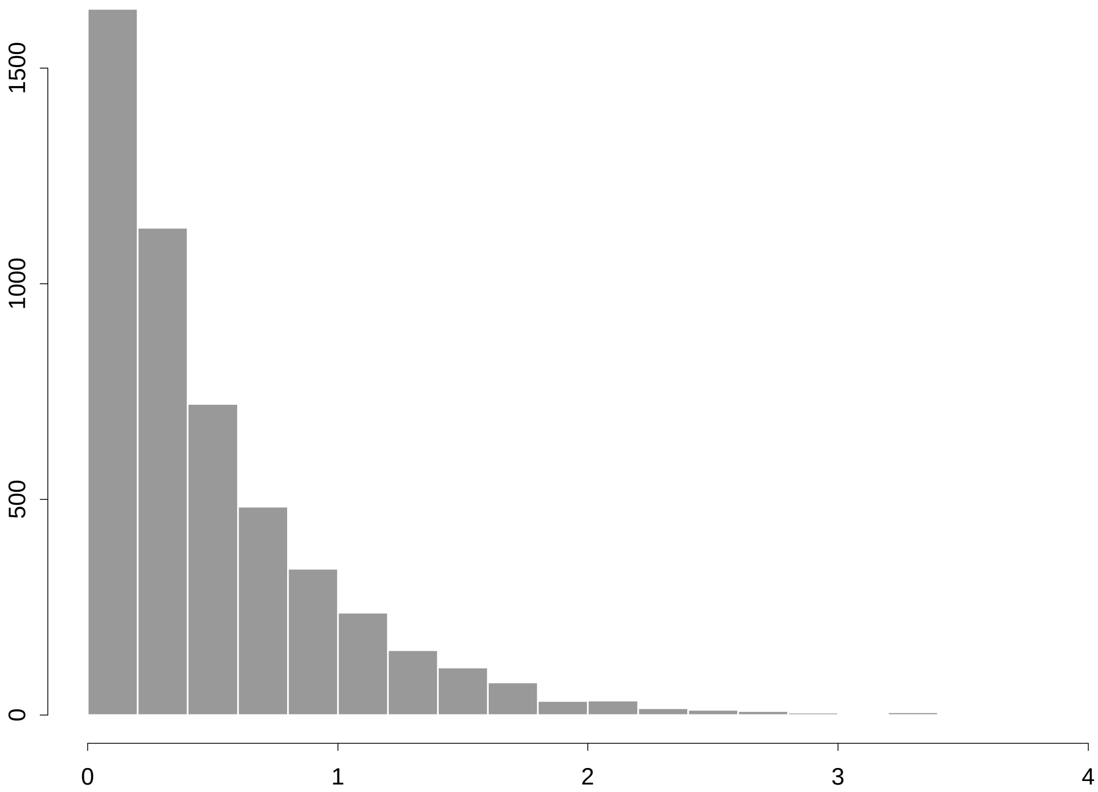


Figure. 2.3: Histogram of 5000 simulated exponential variates

It may seem that the inversion method is sufficiently universal to be the only method required. In fact, there are many situations in which the inversion method is either complicated to program or inefficient to run (or both). The inversion method is only really useful in cases where the inverse cdf is easy to compute. This is not the case if we wish to sample from a Normal distribution for example. in such cases, alternative simulation methods must be used.

2.3.4 Composition

We can extend these techniques to composite distributions, by first randomly choosing a distribution and then sampling from it. Suppose that we have

$$f = \sum_{i=1}^{r} p_i f_i$$

for pdf's f_i and probability distribution $\{p_1, ..., p_r\}$. Then we can sample from f by first choosing $I = i$ with probability p_i and then taking a sample from f_I. The density f is said to be a mixture distribution, and this method of sampling is known as composition.

Exercise Check.

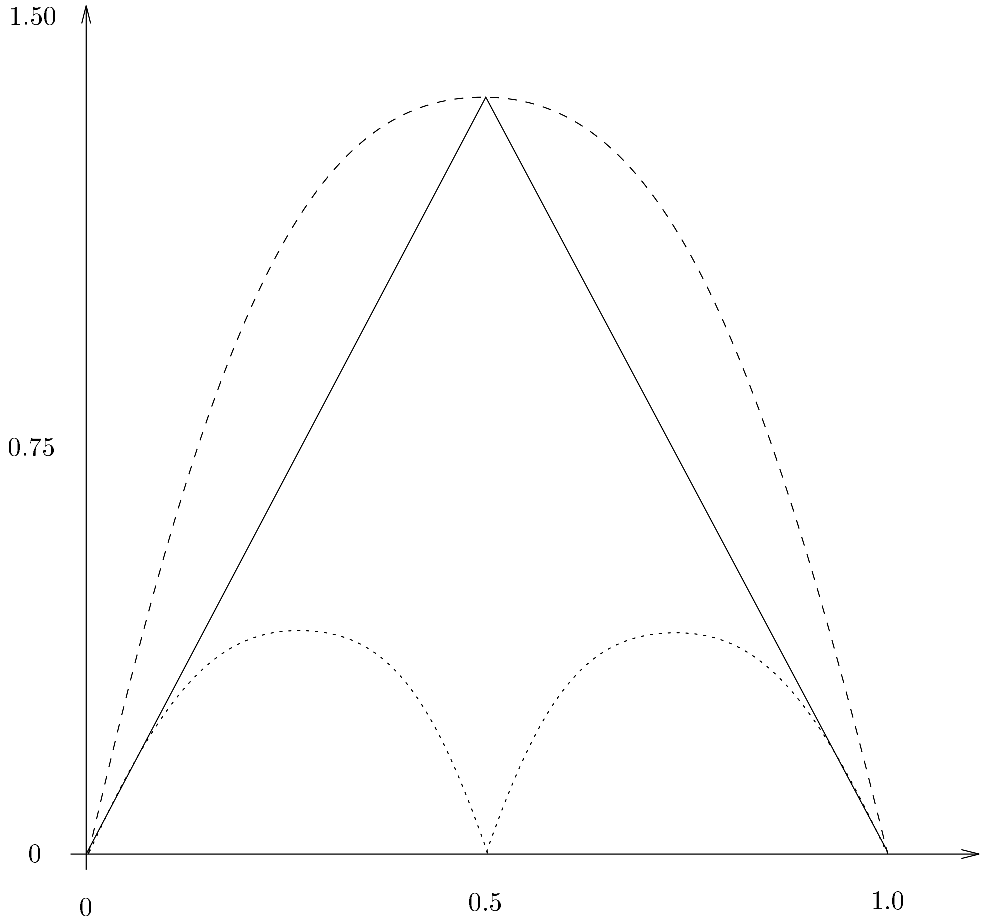


Figure. 2.4: Composition Example. Solid line: $\alpha f_1(x)$. Dashed line: $f(x) = 6x(1-x)$. Dotted line: $f(x) - \alpha f_1(x)$.

Example

A random variable X with the beta probability density function,

$$f(x) = 6x(1-x) \quad 0 \le x \le 1$$

can be simulated quite easily by either the table look-up method, or the rejection method. However, if we wish to use the composition method, we can split f into two components,

$$f(x) = \alpha f_1(x) + (1-\alpha)\left(\frac{f(x) - \alpha f_1(x)}{1-\alpha}\right),$$

for any function f_1 and constant, $0 < \alpha < 1$, and we can let

$$f_2(x) = \left(\frac{f(x) - \alpha f_1(x)}{1-\alpha}\right).$$

Thus, $f(x)$ can always be written as a composite density, by a split of this sort.

Now, we can define f_1 to be the symmetric triangular density over $[0,1]$, i.e.,

$$f_1(x) = \begin{cases} 4x & 0 \le x \le 0.5 \\ 4(1-x) & 0.5 \le x \le 1, \end{cases}$$

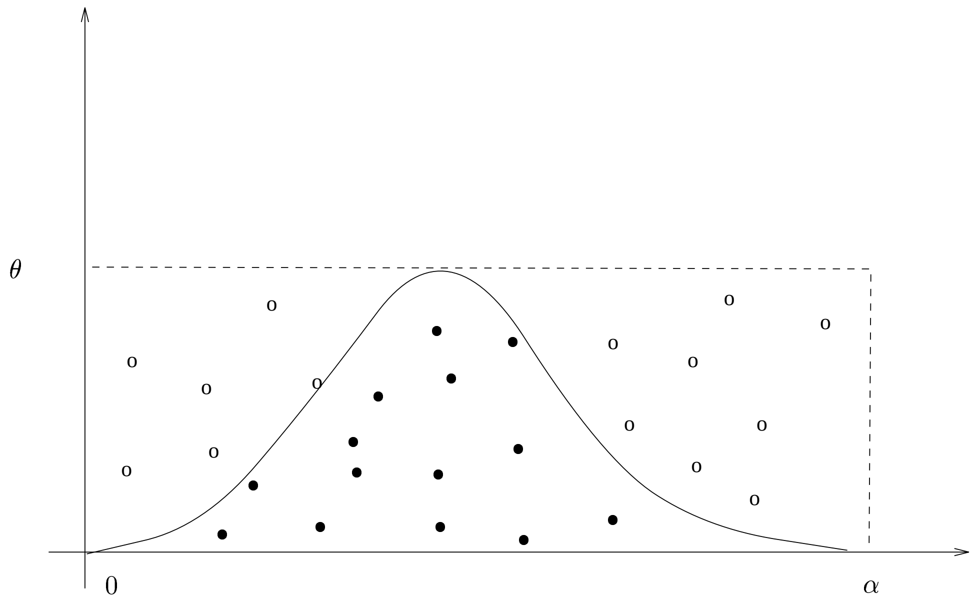


Figure. 2.5: Simulation by Rejection Sampling

from which it is straightforward to sample, then

$$f_2(x) = \begin{cases} 12x(1-2x) & 0 \leq x \leq 0.5 \\ 12(1-2x)(x-1) & 0.5 \leq x \leq 1 \end{cases}$$

which is somewhat more difficult to sample from than f_1. See Figure 2.4.

If we are to use the composition method, we would like α, the probability of sampling from f_1 to be as large as possible. However, we do have the constraint that $f_2(x) \geq 0$, which implies that $f(x) - \alpha f_1(x) \geq 0$, which in this case implies that $\alpha \leq 0.75$. Hence, the most efficient way to sample from f, using the method of composition, is to set $\alpha = 0.75$, i.e., we take

$$\alpha = \inf_x \left(\frac{f(x)}{f_1(x)} \right).$$

We can think of α as a shrinkage factor, applied to f_1, so that it just fits under f. This also implies that we should choose a function f_1 to be as close in shape to f as possible, so that α can be maximised.

2.3.5 The Rejection Method

Suppose that we wish to generate observations $X_1, X_2, \ldots$ from a distribution with density function $f_X(x)$. One way to do this is to enclose the density function within a rectangular box and generate points uniformly at random over this region. See Figure 2.5.

Any points outside the density function are rejected and any underneath are accepted. We take the abscissa (or x-coordinate) of the accepted points to be our required random number.

Thus, we use the following algorithm. Take $0 \leq x \leq \alpha$ and $0 \leq f_X(x) \leq \theta$.

Step 1. Generate $U \sim U[0,1]$ and let $X = \alpha U$.
Step 2. Generate a second uniform random variable, $V \sim U[0,1]$ and set $Y = \theta V$.
Step 3. If $Y > f_X(X)$ go back to Step 1 and repeat, otherwise accept the random variable X.

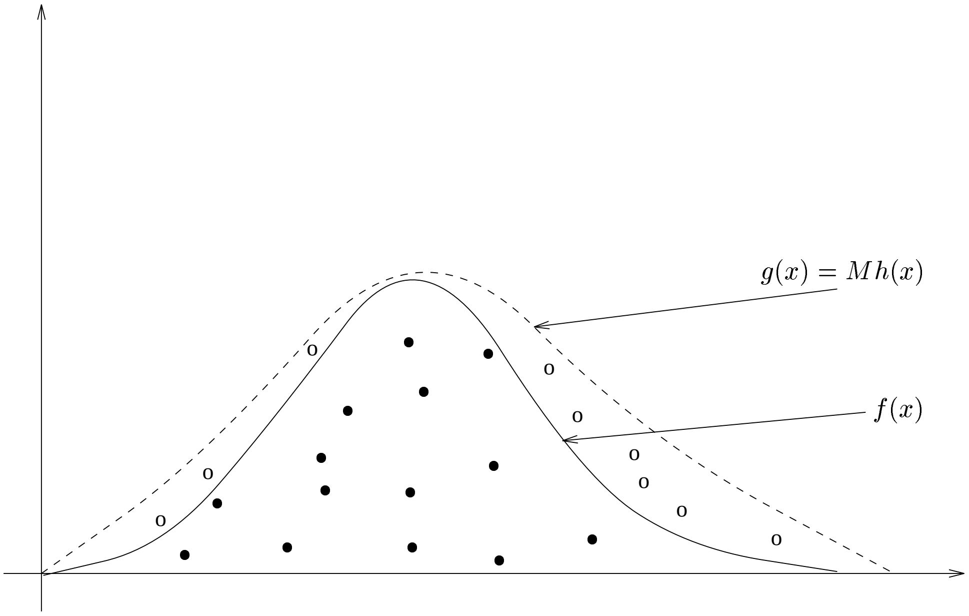


Figure. 2.6: Simulation by Rejection Sampling, with general envelope g

There are a number of problems associated with this method.

1. A rectangle cannot be used when $f_X(x)$ has an infinite range, and
2. The probability of rejection can become quite large.

A way of overcoming these problems is to *envelope* the density function, $f_X(x)$ *not* by a rectangle, but some some other (more general) curve, $g(x)$. $g(x)$ is often some multiple of a second p.d.f. $h(x)$ i.e., $g(x) = Mh(x)$, from which it easy to sample.

If we let $g(x) = Mh(x)$, where $M > 1$, then we can use the following algorithm.

Step 1. Simulate $X \sim h(X)$.
Step 2. Generate $U \sim U(0,1)$ and define $Y = Ug(X)$.
Step 3. Accept X as a realisation from $f_X(X)$ if and only if $Y \leq f_X(X)$.

The reason this works is clear from the following argument.

$$\mathbb{P}(X \text{ accepted}) = \int_{-\infty}^{\infty} h(y)\frac{f_X(y)}{g(y)}dy$$

and

$$\mathbb{P}(X \leq x \text{ and } X \text{ accepted}) = \int_{-\infty}^{x} h(y)\frac{f_X(y)}{g(y)}dy.$$

Then,

$$\begin{aligned}\mathbb{P}(X \leq x | X \text{ accepted}) &= \frac{\int_{-\infty}^{x} h(y)\frac{f_X(y)}{g(y)}dy}{\int_{-\infty}^{\infty} h(y)\frac{f_X(y)}{g(y)}dy} \\ &= \frac{\int_{-\infty}^{x} f_X(y)dy}{\int_{-\infty}^{\infty} f_X(y)dy},\end{aligned}$$

so that accepted values have pdf f.

Note that the probability of rejection is given by

$$\begin{aligned}
\mathbb{P}(rejection) &= \frac{\int_{-\infty}^{\infty}(g(x)-f(x))dx}{\int_{-\infty}^{\infty}g(x)dx} \\
&= \frac{\int_{-\infty}^{\infty}g(x)dx}{\int_{-\infty}^{\infty}g(x)dx} - \frac{\int_{-\infty}^{\infty}f(x)dx}{\int_{-\infty}^{\infty}g(x)dx} \\
&= 1 - \frac{1}{\int_{-\infty}^{\infty}Mh(x)dx} \\
&= 1 - \frac{1}{M}, \text{ since } h \text{ is a pdf.}
\end{aligned}$$

Therefore, we would like M to be as close to 1 as possible, subject to $M \geq 1$. But, we also need that $Mh(x) \geq f(x)$, for all x, since g must "envelope" f. Therefore,

$$M \geq \frac{f(x)}{h(x)} \ \forall x,$$

so that the optimal M is simply

$$M^* = \sup_x \left(\frac{f(x)}{h(x)}\right),$$

where this maximum is finite.

Once again, the rejection sampling algorithm can be easily implemented in S-Plus. As an example, consider the distribution with density

$$f(x) \propto x^2 e^{-x}; \quad 0 \leq x \leq 1,$$

a truncated Gamma distribution. Then, since $f(x) \leq e^{-x}$ everywhere, we can set $g(x) = \exp(-x)$ and so simulate from an exponential distribution, rejecting according to the above algorithm. In this case, we can use the following code.

```
rejection <- function(n) {
     r <- NULL
     for(i in 1:n) {
         t <- -1
         while(t < 0) {
               x <- rexp(1,1)
               y <- runif(1,0,exp(-x))
               if(x > 1)
                    t <- -y
               else t <- x^2 * exp(-x) - y
         }
         r[i] <- x
     }
     r
}
```

Example

Suppose we wish to use the rejection algorithm to sample from a $\Gamma(k,1)$ distribution i.e.,

$$f_X(x) = \frac{x^{k-1}e^{-x}}{\Gamma(k)} \quad x \geq 0, \ k > 1.$$

Take $g(x) = Mh(x)$, where

$$h(x) = \frac{e^{-x/k}}{k}, \text{ for } x \geq 0$$

i.e., exponential with parameter $1/k$, from which we can sample via the method of inversion.

Note that, since $k > 1$, as $x \to \infty$, $f(x) \to 0$ faster than $h(x)$, so $Mh(x)$ is a suitable envelope function for $f(x)$ i.e., $Mh(x) \geq f(x)\ \forall x \geq 0$. We now need to find a suitable value of M.

Let $y = f(x)/h(x)$, and maximise y with respect to x.

$$\begin{aligned} \log_e y &= \log_e f(x) - \log_e h(x) \\ &= -x + (k-1)\log_e x - \log_e \Gamma(k) - \log_e 1/k + x/k. \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{d}{dx}\log_e y &= \frac{k-1}{x} - \frac{1}{k} - 1, \text{ and} \\ \frac{d^2}{dx^2}\log_e y &= -\frac{k-1}{x^2}. \end{aligned}$$

Therefore, we maximise y when $x = k(k-1)/(k+1)$.

2.3.6 Ratio of Uniforms

Suppose that we wish to generate random observations from a distribution with density function $f_X(x)$, $-\infty < x < \infty$. Then, if we can write

$$f_X(x) = \frac{h(x)}{\int_{-\infty}^{\infty} h(x)dx}$$

for some non-negative function h with finite integral, we can use the following theorem.

Theorem 2.1: *If the pair of random variates (u, v) is uniformly distributed in the region*

$$C_h = \left\{(u,v) : 0 \leq u \leq \left[h\left(\frac{v}{u}\right)\right]^{1/2}\right\}$$

then the variable $X = \frac{V}{U}$ has probability density function $f(x)$.

Proof:

Recall the transformation of variables results, that if we let S_1 and S_2 be jointly continuous random variables with pdf $f_S(s_1, s_2)$ and we let $T_1 = g_1(S_1, S_2)$ and $T_2 = g_2(S_1, S_2)$, then the joint pdf of $T = (T_1, T_2)$ is given by

$$f_T(t_1, t_2) = f_S(S_1(t_1, t_2), S_2(t_1, t_2))|J|,$$

where J is the Jacobian of the transformation, given by

$$J = \begin{pmatrix} \frac{\partial S_1}{\partial T_1} & \frac{\partial S_1}{\partial T_2} \\ \frac{\partial S_2}{\partial T_1} & \frac{\partial S_2}{\partial T_2} \end{pmatrix}.$$

Now, let $Y = (U, V)$ be uniformly distributed on C_h. Then,

$$f_Y(u, v) = \frac{1}{A} \text{ where } A \text{ is the area of } C_h,$$

for $0 \le u \le \sqrt{h(v/u)}$. What is the distribution of $X = V/U$?

Let $X_1 = V/U$ and $X_2 = U$, then $U = X_2$, $V = X_1 X_2$ and the Jacobian, $|J|$, is given by

$$\begin{vmatrix} \frac{\partial u}{\partial x_1} & \frac{\partial u}{\partial x_2} \\ \frac{\partial v}{\partial x_1} & \frac{\partial v}{\partial x_2} \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ x_2 & x_1 \end{vmatrix} = -x_2.$$

Therefore, the joint density is given by

$$\begin{aligned} f_X(x_1, x_2) &= f_Y(x_2, x_1 x_2)|J| \\ &= \frac{x_2}{A} \quad \text{for } 0 \le x_2 \le \sqrt{h(x_1)}. \end{aligned}$$

Finally, the marginal density of X_1 is given by

$$\begin{aligned} \int_{x_2} f_X(x_1, x_2) dx_2 &= \int_0^{\sqrt{h(x_1)}} \frac{x_2}{A} dx_2 \\ &= \frac{h(x_1)}{2A}. \end{aligned}$$

However,

$$\int \frac{h(x_1)}{2A} = 1,$$

since it is a density, therefore,

$$2A = \int h(x_1) dx_1,$$

so the marginal density for $X_1 = V/U$, is given by

$$\frac{h(x_1)}{\int h(x_1) dx_1},$$

as required. □

This result is most useful when the set C_h is contained in a rectangle $[0, a] \times [b_1, b_2]$ in which case, we can use rejection sampling.

STEP 1. GENERATE $U_1, U_2 \sim U(0, 1)$.
STEP 2. LET $U = aU_1$ AND $V = b_1 + (b_2 - b_1)U_2$, THEN (U, V) IS A POINT RANDOMLY CHOSEN IN THE RECTANGLE $[0, a] \times [b_1, b_2]$.
STEP 3. IF $(U, V) \in C_h$, THEN RETURN $X = V/U$, ELSE REPEAT FROM STEP 1.

So, what conditions do we need for C_h to be contained within a rectangle?

Theorem 2.2: *If $h(x)$ and $x^2 h(x)$ are bounded then $C_h \in [0, a] \times [b_1, b_2]$ where*

$$\begin{aligned} a &= \sup_x \sqrt{h(x)} \\ b_1 &= -\sqrt{\sup_{x \le 0}(x^2 h(x))} \\ b_2 &= \sqrt{\sup_{x \ge 0}(x^2 h(x))}. \end{aligned}$$

Hence, we can re-write Step 3 as

STEP 3. ACCEPT $X = V/U$ IF AND ONLY IF

$$U^2 \le h\left(\frac{V}{U}\right)$$

OR EQUIVALENTLY

$$V^2 \leq X^2 h(X),$$

WHICHEVER IS COMPUTATIONALLY EASIER.

Example

To generate observations $X_1, X_2, ...$ from a $N(0,1)$ density i.e.,

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \quad -\infty < x < \infty,$$

we set

$$h(x) = e^{-\frac{1}{2}x^2} \quad -\infty < x < \infty.$$

Then,

$$\int_{-\infty}^{\infty} h(x)dx = \sqrt{2\pi} < \infty.$$

Next, we need to find a, b_1 and b_2.

$$a = \sup_x \sqrt{h(x)} = \sqrt{1} = 1 \text{ (when } x = 0).$$

Now, consider

$$y = x^2 h(x) = x^2 e^{-\frac{1}{2}x^2}.$$

$$\begin{aligned} \frac{dy}{dx} &= 2xe^{-\frac{1}{2}x^2} + x^2 e^{-\frac{1}{2}x^2}(-x) \\ &= e^{-\frac{1}{2}x^2} x(2 - x^2) \\ &= 0, \text{ when } x = \pm\sqrt{2} \text{ or } x = 0. \end{aligned}$$

Now

$$\begin{aligned} \frac{d^2y}{dx^2} &< 0 \quad \text{when } x = \pm\sqrt{2} \\ \text{and } \frac{d^2y}{dx^2} &> 0 \quad \text{when } x = 0 \end{aligned}$$

Therefore,

$$\begin{aligned} \sup_x (x^2 h(x) | x \leq 0) &= 2e^{-\frac{1}{2}(2)} = 2e^{-1} \\ \text{and } \sup_x (x^2 h(x) | x \geq 0) &= 2e^{-1} \end{aligned}$$

Hence $b_1 = -\sqrt{2}e^{-\frac{1}{2}}$ and $b_2 = \sqrt{2}e^{-\frac{1}{2}}$.

So, what condition must be satisfied for $(u, v) \in C_h$?

$$C_h = \left\{ (u, v) : 0 \leq u \leq \sqrt{h(v/u)} \right\}.$$

Therefore,

$$C_h = \left\{ (u, v) : 0 \leq u \leq \sqrt{e^{-\frac{1}{2}(v/u)^2}} \right\}.$$

Or, equivalently,

$$v^2 \leq -4u^2 \log_e u.$$

Thus, we can use the following algorithm.

STEP 1. GENERATE $U_1, U_2 \sim U(0,1)$.
STEP 2. LET $U = aU_1 = U_1$ AND $V = b_1 + (b_2 - b_1)U_2 = \sqrt{2}e^{-\frac{1}{2}}(2U_2 - 1)$.
STEP 3. IF $V^2 \leq -4U^2 \log_e U$, THEN RETURN $X = V/U$, OTHERWISE REPEAT FROM STEP 1.

This can be easily implemented in S-Plus as follows.

```
ratio <- function(n){
     r <- NULL
     for(i in 1:n){
        t <- 1
        while(t > 0) {
             u <- runif(1,0,1)
             v <- sqrt(2)*exp(-1/2)*(2*runif(1,0,1)-1)
             t <- v^2 + 4*u^2*log(u)
             }
             r[i] <- v/u
        }
        r
}
```

and a histogram of 5000 simulated values is give in Figure 2.7.

A number of modifications of the ratio method have been suggested, to improve upon the efficiency of this procedure, which amount to rescaling and locating distributions before applying the method.

Another method for improving efficiency is by a process known as "squeezing" or "pre-testing". This applies to both the ratio and the rejection methods. The point is that, in the ratio method for example, the slowest part of the algorithm can be the check of whether or not $(u, v) \in C_h$. However, there may be similar regions C_1 and C_2 such that $C_1 \subset C_h \subset C_2$ so that if (u, v) is found to lie inside C_1 or outside C_2, then we immediately know whether (u, v) lies in C_h or not. Finding suitably simple sets C_1 and C_2 can greatly improve the efficiency of the methods.

2.4 Exercises

Exercise 2.1: *In Buffon's needle experiment, where we have a needle of length l and lines equally spaced a distance $d \geq l$ apart, let x denote the distance from the centre of the needle to the nearest upper line and let θ denote the angle between the needle and that line. Then, under the assumption of random needle throwing, we have $x \sim U[0, d]$ and $\theta \sim U[0, \pi]$. Calculate p, the probability that the needle touches one of the lines, in terms of π, l and d.*

Say we repeat the experiment n times, and that the needle touches a line R of those times, then if we let $\hat{p} = R/n$, show that $Var(\hat{p}) = p(1-p)/n$. Finally, if we let $\phi = 1/\pi$, and define an estimate $\hat{\phi}$ of ϕ by $\hat{\phi} = \hat{p}d/2l$, show that the variance of the estimator $\hat{\phi}$ is minimised by setting $l = d$.

Exercise 2.2: *The following recursive formula is known as the* Fibonacci *recursion algorithm.*

$$U_i = (U_{i-1} + U_{i-2}) \mod 1. \qquad U_i \in \mathbb{R},\ 0 \leq U_i < 1.$$

Show that the resulting sequence is not random since the event $U_{i-2} < U_i < U_{i-1}$ $(i \geq 3)$ occurs with probability zero, whereas in a truly random sequence, this event occurs with probability $1/6$.

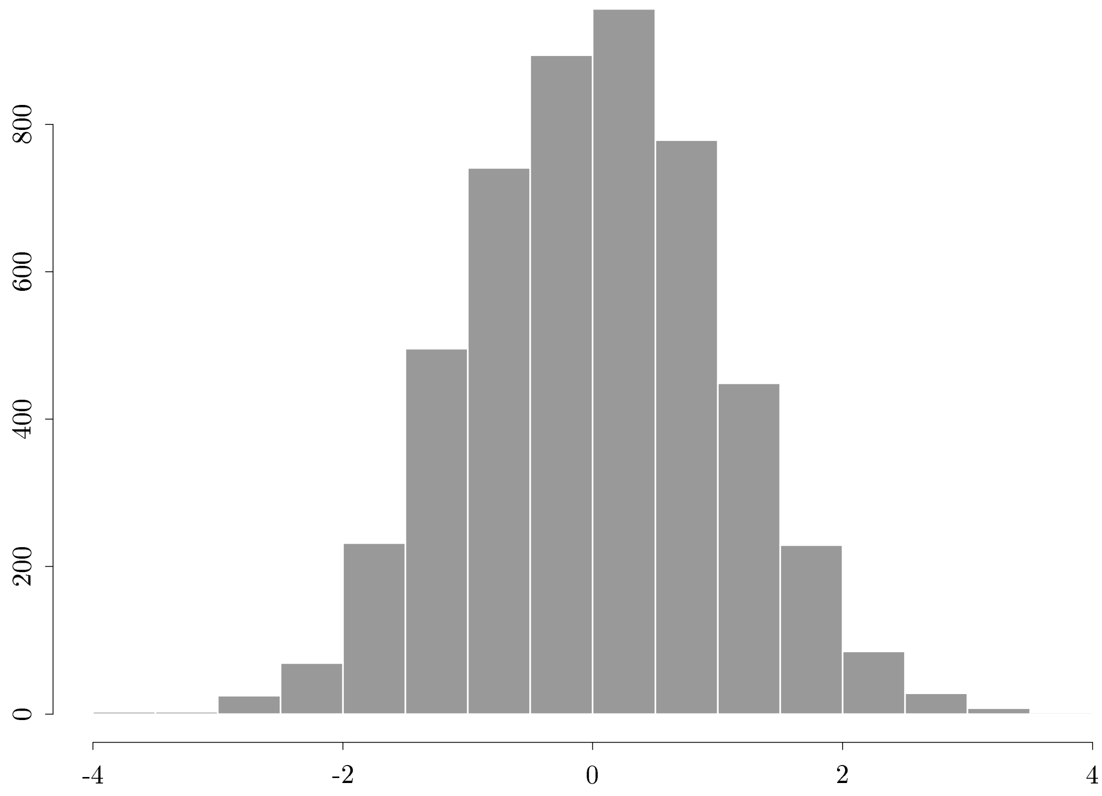


Figure. 2.7: Histogram of 5000 draws from a $N(0,1)$ distribution using the ratio method

Exercise 2.3: *(a) In a congruential generator, show that if $M = 10^k$, for some positive integer $k > 1$, then for the cycle length to equal M, we need to set $a = 20d + 1$, where d is a positive integer.*

(b) Prove by induction (or otherwise) that for a congruential generator with $a > 1$,

$$X_{n+k} = \left[a^k X_n + (a^k - 1)c/(a-1)\right] (mod\, M),$$

where $k \geq 0$ and $n \geq 0$. Hence show that if there exists j such that $a^j = 1(mod\, M)$ then there exists a cycle of length j.

Exercise 2.4: *Devise a table look-up method for generating geometric random variables with parameter p. (Hint: show that the cumulative sums have a simple form and hence that our geometric random variable X can be written explicitly in terms of a uniform U(0,1) random variable.)*

Exercise 2.5: *If X is a random variable with the exponential, $\lambda e^{-\lambda x}$ p.d.f., for $x \geq 0$, deduce the distribution of the integer part of X, i.e. $Y = [X]$. Hence, explain why we can obtain a geometric random variable by rounding up an exponential random variable.*

Exercise 2.6: *The following is the Box-Muller algorithm.*

STEP 1. GENERATE $U_1 \sim U(0,1)$ AND $U_2 \sim U(0,2\pi)$ RESPECTIVELY, I.E. $U_2 = 2\pi U$, WHERE $U \sim U(0,1)$.

STEP 2. LET

$$\begin{aligned} X_1 &= \sqrt{-2\log_e U_1}\cos U_2, \text{ AND} \\ X_2 &= \sqrt{-2\log_e U_1}\sin U_2. \end{aligned}$$

Show that this algorithm produces random variables X_1 and X_2 which are independent $N(0,1)$ random variables. (Hint: Use transformation of variables to obtain the joint distribution of X_1 and X_2.)

Exercise 2.7: *Simulate the Buffon's needle experiment discussed in Exercise 2.1, using the code provided in Section 2.1.4 to estimate π. (Don't forget to use the* `attach` *command to get the functions, and then the* `motif()` *command to set up a graphics window.) Type* `par(mfrow=c(2,2))` *and run the function four times with $n = 2000$. What can you say about the graphs? How good is this experiment as a method for estimating π?*

Exercise 2.8: *Write an S-Plus function* `rand.gen(x,n)` *which simulates n observations via the recurrence*

$$X_t = f(X_{t-1}),$$

where

$$f(X) = \text{ the nearest integer to } (100 \times \text{ the fractional part of } \log_e),$$

where x is the "seed" for the sequence. Verify that if $x = 82$, then the sequence gets "stuck" after 3 iterations. Try $x = 39$ and $n = 96$ and plot the output. What (if any) patterns do you see?

Useful commands may include

`as.integer(a)`	*Returns the integer part of* `a`.
`round(a)`	*Rounds* `a` *to the the nearest integer.*
`log(a)`	*Returns $\log_e$ of* `a`.

Chapter 3

MONTE CARLO METHODS AND NONPARAMETRIC INFERENCE

3.1 Monte Carlo Integration

In one form or another, the reason for simulation can often be formulated as an integral. This is obviously the case for expectations: $\mathbb{E}(X) = \int x f(x)dx$, so if we have a sample $x_1, ..., x_n$ from the distribution of X, then we can approximate the theoretical mean by the sample mean to obtain the approximation:

$$\mathbb{E}(X) \approx \frac{1}{n}\sum_{i=1}^{n} x_i.$$

But this argument can be generalised. Suppose that we wish to calculate

$$\theta = \int \phi(x)f(x)dx$$

which is, of course, $\mathbb{E}(\phi(x))$, where expectation is with respect to the distribution f. Then, if $x_1, ..., x_n$ is a sample from this distribution,

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} \phi(x_i)$$

is an unbiased estimate for θ. This approach is remarkably easy to use, even in high dimensions. However, the cost for this simplicity is that the variance of the resulting estimators is high.

Example

Suppose that we wish to calculate $\mathbb{P}(X < 1, Y < 1)$, where (X, Y) are bivariate standard normal, with correlation 0.5. This can be written as

$$\int I_A(x, y)f(x, y)dxdy,$$

where f is the bivariate normal density and I_A is the indicator function for the set $A = \{(x, y) : x < 1,\ y < 1\}$. Thus, provided we can simulate from the bivariate normal, we can estimate this probability as

$$\frac{1}{n}\sum_{i=1}^{n} I_A(x_i, y_i),$$

which is simply the proportion of simulated points falling in A. There are various approaches to simulating bivariate normals, but conceptually easiest is to simulate each pair (x, y) by first simulating x, and then simulating y from the conditional distribution of $Y|X$ which is also normal. The following S-Plus code can be used to obtain such a sample.

```
bvnorm <- function(n,mu,sd,rho){
     x <- rnorm(n,mu[1],sd[1])
     y <- rnorm(n,mu[2] + (rho*sd[2])/sd[1] *
                (x-mu[1]),sd[2] * sqrt(1 - rho*rho))
     cbind(x,y)
}
```

Hence to obtain an estimate of the required probability on the basis of, say, 1000 simulations, we simply need

```
data <- bvnorm(1000,c(0,0),c(1,1),0.5)
```

(see Figure 3.1) and then

```
sum(data[,1] < 1 & data[,2] < 1)/1000
```

Using this code, we got the answer 0.764.

3.2 Variance Reduction

A number of techniques are available for improving the precision of Monte Carlo integration. We will look at one of these in detail, and describe the idea behind the other two.

3.2.1 Importance Sampling

Suppose again that the ratio f/g is not bounded, so that we are unable to use rejection sampling and that both f and g are normalised densities. Suppose further that we are interested in obtaining an estimate of $\theta = \mathbb{E}_f[\phi(X)]$. Let $\{x_1, x_2, ...\}$ be a sample from g and define "importance" weights

$$w(x_i) = \frac{f(x_i)}{g(x_i)}.$$

Then, if

$$\hat{\theta}_g = \frac{1}{n}\sum_{i=1}^{n} w(x_i)\phi(x_i),$$

$\mathbb{E}(\hat{\theta}_g) = \mathbb{E}_f[\phi(X)]$.

The advantage of this method is that we can use it for any densities provided that they are continuous and have the same support. However there are several disadvantages. Firstly, the variance of $\hat{\theta}_g$ can be very large leading to unrealistic estimates. Secondly, without normalisation constants for f and g, we need to use

$$\frac{\sum_1^n w(x_i)\phi(x_i)/n}{\sum_1^n w(x_i)/n}$$

so that the numerator tends to $c\mathbb{E}_f[\phi(X)]$ and the denominator to c. This can make the variance infinite.

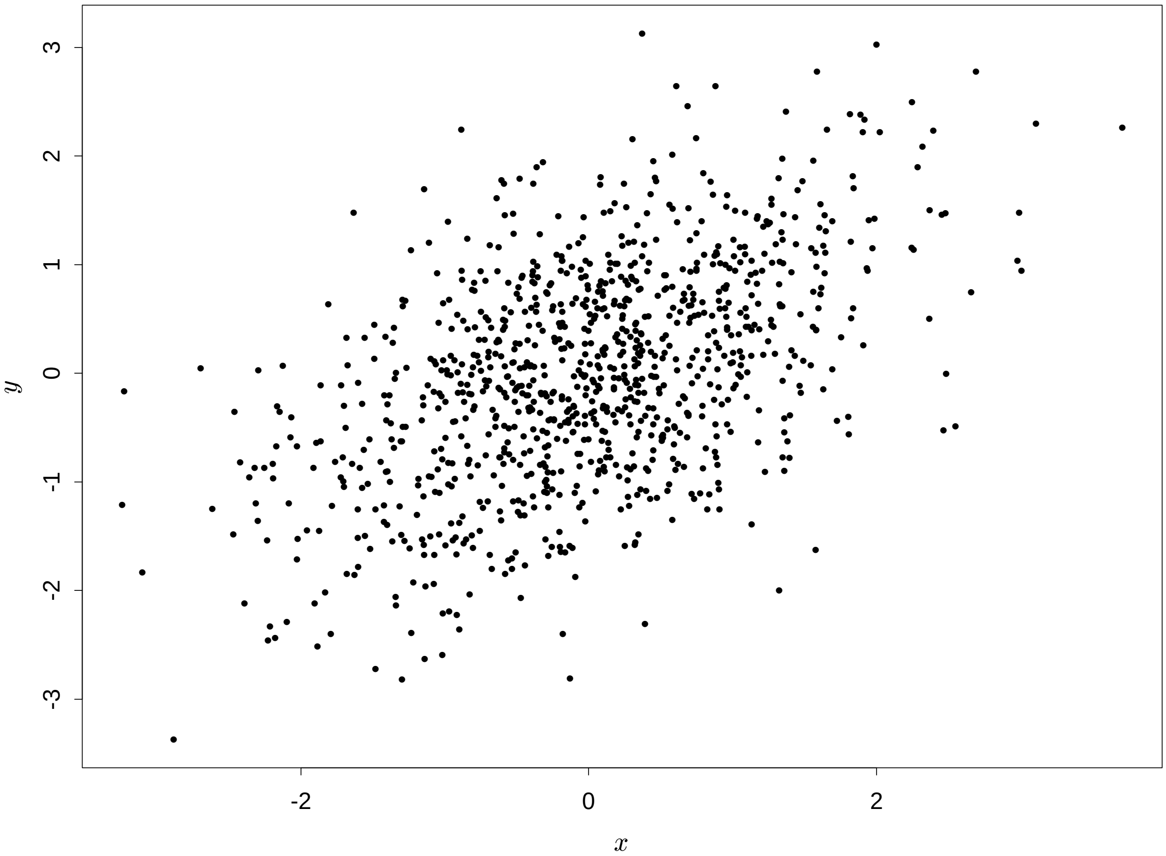


Figure. 3.1: Scatter plot of bivariate normal sample.

$$\begin{aligned}
Var(\hat{\theta}_g) &= \frac{1}{n}\mathbb{E}[(w(x)\phi(x) - \theta)^2] \\
&= \frac{1}{n}\int (w(x)\phi(x) - \theta)^2 g(x)dx \\
&= \frac{1}{n}\int \left(\frac{\phi f}{g} - \theta\right)^2 g(x)dx
\end{aligned}$$

Now $Var(\hat{\theta}_g)$ is minimised by

$$g \equiv g_0 = \frac{|\phi f|}{\int |\phi f|}$$

Proof:

$$Var(\hat{\theta}_g) = \frac{1}{n}\left(\int \frac{\phi^2 f^2}{g}dx - \theta^2\right)$$

So

$$Var(\hat{\theta}_{g_0}) = \frac{1}{n}\left[\left(\int |\phi f|dx\right)^2 - \theta^2\right]$$

but

$$\left(\int |\phi f| dx\right)^2 = \left(\int \frac{|\phi f|}{g^{\frac{1}{2}}} g^{\frac{1}{2}} dx\right)^2 \tag{3.1}$$

$$\leq \int \frac{\phi^2 f^2}{g} dx \cdot \int g dx \tag{3.2}$$

$$= \int \frac{\phi^2 f^2}{g} dx \tag{3.3}$$

by the Cauchy-Schwartz inequality.

Thus, importance sampling can be used to reduce the variance of estimates. However the optimal importance sampling density varies with different functions of interest.

Example

Suppose that we wish to estimate the probability $\mathbb{P}(X > 2)$, where X follows a Cauchy distribution, with density

$$f(x) = \frac{1}{\pi(1+x^2)}, \qquad x \in \mathbb{R}$$

so we require

$$\int_2^\infty f(x)dx = \int I_A(x) f(x) dx,$$

where $A = \{x : x > 2\}$. We could simulate from the Cauchy distribution directly, but the variance of the ergodic average in this case, is very large. Alternatively, we observe that, for large x, $f(x)$ is similar in behaviour to the density

$$g(x) = 2/x^2 \quad x > 2.$$

By inversion, we can simulate from g by letting $x_i = 2/u_i$, where $U_i \sim U(0,1)$. Thus, our estimator becomes:

$$\hat{\theta}_g = n^{-1} \sum_{i=1}^n \frac{x_i^2}{2\pi(1+x_i^2)},$$

where $x_i = 2/u_i$.

We can implement this via the S-Plus function

```
imp.samp <- function(n){
     x <- 2/ runif(n)
     psi <- x^2/(2*pi*(1+x^2))
     mean(psi)
}
```

The exact value of $\phi = \mathbb{P}(X > 2)$ is $0.5 - \pi^{-1} \tan 2 = 0.1476$. And Figure 3.2a plots the estimated value of ϕ as a function of n. Figure 3.2b plots the corresponding estimated values of ϕ obtained by using the `rcauchy()` command in S-Plus. Clearly, the reduction in variability is substantial.

3.2.2 Control and Antithetic Variates

Let Y be an unbiased estimate of θ. A random variate C with mean μ_C is a control variate for Y if it is correlated with Y. We can construct an estimator for θ that has smaller variance than Y as follows. For any β,

$$Y(\beta) = Y - \beta(C - \mu_C)$$

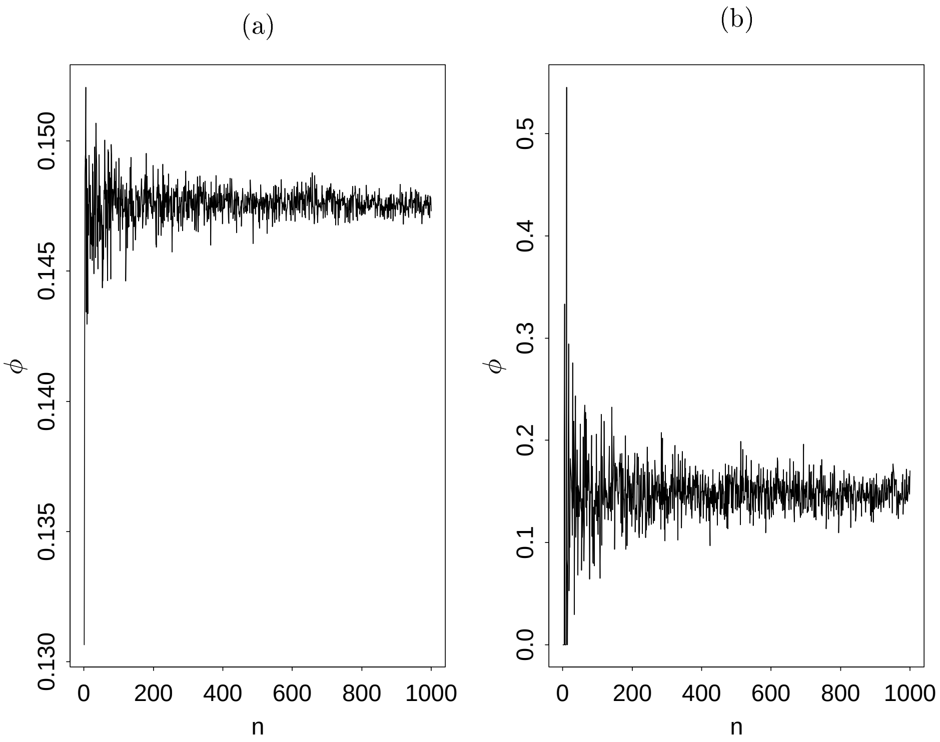


Figure. 3.2: Convergence of Importance sampled mean (a) and standard estimator (b), for $\phi = \mathbb{P}(X > 2)$.

is an unbiased estimator of θ. Now

$$Var[Y(\beta)] = Var[Y] - 2\beta Cov[Y, C] + \beta^2 Var[C].$$

So if

$$2\beta Cov[Y, C] > \beta^2 Var[C]$$

then variance reduction is achieved. The value of β that minimises $Var[Y(\beta)]$ is (differentiate w.r.t. β and set to zero) found to be

$$\beta^* = \frac{Cov[Y, C]}{Var[C]}$$

and the minimum variance is

$$Var[Y(\beta^*)] = (1 - \rho_{YC}^2)Var[Y]$$

with ρ_{YC} the correlation between Y and C. The more C is correlated with Y, the greater the reduction in variance. The practical problem with this method is in identifying suitable control variates.

Returning to the Cauchy example of the last section,

$$\mathbb{P}(X > c) = \int_2^\infty f(x)dx = 1 - \int_{-\infty}^2 f(x)dx = \frac{1}{2} - \int_0^2 f(x)dx$$

and this can be estimated by

$$\hat{\theta} = \frac{1}{2} - \frac{1}{n}\sum_{i=1}^n f(x_i)/g(x_i) = \frac{1}{2} - 2 \times \frac{1}{n}\sum_{i=1}^n f(x_i)$$

where the $x_i \sim U[0, 2]$ i.e., $g(x_i) = 1/2$, $x_i \in [0, 2]$.

To estimate the integral using a control variate we seek a function (or functions) with known mean which varies with $f(x)$ over $[0,2]$. A Taylor series expansion of $f(x)$ suggests control variates of x^2 and x^4, whose means, with respect to the $U[0,2]$ distribution, are easily evaluated as 8/6 and 32/10. Now, in principle, any function of the form $\beta_1 x^2 + \beta_2 x^4$ suffices as a control variate. However, to minimize the variance we need to choose the β's so as to optimise agreement between $f(x)$ and $\beta_1 x^2 + \beta_2 x^4$. This is achieved through simple regression. Empirically, you can simulate 10 observations uniformly over $[0,2]$, leading to the regression equation

$$W(x) = 0.288 - 0.143x^2 + 0.024x^4$$

and hence,

$$\hat{\theta} = \frac{1}{2} - 2\left[\frac{1}{n}\sum_{i=1}^{n}\{f(X_i) - W(X_i)\} + 0.288 \times 2 - 0.143 \times 8/6 + 0.024 \times 32/10\right].$$

With $n = 1000$, we got an estimate of $\hat{\theta} = 0.1474$.

Example

Suppose that we are interested in the range of a sample of five observations from a N(0,1) distribution. If we arrange the possible values of X, a N(0,1) random variate into ranges, 0-1, 1-2, 2-3, 3-4, 4-5 and $5 - \infty$, then the probability that X lies in range i is given by p_i, say. We can estimate p_i by

$$\hat{p_i} = \frac{n_i}{n}, \qquad \mathbb{E}(\hat{p_i}) = p_i \text{ and } Var(\hat{p_i}) \approx n_i(n - n_i)/n^3$$

where a sample of n X-values are obtained for which n_i of the n values are found to lie in the i^{th} category.

Now introduce an auxiliary variable,

$$Y_i = (X_i - \bar{X})^2$$

and divide the possible Y-values into ranges, 0-0.71, 0.71-2.57, 2.75-4.88, 4.88-9.49, 9.49-14.86 and $14.84 - \infty$. Then each observation lies in range i of X-values and range j of Y-values, say. If we let n_{ij} denote the number of such observations occurring, then

$$\hat{p_{ij}} = \frac{n_{ij}}{n}$$

is an estimate of the probability of an observation lying in ranges (i,j) and an estimator for the probability of an observation X taking a value in range i is given by

$$\hat{p_{i\cdot}} = \sum_j p_{ij} = \sum_j \left(\frac{p_{ij}}{p_{\cdot j}}\right) p_{\cdot j} = \sum_j \left(\frac{n_{ij}}{n_{\cdot j}}\right) p_{\cdot j}$$

with

$$\mathbb{E}(\hat{p_{i\cdot}}) = p_{i\cdot} - \sum_j (1 - p_{\cdot j})^{n-1}(p_{ij} - p_{\cdot j}p_{i\cdot})$$

and

$$Var(\hat{p_{i\cdot}}) \approx p_{i\cdot}(1 - p_{i\cdot})/n - \sum_j (p_{ij} - p_{i\cdot}p_{\cdot j})^2/np_{\cdot j}$$

(See Fieller and Hartley (1954) for derivations).

Therefore if X and Y are independent, the use of the variate Y accomplishes nothing, however if $p_{ij} \neq p_{i\cdot}p_{\cdot j}$, then the variance of $\hat{p_{i\cdot}}$ is reduced, compared with that obtained when Y is not used.

However, this decrease in variance is at the expense of introducing bias to the estimator. In practice, this is often a reasonable trade-off so long as the bias is not "too" large.

The method of antithetic variables finds two correlated estimators of θ and combines them. Suppose that we have two estimates $\hat{\theta}_1$ and $\hat{\theta}_2$ of the same parameter θ, with identical means and variances, but negative covariance. Assume that the "costs" of computing the two are identical. Define $\hat{\theta}_3 = \frac{1}{2}(\hat{\theta}_1 + \hat{\theta}_2)$. Then

$$\begin{aligned} Var(\hat{\theta}_3) &= \frac{1}{4}\{Var(\hat{\theta}_1) + Var(\hat{\theta}_2) + 2Cov(\hat{\theta}_1, \hat{\theta}_2)\} \\ &= \frac{1}{2}\{Var(\hat{\theta}_1) + Cov(\hat{\theta}_1, \hat{\theta}_2\} \\ &< \frac{1}{2}Var(\hat{\theta}_1) \end{aligned}$$

The cost of computing $\hat{\theta}_3$ is scarcely more than twice the cost of computing either of the original two estimates, but the variance is more than halved, so there is an advantage in terms of cost-effectiveness in using $\hat{\theta}_3$.

Of course we need to find two estimators which are negatively correlated, but this can be done as follows. Given a distribution function F, with inverse F^{-1}, if $U \sim U[0,1]$, then $1-U \sim U[0,1]$. So $X_1 = F^{-1}(U)$ and $X_2 = F^{-1}(1-U)$ are distributed with distribution function F, but $Corr(X_1, X_2) < 0$.

Proof: (brief)

We know that $F^{-1}(U)$ is an increasing function in U. Let $G = \mathbb{E}(F^{-1}(1-U))$, and V $= 1 - \inf\{u | F^{-1}(U) > G\}$, i.e. $V = 1 - F(G)$. Then

$$\begin{aligned} Cov(F^{-1}(U), F^{-1}(1-U)) &= \mathbb{E}\left(F^{-1}(U)[F^{-1}(1-U) - G]\right) \\ &= \int_0^V F^{-1}(u)[F^{-1}(1-u) - G]du + \int_V^1 F^{-1}(u)[F^{-1}(1-u) - G]du \\ &< F^{-1}(V)\int_0^1 [F^{-1}(1-u) - G]du \\ &= 0. \end{aligned}$$

□

For example, returning to the Cauchy example the antithetic variable estimator for $\mathbb{P}(X > 2)$ is given by

$$\frac{1}{2} - \frac{2}{n}\sum_{i=1}^{n}\left[\frac{1}{\pi(1+u_i^2)} + \frac{1}{\pi(1+(2-u_i)^2)}\right]$$

where $u_i \sim U[0,2]$.

Exercise: Check.

Example

Say that we wish to estimate the value of π. We know that

$$\frac{\pi}{4} = \int_0^1 \sqrt{1-x^2}dx \tag{3.4}$$

since each side of equation (3.4) represents the area of a quadrant of a circle of radius unity.

If we let $P = \sqrt{1-U^2}$, where $U \sim U(0,1)$, then $\mathbb{E}(P) = \frac{\pi}{4}$. Thus, if we sample $U_1, ..., U_n$ from $U(0,1)$, then

$$\bar{P} = \frac{1}{n}\sum_{i=1}^{n} P_i = \frac{1}{n}\sum_{i=1}^{n}\sqrt{1-U_i^2}$$

is an estimate of $\frac{\pi}{4}$. We would term this a "crude" Monte Carlo estimate of π. Clearly,

$$\begin{aligned}
Var(\bar{P}) &= \frac{1}{n}Var(\sqrt{1-U^2}) \\
&= \frac{1}{n}\left(\int_0^1 (1-x^2)dx - \left[\int_0^1 \sqrt{1-x^2}dx\right]^2\right) \\
&= \frac{1}{n}\left(\frac{2}{3} - \frac{\pi^2}{16}\right) \\
&\approx 0.498/n
\end{aligned}$$

To improve on the variance of this estimator, we can introduce an antithetic variable, as discussed in Section 3.2.2. The antithetic variable for U is $1-U$, since they both have the same distribution and are negatively correlated (therefore U^2 and $(1-U)^2$ bust be positively correlated). If we let

$$Q = \frac{1}{2}\left(\sqrt{1-U^2} + \sqrt{1-(1-U)^2}\right)$$

then $\mathbb{E}(Q) = \frac{\pi}{4}$, as before but

$$\begin{aligned}
Var(Q) &= \frac{1}{2}Var(\sqrt{1-U^2}) + \frac{1}{2}Var(\sqrt{1-[1-U]^2}) + Cov(\sqrt{1-U^2}, \sqrt{1-(1-U)^2}) \\
&= \frac{1}{2}\left(\frac{2}{3} - \frac{\pi^2}{16}\right) + \mathbb{E}\left(\sqrt{1-U^2}\sqrt{1-(1-U)^2} - \frac{\pi^2}{16}\right) \\
&\leq \frac{1}{2}\left(\frac{2}{3} - \frac{\pi^2}{16}\right)
\end{aligned}$$

Therefore,

$$Var(\bar{Q}) \leq Var(\bar{P})$$

and in fact $Var(\bar{Q}) = 0.084/n$, a considerable improvement upon the "crude" method.

Note that Buffon's needle experiment is another example of a Monte Carlo simulation method.

3.3 Monte Carlo Tests

Suppose that we have a null hypothesis H_0, represented by a completely specified model, and a test statistic T, such as a goodness-of-fit statistic, for which small values indicate departure from H_0.

To perform a pure significance test, we need to know the distribution of T under H_0. This may be difficult or impossible to obtain analytically, but it may be possible to simulate from the model to produce m samples and the corresponding values $t_1, ..., t_m$ of T. We could then estimate the distribution of T under H_0 by the empirical distribution of $\{t_1, ..., t_m\}$ or a smoothed version thereof. Informally, we estimate the critical point of a level α test by the $100\alpha^{th}$ percentile of $\{t_1, ..., t_m\}$.

Monte Carlo tests were introduced by Barnard (1963) and are closely related to this idea. If H_0 is true, and we have $m+1$ samples from T, m by simulation, and one by observation, t, say. Then the probability that t is the k^{th} smallest or smaller, is given by $k/(m+1)$, ignoring ties. [Assume

k \ u	0.1	0.075	0.05	0.025	0.01
1	0.135	0.227	0.377	0.618	0.826
2	0.088	0.199	0.413	0.745	0.942
3	0.057	0.171	0.429	0.817	0.979
4	0.038	0.148	0.438	0.864	0.992
5	0.025	0.128	0.445	0.897	0.997
10	0.004	0.065	0.461	0.971	-

Table. 3.1: Table of values of $p(u)$ with $\alpha = 0.05$.

that T has a continuous distribution]. If we choose k and m so that $\alpha = k/(m+1)$ is a conventional significance level (i.e.: 0.05 or 0.01), then we have a "Monte Carlo Test".

The Monte Carlo Test has a random critical point and so "blurs" the critical region, though the test is "exact": $\mathbb{P}(Type\,I\,error) = \alpha$. Let F be the cdf of T under H_0 recall that $U = F(T) \sim U[0,1]$. A "conventional" test of size α, rejects the null hypothesis if

$$u = F(t) = \mathbb{P}(T \leq t|H_0) < \alpha. \tag{3.5}$$

The Monte Carlo Test rejects H_0 if

$$\begin{aligned} t &< t_{(k)} && (3.6) \\ \text{or } u &\leq F(t_{(k)}) && (3.7) \end{aligned}$$

where $t_{(k)}$ is the t^{th} order statistic of the t_i and u is a standard Uniform random variate. So, the Monte Carlo Test rejects H_0 with probability $p(u)$, where

$$p(u) = \sum_{r=0}^{k-1} \binom{m}{r} u^r (1-u)^{m-r}.$$

This should be interpreted as the proportion of times the Monte Carlo Test will reject H_0 with $t = F^{-1}(u)$ as observation. Marriott (1979) tabulates $p(u)$, for a variety of α-values. See Table 3.1.

For small k the effect of blurring can be substantial. However, the conventional test in (3.5) and the Monte Carlo test will only give different decisions a significant proportion of the time when t corresponds to a p-value close to α.

3.3.1 Power Loss for Monte Carlo Tests

If we compare the power function $\beta^m(\alpha)$, of the Monte Carlo Test with the power $\beta(\alpha)$, of the conventional test, we might expect that $\beta^m < \beta$. In fact this is true, but how large is the loss in power?

The conventional test in (3.5) rejects H_0 if $T < F^{-1}(\alpha)$, and has power against an alternative denoted by F_θ of

$$\begin{aligned} \beta(\alpha) &= \mathbb{P}(T < F^{-1}(\alpha)|T \sim F_\theta) \\ &= F_\theta(F^{-1}(\alpha)). \end{aligned}$$

Similarly, the power of the Monte Carlo test of equation (3.6) against this alternative is given by

$$\begin{aligned} \beta^m(\alpha) &= \mathbb{P}(T < T_{(k)} | T \sim F_\theta, T_1, ..., T_m \sim F) \\ &= \int_{-\infty}^{\infty} \sum_{r=0}^{k-1} \left(\begin{array}{c} m \\ r \end{array} \right) F(t)^r [1 - F(t)]^{m-r} dF_\theta(t). \end{aligned}$$

Exercise. (Ripley 1987, p.172).

Given that $F^{-1}(T_{(k)})$ has a beta distribution with parameters $(k, m+1-k)$

$$\beta^m(\alpha) = \int_0^1 \beta(\xi) b(\alpha, m, \xi) d\xi \tag{3.8}$$

where $b(\alpha, m, \cdot)$ is the pdf of the beta distribution with parameters $\alpha(m+1)$ and $(1-\alpha)(m+1)$.

Theorem 3.1 (Jockel, 1986): *Suppose that $\beta(\cdot)$ is concave on $[0,1]$. Then*

$$\beta^m(\alpha) \uparrow \beta(\alpha) \text{ as } m \to \infty.$$

Proof:

Equation (3.8) implies that

$$\beta^{m+1}(\alpha) - \beta^m(\alpha) = \int_0^1 \{b(\alpha, m+1, \xi) - b(\alpha, m, \xi)\} \beta(\xi) d\xi.$$

There are points ξ_1 and ξ_2 with $0 < \xi_1 < \xi_2 < 1$ such that the integrand is zero at these points, positive on (ξ_1, ξ_2) and negative on $(0, \xi_1)$ and $(\xi_2, 1)$. Let

$$L(\xi) = \beta(\xi_1) + [\beta(\xi_2) - \beta(\xi_1)][\xi - \xi_1]/(\xi_2 - \xi_1),$$

so $L \leq \beta$ on (ξ_1, ξ_2) and $L \geq \beta$ elsewhere (by concavity). Then

$$\beta^{m+1}(\alpha) - \beta^m(\alpha) = \int_0^1 L(\xi) \{b(\alpha, m+1, \xi) - b(\alpha, m, \xi)\} d\xi = 0,$$

as required, since both densities have mean α □

So if m is big, the Monte Carlo Test is a "good" test. The amount we lose by using a Monte Carlo test can be bounded:

Theorem 3.2: *Suppose that $\beta(\cdot)$ is concave on $[0,1]$ with $\beta(0) = 0$ and $\beta(1) = 1$. Then*

$$\begin{aligned} \frac{\beta^m(\alpha)}{\beta(\alpha)} &\geq 1 - \frac{\mathbb{E}(|Z - \alpha|)}{2\alpha} \\ &\simeq 1 - \left[\frac{(1-\alpha)}{2\pi m \alpha} \right]^{\frac{1}{2}}, \end{aligned}$$

where $Z \sim beta(\alpha(m+1), (1-\alpha)(m+1))$.

For $\alpha = 0.05$, the approximate bound on $\beta^m(\alpha)/\beta(\alpha)$ is 0.64 at $m = 19$, 0.83 at $m = 99$ and 0.95 at $m = 999$. Jockel (1986) gives more specific results on asymptotic relative efficiency.

In summary, study of Monte Carlo Tests suggest avoiding small values of k, which blur the critical region and lose power. Since $\alpha = k/(m+1)$, this implies that m is large. Usually $m = 99$ will be sufficient.

3.4 An Introduction to the Bootstrap

A study was performed to see if regular, small doses of Aspirin were an effective means of preventing strokes. The study provided the following results.

	strokes	subjects
Aspirin Group	119	11037
Placebo Group	98	11034

We can compare the effect of the two treatments (Aspirin and Placebo) by looking at the ratios of the proportions of strokes (or the rate of strokes) in the two groups. In this case, the ratio of the rates is given by

$$\hat{\theta} = \frac{119/11037}{98/11034} = 1.21,$$

and this suggests that taking Aspirin is actually harmful. However, the interval for the true stroke ratio, θ, is given by

$$0.93 < \theta < 1.59, \tag{3.9}$$

with 95% confidence. This includes the neutral value $\theta = 1$, which would mean that Aspirin takers were no worse nor better off than those taking the Placebos.

The Bootstrap is a data-based simulation method for statistical inference, which can be used to produce inferences like (3.9). The use of the term *bootstrap* derives from the phrase "to pull oneself up by ones bootstrap", widely attributed to Rudolph Raspe, who wrote the *Adventures of Baron Munchausen* in which the "baron falls to the bottom of a well. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps".

Here is how the bootstrap works in the stroke example. First, we create two populations, the first consisting of 119 1's and 10918 (11037-119) 0's, and the second consisting of 98 1's and 10936 0's. We draw with replacement, 11037 items from the first population and 11034 from the second, forming two *bootstrap samples*. From these, we derive the bootstrap replicate of $\hat{\theta}$ given by

$$\hat{\theta}^* = \frac{\text{Proportion of 1's in bootstrap sample number 1}}{\text{Proportion of 1's in bootstrap sample number 2}}.$$

We repeat this process a large number (B) of times to obtain B bootstrap replicates $\hat{\theta}^*$. We can then use these replicates to form a confidence interval for θ. For example, a rough 95%, non-parametric interval can be derived by taking the $0.025B$th and $0.975B$th largest of the B replicates. In a simulation with $B = 1000$, the bootstrap confidence interval was given as $(0.93, 1.60)$, which agrees very closely with the theoretical confidence interval.

Bootstrap methods are intended to simplify the calculation of inferences like that of (3.9), producing them in an automatic manner, even in situations which are much more complicated than this simple Aspirin study.

Most data analyses take the form of obtaining observations $\boldsymbol{x} = \{x_1, ..., x_n\}$ from some unknown probability distribution function F and constructing an estimate $\hat{\theta}(\boldsymbol{x})$ for some parameter of interest θ, given these observations. Having obtained $\hat{\theta}(\boldsymbol{x})$, we then wish to assess it's accuracy as an estimate of the true value, θ. If $\hat{\theta}$ is unbiased, then the most common measure of this accuracy is the standard error given by

$$SE(\hat{\theta}) = \sqrt{(Var_F(\hat{\theta}))}.$$

For example if $\theta = \mathbb{E}_F(\boldsymbol{X})$, then

$$\hat{\theta}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

is an unbiased estimator for θ. Further, if

$$\sigma^2(F) = Var_F(X),$$

then

$$SE(\hat{\theta}) = \left(\frac{\sigma^2(F)}{n}\right)^{1/2}. \tag{3.10}$$

In this case, $\sigma^2(F)$ is a function of the unknown distribution function F, but this can also be estimated from the data by

$$\hat{\sigma}^2(\boldsymbol{x}) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}.$$

So that an estimate of $SE(\hat{\theta})$ is given by

$$\widehat{SE}(\hat{\theta}) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n(n-1)}}. \tag{3.11}$$

Thus, if $\theta = \mathbb{E}_F(X)$, the formula for the standard error is quite simple and easily estimable. However, for other estimators such as the sample correlation coefficient, such formulae do not exist. Traditionally the statistical literature has concentrated upon obtaining approximations to (3.10) for particular estimators, avoiding estimators which do not have such formulae. However, over the past 30 years significant advances have been made in this area, firstly by the introduction of the Jackknife estimate of error, and then by the Bootstrap, both of which we shall discuss in the next section.

3.5 Empirical Distributions, the Jackknife and Bootstrap

3.5.1 Empirical Distributions

As we have already seen, problems of statistical inference often involve estimating some aspect of a probability distribution F on the basis of a random sample from that distribution. The empirical distribution function, which we denote by $\hat{F}$, is a simple estimate of the entire distribution function, F. An obvious way to estimate some interesting aspect of F, for example its mean or median, is to use the corresponding aspect of $\hat{F}$. We call this the *plug-in principle*.

Having observed a random sample, $(x_1, ..., x_n)$, of size n from a population with distribution function F, the empirical distribution function is defined to be the discrete distribution that puts probability $1/n$ on each value x_i, $i = 1, ..., n$. Thus, for example, we estimate $\mathbb{P}(x \in A)$, for some A by $\mathbb{P}_{\hat{F}}(x \in A)$, given by

$$\begin{aligned}\mathbb{P}_{\hat{F}}(x \in A) &= (\text{Number of } x_i \text{ in } A)/n \\ &= \frac{1}{n}\sum_{i=1}^{n} I_A(x_i)\end{aligned}$$

i.e., the proportion of the sample occurring within the set A.

An example of the empirical distribution function for 20 values simulated from a standard normal distribution is shown in Figure 3.3, together with the true standard normal distribution function.

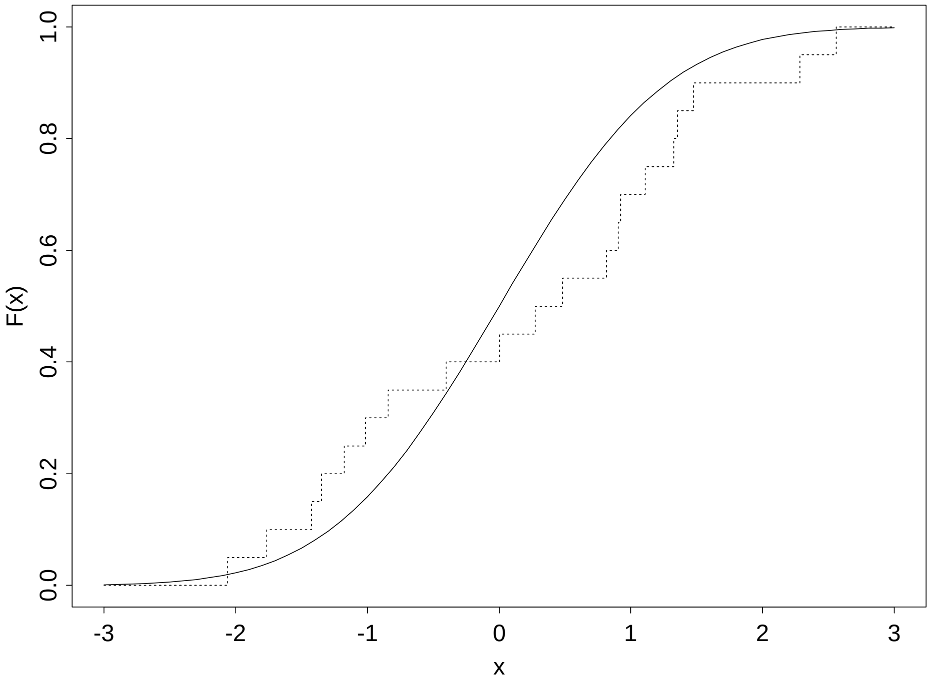


Figure. 3.3: Comparison of the true and empirical distribution functions.

Example

Say we rolled a die 100 times. The outcomes might look as follows.

6	3	2	4	6	6	6	5	3	6	2	2	6	2	3	1	5	1	6	6
4	1	5	3	6	6	4	1	4	2	5	6	6	5	5	3	6	2	6	6
1	4	1	5	6	1	6	3	3	2	2	2	5	2	2	4	1	4	5	6
6	6	2	2	4	6	1	2	2	2	5	1	5	3	5	4	2	1	4	6
6	5	6	4	6	4	3	6	4	1	4	5	4	4	2	3	2	1	4	6

The empirical distribution $\hat{F}$ puts probability $1/100$ on each of the 100 outcomes. In cases like this, where there are repeated values, we can express $\hat{F}$ more economically as the vector of *observed frequencies* $\hat{f}_k$, $k = 1, ..., 6$, where

$$\hat{f}_k = \frac{1}{100} \sum_{i=1}^{100} I_{\{x_i = k\}}(x_i).$$

For the above data, we get

$$\hat{f} = (0.13,\ 0.19,\ 0.10,\ 0.17,\ 0.14,\ 0.27).$$

It can be shown that $\hat{f} = (\hat{f}_1, ..., \hat{f}_k)$ is a *sufficient statistic* for F, thus we lose no information in reducing the empirical distribution to this form by pooling repeated observations.

The most obvious question to ask, given the above data, is whether or not the die is a "fair" one i.e., does $f = (1/6, 1/6, ...)$. We only have the sample data, so any answer to this question, must be based upon the empirical frequencies. Generally, discussions of statistical inference are phrased in terms of *parameters* and *statistics*. A parameter is a function of the probability distribution, F. A statistic is a function of the sample. Thus f_k is a parameter of F, while $\hat{f}_k$ is a statistic.

It is usual to write parameters directly as a function of F, say

$$\theta = \theta(F).$$

This notation emphasises the direct dependence of θ upon F. For example, if F is a probability distribution upon the real line, the expectation can be thought of as the parameter

$$\theta = \theta(F) = \mathbb{E}_F(X).$$

The Plug-in Principle

The plug-in principle is a simple method of estimating parameters from samples. The *plug-in estimate* of a parameter $\theta(F)$ is defined to be

$$\hat{\theta} = \theta(\hat{F}).$$

In other words, we estimate the function $\theta(F)$ of the probability function F by the same function of the empirical distribution, $\hat{F}$. But how good is the plug-in principle?

It is actually quite good if the only available information about F comes from the sample $\boldsymbol{x}$. Under this circumstance, $\hat{\theta} = \theta(\hat{F})$ is "optimal" in the traditional sense. For example, if $\hat{f}_k$ is the plug-in frequency estimate, then

$$\hat{f}_k \sim Bin(n, f_k)/n.$$

In this case, the estimator $\hat{f}_k$ is *unbiased* for f_k (i.e., $\mathbb{E}(\hat{f}_k) = f_k$) and has variance $f_k(1-f_k)/n$. This is the smallest possible variance for *any* unbiased estimator of f_k.

We will use the bootstrap to study the bias and standard error of the plug-in estimator $\hat{\theta}$. The bootstrap's virtue is that it produces biases and standard errors in an automatic way, no matter how complicated the the functional θ may be.

The plug-in principle is less effective in situations where there is information about F other than that provided by the sample. We might know, or assume, that F is a member of a *parametric family* of distributions, such as the family of normal distributions. The plug-in principle can be adapted to incorporate this additional information, but we shall restrict our attention to the case where we have only the one random sample from an unknown distribution, F. This is called the *one-sample non-parametric* setup.

Before we discuss the Bootstrap approach, we will introduce the related idea of the Jackknife.

3.5.2 The Jackknife

This technique was introduced as a method for reducing the bias of a serial coefficient estimator, based upon splitting the sample into two half samples. The original method was subsequently generalised by splitting the sample into g groups of size h, with $g = nh$. The method of the Jackknife was then based upon this generalised method, method with $g = n$ and $h = 1$, and avoided the, often analytically complex, equation (3.10) altogether, going directly to a generalisation of (3.11) as follows. Let $\boldsymbol{x}_{(i)}$ be the data set with the i^{th} datum removed so that

$$\boldsymbol{x}_{(i)} = \{x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n\},$$

and let $\hat{\theta}_{(i)}$ be the statistic $\hat{\theta}$ evaluated for the data set $\boldsymbol{x}_{(i)}$. Then the jackknife estimate of the standard error of $\hat{\theta}$ is given by

$$\widehat{SE}_{jack}(\hat{\theta}) = \left(\frac{n-1}{n} \sum_{i=1}^{n} (\hat{\theta}_{(i)} - \bar{\theta}_{(\cdot)})^2 \right)^{1/2}, \tag{3.12}$$

where

$$\bar{\theta}_{(.)} = \frac{\sum_{i=1}^{n} \hat{\theta}_{(i)}}{n}.$$

Clearly if $\hat{\theta} = \bar{x}$ then

$$\bar{\theta}_{(i)} = \frac{1}{n-1} \sum_{j \neq i} x_j, \text{ and } \bar{\theta}_{(\cdot)} = \bar{x},$$

so (3.12) reduces to (3.11).

As an alternative to the standard error, we might also be interested in the bias of our estimator. The bias of an estimator $\hat{\theta}$ of θ is defined as

$$bias = \mathbb{E}(\hat{\theta}) - \theta.$$

We can define the Jackknife estimate of bias to be

$$bias_{jack} = (n-1)(\mathbb{E}_{\hat{F}}(\hat{\theta}) - \hat{\theta}).$$

That is, the difference between the expected value of the empirical distribution and the estimated value, so this is simply

$$(n-1)(\bar{\theta}_{(.)} - \hat{\theta}).$$

Some theoretical justification of these results can be found in Efron and Tibshirani (1986) on pages 141–143.

Once again, if $\hat{\theta} = \bar{x}$, then the estimate of the bias is zero i.e., we have an unbiased estimator.

The Jackknife method (for standard errors) can be thought of algorithmically as follows.

STEP 1. SET $i = 1$.

STEP 2. REMOVE THE iTH OBSERVATION AND CALCULATE $\hat{\theta}_{(i)}$ FROM THE REMAINING OBSERVATIONS.

STEP 3. INCREASE i AND IF $i \leq n$, RETURN TO STEP 2, ELSE CONTINUE TO STEP 4.

STEP 4. CALCULATE $\widehat{SE}_{jack}(\hat{\theta})$.

This algorithm may be implemented using the following S-Plus function.

```
myjackknife <- function(x, theta) {
        n <- length(x)
        u <- rep(0, n)

        for(i in 1:n) {
                u[i] <- theta(x[ - i])
        }

        thetahat <- theta(x)
        jack.bias <- (n - 1) * (mean(u) - thetahat)
        jack.se <- sqrt(((n - 1)/n) * sum((u - mean(u))^2))

        return(jack.se, jack.bias, jack.val = u)
}
```

where `theta` is the function that we wish to jackknife.

Example

We consider here a set of data relating two score tests, LSAT and GPA, at a sample of 15 American law schools. We are interested in the correlation between these measurements, and we have the following observational data.

LSAT	576	635	558	578	666	580	555	661
GPA	3.39	3.30	2.81	3.03	3.44	3.07	3.00	3.43
LSAT	651	605	653	575	545	572	594	
GPA	3.36	3.13	3.12	2.74	2.76	2.88	2.96	

We can estimate the correlation coefficient between the two scores and this gives $\hat{\theta} = 0.776$, but how accurate is this estimate? We can assess this using the Jackknife algorithm. In this case, we want to sample from the rows of the data matrix; this requires care in the use of the S-Plus algorithm above, since we anticipated that `x` would be a vector. We get around this problem by setting `x` to be the vector `1:n`. Then we place the data matrix inside the function `theta()` directly. Thus, we define

```
theta <- function(x){
         cor(data[x,1],data[x,2])
}
```

where data is the data matrix, defined by

```
> data <- matrix(c(576,635,558,578,666,580,555,661,651,605,653,575,
          545,572,594,3.39,3.30,2.81,3.03,3.44,3.07,3.00,3.43,3.36,
          3.13,3.12,2.74,2.76,2.88,2.96),nrow=15,byrow=F)
```

Then, if we type `myjackknife(1:15,theta)`, we get the following output.

```
$jack.se:
[1] 0.1425188

$jack.bias:
[1] -0.006473279

$jack.val:
 [1] 0.8929472 0.7637067 0.7549983 0.7760967 0.7313195 0.7799686 0.7845358
 [8] 0.7361617 0.7517390 0.7761231 0.8181007 0.7857184 0.7403508 0.7670412
[15] 0.7798725
```

Note how the return command leads to the output above. If we had typed

```
> out <- myjackknife(1:15,theta)
```

then we could refer to the jackknife estimate of the standard error as `out$jack.se`.

The final S-Plus object to be returned is `out$jack.val`, which lists the $\hat{\theta}_{(i)}$ values. By subtracting $\hat{\theta}$ from each of these values, we can see which observations have the largest *influence* upon the correlation between the samples. Note also, that `mean(out$jack.val)`$= \hat{\theta}$.

3.5.3 The Bootstrap

The bootstrap was proposed in an attempt to better understand the jackknife. Instead of bypassing (3.10) as for the jackknife, Efron defined an empirical estimate, $\hat{F}$, given by

$$\hat{F}(x) = \frac{i}{n} \quad x_{(i)} \leq x < x_{(i+1)}.$$

Then

$$\sigma^2(\hat{F}) = \frac{1}{n}\sum_{i=1}^{n}(x_i^* - \bar{x}^*)^2,$$

where the ${x_i}^*$ are independent, identically distributed random variables drawn from $\hat{F}$, i.e.: the data set $\boldsymbol{X}^*$ consists of n observations randomly sampled with replacement from $\{x_1, ..., x_n\}$. Then,

$$\hat{\sigma}_{boot} = \sqrt{Var_{\hat{F}}\hat{\theta}(\boldsymbol{X}^*)}.$$

Therefore, if $\theta = \mathbb{E}_F(\boldsymbol{X})$, then

$$\begin{aligned}\widehat{SE}_{boot}(\hat{\theta}) &= \sqrt{\frac{1}{n^2}\sum_{i=1}^{n}(x_i^* - \bar{x}^*)^2} \\ &= \sqrt{\frac{n-1}{n}}\hat{SE}(\hat{\theta}).\end{aligned}$$

It can be shown that, asymptotically, SE_{boot} agrees with SE_{jack} and further, that the jackknife is in fact a linear approximation to the bootstrap.

Obviously we can gain an empirical estimate of $\hat{\sigma}_{boot}$ by randomly selecting a large number of bootstrap samples, $\boldsymbol{x}_1^*, ..., \boldsymbol{x}_B^*$, and calculating $\hat{\theta}(\boldsymbol{x}_i^*)$ for each. Thus we obtain

$$\widehat{SE}_{boot}(\hat{\theta}) = \sqrt{\frac{\sum_{b=1}^{B}\left(\hat{\theta}(\boldsymbol{x}_b^*) - \bar{\theta}(\cdot)\right)^2}{B-1}}.$$

This approximation to the true standard error can be made arbitrarily accurate by taking suitably large B. This is the essential difference between the jackknife and the bootstrap, since we can take as many bootstrap samples as we like, but we are restricted to only n jackknife samples.

The bootstrap algorithm can be written as follows.

STEP 1. EVALUATE THE EMPIRICAL DISTRIBUTION, $\hat{F}$.
STEP 2. SIMULATE B SAMPLES $\boldsymbol{x}_1^*, ..., \boldsymbol{x}_B^*$, EACH OF SIZE n FROM $\hat{F}$.
STEP 3. ESTIMATE THE POPULATION BOOTSTRAP VARIANCE AS THE SAMPLE VARIANCE

$$Var_{\hat{F}}(\hat{\theta}) = \frac{1}{B-1}\sum_{b=1}^{B}(\hat{\theta}(\boldsymbol{x}_b^*) - \bar{\theta}(\cdot))^2,$$

WHERE

$$\bar{\theta}(\cdot) = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}(\boldsymbol{x}_b^*).$$

So, there are two aspects to the bootstrap procedure: replacement of F by $\hat{F}$; and simulation from $\hat{F}$ to form estimates of functionals of interest. We can implement this algorithm in S-Plus, using the following code.

```
mybootstrap <- function(x, nboot, theta) {
        n <- length(x)

        bootsam <- matrix(sample(x, size = n * nboot, replace = T), nrow =
                nboot)

        thetastar <- apply(bootsam, 1, theta)
        thetastar
}
```

where `theta()` is the function that we wish to bootstrap.

Example

If we return to the earlier example in which we examine the correlation between scores in two different Law exams, we can use the `mybootstrap()` function to obtain a set of bootstrap replicates $\hat{\theta}_i^*$, $i = 1, ..., B$ as follows.

```
> out <- mybootstrap(1:15,200,theta)
> sqrt(var(out))
[1] 0.1294239
```

So, `out` contains the bootstrap replicates $\hat{\theta}_i^*$, $i = 1, ..., B$ (see the histogram given in Figure 3.4), and the bootstrap estimate of the standard error is given as 0.129, as opposed to the 0.143, given by the jackknife method earlier.

Example

Define

$$p_1 = \mathbb{P}[\sqrt{n}(\bar{X} - \mu) \leq \alpha],$$

where

$$\mu = \mathbb{E}(X_i), \text{ and } \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

We wish to estimate p_1, given data $\{x_1, ..., x_n\}$.

Step 1. Construct $\hat{F}$ by assigning mass $\frac{1}{n}$ to each of the x_i, $i = 1, ..., n$.

Step 2. Draw a bootstrap sample from $\hat{F}$, $\boldsymbol{x}^* = \{x_1^*, ..., x_n^*\}$, i.e.: draw each x_i^* independently , with replacement from $\{x_1, ..., x_n\}$. Calculate

$$\hat{\theta}(\boldsymbol{x}^*) = \sqrt{n}(\bar{x}^* - \bar{x}).$$

Step 3. Repeat step 2 B times to get $\hat{\theta}(\boldsymbol{x}_1^*), ..., \hat{\theta}(x_B^*)$, then

$$\hat{p_1} = \frac{1}{B}\sum_{b=1}^{B} I[\hat{\theta}(\boldsymbol{x}_b^*) \leq \alpha].$$

Then $\hat{p_1} \rightarrow p_1$ as $B \rightarrow \infty$.

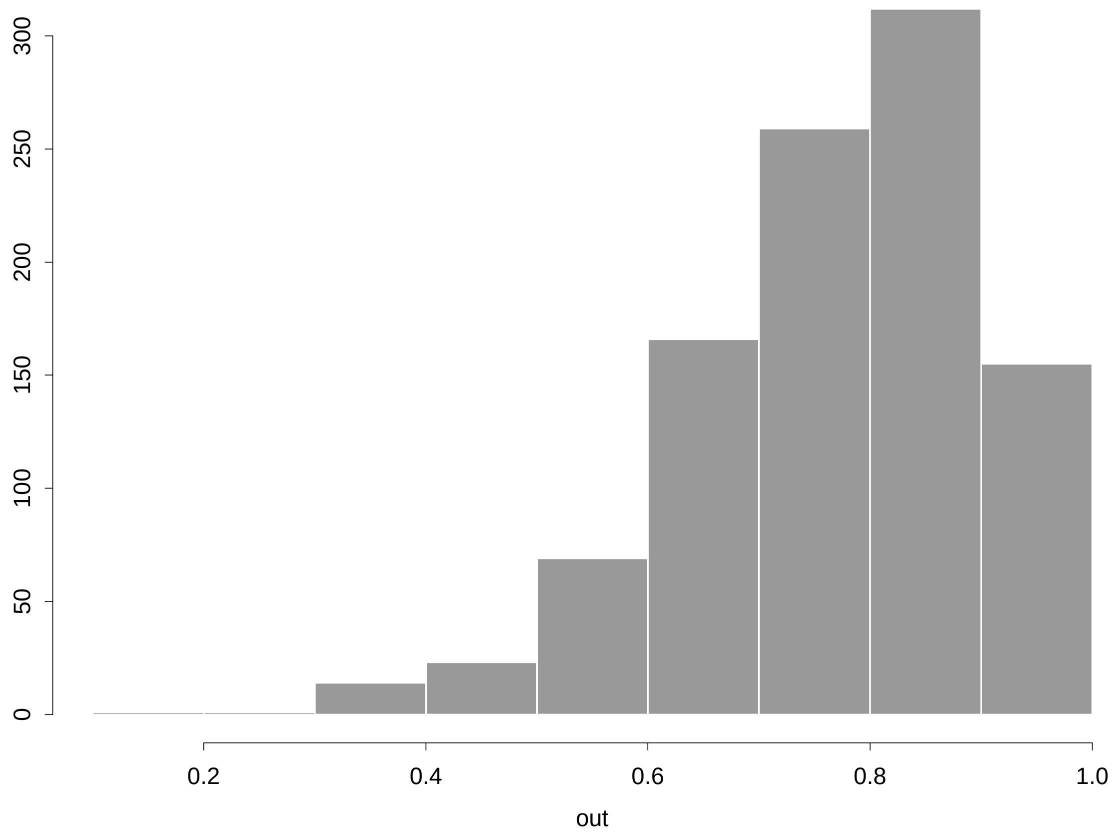


Figure. 3.4: Histogram of the 1000 bootstrap samples for θ.

3.5.4 The Bayesian Bootstrap

The Bayesian bootstrap is analogous to the bootstrap, but simulates the posterior distribution of the parameter θ, rather than the estimated sampling distribution of $\hat{\theta}$ (an estimate of θ).

Each Bayesian bootstrap replication generates a posterior probability for each $\boldsymbol{x}_i$, where the values of $\boldsymbol{X}$ that are not observed have zero posterior in the same way that they have zero probability under the sample cumulative distribution function. The posterior probability is centred at $\frac{1}{n}$, but has some degree of variability. We introduce this variability by generating $n-1$ random variates from a $U(0,1)$ distribution, $u_1, ..., u_{n-1}$, for each replication. Then we let

$$g_i = u_{(i)} - u_{(i-1)}, \ i = 1, ..., n$$

where $u_{(i)}$ is the i^{th} smallest u_i, with $u_{(0)} = 0$ and $u_{(n)} = 1$. Then $\boldsymbol{g} = (g_1, ..., g_n)$ is the vector of probabilities associated with the data values $(x_1, ..., x_n)$ for that replication. Repeated replications will then give the Bayesian bootstrap distribution of $\boldsymbol{X}$, and therefore of any parameter of this distribution.

For example, if $\theta = \mathbb{E}_F(X)$, then at each replication, we calculate the mean of X as if g_i were the probability that $X = x_i$, i.e.: we calculate $\sum_{i=1}^n g_i x_i$. The distribution of the values of $\sum_{i=1}^n g_i x_i$ over all Bayesian bootstrap replications gives the Bayesian bootstrap distribution of the mean of X.

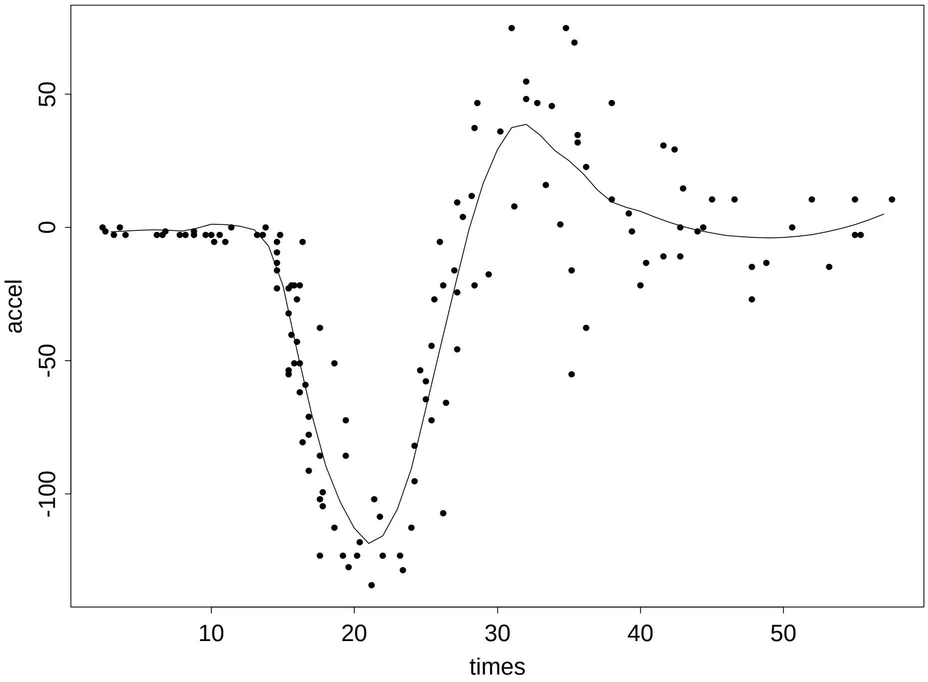


Figure. 3.5: Scatterplot of the motorcycle data.

In practice, the bootstrap and the Bayesian bootstrap differ only in how the probabilities are attached to the x_i. However, the interpretations of the resulting distributions are different. The Bayesian bootstrap generates likelihood statements about parameters rather than frequency statements about statistics under assumed values for parameters, and thus has an inherent advantage over the bootstrap.

3.6 More Bootstrap Examples

The traditional bootstrap method is easily extended to models of more detailed structure. Here we look at a couple of examples to illustrate the general principle.

3.6.1 Time Series

The time series in Figure 3.6 gives the annual number of lynx trappings in the Mackenzie River District of North-West Canada for the period 1821 to 1934 and is stored in Splus as `lynx`.

The objective is to model this time series and use bootstrapping to assess the accuracy of the fitted model. The challenge is to find a way of bootstrapping from the sample series to create new series which have the same stochastic structure as the original series. We will illustrate the procedure using the simple AR(1) model, though this isn't necessarily the best model for these data. The residuals from an AR(1) fit are shown in Figure 3.7; the evident skewness suggests that estimates of variance based on the usual Gaussian error model are unlikely to be valid. Thus, bootstrapping provides an alternative data-based estimate of parameter precision. One approach is given by the following algorithm.

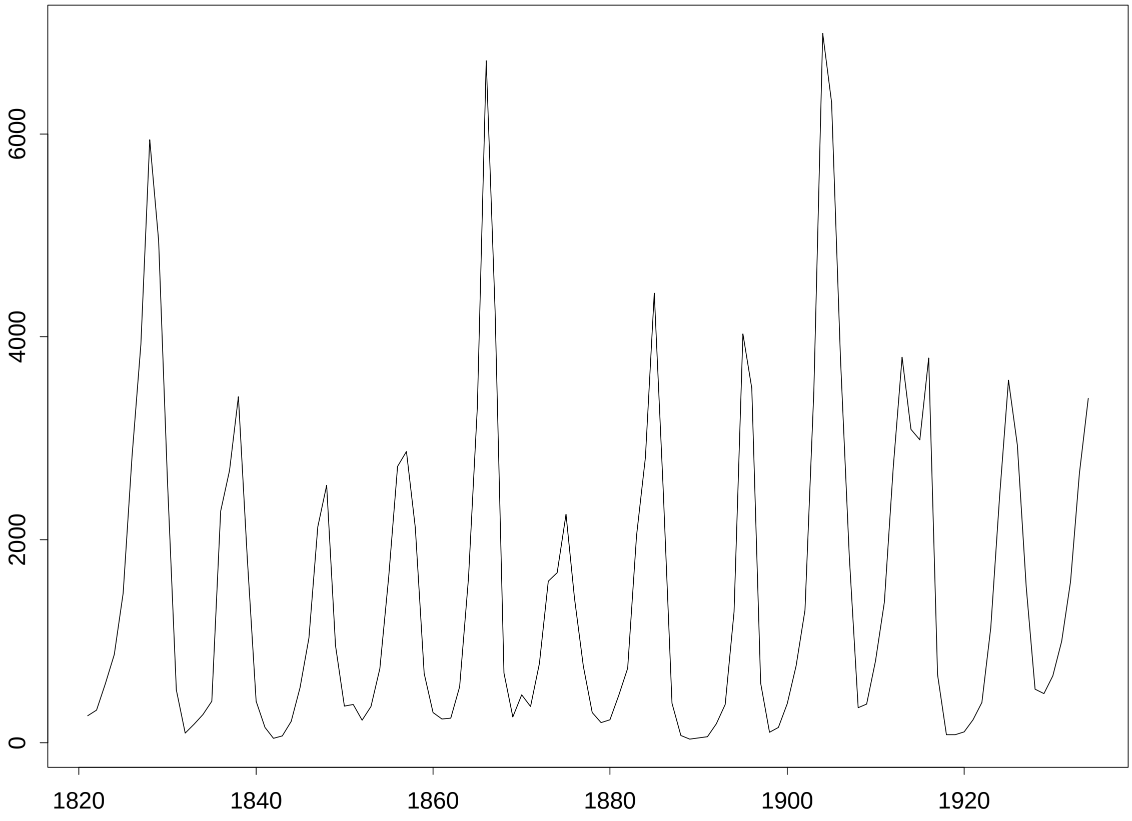


Figure. 3.6: Raw plot of the lynx data.

Step 1 Fit the AR(1) model to the data $x_t = \hat{\beta} x_{t-1} + \epsilon_t$.

Step 2 Obtain the residuals ϵ_t from the fitted model.

Step 3 Obtain the empirical distribution function $\hat{F}$ of the residuals.

Step 4 Simulate B bootstrap replications of the series by:

(a) Set $y_1 = x_1$

(b) Evaluate recursively $y_t = \hat{\beta} y_{t-1} + \epsilon_t^*$ where ϵ_t^* is a realisation from $\hat{F}$.

Step 5 Re-fit the AR(1) model to each bootstrapped time series to obtain B bootstrap realisations of $\hat{\beta}$.

Step 6 Summarise the required characteristics of the bootstrap distribution.

In this way, the bootstrap realisations of the time series are forced to have the correct model structure. Within Splus, this is all easily implemented using the following command

```
> mybootstrap(2:114,200,lynxtheta)
```

where `lynxtheta` is defined as

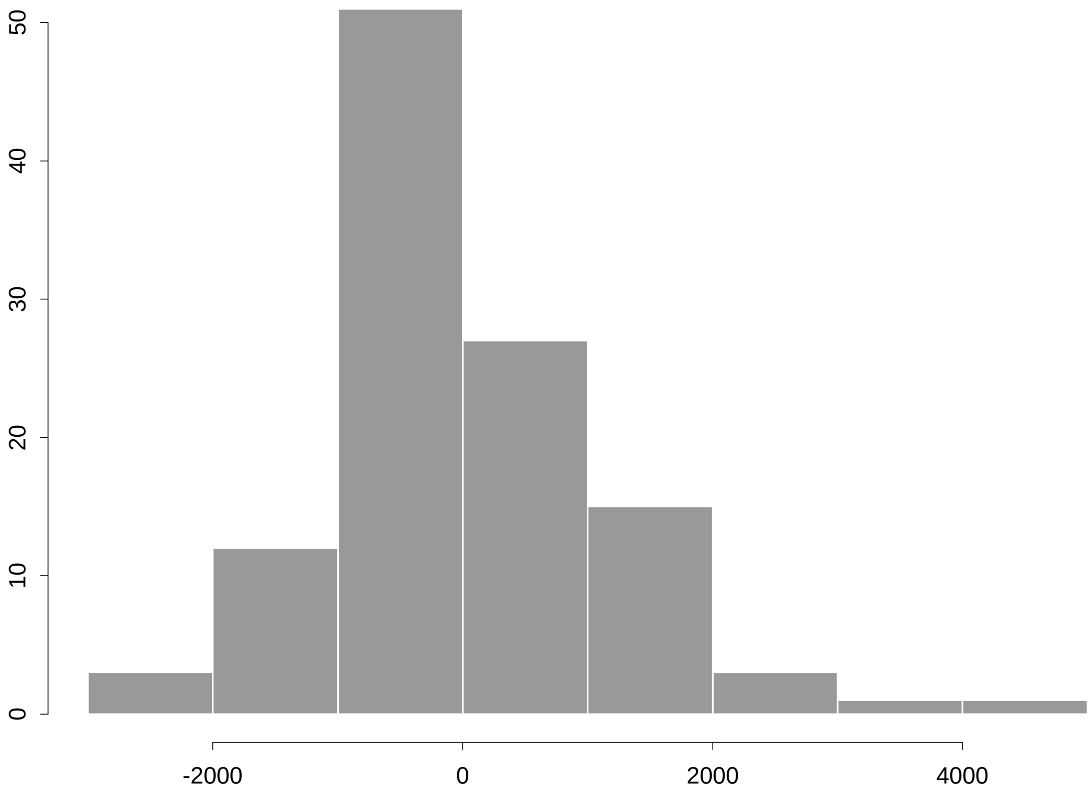


Figure. 3.7: Histogram of residuals of AR(1) fit to the lynx data.

```
lynxtheta <- function(x)
{
        a <- ar(lynx, aic=F, ord = 1)
        beta <- a$ar[1]
        resid <- a$resid
        x1 <- lynx[1]
        y <- NULL
        y[1] <- lynx[1]
        for(i in 1:(length(x)-1)) {
                y[i + 1] <- beta * y[i] + resid[x[i]]
        }
        ar.fit <- ar(y, aic = F, ord = 1)
        plot(y, type = "l")
        ar.fit$ar
}
```

For this example, Figure 3.8 shows 9 bootstrap realisations of the series. Note that the simulated series look somewhat different from the original. This is because the AR(1) model does not describe

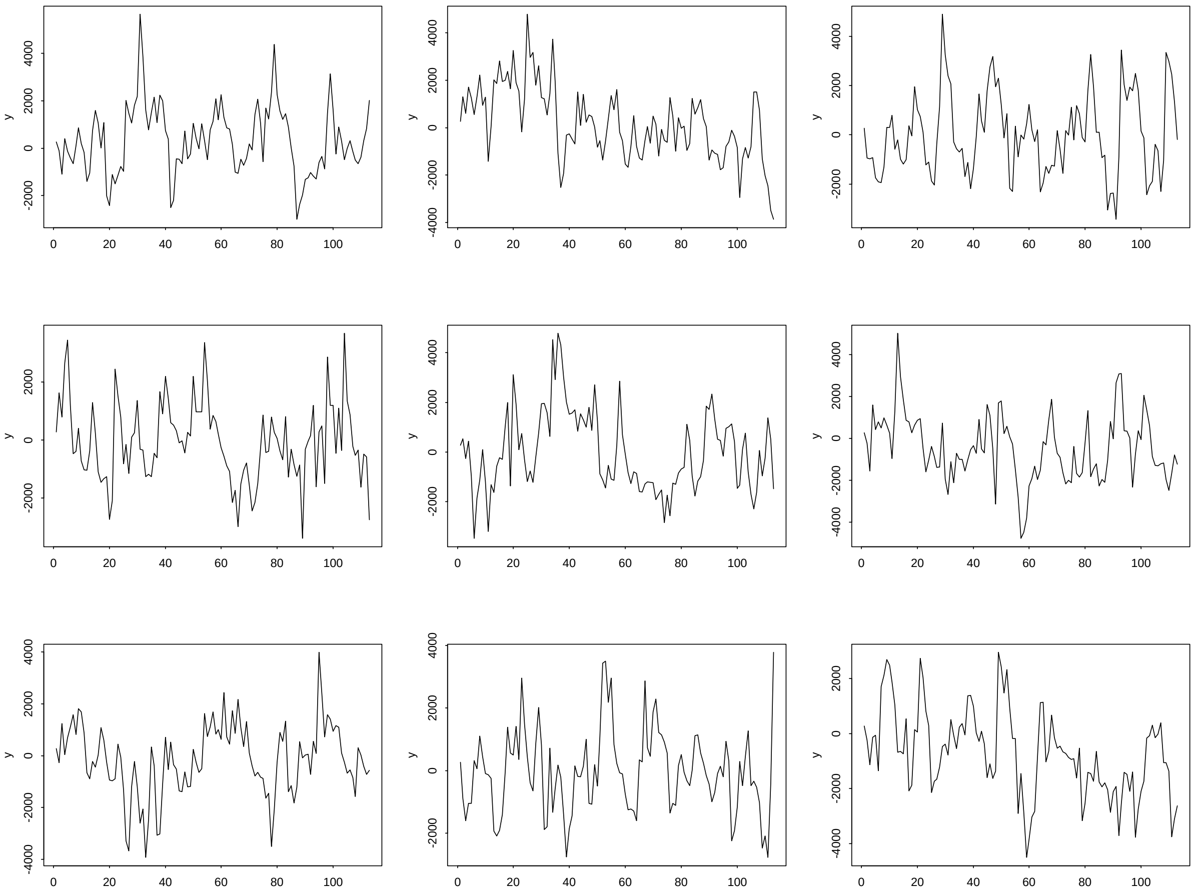


Figure. 3.8: 9 bootstrap series for the lynx data.

the data well (in fact an AR(11) is the preferred model using the AIC – see Section 5.2.4). Figure 3.9 gives the histogram of 200 bootstrap realisations of $\hat{\beta}$, the AR(1) coefficient. The mean and standard deviation of the bootstrap distribution are 0.675 and 0.0678 respectively, suggesting that $\hat{\beta}$ is a somewhat biased estimator of β (bootstrap bias = 0.675 - 0.711 = 0.036) and a bootstrap standard error of 0.0678.

3.6.2 Non-Parametric Regression

Here we consider data providing measurements of acceleration against time for a simulated motorcycle accident. The data are shown in Figure 3.5.

Clearly, the relationship is non-linear, and has structure that will not be easily modelled parametrically. S-Plus provides a number of routines for non-parametric regression, we shall use the `loess()` command. Essentially, this command fits a locally weighted least-squares regression line to the data, given an argument `span`, which determines the proportion of the data to be included in the moving window, which specifies the points to be regressed upon. We set `span = 1/3` to produce the loess fit to the original data in Figure 3.5.

Because of the non-parametric structure of the model, classical approaches to the assessment of parameter precision are not available. Once again, the challenge in bootstrapping is to create

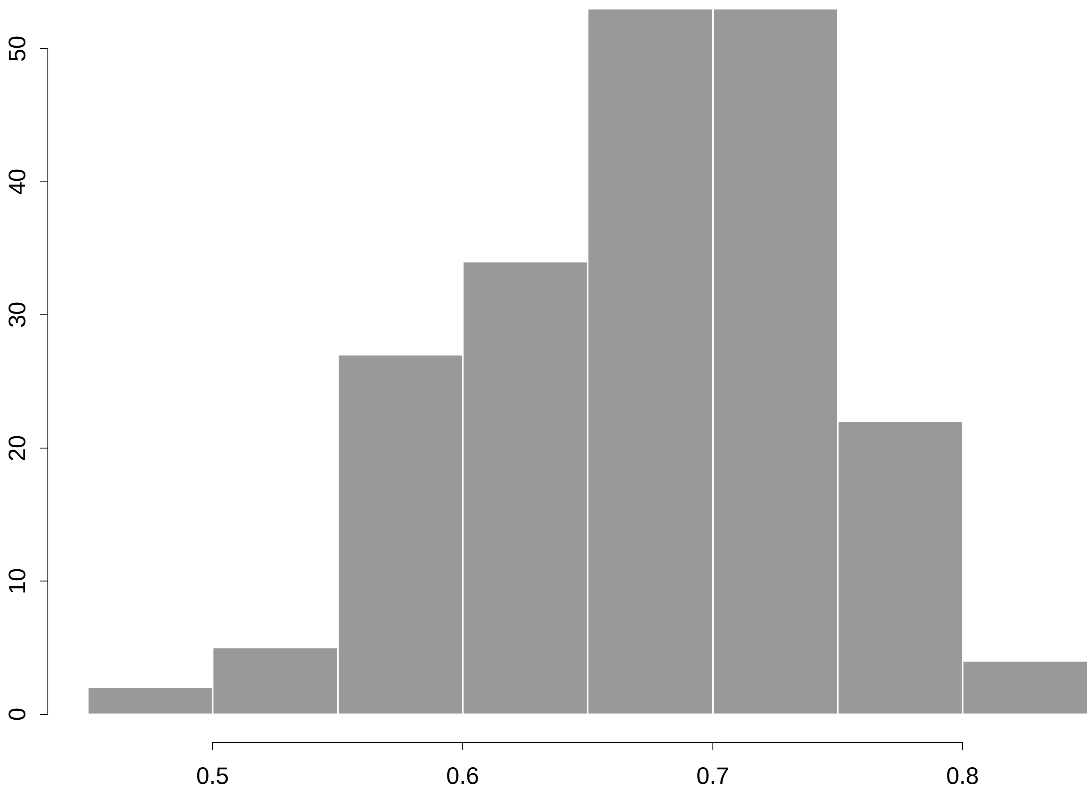


Figure. 3.9: Histogram of the bootstrap realisations of $\hat{\beta}$ for the lynx data.

bootstrap realisations of the data which have similar stochastic structure to the original series. This is achieved by simply bootstrapping the pairs (x, y) in the original plot and fitting `loess` curves to each simulated bootstrap series. Thus, we can define the S-Plus function `theta.cycle` by

```
theta.cycle <- function(x){
        mc.lo <- loess(cycle[x, 2] ~ cycle[x, 1], span = 1/3)
        y <- predict.loess(mc.lo, 1:60)
        lines(1:60, y)
}
```

where `cycle` contains the data, and then use the command

```
> mybootstrap(1:133,30,theta.cycle)
```

to produce 30 bootstrap samples, for example, which results in the `loess` curves in Figure 3.10. In this way we can assess, both pointwise and globally, the variability in the original `loess` fit.

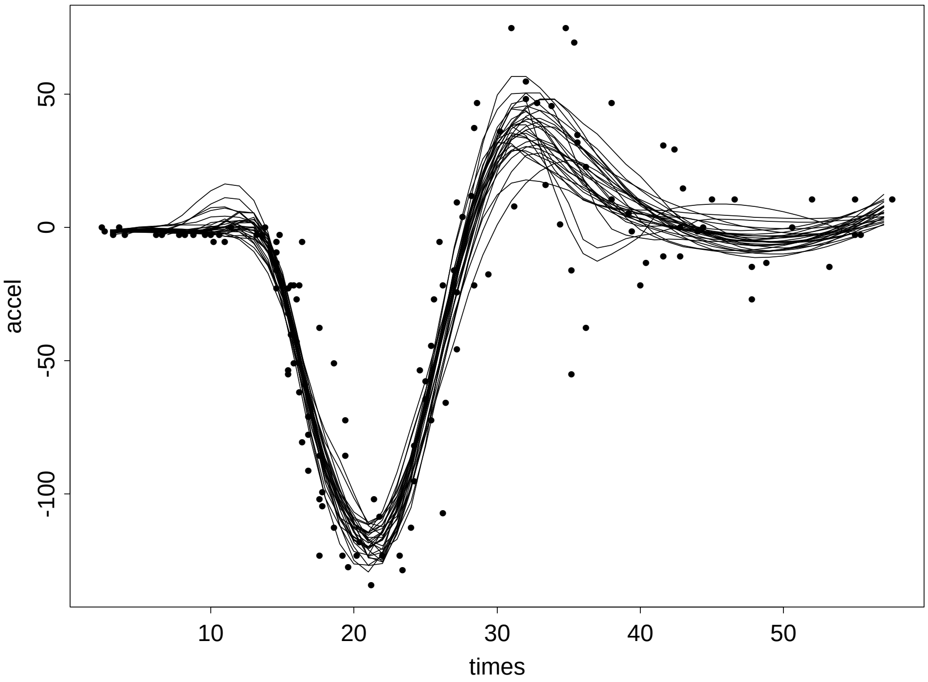


Figure. 3.10: Bootstrap realisations of *loess* fits to the motorcycle data

3.6.3 Regression

A more formal regression-type model has the structure

$$y_i = f(x_i, \boldsymbol{\beta}) + \epsilon_i,$$

where f is a specified function acting on the covariates x_i with parameters $\boldsymbol{\beta}$, and ϵ_i is a realisation from a specified error structure. With this model framework, there are two alternative ways to bootstrap the model.

1. Fit the regression model, form the empirical distribution function of the residuals, generate bootstrap replications of the data by substituting these back into the model, and re-fitting the model to obtain bootstrap distributions of $\boldsymbol{\beta}$. Or,

2. Bootstrap from the pairs (x_i, y_i), re-fit the model to each realisation, form the bootstrap distribution of $\boldsymbol{\beta}$. This is the approach that we adopted in the non-parametric example.

We shall illustrate how the second technique can be used to bootstrap the $\boldsymbol{\beta}$ parameter in a simple example. We return to the American Law school data, where we have fifteen measurements of performance on each of two types of test, LSAT and GPA, and we try to fit the model

$$GPA_i = \beta_0 + \beta_1 LSAT_i + \epsilon_i,$$

without making any distributional assumptions about the ϵ_i.

A least squares fit to these data gives $\beta_0 = 0.3794$ and $\beta_1 = 0.0045$, which leads to the regression line presented in Figure 3.11.

To apply the bootstrap algorithm, we define

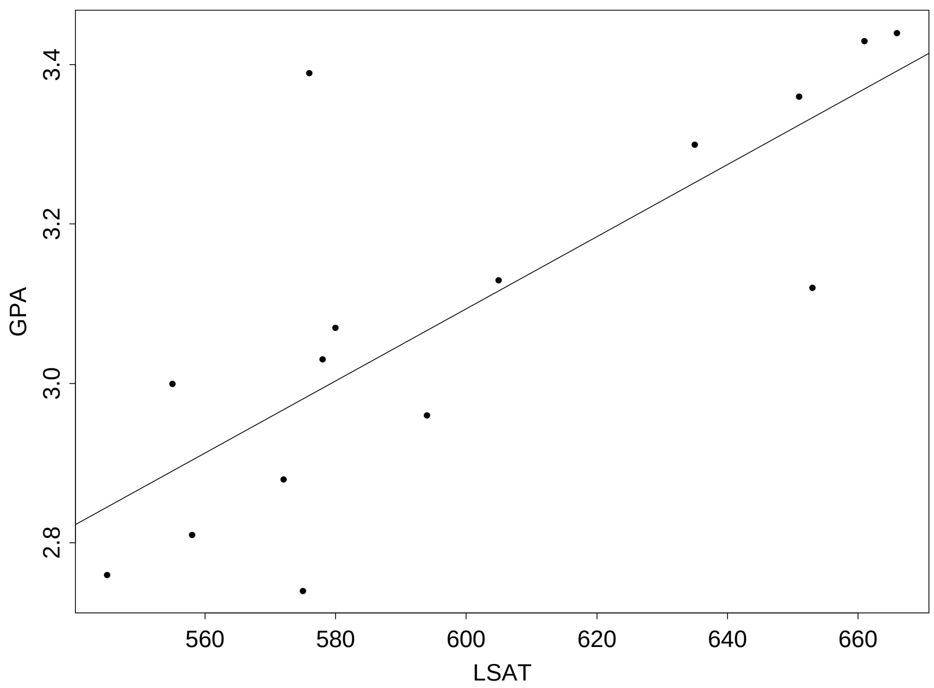


Figure. 3.11: GPA vs LSAT for the American Law School example.

```
theta.regress <- function(x){
      ls <- lsfit(data[x,1], data[x,2])
      abline(ls)
      ls$coeff
}
```

where `data` is the data matrix. We can then use the `mybootstrap()` function,

```
> mybootstrap(1:15,20,theta.regress)
```

to obtain the output presented in Figure 3.12, corresponding to the `lsfit` lines for 20 bootstrap samples.

3.7 Bootstrap Confidence Intervals

Suppose that we are in a one-sample situation, where the data are obtained by random sampling from an unknown distribution, F. Let $\hat{\theta} = \hat{\theta}(F)$ be the plug-in estimate of a parameter of interest, $\theta = \theta(F)$, and let $\widehat{SE}$ denote some reasonable estimate of the standard error for $\hat{\theta}$, based perhaps upon bootstrap or jackknife simulations. Under most circumstances, it turns out that for large sample sizes, the distribution of $\hat{\theta}$ becomes approximately normal, with mean θ and variance approximated by $\widehat{SE}$. Therefore, if we base our estimate upon a sample $x_1, ..., x_n$, then for large n,

$$Z = \frac{\hat{\theta} - \theta}{\widehat{SE}} \sim N(0, 1).$$

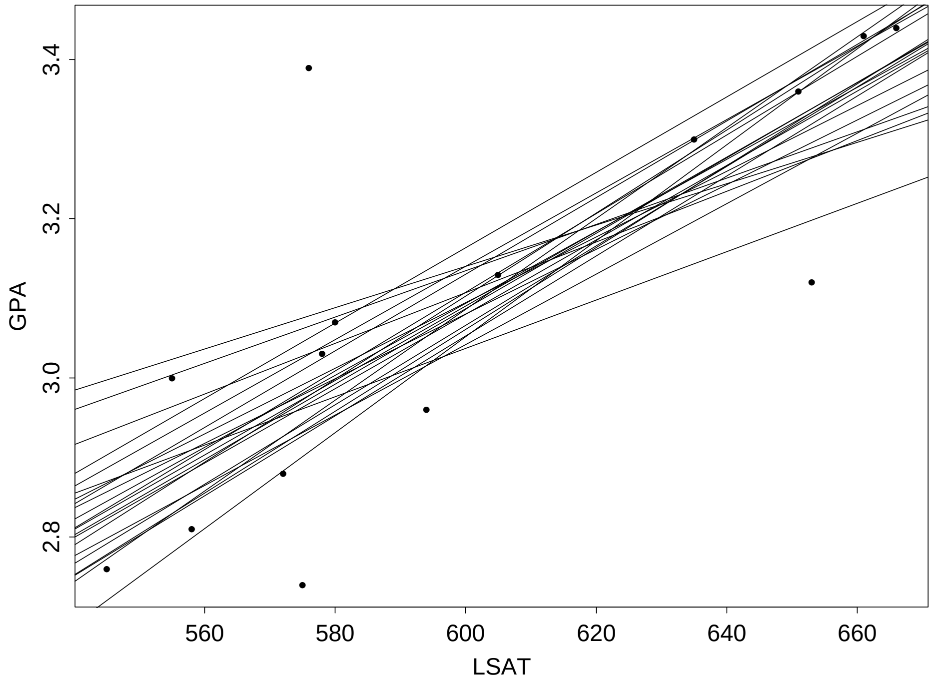


Figure. 3.12: Bootstrap regression lines for the American Law School example.

This result suggests that for large n, a $100(1-\alpha)\%$ confidence interval for θ can be constructed as

$$\left[\hat{\theta} - Z_{(1-\alpha/2)}\widehat{SE}, \hat{\theta} - Z_{\alpha/2}\widehat{SE}\right].$$

We call this the *standard confidence interval* for θ.

However, this interval relies upon the assumption of normality. If this assumption cannot be justified, then we can use the bootstrap to construct non-parametric confidence intervals in two ways.

3.7.1 The Bootstrap-t Interval

In this section, we describe the first of two ways to obtain bootstrap intervals, without resorting to assumptions of normality as discussed above. In essence, we estimate the distribution of the Z-statistic directly from the data, and use this distributional estimate to construct our confidence intervals. We begin by generating B bootstrap samples, and then computing the bootstrap version of Z for each. We then construct percentiles for the Z statistic and use these to construct our intervals.

More formally, we generate B bootstrap samples $\boldsymbol{x}_1^*, ..., \boldsymbol{x}_B^*$, and for each we compute

$$Z^*(b) = \frac{\hat{\theta}^*(b) - \hat{\theta}}{\widehat{SE}^*(b)},$$

where $\hat{\theta}^*(b)$ is the value of $\hat{\theta}$ for the bootstrap sample $\boldsymbol{x}_b^*$ and $\widehat{SE}^*(b)$ is the estimated standard error of $\hat{\theta}^*$ for the bootstrap sample $\boldsymbol{x}_b^*$. The αth percentile of $Z^*(b)$ is then estimated by $\hat{Z}_\alpha$, such that

$$\frac{1}{B}\sum_{b=1}^{B} I_{\{(Z^*(b) \leq \hat{Z}_\alpha)\}}(b) = \alpha.$$

For example, if $B = 1000$, the estimate of the 5% point is the 50th largest value of the $Z^*(b)$'s. Finally, the bootstrap-t confidence interval, is given by

$$\left[\hat{\theta} - \hat{Z}_{(1-\alpha/2)}\widehat{SE}, \hat{\theta} - \hat{Z}_{\alpha/2}\widehat{SE}\right].$$

3.7.2 Percentile Intervals

The percentile interval works as follows. If, as before, we generate B bootstrap samples $\boldsymbol{x}_1^*, ..., \boldsymbol{x}_B^*$, and for each we compute $\hat{\theta}^*(b)$, the estimate of θ based upon the bootstrap sample $\boldsymbol{x}_b^*$, then the percentile interval for θ is defined as

$$\left[\hat{\theta}^*_{\alpha/2}, \hat{\theta}_{(1-\alpha/2)}\right],$$

where $\hat{\theta}^*_\alpha$ is the 100αth percentile of the bootstrap distribution i.e., if $B = 1000$, then $\hat{\theta}^*_{0.05}$ is simply the $B\alpha$th= 50th largest of the $\hat{\theta}^*$ values.

3.7.3 Example

We examine the construction of these two types of intervals in an artificial example, where we know what the correct confidence interval should be. If we generate a sample $X_1, ..., X_{10}$ from a standard normal distribution and choose the parameter of interest to be $\theta = e^\mu$, where μ is the population mean, then the true value of θ is $e^0 = 1$, whilst a sample value $\hat{\theta} = e^{\bar{x}} = 1.508$, say.

We can use S-Plus to generate 1000 bootstrap samples, by defining

```
conf.theta <- function(x){
     y <- exp(mean(x))
     y
}
```

and then running the `bootstrap` command

```
> conf.data <- rnorm(10)
> out.conf <- mybootstrap(conf.data,1000,conf.theta)
```

so that the samples are stored in `out.conf`.

The standard interval, based upon $\widehat{SE} = 0.545$ is given by

$$1.508 \pm 1.96 \times 0.545 = [0.492, 2.670].$$

However, the percentile interval is given by

$$[\hat{\theta}^*_{0.025}, \hat{\theta}^*_{0.975}] = [0.838, 2.911].$$

Finally, the bootstrap-t interval is given by

$$\begin{aligned}\left[\hat{\theta} - \hat{Z}_{(1-\alpha/2)}\widehat{SE}, \hat{\theta} - \hat{Z}_{\alpha/2}\widehat{SE}\right] &= [1.508 - 2.941 \times 0.545,\ 1.508 + 2.499 \times 0.545] \\ &= [-0.095, 2.870].\end{aligned}$$

Note the differences between the three intervals. In this case, θ does not have an approximately normal distribution. This is quite clear from the fact that the two bootstrap intervals are not even close to being symmetric about $\hat{\theta}$. This highlights one of the appealing properties of the bootstrap intervals, in that non-symmetric intervals can be produced in the case of skewed distributions, of which this is an example.

3.8 Exercises

Exercise 3.1: *Show that the variance of the importance sampling estimator, $\hat{\theta}_g$, is minimised by taking $g \propto |\phi f|$.*

Exercise 3.2: *Describe a Monte Carlo procedure for estimating the integral*

$$I = \int_0^a \frac{1}{1+x^2} dx \quad 0 < a < \infty.$$

Exercise 3.3: *A random sample of size n is taken* with *replacement from a population of size N. Show that the probability of having no repetitions in the sample is given by the product*

$$\prod_{j=0}^{n-1} \left(1 - \frac{j}{N}\right).$$

Exercise 3.4: *Given a data set of n distinct values, show that the number of distinct bootstrap samples is*

$$\binom{2n-1}{n}.$$

How many are there for $n = 15$?

Exercise 3.5: *Given a set of observations $x_1, ..., x_n$, what empirical estimator $\hat{\theta}$ would you use to estimate the following θ-functions?*

a) $\theta = \mathbb{E}(X)$.

b) $\theta = \mathbb{P}(10 < X < 25)$.

c) $\theta = \mathbb{E}(\sin X)$.

d) $\theta = \mathbb{E}(X)/Var(X)$.

Exercise 3.6: *Let $x_{(1)} < ... < x_{(7)}$ be an ordered sample of size $n = 7$. Let $\boldsymbol{x}^*$ be a bootstrap sample, and $s(\boldsymbol{x}^*)$ be the corresponding bootstrap replications of the median. Show that*

i) $s(\boldsymbol{x}^*)$ *equals one of the original data values* $x_{(i)}$, $i = 1, ..., 7$.

ii) $s(\boldsymbol{x}^*)$ *equals* $x_{(i)}$ *with probability*

$$p(i) = \sum_{j=0}^{3} \left\{ Bin\left(j : n, \frac{i-1}{n}\right) - Bin\left(j : n, \frac{i}{n}\right) \right\},$$

where $Bin(j :, n, p)$ is the binomial probability $\binom{n}{j} p^j (1-p)^{n-j}$.

Calculate $p(i)$, $i = 1, ..., 7$.

Exercise 3.7: *Show that if $\hat{\theta} = \bar{x}$, the jackknife estimate of standard error is equal to the unbiased estimate,*

$$\sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{(n-1)n}}.$$

Exercise 3.8: *Repeat Exercise 3.7 for the bootstrap estimate of standard error. Hence discuss the relationship between Jackknife and Bootstrap estimates of standard error.*

Exercise 3.9: *Let X_i, $i = 1, ..., n$ denote a set of n random variables from a uniform $U[0, \theta]$ distribution, and consider the functional*

$$T(\boldsymbol{X}, \theta) = \frac{n(\theta - X_{(n)})}{\theta}$$

where $X_{(n)} = \max\{X_1, ..., X_n\}$. Show that for large n, $T(\boldsymbol{X}, \theta) \sim exp(1)$. (Hint: Use the fact that $X_i/\theta \sim U(0,1)$ and that $\left(1 - \frac{x}{n}\right)^n \to e^{-x}$ as $n \to \infty$.)

Show that given a bootstrap sample $\boldsymbol{Y} = (Y_1, ..., Y_n)$, a bootstrap estimate of T is of the form

$$T(\boldsymbol{Y}, \hat{\theta}) = \frac{n(X_{(n)} - Y_{(n)})}{X_{(n)}}.$$

Finally, show that $\mathbb{P}(T(\boldsymbol{Y}, \hat{\theta}) = 0) \to 1 - \exp(-1)$ as $n \to \infty$. Bearing in mind that the distribution of T is continuous, what does this last result tell you about the suitability of the bootstrap method for this particular functional?

Exercise 3.10: *In the table opposite, a group of 30 students have given their attitudes to four questions in a course evaluation. We assume that the students answer independently from one another (although it is not clear whether or not this is a reasonable assumption), but answers from the same student to the four questions are probably dependent and produce a random vector with a 4×4 covariance matrix. Computing the eigenvectors of covariance matrices is the core of some statistical methods like Principal Components Analysis and Factor Analysis. By using some sort of orthogonality properties of the eigenvectors, a random vector with that covariance matrix can be transformed into a vector of uncorrelated variables having certain optimal properties. Analyses of this kind are often used on data from psychological studies, aptitude tests, etc. Since the true covariances of the model are usually unknown, estimated covariance matrices are used instead. Experience shows that results are easily over-emphasised, unless a realistic picture of the uncertainty is also given.*

Use S-Plus, and the data provided, to estimate the eigenvectors of the covariance matrix, perform a bootstrap analysis and find a way to illustrate (perhaps graphically) the uncertainties of the vectors corresponding to the two lowest eigenvalues.

Person	*Question 1*	*Question 2*	*Question 3*	*Question 4*
1	5	5	5	5
2	4	4	5	5
3	4	4	5	3
4	4	4	4	3
5	4	4	3	5
6	2	3	5	1
7	3	4	4	3
8	4	3	3	2
9	3	3	3	3
10	4	4	5	3
11	4	3	4	2
12	4	4	5	3
13	4	4	4	3
14	4	4	2	3
15	5	5	3	3
16	5	5	3	2
17	5	5	4	2
18	4	4	3	4
19	5	5	4	4
20	4	4	2	3
21	5	5	3	3
22	5	5	4	1
23	5	5	4	4
24	5	5	3	4
25	4	3	4	2
26	4	4	4	3
27	5	5	5	4
28	1	2	3	1
29	3	3	3	3
30	3	4	2	5

Chapter 4

BAYESIAN INFERENCE

4.1 Introduction

The integration operation plays a fundamental role in Bayesian statistics. For example, given a sample $\boldsymbol{y}$, from a distribution with joint probability distribution $f(\boldsymbol{y}|\theta)$ (which we will often refer to–slightl incorrectly– as a likelihood[1]), and a prior for θ given by $p(\theta)$, Bayes' Theorem relates the posterior $\pi(\theta|\boldsymbol{y})$ to the prior via the formula

$$\pi(\theta|\boldsymbol{y}) \propto L(\boldsymbol{y}|\theta)p(\theta),$$

where the constant of proportionality is given by

$$\left(\int L(\boldsymbol{y}|\theta)p(\theta)d\theta\right)^{-1}. \tag{4.1}$$

Given the posterior, and in the case where $\boldsymbol{\theta} = (\boldsymbol{\phi}, \boldsymbol{\psi})$ is multivariate, for example, we may be interested in the marginal posterior distributions, such as

$$\pi(\boldsymbol{\phi}|\boldsymbol{y}) = \int \pi(\boldsymbol{\phi}, \boldsymbol{\psi}|\boldsymbol{y})d\boldsymbol{\psi}. \tag{4.2}$$

Alternatively, we might be interested in summary inferences in the form of posterior expectations, for example

$$\mathbb{E}_\pi\left(g(\theta)\right) = \int g(\theta)\pi(\theta|\boldsymbol{y})d\theta. \tag{4.3}$$

Thus, the ability to integrate often complex and multi-dimensional functions is extremely important in Bayesian Statistics, whether it is for calculating the normalising constant in (4.1), the marginal distribution in (4.2) or the expectation in (4.3).

Often, explicit evaluation of these integrals is not possible and, traditionally, we would be forced to use numerical integration or analytic approximation techniques. However, Markov Chain Monte Carlo provides an alternative whereby we sample from the posterior directly, and obtain sample estimates of the quantities of interest.

The idea of Markov Chain Monte Carlo (MCMC) was first introduced by Metropolis *et al* (1953) as a method for efficient simulation of the energy levels of atoms in a crystalline structure, and was subsequently adapted and generalised by Hastings to focus upon statistical problems, such as those described above. The idea is extremely simple. Suppose that we have some distribution,

[1] Recall that in fact $L(\boldsymbol{\theta}|\boldsymbol{y}) = f(\boldsymbol{y}|\boldsymbol{\theta})$.

$\pi(x)$ $x \in E \subseteq \mathbb{R}^d$, which is known only up to some multiplicative constant. We commonly refer to this as the *target* distribution. If π is sufficiently complex so that we cannot sample from it directly, an indirect method for obtaining samples from π is to construct a Markov chain with state space E, and whose stationary (or invariant) distribution is $\pi(x)$. Then, if we run the chain for long enough, simulated values from the chain can be treated as a sample from the target distribution and used as a basis for summarising important features of π.

Under certain regularity conditions, the Markov chain sample path mimics a random sample from π. Given realisations $\{X^t : t = 0, 1, ...\}$ from such a chain, typical asymptotic results include the distributional convergence of the realisations i.e.,

$$X^t \underset{t\to\infty}{\overset{d}{\to}} \pi(x)$$

and the consistency of the ergodic average, for any scalar functional θ, i.e.,

$$\frac{1}{n}\sum_{t=1}^{n}\theta(X^t) \overset{n\to\infty}{\longrightarrow} \mathbb{E}_\pi\left[\theta(X)\right] \quad a.s.$$

There are many important implementational issues associated with MCMC methods. These include (amongst others) the choice of sampler, the number of independent replications to be run, the choice of starting values, how long to run the samplers, and both estimation and efficiency problems. These will be discussed further in Section 4.6.

4.2 Markov Chains and the Monte Carlo Method

Markov Chain Monte Carlo is an example of the Monte Carlo method, which we implement through the use of Markov Chains. Before we take a more detailed look at MCMC, we examine these two elements separately.

4.2.1 The Monte Carlo Method

As we have seen, we are often faced with the problem of evaluating an integral which is too complex to calculate explicitly, however we can use the simulation technique of *Monte Carlo Integration* to gain an estimate of a given integral. The method is based upon drawing observations from the distribution of the variable of interest and simply calculating the empirical estimate of the mean. For example, given a sample of observations, $x_1, ..., x_n \sim \pi(x)$, we can estimate the integral

$$\mathbb{E}_\pi\left[f(X)\right] = \int f(x)\pi(x)dx$$

by the ergodic average

$$\bar{f}_n = \frac{1}{n}\sum_{i=1}^{n} f(x_i). \tag{4.4}$$

This is Monte Carlo integration.

For independent samples, the Law of Large Numbers ensures that

$$\bar{f}_n \to \mathbb{E}_\pi\left[f(x)\right] \quad \text{as } n \to \infty.$$

Independent sampling from $\pi(x)$ may be difficult, however Equation (4.4) still holds if we generate our samples, not independently, but via a Markov chain.

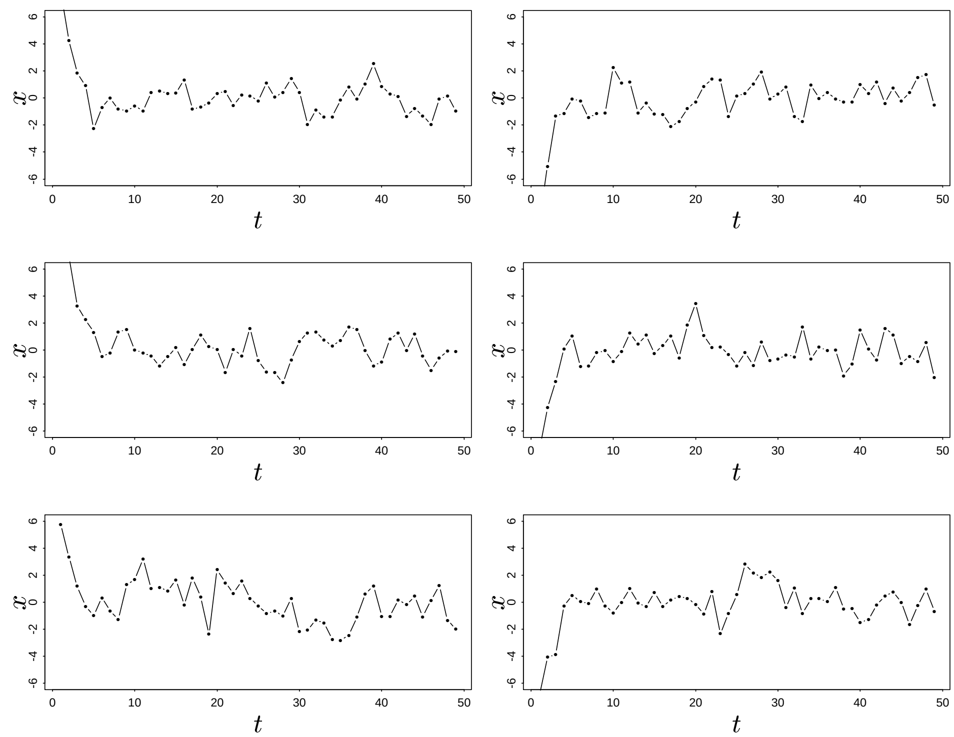


Figure. 4.1: 3 independent runs of a Markov chain from two distinct starting points.

4.2.2 Markov Chains

We are interested in Markov chains defined on *continuous* state spaces, in general. A Markov chain is generated by sampling the new state of the chain, based only upon information regarding the current state i.e., at time t, we generate the new state of the chain from a density dependent only upon x^t;

$$x^{t+1} \sim \mathcal{K}(x^t, x) \quad \Big(= \mathcal{K}(x|x^t)\Big).$$

So x^{t+1} depends only upon x^t, not on $x^0, ..., x^{t-1}$. We call $\mathcal{K}$ the transition kernel for the chain, and it uniquely describes the dynamics of the chain.

Example

Consider the Markov chain with transition kernel given by

$$X^{t+1} \sim N\left(\frac{1}{2}x^t, 1\right).$$

Figure 4.1 shows the resulting sample path for this transition kernel for three runs from each of two starting points, $x^0 = 15$ and $x^0 = -15$. We can see that each of the sample paths is different from the others, but that each settles down to a similar pattern within a few iterations, despite the differing starting values. So does x^t depend upon x^0 and, if so, for what values of t? What happens as $t \to \infty$?

Under certain conditions (the chain is both aperiodic and irreducible), the Markov chain will converge to its *stationary* distribution i.e.,

$$\mathbb{P}(X^t \in A) \to \pi(A) \quad \forall A \in E.$$

In the above example, the stationary distribution is

$$x^t \sim N(0.0, 1.33),$$

which is independent of x^0. Does this happen for all Markov Chains?

Assuming that a stationary distribution exists, it is *unique* if the chain is *irreducible.* Irreducibility is a condition on the chain, which essentially means that any set of states can be reached from any other set of states, within a finite number of moves. We shall come back to this concept later.

Suppose that we have an irreducible Markov chain with stationary distribution, $\pi(x)$. Then the *Ergodic Theorem* states that

$$\bar{f}_n = \frac{1}{n}\sum_{t=1}^{n} f(x^t) \to \mathbb{E}_\pi\left(f(x)\right) \text{ as } n \to \infty.$$

We call $\bar{f}_n$ the *ergodic average.*

Thus, assuming that the chain has stationary distribution $\pi(x)$, and is irreducible, we can use the Markov chain sample path to obtain an estimate of the value of any functional of the variable X. This means that, since many integrals in which we are interested can be thought of as expectations, we can form Monte Carlo approximations to these integrals by sampling via a Markov chain. We call this technique, Markov Chain Monte Carlo.

The usual approach to Markov chain theory is to start with some transition kernel, determine conditions under which there exists an invariant or stationary distribution, and then to identify the form of that limiting distribution. MCMC methods involve the solution of the inverse of this problem whereby the stationary distribution is known, and it is the transition kernel that needs to be identified. There are a number of methods for developing Markov chains with a given stationary distribution, and we discuss these in the next sections.

4.3 MCMC and the Gibbs Sampler

Suppose that $y_1, ..., y_n$ are independent, identically distributed (*iid*) observations from a $N(\mu, \tau^{-1})$ distribution. Then the likelihood is given by

$$L(\mu, \tau; \boldsymbol{y}) = \frac{\tau^{n/2}}{(2\pi)^{n/2}} \exp\left[-\frac{\tau}{2}\sum_{i=1}^{n}(y_i - \mu)^2.\right]$$

Now suppose that we have prior information on μ and τ, indicating that they are independent, μ being Normal (μ_0, σ_0^2) and τ being Gamma (α_0, β_0) so that

$$p(\mu, \tau) = \frac{e^{-(\mu-\mu_0)^2/2\sigma_0^2}}{\sqrt{2\pi\sigma_0^2}} \frac{e^{-\beta_0\tau}\beta_0^{\alpha_0}\tau^{\alpha_0-1}}{\Gamma(\alpha_0)} \qquad \tau > 0 \quad \mu \in \mathbb{R}.$$

Then Bayes' Theorem gives us the posterior,

$$\begin{aligned} p(\mu, \tau | y_1, ..., y_n) &\propto L(\mu, \tau; \boldsymbol{y}) p(\mu, \tau) \\ &\propto \tau^{\frac{n}{2}+\alpha_0-1} e^{-\beta_0\tau} e^{-\frac{\tau S_n}{2} - \frac{(\mu-\mu_0)^2}{2\sigma_0^2}} \end{aligned}$$

where $S_n = \sum_{i=1}^{n}(y_i - \mu)^2$.

This is all very well, but we have two problems: we are unable to compute anything directly, and the normalisation constant (or constant of proportionality) is unknown. However, suppose that we knew τ say, then

$$\pi(\mu | \tau, y_1, ..., y_n) \sim N\left(\frac{n\bar{y}\tau + \mu_0\tau_0}{n\tau + \tau_0}, (n\tau + \tau_0)^{-1}\right)$$

where $\tau_0 = (\sigma_0^2)^{-1}$.

Exercise: Check.

Similarly, if μ is known then

$$\pi(\tau|\mu, y_1, ..., y_n) \sim \Gamma\left(\alpha_0 + \frac{n}{2}, \beta_0 + \frac{S_n}{2}\right).$$

This kind of situation, where the marginal distributions are unknown, but the conditionals are not only known but have a standard form, is (fortunately) very common.

Question: What happens if we start with $\mu(0)$ and $\tau(0)$ and follow the iterative algorithm below?

(I) GIVEN $\tau(t)$, SAMPLE FROM $\pi(\mu|\tau(t), y_1, ..., y_n)$ TO GIVE $\mu(t+1)$.

(II) GIVEN $\mu(t+1)$ SAMPLE $\tau(t+1)$ FROM $\pi(\tau|\mu(t+1), y_1, ..., y_n)$ TO GET $(\mu(t+1), \tau(t+1))$.

What can we say about the distribution of the sequence $\{\mu(i), \tau(i)\}$? How does it depend upon $\mu(0)$ and $\tau(0)$?

In fact, this iterative procedure produces a Markov chain with stationary distribution (for μ and τ) equal to the posterior distribution of these two variables. This algorithm is a simple example of an MCMC sampler, called the *Gibbs Sampler*.

4.3.1 The Gibbs Sampler

The Gibbs Sampler works as follows. Given a vector variable $\boldsymbol{X} = (X_1, ..., X_p) \in \mathbb{R}^p$ with distribution $\pi(x)$, the Gibbs Sampler uses the set of full conditionals of π to sample indirectly from the marginal distributions.

Let $\pi(x_i|x_{(i)})$ denote the induced full conditional of x_i, given the values of the other components $x_{(i)} = (x_1, ..., x_{i-1}, x_{i+1}, ..., x_k)$, $i = 1, ..., k$, $1 < k \leq p$. Then, given an arbitrary starting value $\boldsymbol{x}^0 = (x_1^0, ..., x_k^0)$, the Gibbs Sampler successively makes random drawings from the full conditional distributions $\pi(x_i|x_{(i)})$, $i = 1, ..., k$, as follows:

$$\begin{array}{lll}
x_1^1 & \text{is sampled from} & \pi(x_1|\boldsymbol{x}_{(1)}^0) \\
x_2^1 & \text{is sampled from} & \pi(x_2|x_1^1, x_3^0, ..., x_k^0) \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
x_i^1 & \text{is sampled from} & \pi(x_i|x_j^1,\ j < i \quad x_j^0,\ \text{and } j > i) \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
x_k^1 & \text{is sampled from} & \pi(x_k|\boldsymbol{x}_{(k)}^1).
\end{array}$$

This completes a transition from $\boldsymbol{x}^0$ to $\boldsymbol{x}^1$. Iteration of the full cycle of random variate generations from each of the full conditionals in turn, produces a sequence $\boldsymbol{x}^0, \boldsymbol{x}^1, ..., \boldsymbol{x}^t, ...$ which is a realisation

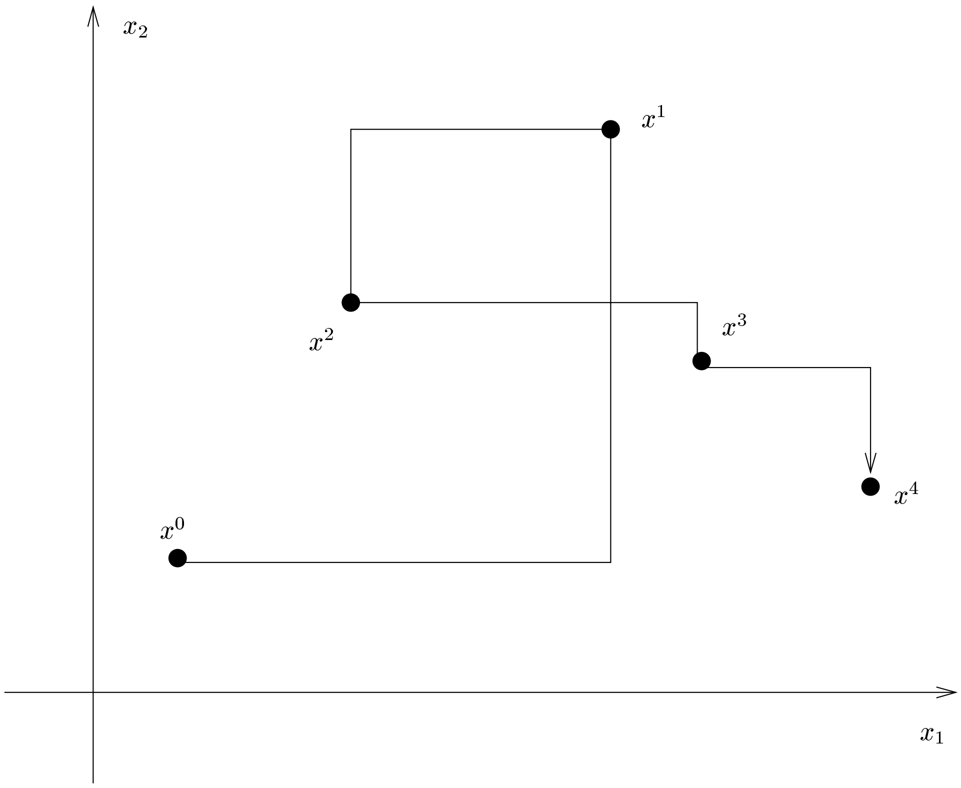


Figure. 4.2: Typical Gibbs sampler sample path.

of a Markov chain with transition probability for going from $\boldsymbol{x}^t$ to $\boldsymbol{x}^{t+1}$ given by

$$\mathcal{K}_G(\boldsymbol{x}^t, \boldsymbol{x}^{t+1}) = \prod_{l=1}^{k} \pi(x_l^{t+1} | x_j^{t+1},\ j < l \text{ and } x_j^t,\ j > l), \tag{4.5}$$

and stationary distribution π.

Thus, in two dimensions a typical trajectory of the Gibbs Sampler may look something like that give in Figure 4.2.

Conceptually, the Gibbs Sampler appears to be a rather straightforward algorithmic procedure. Ideally, each of the conditionals will be of the form of a standard distribution and suitable prior specification often ensures that this is the case. However, in the cases where one or more of the conditionals is non-standard, there are many ways to sample from univariate conditionals, as we discussed in Chapter 2. Alternatively, the introduction of auxiliary variables, whilst increasing the dimensionality, may be used to turn the full conditionals into standard distributions, see Section 4.6.6.

Example

If we return to the example at the beginning of this section, we have $k = 2$ and let $X_1 = \mu$ and $X_2 = \tau$. Then the conditionals are given by

$$\begin{aligned}
\pi(x_1 | \boldsymbol{x}_{(1)}) &= \pi(x_1 | x_2) \sim N\left(\frac{n\bar{y}x_2 + \mu_0\tau_0}{nx_2 + \tau_0}, (nx_2 + \tau_0)^{-1}\right) \\
\pi(x_2 | \boldsymbol{x}_{(2)}) &= \pi(x_2 | x_1) \sim \Gamma\left(\alpha_0 + \frac{n}{2}, \beta_0 + \frac{S_n}{2}\right).
\end{aligned}$$

If we set $n = 10$, and generate 10 observations from a $N(0,1)$ distribution, so that $\mu = 0$ and $\tau = 1$, then we can use the following S-Plus routine to implement the Gibbs Sampler for this example.

```
gibbs1 <- function(nits,y,mu=0,tau=1){
     alpha0 <- 0.1
     beta0 <- 0.01
     mu0 <- 0.0
     tau0 <- 1.0

     x <- array(0,c(nits+1,2))
     x[1,1] <- mu
     x[1,2] <- tau

     n <- length(y)
     ybar <- mean(y)

     for(t in 2:(nits+1)){
          x[t,1] <- rnorm(1,(n*ybar*x[t-1,2] + mu0*tau0)/
                           (n*x[t-1,2]+tau0), 1/(n*x[t-1,2]+tau0))
          sn <- sum((y-x[t,1])^2)
          x[t,2] <- rgamma(1,alpha0+n/2)/(beta0+sn/2)
     }

     par(mfrow=c(1,2))

     plot(1:length(x[,1]),x[,1],type='l',lty=1,xlab='t',ylab='mu')
     plot(1:length(x[,2]),1/x[,2],type='l',lty=1,xlab='t',ylab='sigma')

     x
}
```

Then we can use the commands

```
> y <- rnorm(10)
> out <- gibbs1(100,y,5,0.5)
```

to obtain a sample run of the Gibbs sampler for this particular example.

We can see from Figure 4.3 that the posterior density for μ is centred about zero, and (not quite so obvious) that the density for τ is centred about 1. Running the sampler for 1000 iterations,we get estimates for the means of μ and τ as -0.048 and 1.021, respectively, with associated standard errors of 1.036 and 0.185.

Example - General autoregressive time series

Suppose that we have N observations denoted by $\boldsymbol{x} = (x_1, ..., x_N)$, from an autoregressive (AR) model of order k. Then, the data is generated by the process,

$$x_t = \sum_{\tau=1}^{k} a_\tau x_{t-\tau} + \varepsilon_t, \;\; t = k, ..., N,$$

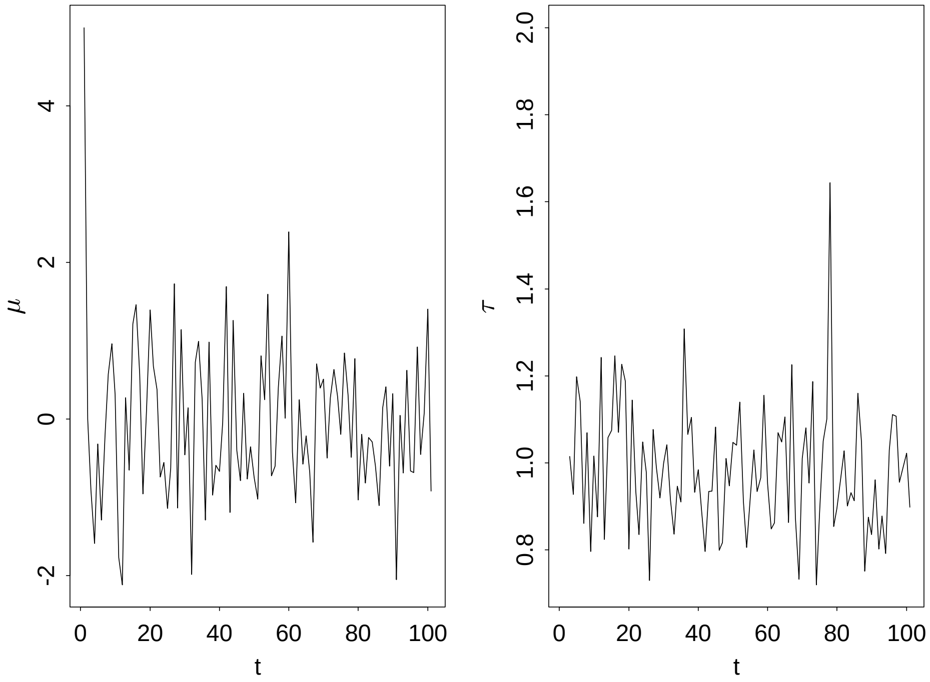


Figure. 4.3: Output from the Gibbs Sampler, showing the sample paths for the variables μ and τ.

where $\varepsilon_t \overset{iid}{\sim} N(0, \sigma^2)$. Alternatively, we can express this process more conveniently in matrix notation. Suppose that we let $\boldsymbol{x}_k$ denote the vector with elements $(x_{k+1}, ..., x_N)$ and X_k denote the matrix of dimension $(N-k) \times k$ with (i,j)th element x_{k+i-j}. Then,

$$\boldsymbol{x}_k = X_k \boldsymbol{a} + \boldsymbol{\varepsilon},$$

where $\boldsymbol{a}$ is the vector with elements $(a_1, ..., a_k)$, and $\boldsymbol{\varepsilon}$ is the vector with elements $(\varepsilon_{k+1}, ..., \varepsilon_N)$.

The corresponding likelihood associated with the data, given the parameters, is approximated by,

$$L(\boldsymbol{x}|\boldsymbol{a}, \sigma^2) = \frac{1}{(\sqrt{2\pi\sigma^2})^{N-k}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_k - X_k\boldsymbol{a})^T(\sigma^2 I)^{-1}(\boldsymbol{x}_k - X_k\boldsymbol{a})\right).$$

Suppose that we have prior distributions,

$$\begin{aligned} \boldsymbol{a} &\sim N(\boldsymbol{\mu}_a, \Sigma_a); \\ \sigma^2 &\sim \Gamma^{-1}(\alpha, \beta). \end{aligned}$$

The full posterior conditional distribution of $\boldsymbol{a}$, given the data and σ^2 is of the form,

$$\boldsymbol{a}|\boldsymbol{x}, \sigma^2 \sim N(\boldsymbol{\mu}, \Sigma), \tag{4.6}$$

where,

$$\Sigma^{-1} = \sigma^{-2} X_k^T X_k + \Sigma_a^{-1}; \quad \boldsymbol{\mu} = \sigma^{-2}\Sigma X_k^T \boldsymbol{x}_k + \Sigma\Sigma_a^{-1}\boldsymbol{\mu}_a.$$

Similarly, the full posterior conditional distribution of σ^2, can be expressed as,

$$\sigma^2|\boldsymbol{x}, \boldsymbol{a} \sim \Gamma^{-1}\left(\alpha + \frac{(N-k)}{2}, \beta + \frac{1}{2}(\boldsymbol{x}_k - X_k\boldsymbol{a})^T(\boldsymbol{x}_k - X_k\boldsymbol{a})\right). \tag{4.7}$$

Proof: See Exercises.

We have explicit expressions for the full conditional posterior distributions of the parameters $\boldsymbol{a}$ and σ^2, which are both of standard form. Thus, we can use the Gibbs sampler to sample from the posterior distribution of interest.

4.3.2 The Hit 'n' Run Algorithm

In the case of the Gibbs Sampler, and where each of the conditionals is univariate, the algorithm can be thought of as sampling once in each of the co-ordinate directions, at each iteration. An alternative is given by the Hit 'n' Run algorithm, which (at iteration t) works as follows.

Step 1. Generate a set of standard uniform random variables $d_1, ..., d_p$ and let

$$\boldsymbol{d}^t = \frac{(d_1, ..., d_p)}{|(d_1, ..., d_p)|}$$

denote a random, unit length, direction in $\mathbb{R}^p$.

Step 2. Generate a signed distance λ^t from the distribution

$$\pi^t(\lambda) = \frac{\pi(\boldsymbol{x}^{t-1} + \lambda \boldsymbol{d}^t)}{\int_{S^t} \pi(\boldsymbol{x}^{t-1} + u\boldsymbol{d}^t) du}, \qquad \lambda^t \in S^t,$$

where $S^t = \{\lambda \in \mathbb{R} : \boldsymbol{x}^{t-1} + \lambda \boldsymbol{d}^t \in E\}$.

Step 3. Set $\boldsymbol{x}^t = \boldsymbol{x}^{t-1} + \lambda^t \boldsymbol{d}^t$.

The function π^t introduced in Step 2, is the (re-parameterised) distribution function of the conditionalisation of π along the line $\boldsymbol{d}^t$. Thus, the algorithm can be thought of as randomly sampling a direction in $\mathbb{R}^p$ and sampling along that conditional.

A typical Hit 'n' Run algorithm sampler path, may look something like that given in Figure 4.4.

4.4 Data Augmentation

Data augmentation is a technique which can be thought of as a special case of the Gibbs sampler. It can be applied equally well to the following two situations.

1. One of the main problems that afflicts statistical computing is that, in the real world, some of our data could be missing. In theory the distribution of the observed data can be obtained by integrating out the missing data. However, this is frequently difficult if not impossible to do.

2. The likelihood of the data is intractable but, conditional on a collection of unobserved data, the likelihood becomes tractable.

These two situations are best illustrated using simple examples.

1. Suppose that $Y_0, ..., Y_n$ is a time series of observations defined by $Y_0 = 0$ and for each $1 \leq i \leq n$,

$$Y_i | Y_0, ..., Y_{i-1} \sim Y_{i-1} + Beta(\theta, \theta),$$

for $\theta > 0$. Then, given the full dataset $Y_0, ..., Y_n$ the likelihood of θ is easy to write down (as a product of beta terms for the successive increments). However, suppose that Y_{i^*} is missing, then the likelihood is now not available in closed form.

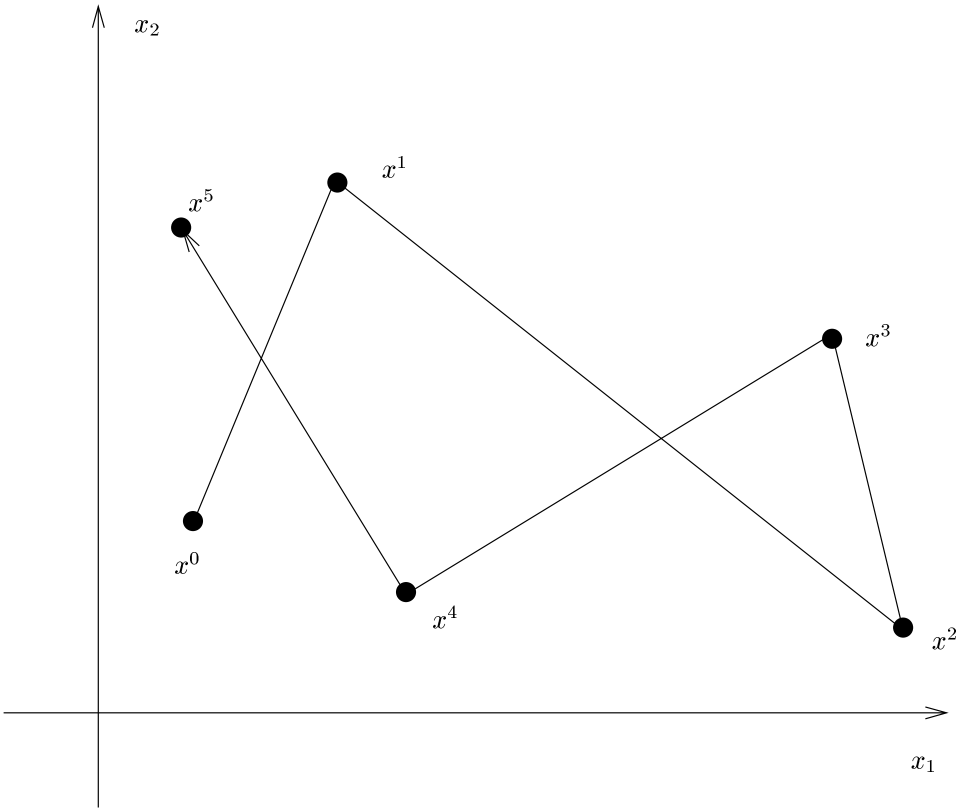


Figure. 4.4: Typical Hit 'n' Run algorithm sample path.

2. Suppose that $Y_1, ..., Y_n$ are *iid* data from the mixture density

$$\frac{1}{2\sqrt{2\pi}}\left(e^{-x^2/2} + e^{-(x-\theta)^2/2}\right).$$

The mixture density can be thought of as the density of the random variable obtained by the following procedure. Toss a fair coin. Given a head, sample from a $N(0,1)$ distribution. Given tail, sample from a $N(\theta, 1)$ distribution. Now, the likelihood of θ can be written explicitly:

$$\prod_{i=1}^{n}\frac{1}{2}\left(e^{-y_i^2/2} + e^{-(y_i-\theta)^2/2}\right).$$

However, this is a difficult function to maximise (for classical inference) or to compute a posterior distribution from. Moreover, it sums $2n$ exponential terms. However, suppose that we knew the sequence of heads and tails, then the likelihood collapses to one exponential term containing two summations, one over those data points corresponding to a head (and therefore containing no information about θ) and those corresponding to a tail, being sampled from the $N(\theta, 1)$ distribution. Then, the inference (both Bayesian and classical) is simplified greatly.

Now introduce some general notation. Let y_{obs} denote the observed data and y_{mis} denote the missing data. Let θ be the unknown parameter(s) with prior $p(\theta)$. Then the situation that the above examples demonstrate is where $L(y_{obs}, y_{mis}|\theta)$ is available. As a result of this, the conditional distribution of θ and y_{mis} is proportional to

$$\pi(\theta, y_{mis}|y_{obs}) \propto L(y_{obs}, y_{mis}|\theta)p(\theta).$$

Data augmentation proceeds by carrying out Gibbs sampling to successively sample from θ and Y_{mis} to produce a sample from this distribution. The marginal distribution of θ is therefore the posterior distribution of interest.

How does this work in the two examples above?

1. Given a uniform prior for θ on $(0,1)$ and given complete data $Y_0, ..., Y_n$, the posterior for θ is explicit and can be sampled by inversion for example. The distribution of Y_{i^*}|everything else, can be seen to have a density proportional to

$$((y_{i^*} - y_{i^*-1})(y_{i^*-1} + 1 - y_{i^*})(y_{i^*} - y_{i^*+1} - 1)(y_{i^*+1} - y_{i^*}))^{\theta-1}$$

   on the region $(y_{i^*-1}, y_{i^*-1} + 1) \cap (y_{i^*+1} - 1, y_{i^*+1})$. This sampling could be carried out via rejection sampling for example.

2. Here Y_{mis} is a sequence of n heads or tails. Suppose that the prior for θ is $N(0,1)$. Then, it's posterior given that Y_{mis} allocates Y_i to the $N(\theta, 1)$ component for all $i \in A$ is

$$N\left(\frac{\sum_{i\in A} Y_i}{1+n}, \frac{1}{n}\right).$$

   On the other hand, if C_i is the result of the coin toss for the ith observation, then it is clear that

$$\mathbb{P}(C_i = T|\theta, \text{ everything else}) = \frac{e^{-(y_i-\theta)^2/2}}{e^{-(y_i-\theta)^2/2} + e^{-y_i^2/2}}.$$

Example: Genetic Linkage

Here we examine a genetic linkage example. The example is well-worn but is useful because it illustrates very simply all of the ideas necessary for this (and subsequent) algorithms. The example concerns the genetic linkage of 197 animals each allocated to one of four categories:

$$Y = (y_1, y_2, y_3, y_4) = (125, 18, 20, 34)$$

with cell probabilities

$$\left(\frac{1}{2} + \frac{\theta}{4}, \frac{1}{4}(1-\theta), \frac{1}{4}(1-\theta), \frac{\theta}{4}\right).$$

Though it is possible to sample the posterior distribution of θ directly by rejection sampling for example, data augmentation brings about a substantial simplification. Specifically, we augment the observed data Y by dividing the first cell into two with respective cell probabilities 1/2 and $\theta/4$, giving an augmented data set $X = (x_1, x_2, x_3, x_4, x_5)$ where $x_1 + x_2 = y_1$, $x_3 = y_2$, etc. Now using a flat prior for θ, we have

$$\pi(\theta|Y) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}$$

whereas

$$\pi(\theta|X) \propto \theta^{x_2+x_5}(1-\theta)^{x_3+x_4}.$$

In this case, writing $X = (y, Z)$ for "missing" data Z (i.e., $x_2 = z$, and $x_1 = y_1 - z$) we have

$$\theta|Z, Y \sim Beta(Z + Y_5 + 1, Y_3 + Y_4 + 1)$$

and

$$Z|\theta, Y \sim Bin\left(125, \frac{\theta}{2+\theta}\right).$$

This is easily coded in S-Plus as

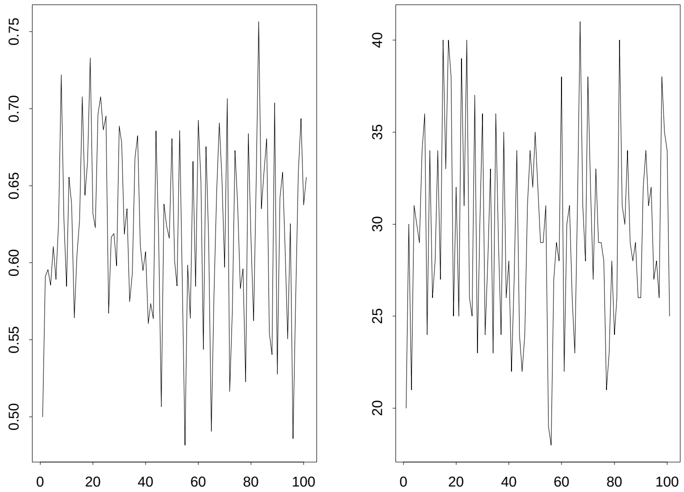


Figure. 4.5: Realisations from the data augmentation algorithm for the genetic example. Left plot: θ, right plot: Z.

```
gibbs2 <- function(n, y)
{
        z.o <- 20
        th.o <- 0.5
        z <- z.o
        for(j in 1:n) {
                th <- rbeta(1, z + y[4] + 1, y[2] + y[3] + 1)
                p <- th/(2 + th)
                z <- rbinom(1, y[1], p)
                z.o <- c(z.o, z)
                th.o <- c(th.o, th)
        }
        cbind(th.o, z.o)
}
```

Note that the initial values `z.o=20` and `th.o=0.5` are chosen arbitrarily. Applying this algorithm with `n=100` and `y=c(125,18,20,34)` gives the output in Figure 4.5.

It seems as though convergence (Section 4.6.1) of the algorithms is almost immediate but there is some evidence that the first few observations are atypical. Thus, we might discard an initial burn-in period of 20 iterations, say. The remaining sample is considered to be a sample from the joint distribution $\pi(\theta, Z|Y)$. In particular, histograms of the separate θ and Z components (Figure 4.6) are estimates of the marginal posterior distributions $\pi(\theta|Y)$ and $\pi(Z|Y)$. Similarly a plot of the (θ, Z) pairs (Figure 4.7) gives an impression of the joint posterior $\pi(\theta, Z|Y)$.

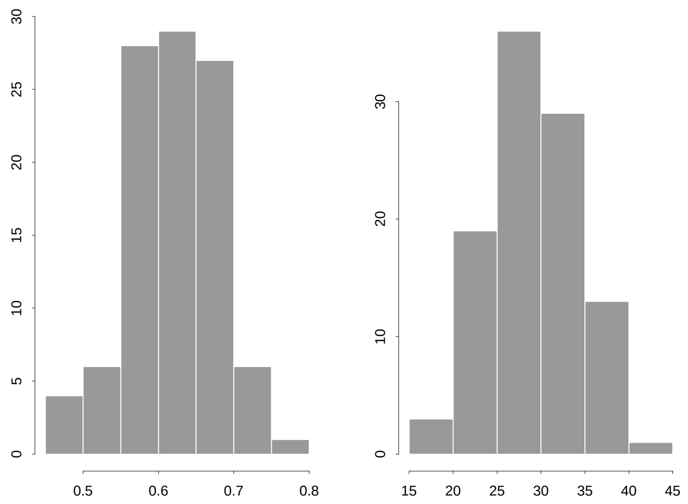


Figure. 4.6: Frequency histogram denoting the shape of the marginal posteriors for the genetic example - LHS: θ, RHS: Z.

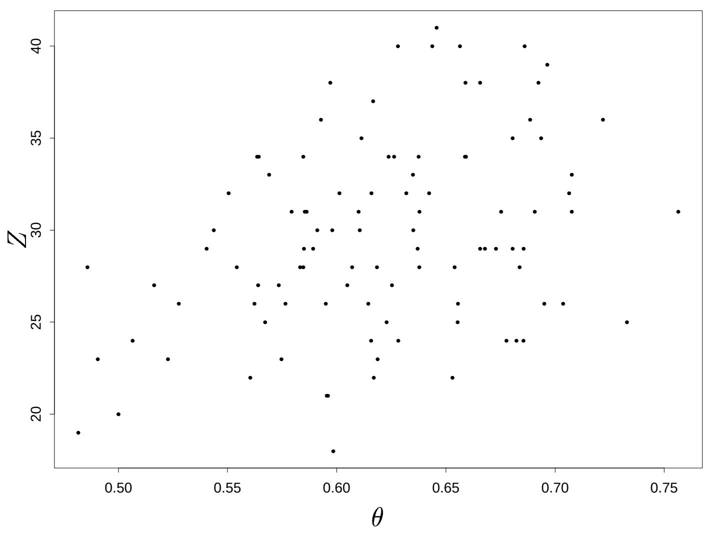


Figure. 4.7: Sample from the joint posterior for the genetic example.

4.4.1 Prediction

Another example of the application of the Gibbs sampler is for prediction. The predictive distribution of a future observation y given past observations x is defined as

$$\pi(y|x) = \int L(y|\theta)\pi(\theta|x)d\theta$$

so that the likelihood $L(y|\theta)$ is averaged across the uncertainty in θ contained in the posterior distribution for θ, $\pi(\theta|x)$. Hence, given a sampled sequence of realisations $\theta^1, ..., \theta^n$ from this posterior, we can estimate

$$\pi(y|x) \approx \frac{1}{n}\sum_{i=1}^{n} L(y|\theta_i).$$

4.5 The Metropolis Hastings Algorithm

An alternative, and more general, way to construct MCMC samplers is as a form of generalised rejection sampling, where values are drawn from approximate distributions and "corrected" in order that, asymptotically, they behave as random observations from the target distribution. This is the motivation for methods such as the Metropolis Hastings algorithm which sequentially draws candidate observations from a distribution, conditional only upon the last observation, thus inducing a Markov chain. The most important aspect of such algorithms is not the Markov property, but the fact that the approximating candidate distributions are improved at each step in the simulation, whereas the distributions used in rejection sampling, for example, remain the same.

The method commonly known as the Metropolis Hastings algorithm, is based upon the observation that given a Markov chain with transition density $\mathcal{K}(\boldsymbol{x}, \boldsymbol{y})$ and exhibiting detailed balance for π i.e.,

$$\pi(\boldsymbol{x})\mathcal{K}(\boldsymbol{x}, \boldsymbol{y}) = \pi(\boldsymbol{y})\mathcal{K}(\boldsymbol{y}, \boldsymbol{x}), \tag{4.8}$$

the chain has stationary density, $\pi(\cdot)$.

The method of rejection sampling begins with a density for generating candidate observations. However, for MCMC, we allow this candidate generating density (or proposal density) to depend upon the current state of the chain, and we denote it by $q(\boldsymbol{x}^t, \boldsymbol{y})$. In general, the induced chain will not satisfy the reversibility condition of (4.8), so we introduce an acceptance function $\alpha(\boldsymbol{x}^t, \boldsymbol{y})$, and accept the candidate observation, so that $\boldsymbol{x}^{t+1} = \boldsymbol{y}$, with probability $\alpha(\boldsymbol{x}^t, \boldsymbol{y})$. However, unlike rejection sampling, if the candidate observation is rejected, the chain remains at $\boldsymbol{x}^t$, so that $\boldsymbol{x}^{t+1} = \boldsymbol{x}^t$.

It can be shown that the optimal form for the acceptance function, in the sense that suitable candidates are rejected least often and computational efficiency is maximised, is given by

$$\alpha(\boldsymbol{x}^t, \boldsymbol{y}) = \min\left(1, \frac{\pi(\boldsymbol{y})q(\boldsymbol{y}, \boldsymbol{x}^t)}{\pi(\boldsymbol{x}^t)q(\boldsymbol{x}^t, \boldsymbol{y})}\right),$$

so that the transition kernel is given by

$$\mathcal{P}_H(\boldsymbol{x}, A) = \int_A \mathcal{K}_H(\boldsymbol{x}, \boldsymbol{y})d\boldsymbol{y} + r(\boldsymbol{x})I_A(\boldsymbol{x}),$$

where

$$\mathcal{K}_H(\boldsymbol{x}, \boldsymbol{y}) = q(\boldsymbol{x}, \boldsymbol{y})\alpha(\boldsymbol{x}, \boldsymbol{y}),$$
$$r(\boldsymbol{x}) = 1 - \int_E q(\boldsymbol{x}, \boldsymbol{y})\alpha(\boldsymbol{x}, \boldsymbol{y})d\boldsymbol{y}, \tag{4.9}$$

and $\mathcal{K}_H$ satisfies the reversibility condition of (4.8), implying that the kernel, $\mathcal{P}_H$ also preserves detailed balance for π.

Thus, the Metropolis Hastings method can be written algorithmically as follows.

STEP 1. GIVEN THE CURRENT POSITION, $\boldsymbol{X}^t = \boldsymbol{x}$, GENERATE A NEW VALUE, $\boldsymbol{y}$, FROM THE DISTRIBUTION $q(\boldsymbol{x}, \boldsymbol{y})$.

STEP 2. CALCULATE

$$\alpha(\boldsymbol{x}, \boldsymbol{y}) = \min\left(1, \frac{\pi(\boldsymbol{y})q(\boldsymbol{y}, \boldsymbol{x}^t)}{\pi(\boldsymbol{x}^t)q(\boldsymbol{x}^t, \boldsymbol{y})}\right).$$

STEP 3. WITH PROBABILITY $\alpha(\boldsymbol{x}, \boldsymbol{y})$, SET $\boldsymbol{X}^{t+1} = \boldsymbol{y}$, ELSE SET $\boldsymbol{X}^{t+1} = \boldsymbol{x}$.

STEP 4. RETURN TO STEP 1.

The Metropolis Hastings algorithm has the advantage over the Gibbs Sampler, in that it is not necessary to know all of the conditional distributions, we need only simulate from q, which we can choose arbitrarily. Moreover, and this can be of crucial importance, we only need to know π up to proportionality, since any constants of proportionality cancel in the numerator and denominator of the calculation of α. The price for this simplicity is that if q is poorly chosen, then the number of rejections can be high, so that the efficiency of the procedure can be low.

Example

Suppose that we were interested in sampling from the standard normal distribution, and that we choose to use a proposal distribution of the form

$$q(x, y) \sim N(x, \sigma^2).$$

Then the acceptance probability is given by

$$\alpha(x, y) = \exp\left(-\frac{1}{2}(y^2 - x^2)\right), \tag{4.10}$$

and we can use the following S-Plus function to simulate the chain.

```
met.has <- function(n,x0,sigma){
    x <- array(0,n+1)
    x[1] <- x0
    for(t in 2:n){
       y <- rnorm(1,x[t-1],sigma^2)
       alpha <- exp((x[t-1]^2 - y^2)/2)
       accept <- min(1,alpha)
       u <- runif(1)
       if (u <= accept) x[t] <- y
       else x[t] <- x[t-1]
    }
    plot(1:length(x),x,type='l',lty=1,xlab="t",ylab="x")
    x
}
```

Figure 4.8 plots the resulting output for $\sigma^2 = 0.1$ and $\sigma^2 = 10$.

Notice that the acceptance function is independent of σ, but that the value of σ has a significant impact upon the acceptance rate of the chain. This is because the proposal with $\sigma^2 = 0.1$ is much closer to the target distribution, so that more sensible candidates are generated and subsequently accepted. However, the proposal with $\sigma^2 = 10$, generates candidate observations too far out in the tail to come from the target distribution and these are subsequently rejected.

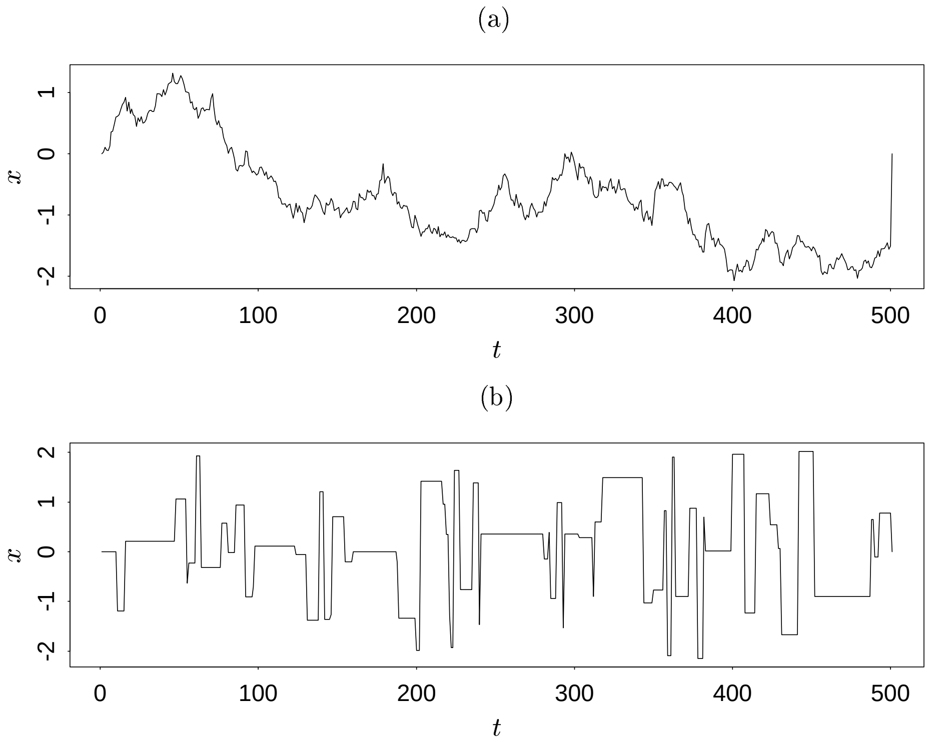


Figure. 4.8: Sample paths for Metropolis Hastings algorithms with (a) $\sigma^2 = 0.1$ and (b) $\sigma^2 = 10$.

In the case where the candidate generating function is symmetric i.e., $q(\boldsymbol{x}, \boldsymbol{y}) = q(\boldsymbol{y}, \boldsymbol{x})$, the acceptance function reduces to

$$\alpha(\boldsymbol{x}, \boldsymbol{y}) = \min\left(1, \frac{\pi(\boldsymbol{y})}{\pi(\boldsymbol{x})}\right). \tag{4.11}$$

This special case is the original Metropolis algorithm. There are a number of other special cases.

4.5.1 Random Walk Metropolis

If $q(\boldsymbol{x}, \boldsymbol{y}) = f(|\boldsymbol{y} - \boldsymbol{x}|)$ for some arbitrary density f, then the kernel driving the chain is a random walk, since the candidate observation is of the form $\boldsymbol{y}^{t+1} = \boldsymbol{x}^t + \boldsymbol{z}$, where $\boldsymbol{z} \sim f$. There are many common choices for f, including the uniform distribution on the unit disk, or a multivariate normal or t-distribution. Note that these are symmetric, so that the acceptance probability is of the simple form given in (4.11). Figure 4.9 highlights how the proposal distribution is centred about the current state of the chain.

Example

In the previous example, we took

$$\pi(x) \propto \exp\left(-\frac{x^2}{2}\right)$$

with $q(x, y) \sim N(x, \sigma^2)$. Suppose that we set $\sigma^2 = 0.25$, then,

$$q(x, y) \propto \exp\left(-\frac{(y-x)^2}{0.5}\right).$$

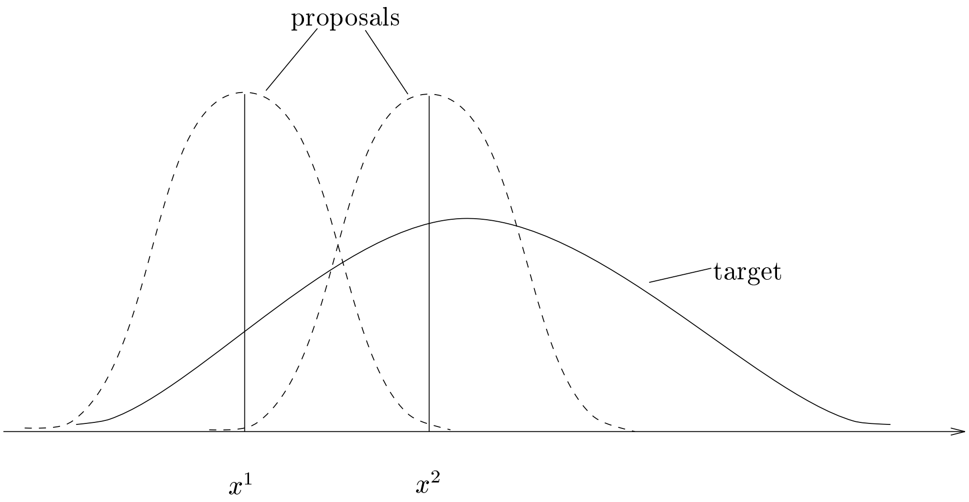


Figure. 4.9: Random Walk proposals.

Thus, at each stage, the proposal distribution depends upon the current state of the chain and is symmetric about that position.

The acceptance probability is given in Equation 4.10, and so the Random Walk Metropolis algorithm may be implemented in S-Plus as follows:

```
rw.met <- function(n,x0){
    x <- array(0,n+1)
    x[1] <- x0
    for(t in 2:n){
       y <- rnorm(1,x[t-1],0.25)
       alpha <- exp((x[t-1]^2 - y^2)/2)
       accept <- min(1,alpha)
       u <- runif(1)
       if (u <= accept) x[t] <- y
       else x[t] <- x[t-1]
    }
    plot(1:length(x),x,type='b',lty=1,xlab="t",ylab="x")
    x
}
```

Which is essentially the same as the `met.has()` function, but with `sigma` set to 0.25. Figure 4.10 plots the output for this example. You can see that "up" and "down" moves appear roughly equally likely and that occasionally the chain gets "stuck" for a few iterations, when proposals are rejected. Though, in this case, the rejection rate seems extremely low.

4.5.2 The Independence Sampler

If $q(\boldsymbol{x},\boldsymbol{y}) = f(\boldsymbol{y})$, then the candidate observation is drawn independently of the current state of the chain. In this case, the acceptance probability can be written as

$$\alpha(\boldsymbol{x},\boldsymbol{y}) = \min\left(1, \frac{w(\boldsymbol{y})}{w(\boldsymbol{x})}\right),$$

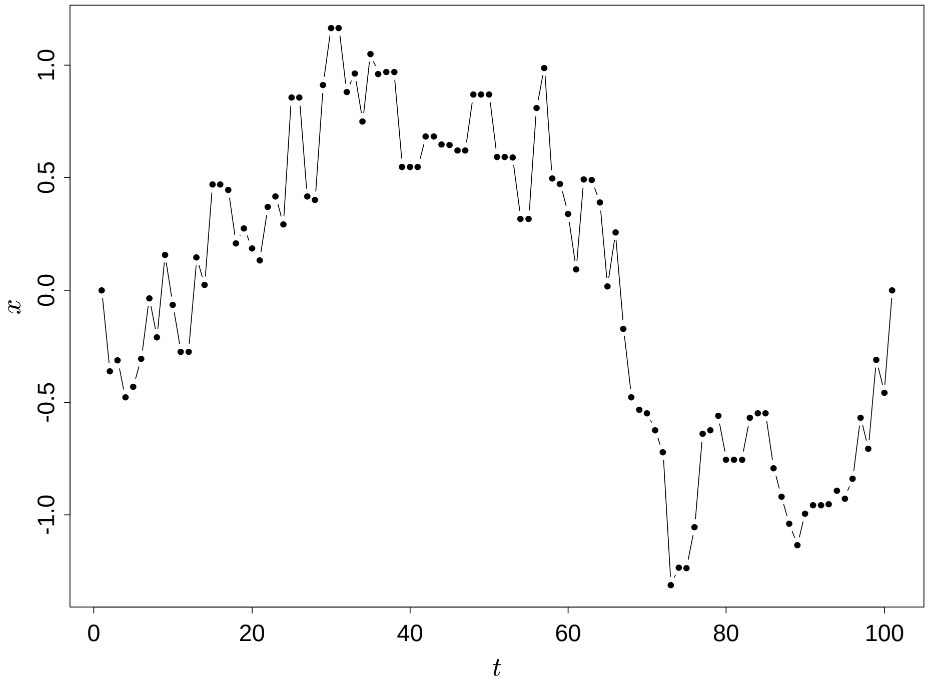


Figure. 4.10: Sample path for Random Walk Metropolis algorithm.

where $w(\boldsymbol{x}) = \pi(\boldsymbol{x})/f(\boldsymbol{x})$ is the importance weight function that would be used in importance sampling given observations generated from f. In some senses, the ideas of importance sampling and the independence sampler are closely related. The essential difference between the two is that the importance sampler builds probability mass around points with large weights by choosing those points relatively frequently. On the other hand, the independence sampler builds up probability mass, on points with high weights, by remaining at those points for long periods of time.

Example

Here, we take

$$\pi(x) \propto \left(1 + \frac{x^2}{3}\right)^{-2},$$

and use proposal

$$q(x, y) \propto \exp\left(-\frac{y^2}{0.5}\right).$$

Clearly, the proposal distribution takes no account of the current state of the chain, but it is essential that the proposal have "heavier tails" than the target distribution i.e., the proposal density covers a wider range than π.

The corresponding acceptance function is,

$$\alpha(x, y) = \left(\frac{3 + x^2}{3 + y^2}\right)^2 \exp\left(2(y^2 - x^2)\right).$$

In this case, the algorithm can be implemented by altering the two lines in `rw.met` which define the proposal and alpha functions. Thus, we use

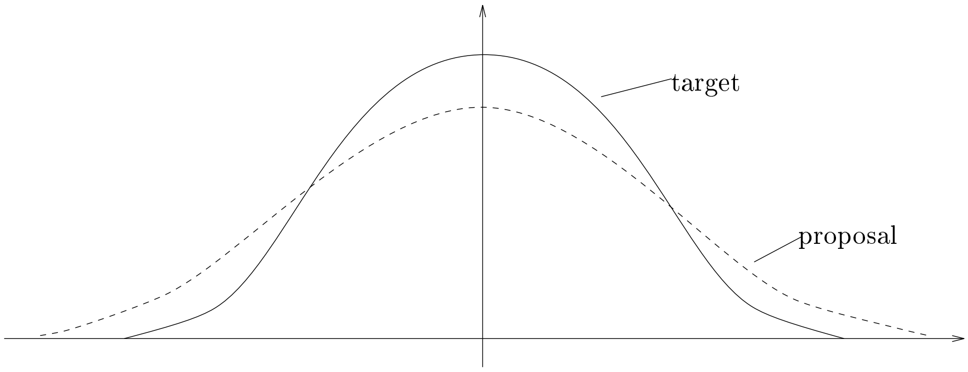


Figure. 4.11: Independence Sampler proposal.

```
is.met <- function(n,x0){
    x <- array(0,n+1)
    x[1] <- x0
    for(t in 2:n){
       y <- rnorm(1,0,0.25) # zero mean, not x[t-1]
       alpha <- ((3+x[t-1]^2)/(3+y^2))^2*exp(2*(y^2 - x[t-1]^2 ))
       accept <- min(1,alpha)
       u <- runif(1)
       if (u <= accept) x[t] <- y
       else x[t] <- x[t-1]
    }
    plot(1:length(x),x,type='b',lty=1,xlab="t",ylab="x")
    x
}
```

Figure 4.12 plots the output for this example. As with Figure 4.10, the chain appears to move around the state space fairly rapidly, but occasionally gets stuck for one or two iterations. As before, the rejection rate appears to be quite low.

4.5.3 Single Updates and the Gibbs Sampler

The Metropolis Hastings algorithm need not update all variables in the sample space simultaneously, it can be used in stages, updating variables one-at-a-time, much like the Gibbs Sampler. This is commonly called the single-update Metropolis Hastings algorithm and, in fact, the Gibbs Sampler is a special case of the single-update Metropolis Hastings algorithm. Suppose that in the single-update Metropolis Hastings algorithm, we break each iteration of the Metropolis Hastings algorithm into k steps, and let q_j denote a proposal for candidates in the jth co-ordinate direction, so that

$$q_j(\boldsymbol{x},\boldsymbol{y}) = \begin{cases} \pi(\boldsymbol{y}_j|\boldsymbol{x}_{(j)}) & \boldsymbol{y}_{(j)} = \boldsymbol{x}_{(j)}, \quad j = 1,...,k. \\ 0 & \text{else} \end{cases}$$

With this proposal, the acceptance probability at the jth step is given by

$$\alpha_j(\boldsymbol{x},\boldsymbol{y}) = \min\left(1, f(\boldsymbol{x},\boldsymbol{y})\right),$$

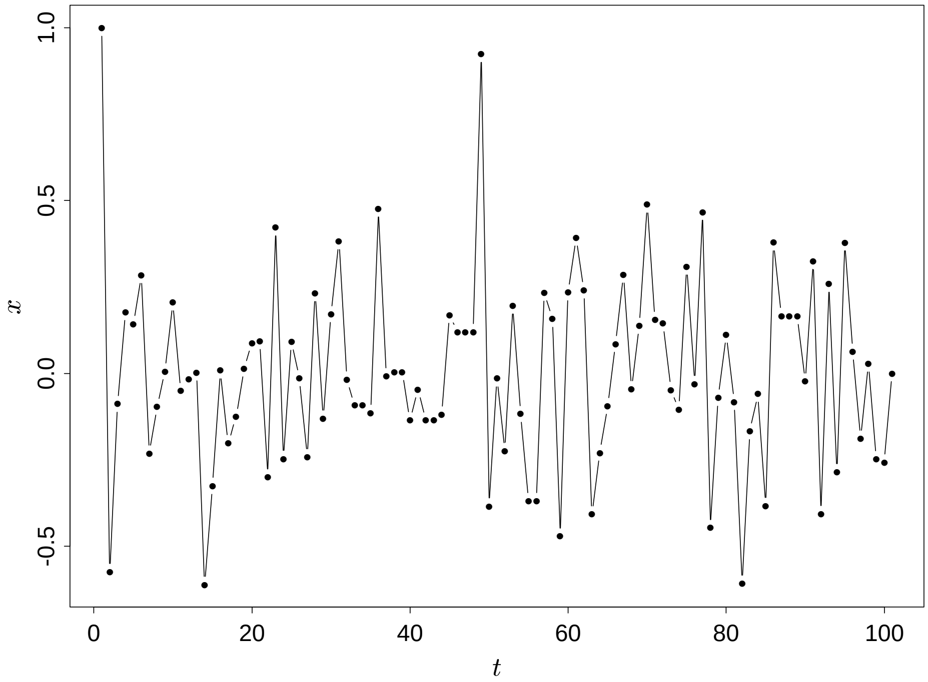


Figure. 4.12: Sample path for Independence Sampler algorithm.

where,

$$\begin{aligned}
f(\boldsymbol{x},\boldsymbol{y}) &= \frac{\pi(\boldsymbol{y})q_j(\boldsymbol{y},\boldsymbol{x})}{\pi(\boldsymbol{x})q_j(\boldsymbol{x},\boldsymbol{y})} \\
&= \frac{\pi(\boldsymbol{y})/\pi(\boldsymbol{y}_j|\boldsymbol{x}_{(j)})}{\pi(\boldsymbol{x})/\pi(\boldsymbol{x}_j|\boldsymbol{y}_{(j)})} \\
&= \frac{\pi(\boldsymbol{y})/\pi(\boldsymbol{y}_j|\boldsymbol{y}_{(j)})}{\pi(\boldsymbol{x})/\pi(\boldsymbol{x}_j|\boldsymbol{x}_{(j)})}, \quad \text{since } \boldsymbol{y}_{(j)} = \boldsymbol{x}_{(j)} \\
&= \frac{\pi(\boldsymbol{y}_{(j)})}{\pi(\boldsymbol{x}_{(j)})}, \\
&\quad \text{by definition of conditional probability for } \boldsymbol{\theta} = (\boldsymbol{\theta}_j, \boldsymbol{\theta}_{(j)}) \\
&= 1, \text{ since } \boldsymbol{y}_{(j)} = \boldsymbol{x}_{(j)}.
\end{aligned}$$

Thus, at each step, the only possible jumps are to parameter vectors $\boldsymbol{y}$, that match $\boldsymbol{x}$ on all components other than the jth, and these are automatically accepted.

Hence,

$$q(\boldsymbol{x},\boldsymbol{y}) = \prod_{j=1}^{p} q_j([\boldsymbol{y}_{<j}, \boldsymbol{x}_{\geq j}], [\boldsymbol{y}_{\leq j}, \boldsymbol{x}_{>j}]),$$

where

$$\boldsymbol{x}_{>j} = (\boldsymbol{x}_i, \;\; i > j),$$

$$\boldsymbol{y}_{<j} = (\boldsymbol{y}_i, \;\; i < j)$$

and, since $\alpha_j(\boldsymbol{x},\boldsymbol{y}) = 1 \quad \forall j$, the transition distribution can, by simple manipulation, be re-written in the same form as the Gibbs transition density given in (4.5).

4.5.4 Hybrid Algorithms

Hybrid algorithms involve merging two or more algorithms in order to improve desirable aspects of convergence. Given a set of Markov kernels $\mathcal{K}_1, ..., \mathcal{K}_m$, each with invariant distribution π, it can be shown that, under weak conditions on the $\mathcal{K}_i$, a hybrid algorithm consisting of *mixtures* or *cycles* of these kernels will produce a chain with the desired stationary distribution. In a mixture, probabilities $\alpha_1, ..., \alpha_m$ are specified and, at each step, one of the kernels is selected according to these probabilities whilst, in a cycle, each of the kernels are used in a pre-determined sequence which is repeated until the sampler is stopped. Such algorithms may be particularly useful when the target distribution is multi-modal, since transition kernels can be chosen so as to ensure that each mode is properly "explored".

One particularly useful consequence of this observation is that Metropolis Hastings steps may be introduced into the Gibbs Sampler, so that components whose conditional distributions are of a standard form may be sampled directly from the full conditional, whilst those with non-standard distributions are updated via a Metropolis Hastings step. Generally, this is simpler to implement than the alternative methods discussed at the end of Section 4.3.1, but may result in a "slow" Markov chain, due to the rejection of Metropolis proposals restricting "movement" around the state space in these directions.

Example

If we have a target distribution of the form

$$\pi(x) \propto \left(1 + \frac{x^2}{3}\right)^{-2},$$

then we might choose the two proposals

$$q_1(x, y) \propto \exp\left(-\frac{y^2}{0.5}\right)$$

and

$$q_2(x, y) \propto \exp\left(-\frac{(y-x)^2}{0.5}\right).$$

Then, at each iteration, we can choose one of these two proposals with equal probability, for example. We then get the following acceptance probabilities associated with the respective proposals,

$$\alpha_1(x, y) = \min\left[1, \left(\frac{3+x^2}{3+y^2}\right)^2 \exp\left(2(y^2 - x^2)\right)\right]$$

and

$$\alpha_2(x, y) = \min\left[1, \left(\frac{3+x^2}{3+y^2}\right)^2\right].$$

We can then use the following S-Plus code to implement the sampler.

```
hyb.met <- function(n,x0){
    x <- array(0,n+1)
    x[1] <- x0
    for(t in 2:n){
       u1 <- runif(1)
       if(u1 <= 0.5){
```

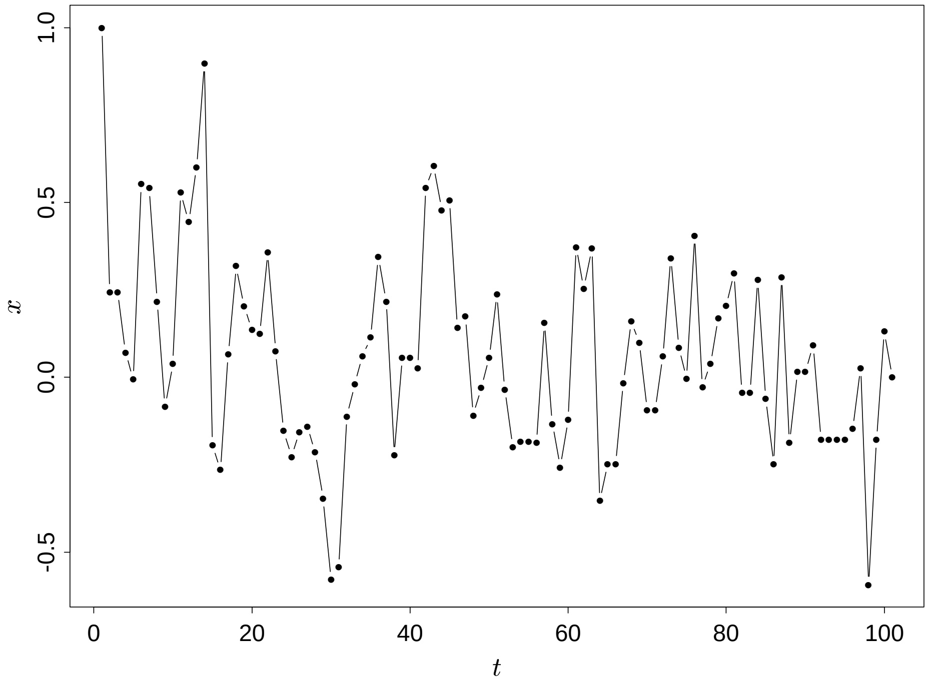


Figure. 4.13: Sample path for hybrid Independence and Random Walk samplers.

```
            y <- rnorm(1,0,0.25)
            alpha <- ((3+x[t-1]^2)/(3+y^2))^2*exp(2*(x[t-1]^2 - y^2))
        }
        else{
            y <- rnorm(1,x[t-1],0.25)
            alpha <- ((3+x[t-1]^2)/(3+y^2))^2
        }
        accept <- min(1,alpha)
        u <- runif(1)
        if (u <= accept) x[t] <- y
        else x[t] <- x[t-1]
    }
    plot(1:length(x),x,type='b',lty=1,xlab="t",ylab="x")
    x
}
```

Figure 4.13 provides an example of the output arising from this hybrid sampler, and it is clear that it's behaviour is not dissimilar to that of its constituent samplers.

4.6 Implementation

In this section we discuss various issues associated with the practical implementation of the MCMC method.

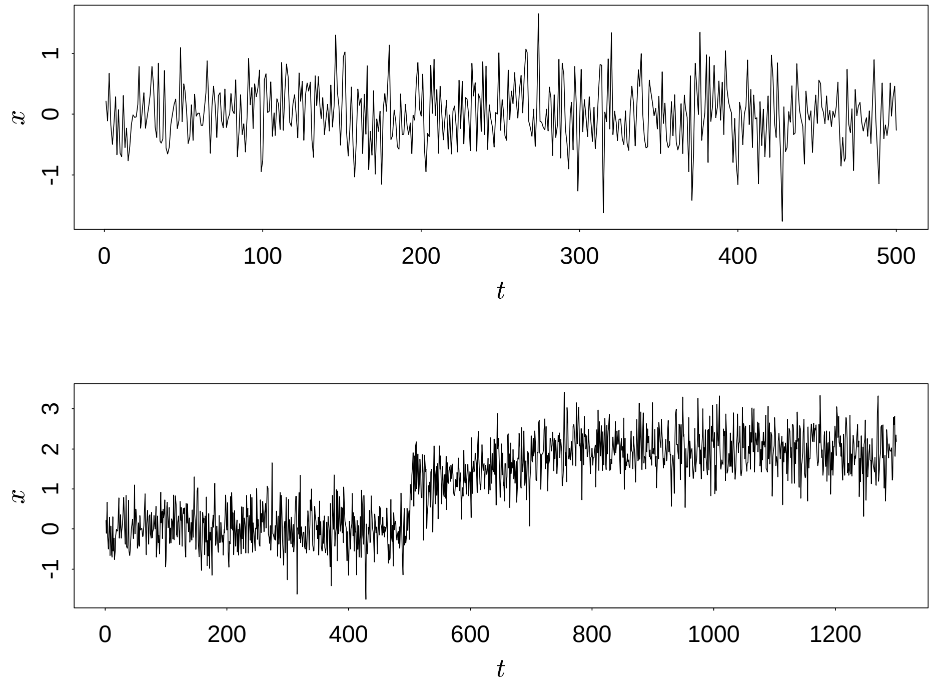


Figure. 4.14: MCMC sample paths.

4.6.1 How Many iterations?

In general, we use ergodic averages over realisations of our Markov chain to estimate functionals of interest. In order to reduce the possibility of bias caused by the effect of starting values, observations within an initial transient or *burn-in* period are discarded, and the remaining output from the chain used as a basis for inference.

As we discussed earlier, we wait for our Markov chain to reach a state of equilibrium, before we use the output from the chain as the basis for inference. Since the equilibrium distribution is independent of the initial state of the chain, one way to determine whether or not the chain has *converged* to its equilibrium distribution is to try and measure the chains dependence on its starting value. When this dependence is essentially zero, we say that the chain has *converged.*

In practice, we would run our Markov chain for $n + n_0$ iterations. We discard observations $x^0, ..., x^{n_0}$ and use observations $x^{n_0+1}, ..., x^n$ for inference. But how do we determine how large n_0 should be? There are a number of techniques which have been developed. One of the earliest was called the "eye-balling technique" and involved simply looking at the output to determine whether or not the chain had converged.

Figure 4.14 plots the output from an MCMC sampler. The first 500 iterations do appear to have already "settled down" to a constant distribution or pattern. If you eye-balled only the first 500 iterations, you might well say that the chain has reached convergence. However, by plotting more iterations (the lower figure), you can see that this is not the case.

Another early technique is the so-called "thick-pen" technique. This involves running two chains (started in two different states) and plotting the output on a single graph. You then take a "thick pen" and when your pen touches both lines of the plot, you say that the chains have converged.

However, a number of more elaborate (mathematical) techniques have been developed. The most popular is based upon running a number of chains in parallel, and from different starting points, and then essentially using an *Analysis of Variance* technique to assess whether or not each of the chains have the same distribution. Specifically, the method involves calculating the variance of the means of the chains,

$$B = \frac{1}{m-1}\sum_{j=1}^{m}(\bar{\theta}_{\cdot j} - \bar{\theta}_{\cdot\cdot})^2,$$

where θ_{ij} denotes the state of chain j at time i, and we run m chains. We call this the *between chain variance*. We then calculate the *within chain variance*, W, which is the mean of the variances of each of the chains i.e.,

$$W = \frac{1}{m}\sum_{j=1}^{m} s_j^2,$$

where

$$s_j^2 = \frac{1}{n-1}\sum_{i=1} n(\theta_{ij} - \bar{\theta}_{\cdot j})^2.$$

Finally, we calculate the weighted average

$$\hat{V} = \frac{n-1}{n}W + \frac{B}{m},$$

and monitor the ratio

$$R = \frac{\hat{V}}{W}$$

until it reduces to the value 1.

In essence, this method examines the different chains and tries to measure the variation associated with the chains. There is variation associated with each chain individually (since these are stochastic simulations) but there is also some variation between the chains, because they were started in different states. The chains have converged when the variation between the chains is very small (since they will all be samples from the target distribution), so the ratio R will be close to 1.

4.6.2 How Many More Iterations?

In any practical application of MCMC, it is necessary to determine how long the simulations need to be run. Since the sampler output is generally used as a basis for inference, the number of iterations required will be dependent upon the problem at hand. However, computation time and storage limits may also be a consideration. For example, it may often be necessary to *thin* the observations by saving only every kth observation, for some suitable value of k.

For an *iid* sample of size n from a posterior distribution π, the standard deviation of the sample mean of a scalar functional $\theta(\boldsymbol{X})$ is $\sigma/\sqrt{n}$, where σ is the posterior standard deviation of θ. If a preliminary estimate of σ is available, perhaps from some pilot run of the chain, then this can be used to estimate the sample size that would be required for an *iid* sample.

Alternatively, if the series can be approximated by a first order autoregressive process, then the asymptotic standard deviation of the sample mean is given by

$$\frac{\sigma}{\sqrt{n}}\sqrt{\frac{1+\rho}{1-\rho}},$$

where ρ is the autocorrelation of the series $\theta(\boldsymbol{X}^t)$, and this may be used to estimate the sample size required. However, for many MCMC samplers, the variance does not decrease with $1/\sqrt{n}$.

An alternative approach uses the Central Limit Theorem to justify the assumption that the ergodic average, given by

$$\bar{\theta}_n = \frac{1}{n}\sum_{t=1}^{n}\theta(\boldsymbol{x}^t) \overset{n\to\infty}{\longrightarrow} \theta$$

for some functional $\theta(\cdot)$, will be asymptotically normal for large n. We then estimate the value of n necessary to ensure that

$$\mathbb{P}[\theta - \epsilon \leq \bar{\theta}_n \leq \theta + \epsilon] = 1 - \alpha,$$

for some $\epsilon > 0$ and $0 < \alpha < 1$. This suggests taking

$$n \approx \widehat{Var}(\bar{\theta})\left(\frac{\Phi^{-1}(1-\alpha/2)}{\epsilon}\right)^2,$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function and $\widehat{Var}(\bar{\theta})$ is an estimate of the variance of the ergodic average, based upon the observations gained from the pilot run.

4.6.3 One Run or Many?

One of the most contentious issues associated with the implementation of MCMC algorithms, is in choosing whether to run one long chain or several shorter chains in parallel. The argument for taking a single long run, is that the chain will be "closer" to the target distribution at the end of one long run than it would be at the end of any number of shorter ones, and that several shorter runs may be wasteful, in that initial "warm-up" periods must be discarded from each. On the other hand, proponents of the "many replications" approach argue that, whilst a single run will eventually cover the entire sample space, by taking a number of parallel replications, we can guard against a single chain leaving a "significant proportion" of the sample space unexplored. By taking several chains, each started in different states, it is also possible to monitor the sample paths in order to determine how well the chains have *mixed* i.e., to what extent the output from the different chains are indistinguishable. In essence, multiple replications protect against bias by attempting to ensure that the sampler output covers the entire sample space, whilst one long run provides less variable estimates, since ergodic averages are based upon a larger sample.

Note that an alternative to running multiple replications is to use regenerative methods to *restart* the chain at appropriate *regeneration times*. In this manner, a single long chain may be run, and then split, to produce multiple replications which will be closer to the stationary distribution than would many independent shorter chains. However, one advantage of the use of traditional parallel runs is that, in certain applications, the implementation can take advantage of parallel computing technology, whilst the computation of regenerations can be both difficult and expensive.

4.6.4 Starting Point Determination

Another important issue is the determination of suitable starting points. If we choose to run several replications in parallel, some people argue that the distribution of starting points should be over-dispersed with respect to the target distribution. A number of methods have been proposed for generating initial values for MCMC samplers. One possibility is to use a simple mode-finding algorithm to locate regions of high density and sampling from a mixture of t-distributions located at these modes, to generate suitable starting values. A number of alternative approaches have also been suggested. For example, a *Simulated Annealing* algorithm can be used to sample initial values, see Section 5.2.

4.6.5 Alternative Parameterisations

Another important implementational issue is that of *parameterisation*. High correlations between variables can slow down the convergence of an MCMC sampler, considerably. Thus, it is common to re-parameterise the target distribution in order that these correlations be reduced.

It should also be noted that by re-parameterising, it is sometimes possible to ensure that the full conditionals conform to standard distributions. This simplifies the random variate generation in the Gibbs Sampler, for example.

4.6.6 Auxiliary Variables

A related issue (also related to the idea of data augmentation) is that of the introduction of *auxiliary variables* to improve convergence, particularly in the case of highly multi-modal target densities. This method involves the introduction of one or more additional variables $\boldsymbol{u} \in U$ to the state variable, $\boldsymbol{x}$. The joint distribution of $\boldsymbol{x}$ and $\boldsymbol{u}$ is defined by taking $\pi(\boldsymbol{x})$ to be the marginal for $\boldsymbol{x}$ and specifying the conditional, $\pi(\boldsymbol{u}|\boldsymbol{x})$, arbitrarily. We then construct a Markov chain on $E \times U$, which (at iteration t) alternates between two types of transition. First, $\boldsymbol{u}^{t+1}$ is drawn from $\pi(\boldsymbol{u}|\boldsymbol{x}^t)$. Then, $\boldsymbol{x}^{t+1}$ is generated by any method which preserves detailed balance for the conditional $\pi(\boldsymbol{x}|\boldsymbol{u})$ i.e., from some transition function $p(\boldsymbol{x}^t, \boldsymbol{y})$, such that

$$\pi(\boldsymbol{x}|\boldsymbol{u})p(\boldsymbol{x}, \boldsymbol{y}) = \pi(\boldsymbol{y}|\boldsymbol{u})p(\boldsymbol{y}, \boldsymbol{x}).$$

It can be shown that such an algorithm produces observations $\{\boldsymbol{x}^t\}$, with stationary distribution π, and that these techniques can be used to overcome problems associated with both slow convergence and multi-modality.

4.7 Model Uncertainty and Reversible Jump MCMC

The previous Markov chain Monte Carlo approaches depend on parameter spaces of fixed dimension, and summarise these in the form of various statistics, obtained as sample estimates. However, what happens when there is uncertainty regarding the model? To illustrate where such a problem may occur, we return to the autoregressive time series example that we discussed earlier.

Autoregressive example - revisited

Recall that there are N observations, denoted by $\boldsymbol{x} = (x_1, ..., x_N)$, generated from an AR process of order k. Then,

$$\boldsymbol{x}_k = X_k \boldsymbol{a} + \varepsilon,$$

where,

$$\varepsilon \sim N(\boldsymbol{0}, \sigma^2 I),$$

and we have the priors,

$$\boldsymbol{a} \sim N(\boldsymbol{\mu}_a, \Sigma_a); \quad \sigma^2 \sim \Gamma^{-1}(\alpha, \beta).$$

However, within this analysis we are implicitly conditioning on k, the order of the AR process. What happens if we do not know the order of the process? Often, not only are the unknown parameters of interest, but also the order of the process, which tells us about the dependence structure of the data on previous observations.

Thus, we now consider an additional unknown within the analysis, that of the underlying order of the AR process. Within a Bayesian analysis we can incorporate this model uncertainty by considering the model itself to be an unknown parameter to be estimated. There may be many questions of interest within an analysis of this form. For example, we may not only be interested in the posterior distribution of $\boldsymbol{a}$, conditional on the order of the process, but also the probability that the data fits a given AR(k) process. The issue relating to the different models is often of particular interest, since insight can often be obtained regarding the underlying dynamics of the system of interest.

4.7.1 Posterior distribution

Within the Bayesian framework where there is model uncertainty, as the parameters do not have a fixed value, but some distribution, so the different possible models have a distribution. Thus, we can extend the posterior distribution to be jointly over the parameters and the models. We let the set of all different possible models be denoted by $\boldsymbol{m}$, and the individual models by $m \in \boldsymbol{m}$. We can then calculate the joint posterior distribution over the model and parameter space. If we let $\boldsymbol{\theta}$ denote the set of parameters, and $\boldsymbol{x}$ the data, then,

$$\pi(\boldsymbol{\theta}, m|\boldsymbol{x}) \propto f_m(\boldsymbol{x}|\boldsymbol{\theta})p_m(\boldsymbol{\theta})p(m),$$

where $f_m(\boldsymbol{x}|\boldsymbol{\theta})$ is the joint probability distribution under model m, $p_m(\boldsymbol{\theta})$ is the prior of the parameters under model m, and $p(m)$ is the prior for model m.

There are generally many different posterior summary statistics of interest, and we discuss the most common that are used in practice.

4.7.2 Posterior model probabilities

When there is uncertainty regarding the model, the posterior probabilities of the different models are often of particular interest, since they tell us something about the underlying dynamics that we are modelling. The posterior model probabilities are able to quantitatively discriminate between the different competing models, and so measure the degree of uncertainty regarding the different models. For example, in the autoregressive example, the order of the process tells us the dependence structure of the data, in terms of its dependence on previous values. In addition, we may also be interested in the posterior probability, say, of a particular feature within the different models. Such statistics may often be of interest within such areas as medical or social statistics, where we may wish to ascertain whether particular factors influence the spread of certain diseases, or different behavioural traits.

If we let $\boldsymbol{m} = (m_1, ..., m_k)$ denote the set of possible models, the corresponding posterior model probability for model m_i is therefore given by,

$$\pi(m_i|\boldsymbol{x}) = \frac{L(\boldsymbol{x}|m_i)p(m_i)}{\sum_{i=1}^{k} L(\boldsymbol{x}|m_i)p(m_i)},$$

where,

$$L(\boldsymbol{x}|m_i) = \int f_{m_i}(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|m_i)d\boldsymbol{\theta},$$

and $f_{m_i}(\boldsymbol{x}|\boldsymbol{\theta})$ denotes the joint probability distribution under model m_i and corresponding parameters $\boldsymbol{\theta}$. We are also able to average the parameters across all the different plausible models, and so including model-uncertainty into our parameter estimation procedure.

Bayesian model-averaging overcomes the problem associated with different plausible models giving very different parameter estimates, which can often occur in practice. The model-averaging approach obtains a single parameter estimate based on all plausible models by weighting each according to their corresponding posterior model probability, and so allows us to incorporate model uncertainty into the estimate of the parameter. Formally, the model-averaged posterior distribution of some parameter vector $\boldsymbol{\theta}$ common to all models m_i, $i = 1, ..., k$ is given by,

$$\pi(\boldsymbol{\theta}|\boldsymbol{x}) = \sum_{i=1}^{k} \pi(\boldsymbol{\theta}|\boldsymbol{x}, m_i)\pi(m_i|\boldsymbol{x}).$$

The posterior distribution is, in general, very complex with little inference directly obtainable, and so ideally we would be able to sample from the distribution using an MCMC approach. However, generally, the different models contain different sets of parameters, so that the models are of varying dimensions, and the posterior distribution is multi-dimensional. In such situations, the Gibbs sampler and Metropolis Hastings algorithm cannot be applied, as they are only defined for fixed dimensional distributions. Thus we need to be able to construct a Markov chain whose states can be of different dimensions, and has the correct stationary distribution. If we can construct a Markov chain that is able to move between such possible states, then once the chain has reached its stationary distribution, we can obtain a sample from the posterior distribution and obtain corresponding summary statistics of interest. We are able to do this using the following reversible jump MCMC algorithm.

4.7.3 Reversible jump algorithm

Reversible jump MCMC allows us to construct a Markov chain with stationary distribution equal to the joint posterior distribution of both models and parameters, $\pi(\boldsymbol{\theta}, m|\boldsymbol{x})$. This algorithm extends the basic MCMC algorithms to deal with moves between models of different dimensions. It is therefore ideally suited to problems where we wish to explore both parameter and model space simultaneously. We are able to traverse the posterior model space, in terms of the models and corresponding parameters, in a single Markov chain. Within each iteration of the Markov chain, the algorithm involves updating the parameters, given the model, using the standard MCMC algorithms discussed previously and then updating the dimension (or model) using the following reversible jump procedure.

We construct a Markov chain which moves between models as follows. Suppose that the chain is in model m, with parameter vector $\boldsymbol{\theta}$, and that we propose to move to the new model m' with associated parameter vector $\boldsymbol{\theta}'$. Generally, there are several different move types that can be used to move between the different models in order to traverse the state space. These are often in the form of adding or deleting a given number of parameters from the model. We let the probability of proposing to move to a model m', say, given that the chain is currently in state m, be denoted by $P(m'|m)$, with an analogous notation for the reverse move.

In addition, we let $|\boldsymbol{\theta}|$ and $|\boldsymbol{\theta}'|$ denote the dimensions of the corresponding parameter vectors. We initially assume that the proposed move increases the dimension of the model, so that $|\boldsymbol{\theta}'| > |\boldsymbol{\theta}|$, and consider the alternative case later. Then, suppose that within this proposed move from model m to m', we are able to generate the new parameter vector, $\boldsymbol{\theta}'$, as a function of the current parameters, $\boldsymbol{\theta}$, and a vector of random variables $\boldsymbol{u}$, where $|\boldsymbol{u}| = |\boldsymbol{\theta}'| - |\boldsymbol{\theta}|$ with some given proposal distribution $q(\boldsymbol{u})$. Thus, $\boldsymbol{\theta}' = g(\boldsymbol{\theta}, \boldsymbol{u})$, for some known deterministic function g. Then, we accept this proposed model update with probability,

$$\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}') = \min(1, A),$$

where

$$A = \frac{\pi(m', \boldsymbol{\theta}'|\boldsymbol{x})P(m|m')}{\pi(m, \boldsymbol{\theta}|\boldsymbol{x})P(m'|m)q(\boldsymbol{u})} \left| \frac{\partial \boldsymbol{\theta}'}{\partial(\boldsymbol{\theta}, \boldsymbol{u})} \right|,$$

where the final term is a Jacobian term. Essentially, we move from $(\boldsymbol{\theta}, \boldsymbol{u})$ to $\boldsymbol{\theta}' = g(\boldsymbol{\theta}, \boldsymbol{u})$ and the Jacobian term results from the transformation of variables. For example, in the simple case, where $|\boldsymbol{\theta}| = 1$ and $|\boldsymbol{u}| = 1$, so that $\boldsymbol{\theta} = \theta$ and $\boldsymbol{u} = u$, and we let $\boldsymbol{\theta}' = (\theta_1', \theta_2')$, then the Jacobian term is equal to,

$$\left|\frac{\partial \boldsymbol{\theta}'}{\partial(\theta, u)}\right| = \begin{vmatrix} \frac{\partial \theta_1'}{\partial \theta} & \frac{\partial \theta_1'}{\partial u} \\ \frac{\partial \theta_2'}{\partial \theta} & \frac{\partial \theta_2'}{\partial u} \end{vmatrix}.$$

The reverse move, for moving between models m' and m is completely defined by the function g, such that $(\boldsymbol{\theta}, \boldsymbol{u}) = g^{-1}(\boldsymbol{\theta}')$. The corresponding acceptance probability of this move is simply,

$$\alpha(\boldsymbol{\theta}', \boldsymbol{\theta}) = \min\left(1, \frac{1}{A}\right),$$

for A given above.

The standard Gibbs sampler, or Metropolis Hastings algorithm are used to update the parameters, given the model, and the reversible jump step presented above is used to move between models within the Markov chain. In practice, several different move types may be used within a single problem, usually of different sizes, since there is a general trade-off between having a high acceptance probability for the proposed model moves and the speed with which the chain can move around the state space; small (or local) moves are generally accepted with high probability, but traverse the model space slowly, whereas large (or global) jumps have a lower acceptance rate, but have the potential to move the chain further from the present state. Since we are able to move between different states, or models, within the reversible jump algorithm, only a single chain needs to be run to obtain estimates of the posterior distribution, irrespective of the number of possible models within an analysis (although the length of time for the estimates of the posterior model probabilities to converge is generally dependent on the number of models). Thus, this approach can be implemented even when there are a large number of models, possibly hundreds of thousands, so that consideration of all possible models individually is not feasible. The reversible jump MCMC algorithm can then be used to estimate posterior summary statistics of interest. In particular, the posterior model probabilities are simply estimated as the proportion of times that the Markov chain is in the corresponding model, (effectively integrating out the parameters). In addition, the reversible jump algorithm described above is able to simultaneously calculate the model-averaged estimates of the parameters within the Markov chain. These are simply obtained as the estimates of the parameters, irrespective of the model that the chain is in, i.e. we "sum" out the different models.

Example

Suppose that we observe data $\boldsymbol{y} = (y_1, ..., y_N)$, which are assumed to be *iid*, but where there is uncertainty regarding the distribution from which the data have been generated. There are two possible distributions (or models), from which the data may have been generated, an $Exp(\lambda)$ or a $\Gamma(\alpha, \beta)$ distribution, where the parameters are unknown, (recall that an $Exp(\lambda) \equiv \Gamma(1, \lambda)$). To discriminate between these two models, we can calculate the joint posterior distribution of the models and parameters. To explore this posterior distribution, we construct a Markov chain that can traverse both parameter and model space. Within each step of the Markov chain we propose to update both the parameters within the model, and then the model itself, using the reversible jump step.

We suppose that we have priors,

$$\lambda \sim \Gamma(2, 1); \quad \alpha \sim \Gamma(4, 2); \quad \beta \sim \Gamma(4, 2)$$

for each of the parameters, for the corresponding models. Then, for the $Exp(\lambda)$ model, the posterior conditional distribution of the parameter is,

$$\lambda \sim \Gamma\left(2+N, 1+\sum_{i=1}^{N} y_i\right).$$

Exercise: Check.

Similarly, for the $\Gamma(\alpha, \beta)$ model, we have that the posterior conditional distribution of the parameters can be expressed as,

$$\begin{aligned} \pi(\alpha|\boldsymbol{y}, \beta) &\propto \alpha \exp(-2\alpha) \prod_{i=1}^{N} \frac{\beta^\alpha}{\Gamma(\alpha)} y_i^{\alpha-1} \\ \beta &\sim \Gamma\left(4+\alpha N, 2+\sum_{i=1}^{N} y_i\right). \end{aligned}$$

Exercise: Check.

Thus, the parameters λ and β are of standard form, and can be updated using the Gibbs sampler, whereas, the posterior conditional distribution of α is non-standard and we use a Metropolis Hastings random walk to update the parameter. The updating of these parameters within a Markov chain can easily be implemented within S-Plus, for a given model. For example, a single update of the parameter α, (for a given value of β) can be obtained using the S-Plus code:

```
updatealpha <- function(y,N,alpha,beta){

    z <- runif(1, alpha - 0.25, alpha + 0.25)
    if (z <= 0) { acc <- 0 }
    else {
        num <- 1
        den <- 1
        for(i in 1:N) {
               num <- num * beta^{z} * y[i]^{z-1}/gamma(z)
               den <- den * beta^{alpha} * y[i]^{alpha-1}/gamma(alpha)
        }
        num <- num * exp(-2*z) * z
        den <- den * exp(-2*alpha) * alpha

        acc <- num/den
    }

    accept <- min(1,acc)
    u <- runif(1)
    if (u <= accept) newalpha <- z
    else newalpha <- alpha

    newalpha
}
```

Note, that we shall use this function in the S-Plus code given later, where we consider the joint posterior distribution over both parameters and model uncertainty.

Finally, we need to incorporate the uncertainty of the model. Then, as we place priors on the parameters, we also need to place a prior probability on each of the possible models, since the model is also regarded as an unknown parameter. We suppose that the prior probability of each model is $\frac{1}{2}$, i.e. each model is equally likely *a priori*.

Then, we can construct a Markov chain that moves over both parameter and model space, to sample from the posterior distribution. Within each step of the Markov chain we update the parameters given the corresponding model, using the Metropolis-Hastings algorithm, (or Gibbs sampler) for the $Exp(\lambda)$ model we update λ; else for the $\Gamma(\alpha,\beta)$ model we update α and β. Then, we propose to update the model, using the following procedure.

Since there are only two models, we always propose to move to the alternative model. Suppose that the current model is $Exp(\lambda)$, and we propose to move to the model $\Gamma(\alpha,\beta)$. Then a possible updating proposal would be to generate a random variable,

$$u \sim \Gamma(a,b),$$

for some $a, b > 0$ and we define the function $g(\lambda, u)$, by,

$$\alpha = u; \quad \beta = \lambda u.$$

This updating procedure uses a moment matching principle, since the mean of both the current and proposed distributions are equal. The corresponding Jacobian term is then given by,

$$\begin{vmatrix} \frac{\partial\alpha}{\partial\lambda} & \frac{\partial\alpha}{\partial u} \\ \frac{\partial\beta}{\partial\lambda} & \frac{\partial\beta}{\partial u} \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ u & \lambda \end{vmatrix} = u.$$

Then, the corresponding acceptance probability for such a move is $\min(1, A)$, for,

$$\begin{aligned} A &= \frac{L_\Gamma(\boldsymbol{y}|\alpha,\beta)p(\alpha)p(\beta)}{L_{Exp}(\boldsymbol{y}|\lambda)p(\lambda)q(u)} \left|\frac{\partial(\alpha,\beta)}{\partial\lambda}\right| \\ &= \frac{\prod_{i=1}^N \frac{\beta^\alpha}{\Gamma(\alpha)} \exp(-\beta y_i) y_i^{\alpha-1} p(\alpha)p(\beta)u}{\prod_{i=1}^N \lambda \exp(-\lambda y_i) p(\lambda) q(u)}. \end{aligned}$$

Conversely, if we propose to move from the $\Gamma(\alpha,\beta)$ model, to the $Exp(\lambda)$ model, then we set,

$$\lambda = \frac{\beta}{\alpha}; \quad u = \alpha,$$

i.e. $(\lambda, u) = g^{-1}(\alpha,\beta)$. The corresponding acceptance probability is $\min(1, A^{-1})$, for A given above.

A single update of the model, can be implemented using the following S-Plus function:

```
updatemodel <- function(y,N,lambda,alpha,beta,model,a,b){

    if (model == 1) {
        v <- rgamma(1,a,b)
        alpha1 <- v
        beta1 <- lambda*v
        lambda1 <- lambda
    }
```

```
    else {
        lambda1 <- alpha/beta
        alpha1 <- alpha
        beta1 <- beta
        v <- alpha
    }

    num <- 1
    den <- 1
    for(i in 1:N){
        num <- num * beta1^{alpha1}*y[i]^{alpha1-1}*
            exp(-beta1*y[i])/gamma(alpha1)
        den <- den * lambda1 * exp(-lambda1*y[i])
    }

    num <- num * 2^4 * alpha1^3 * exp(-2*alpha1)/gamma(4)
    num <- num * 2^4 * beta1^3 * exp(-2*beta1)/gamma(4)
    num <- num * alpha1

    den <- den * lambda1 * exp(-lambda)/gamma(2)
    den <- den * b^a * v^{a-1} * exp(-b*v)/gamma(a)

    acc <- num/den

    u <- runif(1)

    if (model == 1) {
        accept <- min(1,acc)
        if (u <= accept) {
            model <- 2
            alpha <- alpha1
            beta <- beta1
        }
    }
    else {
        accept <- min(1,1/acc)
        if (u <= accept) {
            model <- 1
            lambda <- lambda1
        }
    }
    if (model == 1) {return(model, lambda)}
    else {return(model, alpha, beta)}
}
```

Finally, we can implement this reversible jump procedure, of updating both the parameters and models to sample from the posterior distribution, in S-Plus using the following S-Plus code:

```
rj <- function(nits,y,N,a,b,lambda,alpha,beta){
    x <- array(0,c(nits+1,3))
    no <- array(1,2)

#   set initial values for the parameters:

    x[1,1] <- lambda
    x[1,2] <- alpha
    x[1,3] <- beta
    model <- 1

#   Update the model parameters, given the model
#   (Model 1 - Exp; model 2 - Gamma)

    for(t in 1:nits){

        if (model == 1) {
            x[no[1]+1,1] <- rgamma(1,N + 2, sum(y) + 1) }
        else {
            x[no[2]+1,3] <- rgamma(1,4 + N*x[no[2],2], sum(y) + 2)

            x[no[2]+1,2] <- updatealpha(y,N,x[no[2],2],x[no[2]+1,3])
        }

#   Update the model:

        new <-updatemodel(y,N,x[no(1)+1,1],x[no(2)+1,2],x[no(2)+1,3],model,a,b)

        model <- new$model

        if (model == 1) {
        x[no[1]+1,1] <- new$lambda
        no[1] <- no[1] + 1
        }
        else {
        x[no[2]+1,2] <- new$alpha
        x[no[2]+1,3] <- new$beta
        no[2] <- no[2] + 1
        }
    }
    model
}
```

Suppose that we have data $\boldsymbol{y} = (0.39, 0.60, 3.30, 0.40, 0.47, 0.95, 1.26, 0.93, 0.34, 0.01)$. Then, using the above procedure, (putting `y = c(0.39,...,0.01)`), we can sample from the joint posterior distribution of interest, and obtain sample estimates for the parameters in each of the different models, within the single procedure. For the exponential model, we obtain a posterior estimate of parameter to be,

$$\mathbb{E}_{Exp}(\lambda) = 1.105.$$

Similarly, for the Gamma model, we obtain the estimates,

$$\mathbb{E}_{\Gamma}(\alpha) = 1.246; \quad \mathbb{E}_{\Gamma}(\beta) = 1.601.$$

In addition, we may be interested in the posterior mean of the distribution, rather than the posterior mean of the parameters. These can be clearly obtained, using the analogous procedure, and we obtain, for the exponential model,

$$\mathbb{E}_{\pi}\left(\frac{1}{\lambda}\right) = 1.104,$$

with corresponding 95% symmetric credible interval for the mean of $(0.530, 1.953)$. Similarly for the alternative model,

$$\mathbb{E}_{\Gamma}\left(\frac{\alpha}{\beta}\right) = 0.814,$$

with 95% symmetric credible interval $(0.492, 1.342)$.

We also obtain the posterior model probabilities of 0.665 for the exponential model, and conversely, 0.335 for the gamma model. Thus, the exponential model has much larger posterior support, than the alternative gamma model. In addition, we may wish to calculate the model-averaged estimates of the mean of the distribution of the parameters, denoted by μ_{π}. This is obtained by,

$$\mu_{\pi} = 0.665 \times \mathbb{E}_{Exp}\left(\frac{1}{\lambda}\right) + 0.335 \times \mathbb{E}_{\Gamma}\left(\frac{\alpha}{\beta}\right) = 0.911.$$

We can also obtain the symmetric 95% credible interval for the mean of the population of $(0.504, 1.783)$. Recall that for the individual models, the corresponding 95% credible intervals are $(0.530, 1.953)$ for the exponential model and $(0.492, 1.342)$ for the gamma model.

Exercise: How would the S-Plus code given above be changed, if we assume general Gamma priors on the parameters λ, α and β?

Often, we may wish to consider more general moves, possibly where we do not have a nested structure between models, so that we may wish to propose to simultaneously add and remove parameters within the reversible jump step. Alternatively, we may wish to simultaneously propose to update the parameters within the reversible jump step, independently of their current value. Such moves are not possible in the previous move types considered, where we assume that we can define the proposed parameters in the form $\boldsymbol{\theta}' = g(\boldsymbol{\theta}, \boldsymbol{u})$, where g is a bijective function. Thus we now consider a more general form of the reversible jump algorithm.

General reversible jump updates

We consider a generalisation of the reversible jump updates described above, where, for example, for the given model, we may propose to simultaneously add, remove and/or update existing parameters. These moves can all be incorporated into the set up, where we propose to move from model m to m', where we initially generate a set of random variables $\boldsymbol{u}$ and define $\boldsymbol{\theta}'$ to be a deterministic function of $\boldsymbol{\theta}$ and $\boldsymbol{u}$. Conversely, in the reverse move, when we propose to move from model m' to model m, we simulate a set of $\boldsymbol{u}'$ random variables, and set $\boldsymbol{\theta}$ to be some function of $\boldsymbol{\theta}'$ and $\boldsymbol{u}'$. We let this bijective function from $(\boldsymbol{\theta}, \boldsymbol{u})$ to $(\boldsymbol{\theta}', \boldsymbol{u}')$ be denoted by g, and note that $|\boldsymbol{\theta}| + |\boldsymbol{u}| = |\boldsymbol{\theta}'| + |\boldsymbol{u}'|$. (The previous reversible jump procedure described in Section 4.7.3 is obtained when $|\boldsymbol{u}'| = 0$). We let the proposal densities of $\boldsymbol{u}$ and $\boldsymbol{u}'$ be denoted by $q(\boldsymbol{u})$ and $q'(\boldsymbol{u}')$, respectively. Then, we accept this proposed model update with probability,

$$\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}') = \min(1, A),$$

where

$$A = \frac{\pi(m', \boldsymbol{\theta}'|\boldsymbol{x})P(m|m')q'(\boldsymbol{u}')}{\pi(m, \boldsymbol{\theta}|\boldsymbol{x})P(m'|m)q(\boldsymbol{u})} \left| \frac{\partial(\boldsymbol{\theta}', \boldsymbol{u}')}{\partial(\boldsymbol{\theta}, \boldsymbol{u})} \right|.$$

The reverse move, for moving between models m' and m is completely defined by the function g, and the corresponding acceptance probability is simply,

$$\alpha(\boldsymbol{\theta}', \boldsymbol{\theta}) = \min\left(1, \frac{1}{A}\right),$$

for A given above.

The Metropolis Hastings algorithm is a special case of this reversible jump procedure, where the proposed move does not change the dimension of the model. (See exercises for proof of this statement).

Example

We reconsider the AR(k) process, where we assume that the k is not known *a priori*. Recall that for a given order k,

$$\boldsymbol{x}_k = X_k \boldsymbol{a}_k + \boldsymbol{\varepsilon},$$

where we now include the dependence on k for the parameters $\boldsymbol{a}_k$, for clarity. We have the prior distributions,

$$\begin{aligned} \boldsymbol{a} &\sim N(\boldsymbol{\mu}_a, \Sigma_a); \\ \sigma^2 &\sim \Gamma^{-1}(\alpha, \beta). \end{aligned}$$

Then, the posterior conditional distributions for $\boldsymbol{a}_k$ and σ^2 are of the form,

$$\begin{aligned} \sigma^2|\boldsymbol{x}, k &\sim \Gamma^{-1}\left(\alpha + \frac{(N-k)}{2}, \beta + \frac{1}{2}(\boldsymbol{x}_k - X_k\boldsymbol{a}_k)^T(\boldsymbol{x}_k - X_k\boldsymbol{a}_k)\right); \\ \boldsymbol{a}_k|\boldsymbol{x}, \sigma^2, k &\sim N(\boldsymbol{\mu}_k, \Sigma_k), \end{aligned}$$

where,

$$\boldsymbol{\mu}_k = \sigma^{-2}\Sigma_k X_k^T \boldsymbol{x}_k + \Sigma_k \Sigma_a^{-1} \boldsymbol{\mu}_a; \qquad \Sigma_k^{-1} = \sigma^{-2} X_k^T X_k + \Sigma_a^{-1}.$$

Within each step of the Markov chain we update the σ^2 and $\boldsymbol{a}_k$, using the Gibbs sampler, and propose to update the order of the process using the following reversible jump procedure.

We shall assume that we are only interested in autoregressive models of order $k = 1, ..., k_{max} < N$. Then, within the reversible jump update, we shall only propose to increase or decrease the order of the process by one, and let the probability that we move from the model of order k to k' be denoted by $p(k \to k')$. Suppose that the current model is of order k, with parameters $\boldsymbol{a}_k$, and that we propose to move to the model of order $k+1$, with parameters $\boldsymbol{a}'_{k+1}$. Then, we generate a set of random variables $\boldsymbol{u}$, and $\boldsymbol{u}'$, (of dimensions $k+1$ and k, respectively), from some distributions q and q', respectively, such that there is a bijective map between $(\boldsymbol{a}_k, \boldsymbol{u})$ and $(\boldsymbol{a}'_{k+1}, \boldsymbol{u}')$, i.e. $g(\boldsymbol{a}_k, \boldsymbol{u}) = (\boldsymbol{a}'_{k+1}, \boldsymbol{u}')$, where g is a bijective function. Then, we set,

$$\boldsymbol{a}'_{k+1} = \boldsymbol{u}; \quad \boldsymbol{u}' = \boldsymbol{a}_k,$$

where,

$$\boldsymbol{u} \sim N(\boldsymbol{\mu}_{k+1}, \Sigma_{k+1}),$$

for $\boldsymbol{\mu}_{k+1}$ and Σ_{k+1}, given above. Thus our proposal distribution for the parameters in the new model is the posterior conditional distribution of the parameters. Conversely, in the reverse move, where we propose to decrease the order of the process, from $k+1$ to k, we set $(\boldsymbol{a}_k, \boldsymbol{u}) = g^{-1}(\boldsymbol{a}'_{k+1}, \boldsymbol{u}')$, so that,

$$\boldsymbol{a}_k = \boldsymbol{u}'; \quad \boldsymbol{u} = \boldsymbol{a}'_{k+1}.$$

We once more set the proposal distribution of $\boldsymbol{u}'$ to be the conditional distribution of $\boldsymbol{a}_k$, i.e.,

$$\boldsymbol{u}' \sim N(\boldsymbol{\mu}_k, \Sigma_k).$$

This updating procedure of the parameters, given the proposed model, is similar to the independent Metropolis-Hastings algorithm, since the values of the parameters in the proposed model are independent of their current values. The corresponding Jacobian term for such a move is clearly equal to unity, since the map from $(\boldsymbol{a}_k, \boldsymbol{u})$ to $(\boldsymbol{a}'_{k+1}, \boldsymbol{u}')$ is essentially the identity.

Then, we accept the move with probability, $\min(1, A)$, where,

$$A = \frac{\pi(\boldsymbol{a}'_{k+1}, \sigma^2, k+1|\boldsymbol{x})p(k+1 \to k)q'(\boldsymbol{u}')}{\pi(\boldsymbol{a}_k, \sigma^2, k|\boldsymbol{x})p(k \to k+1)q(\boldsymbol{u})}$$

else, we remain in the current model for the iteration of the Markov chain. The corresponding probability for moving from the model of order $k+1$ to order k is simply $\min(1, A^{-1})$, where A is as above.

4.8 Exercises

Exercise 4.1: *A physical system has two possible states θ_1 and θ_2 and its progression from one state to another is described by a homogeneous Markov chain with transition matrix*

$$P = \begin{bmatrix} \alpha & 1-\alpha \\ 1-\beta & \beta \end{bmatrix},$$

where $0 < \alpha,\ \beta < 1$. If the prior probability is π_1 that the initial state is θ_1, determine the posterior probability that the initial state was θ_1, given that the system occupies state θ_1 after n steps. Explain why this posterior probability approaches π_1 as n becomes large.

Exercise 4.2: *If the stationary distribution for the Markov chain with the transition matrix given in Exercise 4.1 is of the form*

$$\pi = \begin{pmatrix} \pi_1 \\ \pi_2 \end{pmatrix},$$

calculate the values of π_1 and π_2, if $\alpha = 0.2$ and $\beta = 0.4$.

Exercise 4.3: *Suppose that we wish to use the Metropolis Hastings algorithm to generate from $\pi(x) = N(0, \sigma^2)$, and that we use the proposal $q(x, y) = N(ax, \tau^2)$. What is the corresponding acceptance function $\alpha(x, y)$? For what value of a would this particular sampler* never *reject?*

Exercise 4.4: *Show that the Metropolis Hastings algorithm generates a reversible Markov chain i.e.,*

$$\pi(x)\mathcal{K}_H(x,y) = \pi(y)\mathcal{K}_H(y,x)$$

and show that $\pi(x)$ is the stationary distribution of the algorithm i.e.,

$$\int \pi(x)\mathcal{K}_H(x,y)dx = \pi(y).$$

Exercise 4.5: *Suppose we wish to use the Gibbs Sampler to sample from the distribution*

$$\pi(x_1, x_2) = \exp\left\{-\frac{1}{2}(x_1-1)^2(x_2-2)^2\right\}.$$

Write down the two full conditionals for this distribution. How would you sample from these?

Exercise 4.6: *The Gibbs Sampler can be thought of as a special case of the Metropolis Hastings algorithm. If we break each iteration of the Metropolis Hastings algorithm into k steps and let q_j denote a proposal density for candidates in the jth co-ordinate direction. Show that if*

$$q_j(\boldsymbol{x},\boldsymbol{y}) = \begin{cases} \pi(\boldsymbol{y}_j|\boldsymbol{x}_{(j)}) & \boldsymbol{y}_{(j)} = \boldsymbol{x}_{(j)}, \quad j = 1, ..., k, \\ 0 & \textit{else}, \end{cases}$$

then the corresponding acceptance probability $\alpha(\boldsymbol{x},\boldsymbol{y})$ is equal to 1 for all $\boldsymbol{x}$ and $\boldsymbol{y}$. Hence, show that the transition density for k iterations of this single-update Metropolis Hastings algorithm is the same as that of the Gibbs Sampler.

Exercise 4.7: *Verify the conditional distributions given in (4.6) and (4.7).*

Exercise 4.8: *Show that the Metropolis Hastings algorithm is a special case of the reversible jump algorithm, where there is no dimensional change.*

Exercise 4.9: *Suppose that we observe a series of data $\boldsymbol{x} = (x_1, ..., x_N)$, where each x_i are* iid *from a $Bin(n,p)$, where p is unknown. A natural prior for p is a $Beta(\alpha,\beta)$ distribution, but suppose we are unsure as to whether α and β are fixed or should themselves have priors. If $x \sim Bin(n,p)$ and $p \sim Beta(\alpha,\beta)$, we say that $x \sim Beta - Bin(n,\alpha,\beta)$. Thus, each x_i either has a $Bin(n,p)$ distribution, with some prior for p, or the over-dispersed, $Beta - Bin(n,\alpha,\beta)$ distribution, where α and β are unknown and have some prior. Suggest a reversible jump step that can move between the different possible models, and calculate the corresponding acceptance probability.*

Note that the $Beta - Bin(n,\alpha,\beta)$ has density,

$$p(x) = \frac{n!}{x!(n-x)!}\frac{\Gamma(\alpha+x)\Gamma(n+\beta-x)}{\Gamma(\alpha+\beta+n)}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}.$$

Exercise 4.10: (a) *Suppose that within an experiment N individuals are recorded by two sources, each with two levels (1 and 2), denoting the presence or absence of a particular characteristic. Then, we can construct a 2×2 contingency table, with each cell (i,j) denoting the number of individuals with the given characteristics from sources A and B, respectively. We model the probability of being in cell (i,j) to be of log-linear form, so that,*

$$\ln \tilde{p}_{ij} = \theta^{\phi} + \theta_i^A + \theta_j^B.$$

The corresponding cell probabilities are given by,

$$p_{ij} = \frac{\tilde{p}_{ij}}{\sum_{ij}\tilde{p}_{ij}}.$$

For identifiability we set,

$$\theta_2^A = -\theta_1^A; \quad \theta_2^B = -\theta_1^B.$$

For each of the log-linear parameters, θ^ϕ, θ_1^A *and* θ_1^B *we place independent Normal priors, with common known variance* σ^2*, and some mean, which may differ between terms. Finally, we assume the data has a Multinomial(*$N, \boldsymbol{p}$*) distribution.*

Calculate the posterior conditional distribution for the log-linear parameters. How could we sample from the posterior distribution, in order to estimate summary statistics?

(b) Now, suppose that there is some model uncertainty, and we wish to be able to consider a dependence between sources A *and* B*. We can do this by considering an additional model, which contains a further log-linear parameter, accounting for the interaction between the sources, so that, under this model, the cell probabilities can be expressed (upto proportionality) in the form,*

$$\ln p_{ij} \propto \theta + \theta_i^A + \theta_j^B + \theta_{ij}^{AB},$$

where once again we have each of the log-linear parameters have independent Normal priors, with common variance and some known mean, μ*. Note that once again, for identifiability, we set,*

$$\sum_{i=1}^{2} \sum_{j=1}^{2} \theta_{ij}^{AB} = 1.$$

Describe how we can construct a Markov chain with stationary distribution equal to the joint posterior distribution over both parameter and model space.

Exercise 4.11: *Suppose that we observe data* $\boldsymbol{x} = (x_1, ..., x_N)$*, such that each* $x_i \overset{iid}{\sim} Exp(\lambda)$*, where* λ *is unknown* a priori*, and has prior,*

$$\lambda \sim \Gamma(a, b).$$

What is the posterior conditional distribution of λ*? Alternatively, suppose that* $x_i \overset{iid}{\sim} Pareto(\alpha, \beta)$*, where,*

$$\alpha \sim Exp(m); \quad \beta \sim Exp(n).$$

The Pareto(α, β*) distribution has density*

$$p(x) = \beta \alpha^\beta x^{-(\beta+1)},$$

where $0 < \alpha \leq x$ *and* $\beta > 0$*, and has mean* $\beta\alpha/(\beta - 1)$*. How can we sample from the posterior distribution of the parameters, given the data?*

Now suppose that there is uncertainty regarding the model, where the prior probabilities of each the models are equal. How can we incorporate this into the analysis, and obtain the posterior probability of each of the models?

Chapter 5

CLASSICAL INFERENCE

5.1 Introduction

The classical approach to statistical inference is based upon the likelihood function $L(\theta|\boldsymbol{x})$ relating the model parameters $\boldsymbol{\theta}$ to the observed data, $\boldsymbol{x}$. Given a likelihood function, we obtain parameter estimates by maximising the likelihood function with respect to the model parameters to obtain the MLE. Corresponding standard errors are then obtained using the inverse Hessian matrix.

In simple cases, this maximisation can be performed by hand (typically by differentiating and setting to zero) but, in many practical cases, this is impossible. In general, numerical methods are used to maximise the likelihood function and these range from very simple derivative-based search techniques through stochastic methods such as the simplex algorithm to even more sophisticated (so-called genetic) algorithms which try to imitate Darwinian-like dynamics.

With any method, there is always a trade-off between performance and computational expense. Derivative-based methods tend to be very cheap, particularly if the derivative can be calculated analytically, rather than numerically. However, such methods tend to get stuck in local maxima where all local moves reduce the value of the objective function (the function to be maximised). More complex algorithms tend to be more reliable (in theory) but are based upon asymptotic results which don't strictly hold in practical applications, since we don't have an infinite amount of time to wait for our results in general!

In this chapter we introduce two of the most popular optimisation schemes. The first is simulated annealing which has very strong links with the Markov chain Monte Carlo schemes of the previous chapter. Though this is an example of an "asymptotic" scheme it tends to provide reliable results within a reasonable time and some have suggested could be combined with traditional derivative based methods for additional accuracy. Our second scheme is the EM algorithm, which also has some links with the Gibbs sampling ideas of the previous chapter. The EM algorithm is most usefully applied in the presence of missing values which can arise either in the case of data sets in which some data points have been omitted, or when the likelihood function is only available in closed form in the presence of (hidden) auxiliary variables. We begin by introducing the simulated annealing algorithm.

5.2 Simulated Annealing

5.2.1 Introduction to Simulated Annealing

At the heart of simulated annealing is an analogy with Thermodynamics, specifically with the way that both metals, and some liquids, cool and crystallise. If the temperature is lowered slowly then this cooling process is called annealing. At high temperatures molecules are free to move, but as the temperature decreases thermal mobility is lost, and the molecules tend to line themselves up in a rigid structure. This rigid state is, in fact, a state of minimum energy, and so long as the temperature is decreased slowly, nature is almost certain to find it. If the temperature is decreased rather more rapidly, this state may not be achieved, and a polycrystalline or amorphous state with higher energy may be found instead.

Most function minimisation routines would be analogous to this rapid cooling method. In the case where multiple optima are present it is easy for optimisation algorithms to find a local optimum, corresponding to the amorphous state in Nature. Simulated annealing, on the other hand, is analogous to decreasing the temperature slowly, allowing ample time for the redistribution of the molecules. Hence, we might consider simulated annealing to be Nature's own minimisation algorithm. For function optimisation, we would equate function values with energy levels and parameter settings with the positions of molecules.

The annealing process can be described as follows. We consider a succession of decreasing temperatures, starting from some maximum, T_0. At each temperature T, the system is allowed to reach thermal equilibrium in which the probability of the system being in some state $\boldsymbol{\theta}$ with "energy" $f(\boldsymbol{\theta})$ is given by the Boltzmann distribution

$$b_T(\boldsymbol{\theta}) = \frac{1}{Z(T)} \exp\left(-f(\boldsymbol{\theta})/k_b T\right). \tag{5.1}$$

where $Z(T)$ is a normalisation function, and k_b is known as the Boltzmann constant (which we shall ignore and assume is incorporated into the scale of either f or T). This is the prescription of the Boltzmann Theorem of Statistical Physics and can be found in any standard Statistical Physics text. As T decreases, the range of the Boltzmann distribution concentrates upon states with the lowest energy, as high energy states become increasingly unlikely. Eventually, T becomes so low that the system "freezes", and if the temperature is lowered slowly enough this frozen state will have minimum energy.

As an illustration, suppose that we are given a discrete distribution with the following probabilities, $\{0.4, 0.3, 0.3\}$. Suppose that we wish to find the maximal value of this distribution, that is, the value whose corresponding probability is 0.4. Now if we take this distribution and square it (i.e., take $T = 1/2$), and then re-normalise we get the following distribution, $\{0.47, 0.265, 0.265\}$. If we raise the original distribution to the power 10 and re-normalise we get the following distribution $\{0.9, 0.05, 0.05\}$. As the distribution is raised to successive powers, the distribution becomes more and more spiked around it's maximal value. Essentially what we want to do is sample from this distribution in the limit as T approaches zero. Obviously this is a point mass at an unknown point and so this is difficult to do directly so, instead we simulate a series of Markov chains as T decreases, using the end point of the last simulation as the starting point for the next.

5.2.2 The Metropolis and Annealing Algorithms

An algorithm by Metropolis (and others) for the efficient simulation of atoms in equilibrium at a given, fixed, temperature was based upon this process: given a current state of the system with energy $f_0 \equiv f(\boldsymbol{\theta}_0)$, we choose a new state by displacing a randomly chosen particle. If this new state has energy, f, and $f < f_0$, then the system remains in this new state. However, if $f \geq f_0$ then the probability of remaining in this new state, is given by;

$$\exp\left(-\frac{f - f_0}{T}\right).$$

This acceptance rule is known as the *Metropolis criterion.* When this process is repeated, the system eventually reaches a state of equilibrium for that temperature, in which the probability distribution of the accepted points is given by the Boltzmann distribution of Equation (5.1). This process is known as the *Metropolis algorithm* and was the precursor of the Metropolis Hastings algorithm discussed in Chapter 4.

The Metropolis algorithm is concerned with only a single fixed temperature. It was generalised by Kirkpatrick (and others) in 1983, who introduced an *annealing schedule* which defines how the temperature is reduced. Beginning with a high initial temperature T_0, the Metropolis algorithm is followed until equilibrium is achieved. The temperature is then decreased, according to the annealing schedule, and the Metropolis algorithm continues at this new temperature until a new equilibrium is achieved and the temperature is decreased once more. This process is repeated until the system "freezes". If the annealing schedule is sufficiently slow, then the system will freeze in a state of minimum energy, corresponding to the global minimum of our objective function.

Algorithmically, the original simulated annealing scheme proceeds as follows.

STEP 1. CHOOSE SOME INITIAL TEMPERATURE T_0 AND SOME INITIAL STARTING CONFIGURATION $\boldsymbol{\theta}_0$.

STEP 2. PROPOSE A NEW CONFIGURATION, $\boldsymbol{\theta}' = \boldsymbol{\theta} + \boldsymbol{\psi}$ FOR $\boldsymbol{\psi}$ DRAWN FROM SOME SYMMETRIC PROPOSAL DENSITY q.

STEP 3. ACCEPT THE MOVE TO $\boldsymbol{\theta}'$ WITH PROBABILITY

$$\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}') = \min\left(1, \frac{b_T(\boldsymbol{\theta}')}{b_T(\boldsymbol{\theta})}\right).$$

STEP 4. REPEAT STEPS 2 AND 3 FOR A SPECIFIED NUMBER OF ITERATIONS, UNTIL THE CHAIN IS DEEMED TO HAVE REACHED AN EQUILIBRIUM STATE.

STEP 5. LOWER THE TEMPERATURE T ACCORDING TO SOME PREDETERMINED SCHEDULE AND REPEAT STEPS $2-4$, UNTIL SOME STOPPING CRITERION IS MET. THE FINAL CONFIGURATION $\boldsymbol{\theta}$ WILL APPROXIMATE THE MINIMUM OF $f(\boldsymbol{\theta})$.

Example

Consider the Cauchy density function

$$f(x) = \frac{\beta}{\pi[\beta^2 + (x-\alpha)^2]} \qquad -\infty < x, \alpha < \infty \text{ and } \beta \geq 0.$$

The log-likelihood surface for this density has been shown to give various iterative methods of maximum likelihood estimation considerable difficulty when estimating α for some fixed values of β. If we take a random sample $x_1, ..., x_n$, then the log-likelihood can be written as

$$\log L(\alpha, \beta | \boldsymbol{x}) = n \log \beta - \sum_{i=1}^{n} \log[\beta^2 + [x_i - \alpha]^2) - n \log \pi.$$

With β fixed, the maximum likelihood estimate for α is that which minimises

$$f(\alpha) = \sum_{i=1}^{n} \log[\beta^2 + (x_i - \alpha)^2]$$

and, if we fix $\beta = 0.1$, then for the random sample:

$$(-4.20, -2.85, -2.30, -1.02, 0.70, 0.98, 2.72, 3.50)$$

we get the log-likelihood given in Figure 5.1

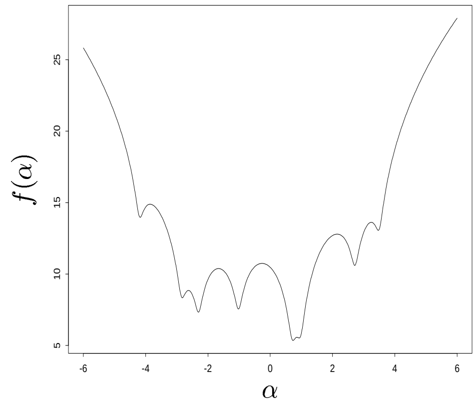


Figure. 5.1: Plot of the objective function to be minimised by the SA algorithm. The true minimum is at $\alpha = 0.7327$

We begin by defining an S-Plus function to calculate the objective function f to be minimised:

```
alphafun <- function(a, b = 0.1, x = c(-4.2, -2.85, -2.3, -1.02, 0.7, 0.98, 2.72, 3.5))
{
        s <- 0
        for (i in 1:length(x)) {
                s <- s + log(b^2 + (x[i] - a)^2)
        }
        s
}
```

We can implement a simulated annealing algorithm in many ways, but let us suppose that we perturb the current state to obtain a new one by generating $u \sim U(-\epsilon, \epsilon)$ and then setting $\alpha' = \alpha + u$. We also need a cooling schedule to reduce the temperature at each stage and a stopping rule to make sure that the simulation actually ends. The simplest cooling schedule is to select some value ρ and to reduce the temperature by this factor at each stage. Thus, the temperature begins at T_0, is then reduced to ρT_0 and then $\rho^2 T_0$ and so on. Finally, we need a stopping rule. The simplest stopping rule is to simply wait until no moves are accepted within a particular temperature sweep and this is the rule adopted in the following S-Plus implementation of the annealing algorithm.

```
simann <- function(T0 = 1, N = 500, rho = 0.90, x0 = 0, eps = 0.1, f)
{
        moving <- 1
```

```
        count <- 0
        Temp <- T0
        x <- x0
        while(moving > 0) {
                moving <- 0
                for(i in 1:N) {
                        y <- x + runif(1, -eps, eps)
                        alpha <- min(1, exp((f(x) - f(y))/Temp))
                        if(runif(1) < alpha) {
                                moving <- moving + 1
                                x <- y
                        }
                }
                Temp <- Temp * rho
                count <- count + 1
        }
        fmin <- f(x)
        return(x, fmin, count)
}
```

Here, `T0` is the initial starting temperature, `N` is the number of iterations at each temperature, `rho` is the factor by which the temperature is decreased after each `N` iterations, `x0` is the starting value for the chain and `eps` is the value of ϵ for the proposal.

Typing `simann(f=alphafun)` we get the following output:

```
$x:
[1] 0.8739446

$fmin:
[1] 5.567428

$count:
[1] 40
```

where `$x` denotes the final state of the chain, `$fmin` denotes the function value at that point and `$count` records the number of temperature changes before no more moves were accepted. Of course, since we are not actually running the simulation for an infinite amount of time, we obtain only an approximate solution. The answer improves as we take larger N, ρ-values closer to 1 and/or smaller ϵ values (within reason).

The plots in Figure 5.2 illustrate how the annealing algorithm behaves at different temperatures. We can see from Figure 5.2a that at high temperatures, the entire space is well explored. However, as we reduce the temperature, accepted points become increasingly focused around (local) minima and eventually only around the true minimum.

5.2.3 General Proposals for Simulated Annealing

Steps 2 and 3 of the annealing algorithm are designed to simulate a Markov chain with invariant distribution b_T. As explained above, simulated annealing algorithms typically use Metropolis jumping

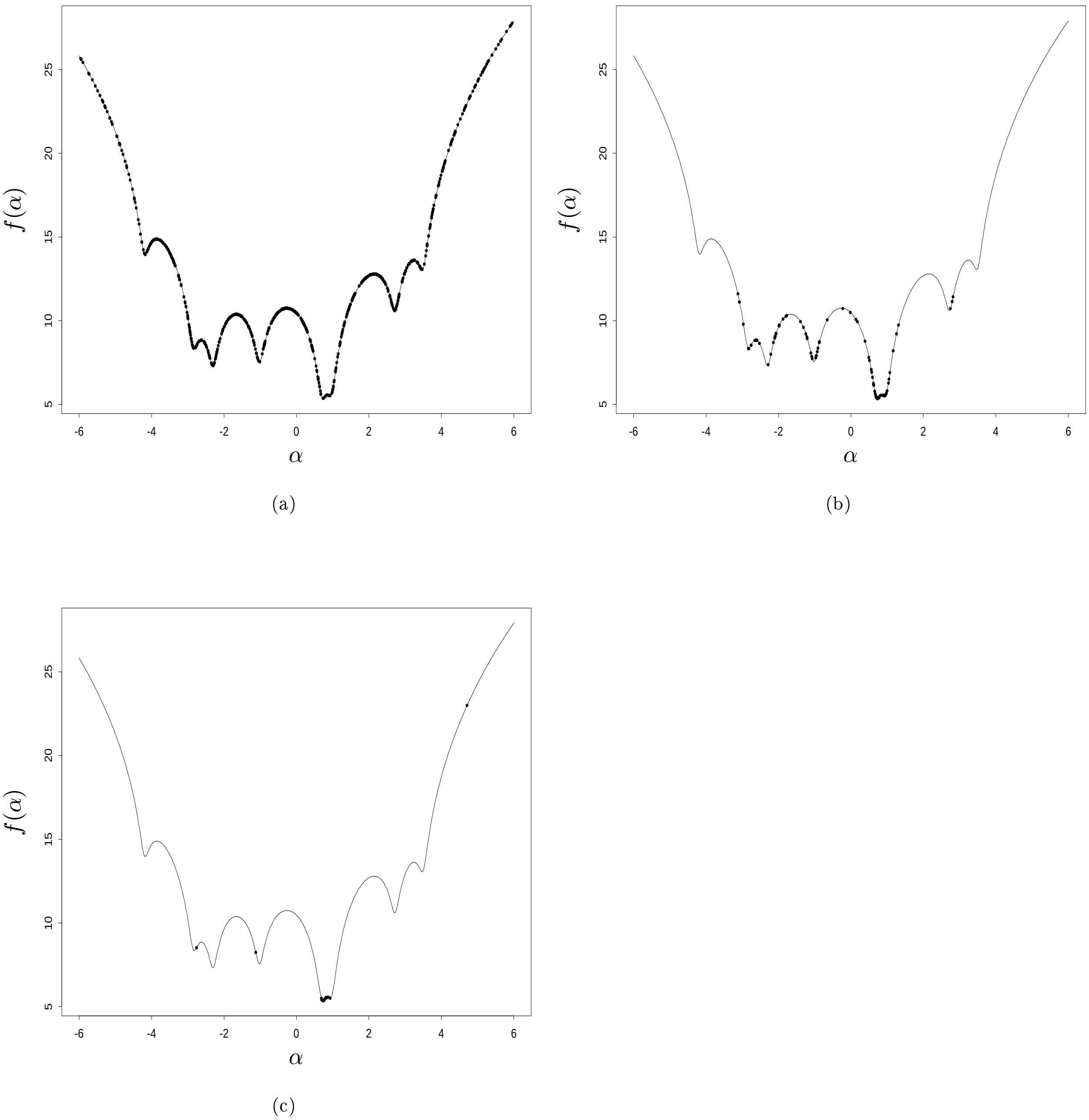


Figure. 5.2: Points accepted in 1000 iterations of the SA algorithm with (a) $T_0 = 10$, (b) $T_0 = 1$ and (c) $T_0 = 0.1$.

rules which essentially perturb the current state and automatically accept any move which reduces the objective function (thereby increasing b). Otherwise the move is accepted with probability $\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}')$. However, as we discussed in Chapter 4 this updating scheme was generalised to obtain the Metropolis Hastings update which may be used instead.

Given a temperature T with corresponding stationary density b_T and current state $\boldsymbol{\theta}$, we generate a candidate value for the next state $\boldsymbol{\theta}'$ from an essentially arbitrary proposal distribution, $q(\boldsymbol{\theta}, \boldsymbol{\theta}')$. If this move is accepted with probability

$$\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}') = \min\left(1, \frac{b_T(\boldsymbol{\theta}')q(\boldsymbol{\theta}', \boldsymbol{\theta})}{b_T(\boldsymbol{\theta})q(\boldsymbol{\theta}, \boldsymbol{\theta}')}\right), \tag{5.2}$$

then the resulting Markov chain has invariant distribution b_T (assuming that if the proposal is rejected, the chain simply remains in the current state). Thus, there are as many implementations of the annealing algorithm as there are choices of q. Here are a few popular examples.

The Single Update Metropolis Hastings Algorithm

The single update Metropolis Hastings algorithm, splits the state vector into blocks of parameters $(\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_k)$, $k \leq p$ and then a single iteration of the algorithm comprises a series of updates in which individual blocks are updated separately using the Metropolis Hastings transition rule.

Gibbs Updates

As we saw in Chapter 4, the Gibbs sampler is in fact a special case of the single update Metropolis Hastings algorithm in which the proposal used for updating parameter $\boldsymbol{\theta}_i$ is simply the conditional distribution of that parameter given the others. Suppose we split the state vector into $k \leq p$ components, and that we are currently in state $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_k) \in \mathbb{R}^p$. Then, having selected component $\boldsymbol{\theta}_i$ to be updated, the Gibbs transition kernel involves sampling a new state $\boldsymbol{\theta}' = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_{i-1}, \boldsymbol{\phi}, \boldsymbol{\theta}_{i+1}, ...\boldsymbol{\theta}_k)$, by sampling $\boldsymbol{\phi}$ from $q(\boldsymbol{\theta}, \boldsymbol{\theta}') = b_T(\boldsymbol{\theta}_i|\boldsymbol{\theta}_{(i)})$, where $\boldsymbol{\theta}_{(i)}$ denotes the state vector with element i removed. With this proposal it is easy to show that the corresponding acceptance ratio is always 1, so we dispense with the accept-reject step altogether (see Section 4.5.3).

Diffusion-Based Updates

Continuous-time Markov processes were motivated by the need to produce "intelligent" and problem-specific proposal distributions for the Metropolis Hastings algorithm. Many methods come from the Physics literature, where the idea of energy gradients are used to choose the most suitable directions for updating. The essential idea of continuous-time Markov chains is that jumps from state to state occur at random times which are distributed exponentially. There has been considerable interest in the use of chains which approximate continuous time Markov chains. One approach to the problem of designing problem-specific proposal distributions for Metropolis Hastings, is to use what are known as Langevin algorithms. Such algorithms (which are derived from approximated diffusion processes), use information about the target distribution, b_T, in the form of the gradient of $\log b_T$, to generate a candidate observation for the next state of the chain.

In order to describe these algorithms, let us first describe the underlying diffusion process. Consider a p-dimensional continuous-time Markov chain, whose state at time t is denoted by $\boldsymbol{\theta}^t$, and which has stationary density b_T. We describe the diffusion process in terms of a stochastic differential equation, of the form

$$d\boldsymbol{\theta}^t = d\boldsymbol{B}^t + \frac{1}{2}\nabla \log b_T(\boldsymbol{\theta})dt, \tag{5.3}$$

where ∇ denotes the vector of partial derivatives with ith element, $\partial/\partial\theta_i$ and $d\boldsymbol{B}^t$ is an increment of p-dimensional Brownian motion. Essentially, (5.3) says that, given $\boldsymbol{\theta}^t$, $\boldsymbol{\theta}^{t+dt}$ follows a multivariate normal distribution with mean $\boldsymbol{\theta}_t + \frac{1}{2}\nabla \log b_T(\boldsymbol{\theta})dt$ and variance matrix $\boldsymbol{I}_p dt$, where $\boldsymbol{I}_p$ denotes the p-dimensional identity matrix.

It can be shown that this diffusion process will, under certain regularity conditions, have unique stationary density b_T. The process $\boldsymbol{\theta}$ is called the *Langevin* diffusion for b_T and, in the context of MCMC, is referred to as the unadjusted Langevin algorithm, or ULA. Clearly, direct simulation of this process, in terms of infinitesimal time-intervals, is impossible. Thus, we *discretise* the process, and take time-intervals of length $\delta > 0$. However, this discretisation disturbs the convergence properties of the process, so it is necessary to "correct" the algorithm by means of a Metropolis Hastings rejection step, leading to the Metropolis-adjusted Langevin algorithm, or MALA.

Trans-Dimensional Jumps

In the same way that MCMC can be extended to consider model choice problems, so can simulated annealing. Suppose that we have a collection of models $M_1, M_2, ...$ indexed by the parameter $k \in \mathcal{K}$ and that under each model we have an associated parameter vector $\boldsymbol{\theta}_k$ of length n_k with corresponding likelihood function $L_k(\boldsymbol{\theta}_k|\boldsymbol{x})$. Suppose also that we wish to find the minimum of some general function $f(\boldsymbol{\theta}_k, k)$ over values of $(\boldsymbol{\theta}_k, k)$. For example, if we take $f(\boldsymbol{\theta}_k, k) = -2\log L_k(\boldsymbol{\theta}_k|\boldsymbol{x}) + 2n_k$ then we have the usual Akaike's Information Criterion (AIC). The Boltzmann distribution associated with this objective function can then be defined in the usual way, and we set

$$b_T(\boldsymbol{\theta}_k, k) \propto \exp(-f(\boldsymbol{\theta}_k, k)/T).$$

However, b_T is no longer a density but a distribution over $\mathcal{K} \times \boldsymbol{\Theta}$. We therefore require the reversible jump MCMC updates as described in the previous chapter.

5.2.4 Example: Autoregressive Time Series

Suppose we are given data $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ from an autoregressive (AR) process of order k so that the data is generated by the process:

$$X_t = \sum_{\tau=1}^{k} a_\tau X_{t-\tau} + \varepsilon_t, \;\; t = k+1, \ldots, N, \tag{5.4}$$

where $\varepsilon_t \sim N(0, \sigma^2)$. Then, the likelihood associated with these data is approximated by

$$L(\boldsymbol{\theta}|\boldsymbol{x}) = \prod_{t=k+1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left[\frac{x_t - \sum_{\tau=1}^{k} a_\tau x_{t-\tau}}{\sigma}\right]^2\right), \tag{5.5}$$

where $\boldsymbol{\theta} = (a_1, ..., a_k, \sigma^2)$ denotes the vector of model parameters. In order to fit this model, we need to locate the value of $\boldsymbol{\theta}$ which maximises the likelihood function, $L(\boldsymbol{\theta}|\boldsymbol{x})$. Thus we might simply take $f(\boldsymbol{\theta}) = -L(\boldsymbol{\theta}|\boldsymbol{x})$ as our objective function but, as we shall see shortly, the MCMC computation in steps 2–4 of the annealing algorithm is simplified greatly by taking $f(\boldsymbol{\theta}) = -\log L(\boldsymbol{\theta}|\boldsymbol{x})$. Given the latter objective function, we obtain a target distribution at any temperature T given by

$$b_T(\boldsymbol{\theta}) \propto \exp(\log L(\boldsymbol{\theta}|\boldsymbol{x})/T) = L(\boldsymbol{\theta}|\boldsymbol{x})^{1/T}.$$

The simulated annealing algorithm defined above requires the construction of a transition mechanism to create a Markov chain with stationary density $b_T(\boldsymbol{\theta})$. The algorithm would then proceed by taking some arbitrary starting values for $\boldsymbol{\theta}$ and some initial temperature T_0 and simulating a chain with stationary density $b_{T_0}(\boldsymbol{\theta})$. The temperature would then be reduced to $T_1 < T_0$, say, and a new chain started with stationary density $b_{T_1}(\boldsymbol{\theta})$. The starting point for the new chain would be the final state of the previous chain . This process would be repeated until the system freezes.

Thus, there remain several implementation problems: first, how do we choose the sequence of temperatures; and second, how do we simulate the chains within each temperature? We shall address the first of these in the context of a particular data set, since the temperature sequence is often data-dependent. The second problem is also somewhat data-dependent, but we shall discuss several general approaches which we shall compare in the context of the analysis of a particular data set below.

Here we have target density

$$b_T(\boldsymbol{\theta}) \propto \left\{ \sigma^{-(N-k)} \exp\left(-\frac{1}{2} \sum_{t=k+1}^{N} \left[\frac{x_t - \sum_{\tau=1}^{k} a_\tau x_{t-\tau}}{\sigma} \right]^2 \right) \right\}^{1/T}.$$

Therefore, the conditional distribution of the error variance given the autoregressive parameters is given by

$$b_T(\sigma^2 | a_1, ..., a_k, \mathbf{x}) \sim IG\left(1 + \frac{N-k}{2T}, \frac{\sum_{t=k+1}^{N} (x_t - \sum_{\tau=1}^{k} a_\tau x_{t-\tau})^2}{2T} \right),$$

where IG denotes the inverse Gamma distribution. Thus, we can use a Gibbs update for the error variance. We consider four alternatives for the autoregressive parameters: single update Metropolis Hastings, single-parameter Gibbs, a multivariate Gibbs update in which all autoregressive parameters are updated simultaneously and the Langevin algorithm.

Metropolis-Hastings Updates for the AR Parameters

Suppose that we are currently in state $\boldsymbol{\theta}$ at temperature T and that we wish to update parameter θ_i corresponding to the ith autoregressive parameter, $i = 1, ..., p$. An obvious proposal is the random walk proposal under which proposed new values have a Normal distribution with mean θ_i and some variance σ_q^2. Suppose we generate a proposed value ϕ, and let $\boldsymbol{\theta}' = (a_1, ..., a_{i-1}, \phi, a_{i+1}, ..., a_p, \sigma^2)$, then we accept this proposed new value with probability $\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}') = \min(1, A)$, where

$$A = \exp\left\{ -\frac{1}{2T\sigma^2} \sum_{t=k+1}^{N} \left[\left(2\left[x_t - \sum_{\tau \neq i} \theta_\tau x_{t-\tau} \right] - (\theta_i + \theta_i') x_{t-i} \right) (\theta_i - \theta_i') x_{t-i} \right] \right\}.$$

Pilot-tuning suggests that a proposal variance of $\sigma_q^2 = 1.0$ appears to work well in practice.

Gibbs Updates for the AR Parameters

At temperature T, for the full conditional distribution $b_T(a_i | \boldsymbol{\theta}_{(i)}, \sigma, \mathbf{x})$ for parameter a_i given the value of the others is a

$$N\left(\frac{\sum_{t=k+1}^{N} x_{t-i}(x_t - \sum_{\tau \neq i} a_\tau x_{t-\tau})}{\sum_{t=k+1}^{N} x_{t-i}^2}, \frac{\sigma^2 T}{\sum_{t=k+1}^{N} x_{t-i}^2} \right)$$

distibution. Here $\boldsymbol{\theta}_{(i)} = \boldsymbol{\theta} \setminus \{a_i\}$ i.e., the vector $\boldsymbol{\theta}$ excluding the coefficient a_i.

Multivariate Gibbs Updates for the AR Parameters

In order to describe the multivariate GS update in which all autoregressive parameters are updated simultaneously, we introduce some additional notation defining two new data-dependent quantities. First, we let $\boldsymbol{x}_k$ denote the vector with elements $(x_{k+1}, ..., x_N)$. Secondly, we let $\boldsymbol{X}_k$ denote the $(N-k) \times k$ matrix with (i,j)th element x_{k+i-j}. Then clearly Equation (5.4) can be rewritten as,

$$\boldsymbol{x}_k = \boldsymbol{X}_k \boldsymbol{a} + \boldsymbol{e},$$

with obvious notation. In this case, the full conditional distribution for the autoregressive parameters $\boldsymbol{a}$ given the error variance σ^2 is given by

$$b_T(\boldsymbol{a}|\sigma^2) = N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where

$$\boldsymbol{\mu} = \frac{\Sigma^T \boldsymbol{X}_k^T \boldsymbol{x}_k}{\sigma^2 T}, \text{ and } \Sigma^{-1} = \frac{\boldsymbol{X}_k^T \boldsymbol{X}_k}{\sigma^2 T}. \tag{5.6}$$

Exercise: Check.

Langevin Proposals for the AR Parameters

The Langevin algorithm described in Section 5.2.3 can be applied to produce either univariate or multivariate updates. However, since the multivariate proposal variance is the identity, the most efficient implementation in terms of Markov chain mixing (though not necessarily in terms of computing time) is to update the AR parameters individually. In this case, if we wish to update parameter a_i, the proposal is a Normal with unit variance and mean given by

$$a_i - \frac{1}{T} \sum_{t=k+1}^{N} \left(x_t - \sum_{\tau=1}^{k} a_\tau x_{t-\tau} \right) x_{t-i}.$$

The corresponding acceptance probability is then simply derived by substitution in Equation (5.2).

Note here, the form of the proposals for the Gibbs updates for the error variance and for both the single and block updates for the autoregressive parameters. It is clear that as $T \to 0$, the variance of the conditionals decrease to zero so that the conditional distributions simply converge to point-masses corresponding to the usual MLE equations. Note also that the Gibbs and Langevin updates vary with temperature. Of course the acceptance step in the Metropolis Hastings update also varies with temperature, but the fact that the proposal for the Gibbs and Langevin updates essentially adapts to the current state and temperature provides more versatile algorithms better capable of exploring more complex distributions, particular as the system begins to freeze.

Results

To illustrate the performance of our algorithm we have taken the well-known lynx data set which records the number of lynxes trapped in the Mackenzie river in Canada from 1821 to 1934 and it has been proposed that this data can be modelled by an AR process. In the Metropolis Hastings algorithm, proposed new parameter values were chosen from a $N(0, 0.1)$ distribution. In each case the temperature schedule began at a temperature of 10 and was reduced by a factor of 0.99 after every 1000 iterations. This was repeated approximately 600 times until a temperature of 0.02 was reached.

All four algorithms provide almost identical final solutions for any model order. For comparison, we take the AR(11) model and provide trace plots of the a_1 and σ^2 parameters in Figures 5.3 and 5.4 respectively.

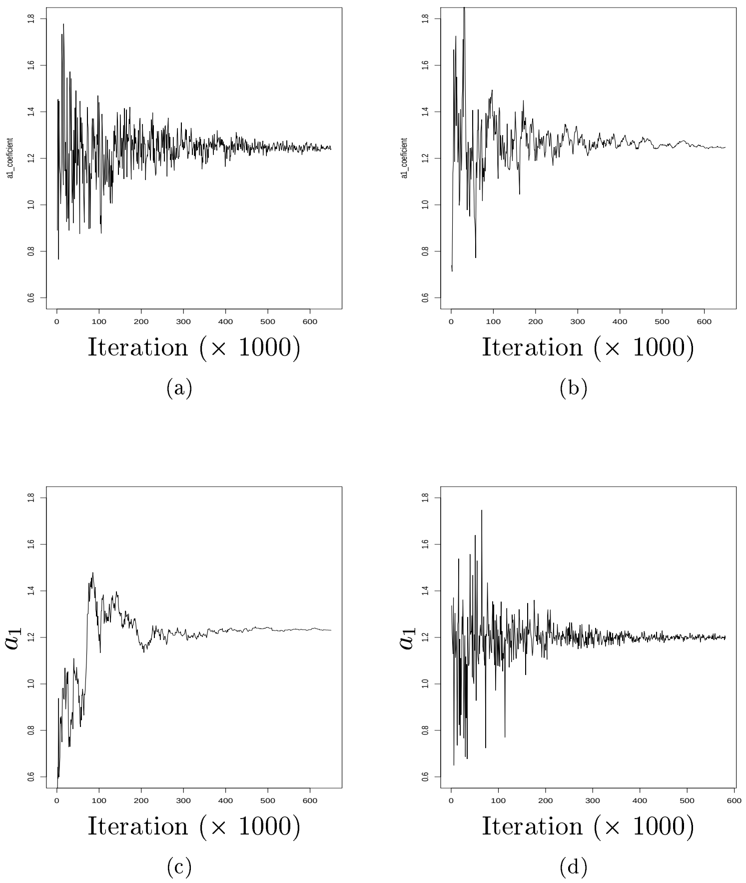


Figure. 5.3: Plot of the trace of the a_1 coefficient under the AR(11) model for Lynx data set. (a) univariate Gibbs updating scheme; (b) MH updates; (c) Langevin proposals; and (d) multivariate Gibbs updates.

We can see from these plots that the algorithms provide identical answers, but differ in their convergence behaviour. In particular, the trace plot of a_1 values with the Langevin algorithm exhibits far less "randomness" than the other three plots. It also appears to settle down fastest and this is due to the use of the derivative to direct proposals towards the mode. This pattern is not apparent in the trace plots of the variance and all four traces look very similar. At this stage, there appears to be no real advantage of any one algorithm over the others though, of course, the Metropolis Hastings algorithm does require the specification of proposal parameters, and the multivariate Gibbs updating scheme requires the inversion of a $k \times k$ matrix at each iteration. Thus, operationally, the univariate Gibbs and the Langevin algorithm appear to have a practical advantage.

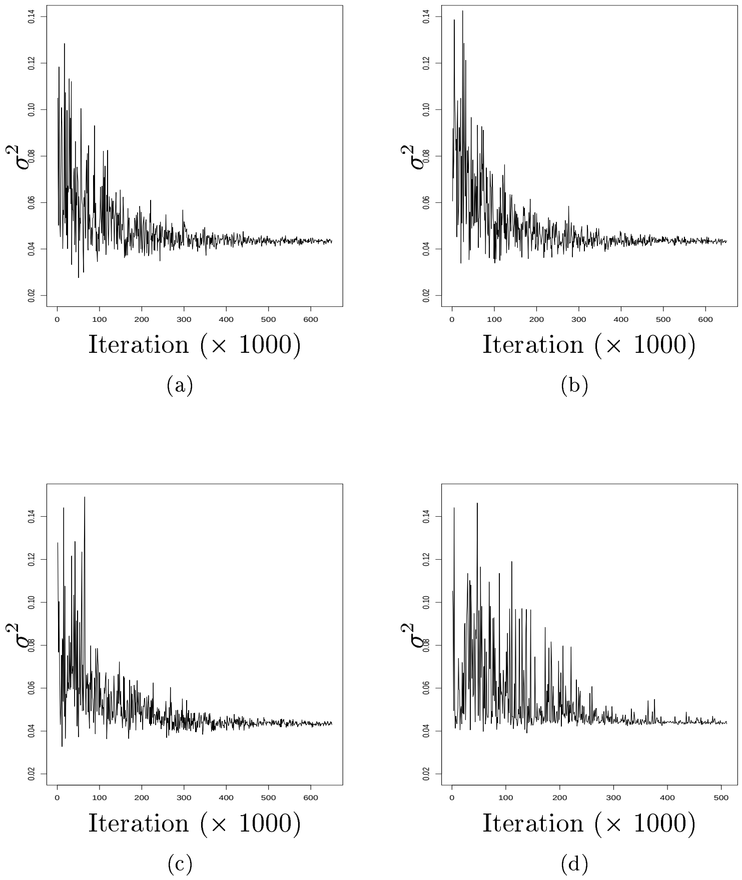


Figure. 5.4: Plot of the trace of the variance under the AR(11) model for Lynx data set (a) univariate Gibbs updating scheme; (b) MH updates; (c) Langevin proposals; and (d) multivariate Gibbs updates.

Model Choice

Now let us consider the problem of model choice. Given data $x_1, ..., x_N$, we shall assume that we are interested in autoregressive models of order $k = 1, ..., k_{max} < N$. We suppose also that in order to retain a fair comparison between models, we use the following approximation to the likelihood in (5.5):

$$L_k(\boldsymbol{\theta}_k) = \prod_{t=k_{max}+1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left[\frac{x_t - \sum_{\tau=1}^{k} a_\tau x_{t-\tau}}{\sigma}\right]^2\right),$$

where $\boldsymbol{\theta}_k$ denotes the $k+1$ dimensional vector of model parameters $(a_1, ..., a_k, \sigma^2)$. Thus, if we use the AIC to distinguish between competing models, then

$$b_T(\boldsymbol{\theta}_k, k) \propto \exp(\log L_k(\boldsymbol{\theta}_k|\boldsymbol{x}) - n_k) = L_k(\boldsymbol{\theta}_k|\boldsymbol{x})^{1/T} \exp(-n_k). \tag{5.7}$$

Suppose that we are currently in state $\boldsymbol{\theta}$ in model M_k and that we propose a move to model M_{k+1}. Then (following the reversible jump procedure of Section 4.7), we might do so by generating a single variable u from a Normal distribution with mean m and variance v. Our proposed new point would then be $(\boldsymbol{\theta}_k, u)$ under model M_{k+1} i.e, our function g is simply the identity. We shall take as our proposal the conditional distribution of the new parameters (given the current ones which remain fixed) under the larger model. Thus, the proposal density for u has mean

$$m = \frac{\sum_{t=k_{max}+1}^{N} (x_t - \sum_{\tau=1}^{k} a_\tau x_{t-\tau}) x_{t-k-1}}{\sum_{t=k_{max}+1}^{N} x_{t-k-1}^2}$$

and variance

$$v = \frac{T\sigma^2}{\sum_{t=k_{max}+1}^{N} x_{t-k-1}^2}.$$

This move has corresponding acceptance probability $\min(1, A)$, where

$$A = \frac{b_T([\boldsymbol{\theta}, u], k+1) p(k+1 \to k)}{b_T(\boldsymbol{\theta}, k) p(k \to k+1) q(u)}.$$

Results

To implement the reversible jump simulated annealing algorithm we take an initial temperature of $T_0 = 10$, which was reduced every 1000 iterations by a factor 0.99.

Figure 5.5 provides the trace plots of model order for the RJSA algorithm. The chain clearly converges to an AR(11) process, and this is the model with the smallest AIC, as can be verified using the built in autoregressive model fitting functions in S-Plus.

5.3 The EM Algorithm

5.3.1 Introduction

The EM algorithm is a technique highly related to the data augmentation algorithm in MCMC. However the EM algorithm (in its basic form at least) is a deterministic algorithm, and whereas the data augmentation algorithm simulated alternatively from the missing data and the parameters according to the appropriate conditional distributions, the EM algorithm replaces the simulation steps by an expectation and a maximisation step. The algorithm is applied to the likelihood function and

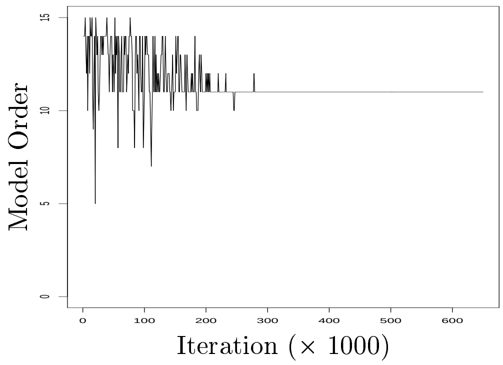


Figure. 5.5: Trace plots of model order for the Lynx data set.

converges to a local maximum of the likelihood of the observed data. Therefore where the likelihood is unimodal it converges to the MLE. Thus it is usually used as an algorithm to estimate the MLE of a statistical problem with missing data. However, it can also be applied in a Bayesian context. For instance by assuming a 'flat' prior on the unknown parameters, the EM algorithm can be used to try and approximate the mode of the posterior distribution.

Typically though, the EM algorithm is used to maximise a likelihood when, with respect to the likelihood specification, some of the data give incomplete information. This might be the case, for example, in a regression model where some of the data is censored. If the data were all complete, maximum likelihood would be a straightforward application of least squares, but the censoring complicates this. The EM algorithm solves this problem iteratively by first using a current estimate of the regression model to estimate the values of the censored data; then re–estimating the regression line on the basis of the original and augmented data. This procedure is then iterated until convergence. The two steps are respectively called **the E–step** (estimation) and **the M–step** (maximisation).

5.3.2 The Algorithm

We suppose that our aim is to find the maximum likelihood estimator $L(\boldsymbol{\theta}|Y)$, where Y represents our observed data. Now, we suppose that it is possible to augment the observed data Y with additional data Z, such that the likelihood $L(\boldsymbol{\theta}|Y,Z)$ is easier to maximise. In essence, we then estimate Z from a current estimate of $L(\boldsymbol{\theta}|Y)$ and then maximise $L(\boldsymbol{\theta}|Y,Z)$ with respect to $\boldsymbol{\theta}$. Then iterate to convergence. More formally, given a current estimate $\boldsymbol{\theta}^i$ of $\boldsymbol{\theta}$, we define the function

$$Q(\boldsymbol{\theta},\boldsymbol{\theta}^i) = \int_Z \log(L(\boldsymbol{\theta}|Y,Z))P(Z|\boldsymbol{\theta}^i,Y)dZ, \tag{5.8}$$

that is

$$Q(\boldsymbol{\theta},\boldsymbol{\theta}^i) = \mathbb{E}_Z(\log(L(\boldsymbol{\theta}|Y,Z)), \tag{5.9}$$

where expectation is with respect to the distribution of Z given the current estimate of $\boldsymbol{\theta}$ and Y, i.e., $P(Z|\boldsymbol{\theta}^i,Y)$. The EM algorithm is defined formally as:

1. **The E–Step**: Calculation of Q;
2. **The M–Step**: Maximisation of Q with respect to $\boldsymbol{\theta}$.

Implicit within the Q–step is the estimation of the augmented data with respect to the current model estimate and complete data. This estimation simply involves taking expectations with respect to the current model (conditional on the observed data). In comparison with the data augmentation algorithm, both of the sampling steps are replaced by deterministic steps (an expectation and a maximisation).

The EM algorithm always increases the likelihood $L(\boldsymbol{\theta}|Y)$. To do this we first note the following inequality that holds for all densities f and g by Jensen's inequality (see any standard probability text book).

$$\mathbb{E}_g\left[\log\frac{f(X)}{g(X)}\right] \leq 0 \ . \tag{5.10}$$

A consequence of this, letting f be $P(Y,Z|\boldsymbol{\theta})$ and g be $P(Y,Z|\boldsymbol{\theta}^i)$ is

$$\int \log\left(\frac{P(Z|Y,\boldsymbol{\theta})}{P(Z|Y,\boldsymbol{\theta}^i)}\right) P(Z|Y,\boldsymbol{\theta}^i)dZ \leq 0 \ . \tag{5.11}$$

Since the second step of the EM algorithm maximises $Q(\boldsymbol{\theta},\boldsymbol{\theta}^i)$ as a function of its second argument, $Q(\boldsymbol{\theta},\boldsymbol{\theta}^i) \geq Q(\boldsymbol{\theta}^i,\boldsymbol{\theta}^i)$, so that

$$\int \log\left(\frac{P(Y,Z|\boldsymbol{\theta})}{P(Y,Z|\boldsymbol{\theta}^i)}\right) P(Z|Y,\boldsymbol{\theta}^i)dZ \geq 0 \ . \tag{5.12}$$

Now subtracting 5.11 from 5.12, we get

$$\int \log\left(\frac{P(Y,Z|\boldsymbol{\theta})}{P(Z|Y,\boldsymbol{\theta})}\frac{P(Z|Y,\boldsymbol{\theta}^i)}{P(Y,Z|\boldsymbol{\theta}^i)}\right) P(Z|Y,\boldsymbol{\theta}^i)dZ \geq 0 \ . \tag{5.13}$$

But the two terms being multiplied together within the log above are just $P(Y|\boldsymbol{\theta})$ and $1/P(Y|\boldsymbol{\theta}^i)$ respectively and these do not depend on Z, so these can be taken outside the integral and we get

$$\log P(Y|\boldsymbol{\theta}) \geq \log P(Y|\boldsymbol{\theta}^i), \tag{5.14}$$

since $\int P(Z|Y,\boldsymbol{\theta}^i)dZ \geq 0$. Thus, the likelihood is always increased.

5.3.3 A genetic example

We return to the genetic example of Section 4.4. concerning the genetic linkage of 197 animals, in which the animals are distributed into 4 categories:

$$Y = (y_1, y_2, y_3, y_4) = (125, 18, 20, 34), \tag{5.15}$$

with cell probabilities

$$\left(\frac{1}{2}+\frac{\theta}{4}, \frac{1}{4}(1-\theta), \frac{1}{4}(1-\theta), \frac{\theta}{4}\right) . \tag{5.16}$$

Though it is by no means impossible to maximise this multinomial likelihood directly, we illustrate how the EM algorithm brings about a substantial simplification, by using the same augmentation method that was used in the data augmentation case. Specifically, we augment the observed data Y by dividing the first cell into two, with respective cell probabilities $\frac{1}{2}$ and $\frac{\theta}{4}$, giving an augmented data set $X = (x_1, x_2, x_3, x_4, x_5)$, where $x_1 + x_2 = y_1$, and $x_3 = y_2, x_4 = y_3, x_5 = y_4$. Now we have

$$L(\theta|Y) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}, \tag{5.17}$$

whereas

$$L(\theta|X) \propto \theta^{x_2+x_5}(1-\theta)^{x_3+x_4}. \tag{5.18}$$

Thus we obtain,

$$\begin{aligned} Q(\theta, \theta^i) &= \mathbb{E}((x_2 + x_5)\log(\theta) + (x_3 + x_4)\log(1-\theta)|\theta^i, Y) \\ &= [\mathbb{E}(x_2|\theta^i, Y) + x_5]\log(\theta) + (x_3 + x_4)\log(1-\theta), \end{aligned}$$

where $x_2|\theta^i, Y \sim Bin(125, \frac{\theta^i}{\theta^i+2})$. Thus

$$Q(\theta, \theta^i) = \left[\frac{125\theta^i}{\theta^i + 2} + 34\right]\log(\theta) + 38\log(1-\theta). \tag{5.19}$$

This is easily maximised to give:

$$\theta^{i+1} = \frac{\mathbb{E}(X_2|\theta^i, Y) + x_5}{\mathbb{E}(X_2|\theta^i, Y) + x_3 + x_4 + x_5}, \tag{5.20}$$

where $\mathbb{E}(X_2|\theta^i, Y) = \frac{125\theta^i}{\theta^i+2}$.

Thus, the alternation between estimation and maximisation is clearly seen. This can all be easily carried out by hand with a simple calculator. Starting with $\theta^1 = 0.5$ we obtain the sequence in Table 5.1. Hence the maximum likelihood estimate (posterior mode) is $\theta = 0.6268$.

i	θ^i
1	0.5
2	0.6082
3	0.6243
4	0.6265
5	0.6268
6	0.6268

Table. 5.1: Sequence of EM iterates

5.3.4 Censored regression example

Here we consider a regression problem involving censored data. Consider the data in Table 5.2. These data represent failure times (hours) of motorettes at four different temperatures (Celsius). The asterisks denote a censored observation, so that for example, 8064* means the item lasted *at least* 8064 hours.

Physical considerations suggest the following model relating the logarithm (base 10) of lifetime (t_i) and $v_i = 1000/(\text{temperature } + 273.2)$:

$$t_i = \beta_0 + \beta_1 v_i + \epsilon_i \tag{5.21}$$

where $\epsilon_i \sim N(0, \sigma^2)$. On this scale a plot of t_i against v_i is given in Figure 5.6 in which censored data are plotted as open circles. The raw data are contained in `motor` and the transformed data are in `motor2`. In each case, there is an additional column representing an indicator variable, which takes the value 1 if the observation has been censored, but is zero otherwise.

Now, in this situation, if the data were uncensored we would have a simple regression problem. Because of the censoring we use the EM algorithm to first estimate where the censored values actually are (the E–step) and then fit the model to the augmented data set (the M–step). Re–ordering the

150	170	190	220
8064*	1764	408	408
8064*	2772	408	408
8064*	3444	1344	504
8064*	3542	1344	504
8064*	3780	1440	504
8064*	4860	1680*	528*
8064*	5196	1680*	528*
8064*	5448*	1680*	528*
8064*	5448*	1680*	528*
8064*	5448*	1680*	528*

Table. 5.2: Censored regression data

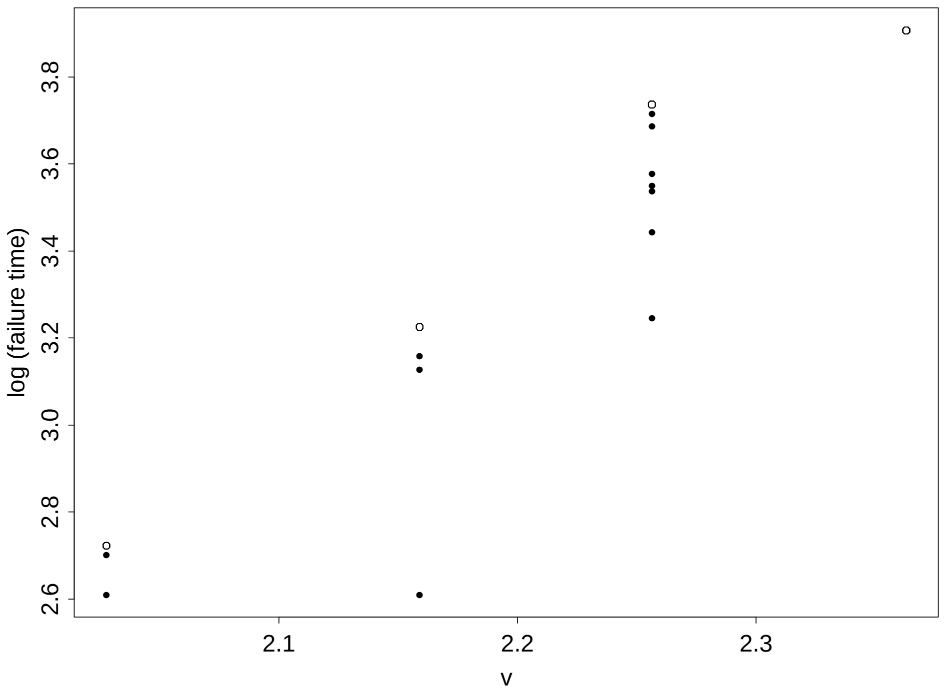


Figure. 5.6: Censored regression data

data so that the first m values are uncensored, and denoting by Z_i the augmented data values, we have the augmented log–likelihood:

$$\log(L(\boldsymbol{\theta}|V,Z)) = -n\log\sigma - \sum_{i=1}^{m}\frac{(t_i-\beta_0-\beta_1 v_i)^2}{2\sigma^2} - \sum_{i=m+1}^{n}\frac{(Z_i-\beta_0-\beta_1 v_i)^2}{2\sigma^2}. \tag{5.22}$$

The E–step requires us to find expectations of $P(Z_i|\beta_0,\beta_1,\sigma,c_i)$ where c_i denotes the censored time. Unconditionally on the censoring, the distribution of each Z_i is normal, so the conditioning is a normal probability, conditioned on $Z_i > c_i$. Thus,

$$\begin{aligned} Q = -n\log\sigma \quad &- \quad \frac{1}{2\sigma^2}\sum_{i=1}^{m}(t_i-\beta_0-\beta_1 v_i)^2 - \frac{1}{2\sigma^2}\sum_{i=m+1}^{n}[\mathbb{E}(Z_i^2|\beta_0,\beta_1,\sigma,Z_i>c_i) \\ &- \quad 2(\beta_0+\beta_1 v_i)\mathbb{E}(Z_i|\beta_0,\beta_1,\sigma,Z_i>c_i)+(\beta_0+\beta_1 v_i)^2]. \end{aligned}$$

Furthermore, it is fairly straightforward to show that

$$\mathbb{E}(Z_i|\beta_0,\beta_1,\sigma,Z_i>c_i) = \mu_i + \sigma H\left(\frac{c_i-\mu_i}{\sigma}\right) \tag{5.23}$$

and

$$[\mathbb{E}(Z_i^2|\beta_0,\beta_1,\sigma,Z_i>c_i) = \mu_i^2+\sigma^2+\sigma(c_i+\mu_i)H\left(\frac{c_i-\mu_i}{\sigma}\right) \tag{5.24}$$

where $\mu_i = \beta_0+\beta_1 v_i$ and $H(x) = \phi(x)/\{1-\Phi(x)\}$.

It follows that maximising Q amounts to solving the usual normal equations for β_0 and β_1, but that maximisation with respect to σ requires solution of the equation:

$$\begin{aligned} &\frac{\sum_{i=1}^{m}(t_i-\beta_0-\beta_1 v_i)^2}{\sigma^4} \\ &+\frac{\sum_{i=m+1}^{n}[\mathbb{E}(Z_i^2|\beta_0,\beta_1,\sigma,Z_i>c_i)-2(\beta_0+\beta_1 v_i)\mathbb{E}(Z_i|\beta_0,\beta_1,\sigma,Z_i>c_i)+(\beta_0+\beta_1 v_i)^2]}{\sigma^4} \\ - \quad &\frac{n}{\sigma^2} = 0 \end{aligned}$$

and so

$$\sigma_{i+1} = \sqrt{\frac{\sum_{j=1}^{m}(t_j-\mu_j^i)^2}{n} + \frac{\sigma_i^2\sum_{j=m+1}^{n}\left[1+\left(\frac{c_j-\mu_j^i}{\sigma_i}\right)H\left(\frac{c_j-\mu_j^i}{\sigma_i}\right)\right]}{n}} \tag{5.25}$$

where $\mu_j^i = \beta_0^i+\beta_1^i v_j$.

This might seem complicated, but all that is going on is:

1. Fitting of regression line by least squares;
2. Maximum likelihood estimate of error variance (allowing for uncertainty in censored observations);
3. Updating of estimates of censored data on basis of fitted model;
4. Iteration of these steps.

This is easily implemented in S-Plus with the following code:

```
em.reg <- function(x, n)
{
        plot(x[, 1], x[, 2])
        a <- lm(x[, 2] ~ x[, 1])
```

```
        b <- coef(a)
        sig <- sqrt(deviance(a)/38)
        cat(b, sig, fill = T)
        for(i in 1:n) {
                mu <- b[1] + b[2] * x[, 1]
                ez <- mu + sig * em.h((x[, 2] - mu)/sig)
                for(j in 1:length(x[, 1])) {
                        if(x[j, 3] == 0) {
                                ez[j] <- x[j, 2]
                        }
                }
                a <- lm(ez ~ x[, 1])
                b <- coef(a)
                ss <- sum((ez[1:17] - mu[1:17])^2)/40
                ss <- ss + (sig^2 * sum(((x[18:40, 2] - mu[18:40])/sig) * em.h(
                        (x[18:40, 2] - mu[18:40])/sig) + 1))/40
                sig <- sqrt(ss)
                cat(b, sig, fill = T)
                abline(a, col = i)
        }
}
```

and the function

```
em.h <- function(x)
{
      dnorm(x)/(1 - pnorm(x))
}
```

Typing `em.reg(motor2,10)` produced the output:

```
-4.9305 3.7470 0.1572
-5.2601 3.9262 0.1998
-5.5112 4.0558 0.2173
-5.6784 4.1398 0.2288
-5.7854 4.1933 0.2371
-5.8552 4.2284 0.2431
-5.9023 4.2521 0.2474
-5.9350 4.2686 0.2506
-5.9581 4.2803 0.2529
-5.9747 4.2887 0.2545
-5.9867 4.2947 0.2558
```

and the sequence of linear fits as shown in Figure 5.7. The first two columns provide the current estimates of the regression parameters β_0 and β_1, the third column gives the corresponding estimate for the error variance. Note that the initial fit was made as if the data were not censored, so the difference between the first and final fits gives an indication of the necessity of taking the censoring into account.

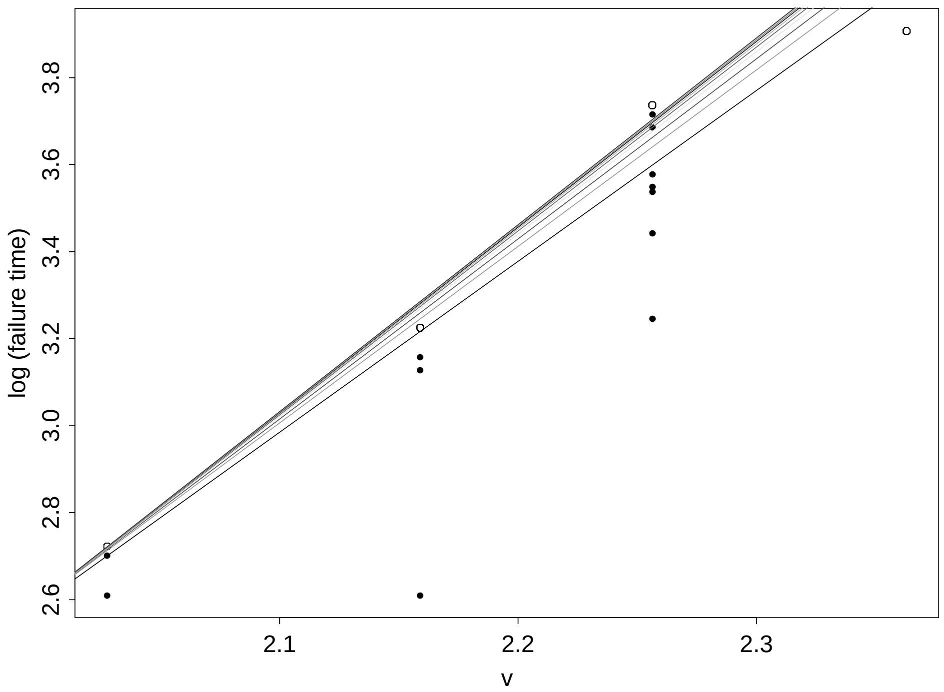


Figure. 5.7: EM iterates to regression fit of censored data

Every iteration of an EM algorithm results in a value of $\boldsymbol{\theta}$ with higher likelihood. Furthermore, it can be shown that if these iterates converge, then they converge to a stationary point of the likelihood function, though this is not necessarily a global maximum. Furthermore, the rate of convergence can be very slow, suggesting that alternative procedures, or techniques which accelerate the convergence, may often be appropriate.

5.3.5 Standard Errors

As usual, calculation of standard errors requires calculation of the inverse Hessian matrix:

$$\left[-\frac{d^2 \log p(\boldsymbol{\theta}|Y)}{d\theta_i d\theta_j}\right]^{-1} \tag{5.26}$$

evaluated at the mode $\boldsymbol{\theta}^*$. In missing data problems this may be difficult to evaluate directly. It is possible however to exploit the structure of the EM algorithm in simplifying this calculation. This uses what is known as the 'missing information principle':

Observed information = Complete information — Missing Information

Explicitly, this takes the form:

$$-\frac{d^2 \log p(\boldsymbol{\theta}|Y)}{d\theta_i d\theta_j} = \left[-\frac{d^2 Q(\boldsymbol{\theta}, \boldsymbol{\phi})}{d\theta_i d\theta_j}\right]_{\boldsymbol{\phi}=\boldsymbol{\theta}} - \left[-\frac{d^2 H(\boldsymbol{\theta}, \boldsymbol{\phi})}{d\theta_i d\theta_j}\right]_{\boldsymbol{\phi}=\boldsymbol{\theta}} \tag{5.27}$$

where

$$H(\boldsymbol{\theta},\boldsymbol{\phi}) = \int_Z \log(p(Z|\boldsymbol{\theta},Y))p(Z|\boldsymbol{\phi},Y)dZ. \tag{5.28}$$

Without the missing data, the first term on the right hand–side of (5.27) would be the Hessian; the second term compensates for the missing information. The proof of (5.27) is almost immediate from the representation

$$\log p(\boldsymbol{\theta}|Y) = \log p(\boldsymbol{\theta}|Y,Z) - \log p(Z|Y,\boldsymbol{\theta}) + C, \tag{5.29}$$

which is essentially a statement of Bayes' theorem. Application of the missing information principle is simplified by using the result:

$$-\frac{d^2H(\boldsymbol{\theta},\boldsymbol{\phi})}{d\theta_i d\theta_j} = Var\left[\frac{d\log p(\boldsymbol{\theta}|Y,Z)}{d\boldsymbol{\theta}}\right], \tag{5.30}$$

a result which mirrors the corresponding result in the classical (full data) theory. We can illustrate this with the genetic linkage example. In that case

$$\begin{aligned} -\left[\frac{d^2Q(\theta,\phi)}{d\theta^2}\right]_{\theta^*} &= \frac{E(X_2|\theta^*,Y)+x_5}{\theta^{*2}} + \frac{x_3+x_4}{(1-\theta^*)^2} \\ &= \frac{29.83}{0.6268^2} + \frac{38}{(1-0.6268)^2} \\ &= 435.3. \end{aligned}$$

This is the information available had the augmented data been genuine data. Now, we also have:

$$\frac{d\log p(\theta|Y,Z)}{d\theta} = \frac{x_2+x_5}{\theta} - \frac{x_3+x_4}{1-\theta}, \tag{5.31}$$

so that,

$$\begin{aligned} Var\left[\frac{d\log p(\theta|Y,Z)}{d\theta}\right] &= \frac{Var(X_2|\theta^*)}{\theta^{*2}} \\ &= 125\left(\frac{\theta^*}{2+\theta^*}\right)\left(\frac{2}{2+\theta^*}\right)\frac{1}{\theta^{*2}} \\ &= 22.71/0.6268^2 = 57.8. \end{aligned}$$

Thus

$$-\frac{d^2\log p(\theta|Y)}{d\theta^2} = 435.3 - 57.8 = 377.5 \tag{5.32}$$

and the standard error of θ^* is $\sqrt{(1/377.5)} = 0.05$.

5.3.6 Monte Carlo EM algorithm

In some situations, direct evaluation of the Q function, or equivalently, estimation of the augmented data as expected values given the current model estimate and complete data, is difficult. In such situations we can employ a Monte Carlo step to approximate Q. Thus, the E–Step of the EM algorithm is replaced with

1. Simulate $z_1, z_2, \ldots, z_m$ from $P(Z|Y,\boldsymbol{\theta}^i)$.

2. Let $\hat{Q}_{i+1} = \frac{1}{m}\sum_{j=1}^m \log L(\boldsymbol{\theta}|z_j,Y)$.

Thus, the exact expectation required for Q is replaced by a Monte Carlo estimate of the integral.

The choice of m here will affect the accuracy of the final result. One approach therefore is to initiate the algorithm with a small value of m, but then to increase m later to minimise variability due to the error in the Monte Carlo integration.

We can apply this again to the genetic linkage example. There we had

$$Q(\theta, \theta^i) = [E(X_2|\theta^i, Y) + x_5]\log(\theta) + (x_3 + x_4)\log(1-\theta), \tag{5.33}$$

where $x_2|\theta^i, Y \sim Bin(125, \frac{\theta^i}{\theta^i+2})$. So, we simply generate $z_1, z_2, \ldots, z_m$ from $Bin(125, \frac{\theta^i}{\theta^i+2})$, and take the mean of these realisations, $\bar{z}$, as the approximation to the expectation in (5.33). Then the M–Step continues as before with

$$\theta^{i+1} = \frac{\bar{z} + x_5}{\bar{z} + x_3 + x_4 + x_5}. \tag{5.34}$$

This is implemented in S-Plus with the following code:

```
em.mc <- function(x, m, th.init)
{
        th <- th.init
        for(i in 1:length(m)) {
                p <- th/(2 + th)
                z <- rbinom(m[i], 125, p)
                me <- mean(z)
                th <- (me + x[4])/(me + x[2] + x[3] + x[4])
                cat(i, round(th, 5), fill = T)
        }
}
```

where `x` is the data, `m` is a vector of the values of m we choose for each step in the iteration, and `th.init` is an initial value for θ. Choosing $m = 10$ for the first 8 iterations, and $m = 1000$ for the next 8, and with an initial value of $\theta = 0.5$ we obtained:

```
1 0.61224
2 0.62227
3 0.62745
4 0.62488
5 0.62927
6 0.62076
7 0.62854
8 0.62672
9 0.62734
10 0.627
11 0.62747
12 0.62706
13 0.62685
14 0.62799
15 0.62736
16 0.62703
```

Note that monitoring convergence is more difficult in this case due to the inherent variation in the Monte Carlo integrals.

5.3.7 Generalising the EM algorithm

A limitation of the EM algorithm is that it gives only the mode of the likelihood surface however it is almost always preferable to obtain a more complete view of the likelihood surface to take account of any skewness in the likelihood when calculating confidence intervals for example.

We look here at a method of modifying the EM algorithm to obtain an estimate of the entire distribution, $L(\boldsymbol{\theta}|Y)$, again based on a simplified augmented distribution, $L(\boldsymbol{\theta}|Y,Z)$, which gives the necessary generalisation. As we shall see, the Gibbs sampler and other Monte Carlo Markov Chain algorithms are further generalisations of this idea.

Consider the following two identities. The **posterior identity**:

$$L(\boldsymbol{\theta}|Y) = \int_Z L(\boldsymbol{\theta}|Y,Z)P(Z|Y)dZ, \tag{5.35}$$

and the **predictive identity**

$$P(Z|Y) = \int_{\boldsymbol{\theta}} P(Z|\boldsymbol{\phi},Y)L(\boldsymbol{\phi}|Y)d\boldsymbol{\phi}. \tag{5.36}$$

These identities suggest the following algorithm:

1. Simulate $z_1, z_2, \ldots, z_m$ from current estimate of $P(Z|Y)$.
2. Update estimate of $L(\boldsymbol{\theta}|Y)$ as
$$\frac{1}{m}\sum_{j=1}^{m} L(\boldsymbol{\theta}|z_j,Y). \tag{5.37}$$
3. Iterate.

At stage 1 of the iteration we need to simulate from $P(Z|Y)$. By the predictive identity, since this is a mixture over the distribution of $\boldsymbol{\theta}$, we perform this step by

1. Sampling $\boldsymbol{\theta}^*$ from current estimate of $L(\boldsymbol{\theta}|Y)$; and
2. Sampling Z from $P(Z|\boldsymbol{\theta}^*,Y)$.

Furthermore, because of the discrete form of the estimate of $L(\boldsymbol{\theta}|Y)$ in (5.37), simulation from $L(\boldsymbol{\theta}|Y)$ is simply a matter of random (uniform) selection of j from the integers, $1,\ldots,m$, and then simulation from $L(\boldsymbol{\theta}|z_j,Y)$.

Before seeing how this works it is worth comparing the approach with the EM algorithm. Here we are simulating the augmented data from the current estimate of the predictive model, we are then integrating to obtain an improved posterior model. Thus we might call this an SI algorithm with the **simulation** and **integration** steps replacing the estimation and maximisation respectively of the EM algorithm.

5.3.8 Example

Again we use the genetic linkage example to illustrate this algorithm. In that situation we had

$$L(\theta|X) \propto \theta^{x_2+x_5}(1-\theta)^{x_3+x_4} \tag{5.38}$$

which is a Beta distribution, and $x_2|\theta,Y \sim Bin(125, \frac{\theta}{\theta+2})$. Thus the SI algorithm becomes:

1. SIMULATE $x_2^1, \ldots, x_2^m$ FROM CURRENT ESTIMATE OF $P(X|Y)$.

2. UPDATE ESTIMATE OF $L(\theta|Y)$ AS

$$\frac{1}{m}\sum_{j=1}^{m} Be(v_1^j, v_2^j)(\theta) \tag{5.39}$$

   WHERE $v_1^j = x_2^j + x_5 + 1$ AND $v_2^j = x_3 + x_4 + 1$.

3. ITERATE.

Moreover, Step 1 in the algorithm consists of repeating m times:

1. Sampling j uniformly from $1, \ldots, m$;

2. Drawing θ^* from $Be(v_1^j, v_2^j)(\theta)$;

3. Drawing x from $Bin(125, \frac{\theta^*}{\theta^*+2})$.

This is implemented in S-Plus with the following code:

```
si.fit <- function(m, n, xa)
{
        th <- runif(m)
        cat(mean(th), sqrt(var(th)), fill = T)
        p <- th/(th + 2)
        x <- rbinom(m, xa[1], p)
        ind <- sample(x, m, replace = T)
        for(j in 1:n) {
                th <- rbeta(m, x[ind] + x[4] + 1, xa[2] + xa[3] + 1)
                p <- th/(th + 2)
                x <- rbinom(m, xa[1], p)
                ind <- sample(x, m, replace = T)
                cat(mean(th), sqrt(var(th)), fill = T)
        }
        xx <- seq(0.01, 0.99, by = 0.01)
        y <- bet.mix(xx, x + xa[4] + 1, rep(xa[2] + xa[3] + 1, m))
        plot(xx, y, type = "n")
        lines(spline(xx, y))
}
```

Here, `m` is the number of evaluations in the Monte Carlo integral, `n` is the number of iterations and `xa` contains the data. The function `bet.mix` simply evaluates a mixture of beta functions:

```
 bet.mix <-function(x, a, b)
{
        s <- (x^(a - 1) * (1 - x)^(b - 1) * gamma(a + b))/gamma(a)/gamma(b)
        mean(s)
}
```

Applying `si.fit` to the original data gives the following output, which are respectively the mean and variance of the predictive distribution estimated at each iteration. Monitoring these gives an assessment of convergence.

mean	s.d.
0.5506	0.2725
0.4747	0.1142
0.5918	0.0555
0.5881	0.0625
0.6171	0.055
0.6071	0.052
0.6046	0.0537
0.5868	0.0555
0.6163	0.0468
0.5827	0.0557
0.597	0.0562

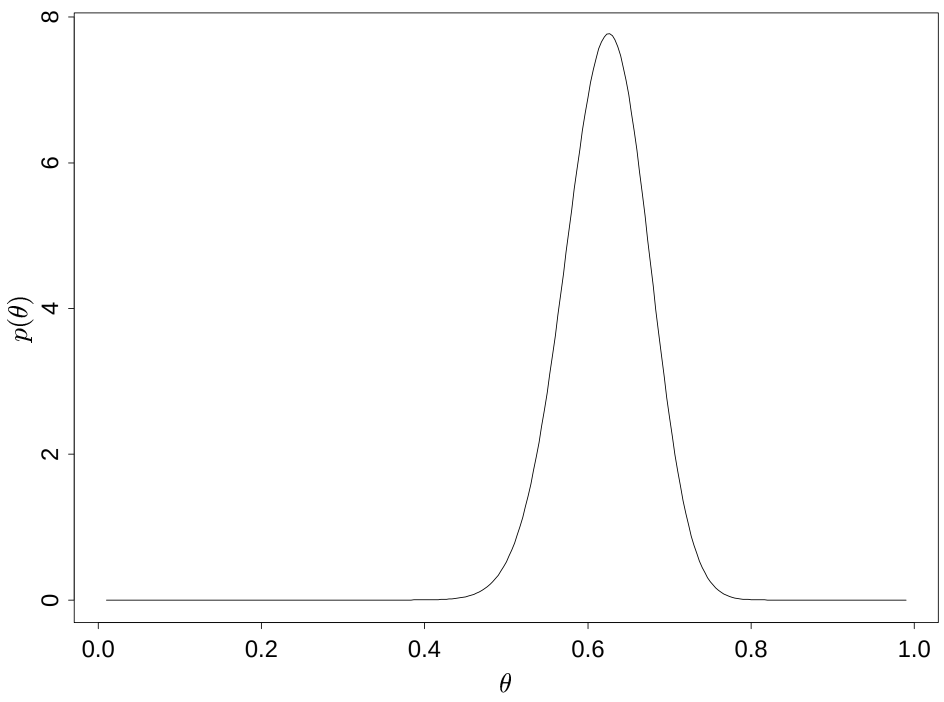


Figure. 5.8: Posterior distribution in genetic linkage example.

The estimated posterior distribution of θ is shown in Figure 5.8. Convergence can also be assessed by plotting the posterior distribution at each iteration as in Figure 5.9.

Clearly, convergence has been achieved after just a couple of iterations of the process. Again, there is some variability due to the Monte Carlo integration step, but this can again be reduced by increasing the value of `m`.

In this example, we see that the posterior distribution (likelihood) is quite symmetric, so that confidence intervals based on the mode and the standard error will be reasonably accurate. However,

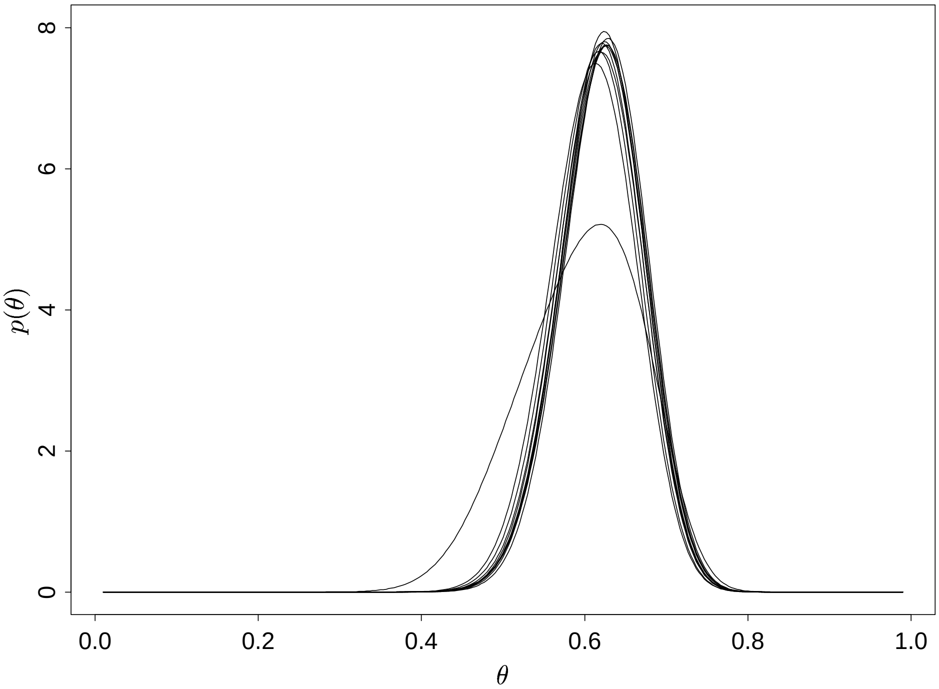


Figure. 5.9: Posterior distributions at each iteration in genetic linkage example.

if we replace the original data `xa_(125,18,20,34)` with `xa_(30,1,1,4)` we obtain the highly skewed posterior density in Figure 5.10.

5.4 Exercises

Exercise 5.1: *Consider the autoregressive example in Section 5.2.4. Four separate within-model mechanisms are derived.*

i) Verify the form of the target density $b_T(\boldsymbol{\theta})$.

ii) Verify the conditional distribution $b_T(\sigma^2|...)$.

iii) Verify the acceptance probability for the Metropolis Hastings update of parameter a_i.

iv) Verify the conditional distribution $b_t(a_i|...)$

v) Verify the multivariate conditional distribution $b_T(\boldsymbol{a}|...)$.

vi) Verify mean for the Langevin proposal, updating parameter a_i. What is the corresponding acceptance probability?

vii What happens to each of these proposals as $T \to 0$? What happens to the acceptance probabilities?

Exercise 5.2: *Consider the 2-dimensional Bohachevsky function*

$$f(x,y) = x^2 + 2y^2 - 0.3\cos(3\pi x) - 0.4\cos(4\pi y) + 0.7 \quad \textit{for } -1 \le x, y \le 1.$$

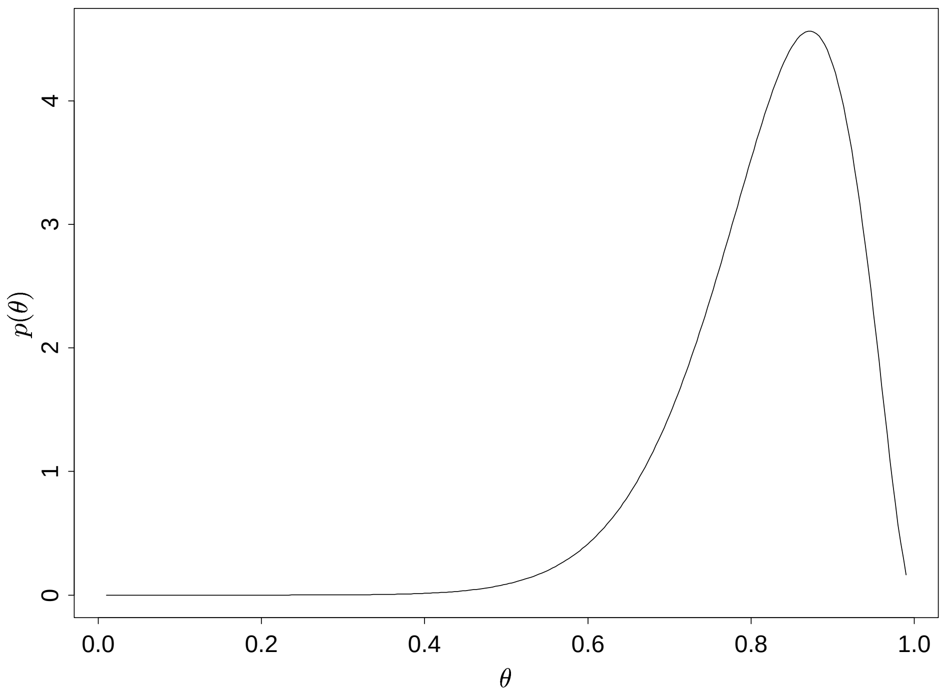


Figure. 5.10: Posterior distribution in genetic linkage example with modified data.

How would you locate the minimum of this function via simulated annealing and using

a) *An independence sampling scheme?*

b) *A (random walk) Metropolis scheme?*

Which do you think might work best? Could you use a Gibbs sampling scheme here?

Exercise 5.3: *Given data $x_1, ..., x_n$, suppose that we wish to decide whether a Normal or a Poisson model provides a better fit. Calculate the MLE's under each model using analytic methods. Describe an annealing algorithm to fit the Normal model using Gibbs updates. What happens to these updates as $T \to 0$? Why can you not implement the Gibbs Sampler for the Possion model?*

Calculate the form of the AIC statistic for each model. Now describe how your annealing algorithm can be extended to distinguish between the two models.

Appendix A

SPLUS NOTES

Throughout this course it will be necessary to use the S-Plus statistical package. The aim of this appendix is to enable those of you who have not used S-Plus before, to follow the examples in the course. Note that there are various S-Plus books held in the library if you require further information. You might also like to refer to the S-Plus Web sites at

http://www.mathsoft.com/splus

or

http://www.utstat.toronto.edu/splus/intro.html

A.1 Accessing S-Plus

First log on to a Linux machine in the computer lab. You should have already been given a userid and password.

To start S-Plus type Splus5 at the command prompt. If you want to exit S-Plus at any time type

```
> q()
```

(just “q()”, without the “>”).

If you need help on a particular command, say plot, type help(plot) for information on how to use the command. If you're not sure what command you need, type help.start() and you'll get a browseable list of commands.

A.2 S-Plus Commands

S-Plus commands are always of the form <command>(<options>). In the cases where there are no options, for example the quit command q(), you must still add the brackets. This is because S-Plus thinks of its commands as functions (more on these later) any of which may require options in order to invoke them properly. If you omit the brackets, then S-Plus thinks that you don't want to execute the command, but simply to see what it does. Thus, typing a command without the brackets, simply lists that commands that would be executed, if the brackets hadn't been omitted.

For example, typing

```
> q
```

results in the following output,

```
function(...)
.Internal(q(...), "S_dummy", T, 33)
```

which makes no sense to you now, but hopefully will later.

A.3 Creating and Deleting Objects

Create a vector `x` and store the integers $5, 6, ..., 15$ in it by typing

```
> x <- 5:15
```

Note the assignment operator "`<-`", which might be interpreted as meaning "comes from", as opposed to the more usual "=" symbol. Now check what is in `x` by typing

```
> x
```

You can also refer to elements of `x`. Try

```
> x[3]
> x[3:5]
```

You can check which objects you have created using `objects()`, and remove unwanted objects using `rm()`. For example, try

```
> xx <- 2*x
> xx
> objects()
> rm(xx)
> objects()
```

A.4 Plotting

Plot the data points in `x` by typing

```
> plot(x)
```

You should see a new window containing a plot of `x[i]` against `i`. You can customise the plot by adding labels and specifying that you want a solid line to join the points, rather than asterisks at each data point. Type

```
> plot(x,type="l",xlab="i",ylab="x[i]")
```

To see what other options are available, you might want to use the help system to investigate.

To send your plot to the printer, you can select "print" from the Graph pull-down menu on the window with the plot you want. Warning! You must set the printer to "ps2", and it might be a good idea to give your plot a title which includes your user id i.e., `title("My Plot.steve")`.

A.5 Arithmetic Operations and S-Plus Objects

You can perform basic arithmetic in S-Plus as you would with a calculator, using the operators `+`, `-`, `*` and `/`, for example:

```
> 2 + 2
[1] 4

> 3 * 8
[1] 24

> 16 ^ (1/2)
[1] 4
```

Note that the answer always has a `[1]` before it. That is because S-Plus usually expects the output of any command to be directed into a S-Plus *object*. The absence of the assignment operator, `<-`, means that the output from the command is dumped to the screen instead. For example, try typing

```
> 2 * 4
> x <- 2 * 4
> x[1]
```

A *vector* in S-Plus is an ordered set of values. Simple numeric values can be created with either the `c()` function, or the sequence operator `:`. Try typing

```
> x <- c(3,1,7)
> x
> y <- 1:5
> y
```

A matrix in S-Plus, is simply a vector with a specified number of rows and columns. You can create matrices with the `matrix()` command. For example, try typing

```
> A <- matrix(c(1:9),nrow=3)
> A
```

You can then refer to the (i, j)th element of your matrix simply as `A[i,j]`.

If we have two vectors, `x` and `y`, of equal length, then there are two ways in which we can multiply them. Typing

```
> xy <- x * y
```

provides a new vector `xy`, which has element `xy[i] = x[i]*y[i]`. In order to treat `x` and `y` as vectors and multiply them, you need to use the matrix product command `%*%`. Type

```
> x <- 1:3
> y <- 2*(1:3)
> xy.vect <- x %*% y
> xy <- x * y
> xy
> xy.vect
```

Another useful command is the transpose command, `t()`, which can be used to transpose either matrices or vectors. Try

```
> B <- t(A)
> B
> z <- t(y)
> z
> yB <- y * B
> yB
```

A.6 Statistical Operations and Functions

S-Plus has a large selection of functions. For example

```
> mean(x)
> sqrt(var(x))
> quantile(x, probs=c(.25,.5,.75))
```

produces the sample mean, standard deviation and quartiles of the data in `x`. Note that the `c()` function collects objects. Try

```
> c(x[1:3],1000,-x[3:5])
```

You can also create your own functions. For example, create a function called `my.plot()` to take a vector `x`, square it and plot it. Type the following lines

```
> my.plot <- function(x){
+ y <- x^2
+ plot(x,y)
+ }
```

You can then invoke this command with

```
> my.plot(x)
```

You will be expected to write some basic functions of your own, during the course. One useful command which can be used to figure out why your function is not working properly, is the `cat()` command, which can be used to print object values to the screen. For example, you could add the line

```
   cat("x =",x,"\n")
   cat("and y =",y,"\n")
```

to the `myplot()` function above, to see the values of `x` and `y`. Note that the `"\n"` command produces a new line in the output.

One final useful tip is that a function will always return the result of the final command in the list of commands that defines the function. For example

```
dummy <- function(n){
         x <- 1:n
         y <- 2*(1:n)-5
         xy <- x*y
         xy
}
```

returns the vector xy, whereas

```
dummy <- function(n){
         x <- 1:n
         y <- 2*(1:n)-5
         xy <- x*y
         x
}
```

returns the vector x. Note that if you fail to end the function definition on such a line i.e., you type

```
dummy <- function(n){
         x <- 1:n
         y <- 2*(1:n)-5
         xy <- x*y
}
```

you'll get an error message

```
> Warning messages:
  Last expression in function is an assignment
        (You probably wanted to return the left-hand side)
 in: xy <- x * y
```

S-Plus functions will always return some object, so you should always make sure that you specify which object you want returned, by simply placing that object's name on the last line of the function definition.

A.7 Editing Objects

Modify the my.plot() function, to plot an additional curve $y = x^3 - 6x^2$, with a dotted line. To invoke the default editor (usually notepad). Type

```
> fix(my.plot2)
```

You should get the editor starting up and creating a new object, called my.plot2. As it is a new function, S-Plus automatically creates a few lines for you

```
function()
{
}
```

All you need to do to define the function my.plot2(), is to add the list of arguments that the function requires, between the () brackets, and then the function commands between the {} brackets.

You can now edit the function, using standard commands. Edit the function to look like this:

```
function(x){
   y <- x^2
   plot(x,y)
   y <- x^3 - 6*x^2
   lines(x,y,lty=2)
}
```

Now save the edited version and quit the editor, to save the function. If S-Plus tells you to re-edit the function because you have made an error, type

```
> fix()
```

This will re-open the editor with the file as you left it.

After quitting the editor, check what you have done by typing

```
> my.plot2
```

(without parentheses). Run `my.plot2`, to check that it works.

Thus, there are two ways to create new functions.

1. Type the function in on the command line i.e., to create the `my.plot2` function, type

   ```
   > my.plot2 <- function(x){
   +        y <- x^2
   +        plot(x,y)
   +        y <- x^3 - 6*x^2
   +        lines(x,y,lty=2)
   + }
   ```

   at the command line. Or,

2. Type `fix(my.plot2)` and edit the file to look like this;

   ```
   function(x){
      y <- x^2
      plot(x,y)
      y <- x^3 - 6*x^2
      lines(x,y,lty=2)
   }
   ```

For shorter functions, the first method may be better, but for more complex functions, the `fix` command is better, since it enables you to re-edit the function if you make a mistake.

A.8 Loops

When vector operations are not convenient, you can operate on vectors element by element, using loops. For example:

```
for(i in 3:n){
   y[i] <- y[i-1] * y[i-2]
}
```

Alternatively, you can use the `while()` command (see the help window). For example you could have written

```
i <- 3
while(i <= n){
   y[i] <- y[i-1] * y[i-2]
i <- i + 1
}
```

A.9 Random Number Generation

Many of the functions that you will be asked to write, will involve the generation of random variables. S-Plus, can generate observations from a large number of common statistical distributions. To see a list of distributions, search the help system under "random numbers".

Each of the random number functions has a name beginning with one of the following four one-letter codes, indicating the type of function.

- r Random number generator. Requires an argument specifying the sample size, plus the parameters associated with the distribution required.
- p Probability function. Requires a vector of quantiles, plus the parameters associated with the distribution required.
- d Density function. Requires a vector of quantiles, plus the parameters associated with the distribution required.
- q Quantile function. Requires a vector of probabilities, plus the parameters associated with the distribution required.

For example, the probability that a value from the standard normal distribution is less than `x` is calculated with the expression `pnorm(x,0,1)`, whereas to compute the 95% quantile from the same distribution, we would use the expression `qnorm(0.95,0,1)`. Finally, to generate `n` observations from the standard normal distribution, we use `rnorm(n,0,1)`.

Try typing

```
> a <- rgamma(200,1)
> a
> par(mfrow=c(1,2))
> plot(a,type=‘‘l’’)
```

You now have a plot of 200 random observations from a Gamma distribution. To see what a Gamma distribution looks like, you could plot a histogram from your sample. Type

```
> hist(a,probability=T)
```

To see how well the histogram approximates the Gamma density, you can now superimpose a plot of the density. Type

```
> b <- dgamma(1:50/10,1)
> lines(1:50/10,b)
```

Don't worry if you get a couple of "Lines out of bounds" messages, just ignore them. These arise because you are laying one plot (the density) on top of another (the histogram) and the two have slightly different axis ranges.

Important Note: In the later stages of the course, we shall make use of the Gamma distribution, which we shall define in terms of two parameters, usually α and β. Older versions of S-Plus automatically assume that the Gamma distribution has the scale parameter set to $\beta = 1$. Thus, to generate an observation from the $\Gamma(\alpha, \beta)$ distribution, you should use a command of the form

```
> x <- rgamma(1,alpha)/beta
```

then `x` will have the required distribution. However, if your version of S-Plus recognises the `rgamma` command with three arguments, then you can simply use `rgamma(1,alpha,beta)`.

A.10 Obtaining my Functions

Several of the exercises will refer to functions that have been used in the lectures. All of the functions provided in the lectures can be obtained by typing

```
> attach("/home/steve/MCI/.Data")
```

You can see what functions you then have by typing

```
> objects(where=2)
```

S-Plus works in a hierarchical manner, with several levels of databases. The top level consists of all of the objects that you have created. The second level will normally consist of the database of standard S-Plus commands. After entering the `attach` command above, my functions get put on the second level and the standard functions get demoted to the third level of the hierarchy.

When looking for a function, S-Plus will look at the top level first and work it's way downwards. Thus, if you have created an object called `my.obj`, then S-Plus will automatically assume that yours (in the top level) is the one you are referring to. If you had not created an object of that name, then S-Plus will automatically assume that you mean ours.

If you edit one of these functions, then the object of ours (in level 2) actually remains unaltered, but S-Plus automatically creates a new object of the same name which it places in level 1 (your level). It is this one that S-Plus will always look at, until it is deleted. However, you will get a warning message saying that you have created a new object (in level 1) of the same name as an existing one (mine in level 2). Something like:

```
Warning messages:
  Conflicting definitions of "<filename>" on databases
    "/home/<your userid>/MySwork" and "/home/steve/MCI/.Data"
    in: assign(name, new.x, where = 1)
```

This is just a warning, ignore it.

In fact you don't need to understand how S-Plus works in this regard, but don't be surprised if, after attaching the files, they don't appear when you type `objects()`. You can still both use and edit them.

A.11 Exercises

1. Try creating an S-Plus function `trunc.norm()`, which takes the arguments `n`, `mu` and `sigma` and which creates a vector of `n` observations from a normal $N(\mu, \sigma^2)$ distribution and returns them.

2. Edit the `trunc.norm()` function, so that `n` positive values are returned and any negative values discarded.

3. Re-edit `trunc.norm()` to add a plot of the resulting vector of observations. Add axis labels to the plot and a title. (Use the help window to find out how titles may be added to plots.)

4. Re-edit your function once again, to calculate the mean and variance of the resulting sequence, and store them in a vector, `moments`, of length 2, which is also returned by the function.

Appendix B

SOLUTIONS TO EXERCISES

Solutions to Exercises of Chapter 2

Solution 2.1:

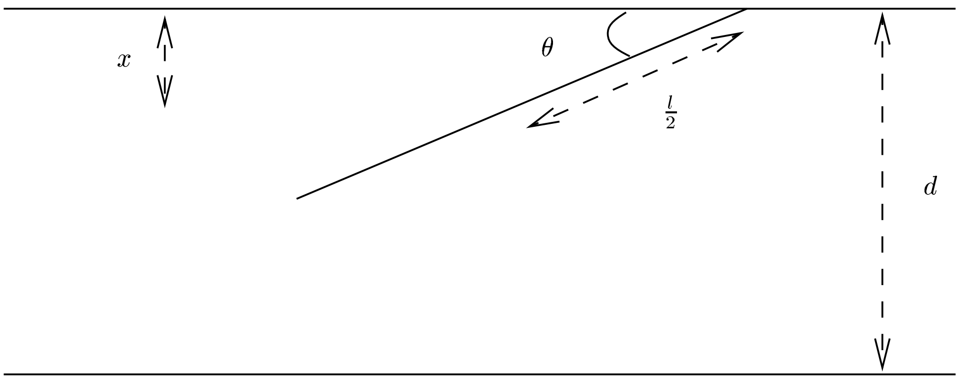


Figure. B.1: Buffon's needle experiment.

$$\mathbb{P}(\text{needle touches next line up}) = \frac{1}{\pi}\int_0^{\pi} \mathbb{P}(\text{touches}|\theta)d\theta. \tag{B.1}$$

Now it is easy to show that,

$$\mathbb{P}(\text{touches } \ |\theta) = \frac{\ell}{d}\sin\theta$$

Therefore,

$$\begin{aligned}
\text{(B.1)} &= \int_0^{\pi} \frac{\ell}{d\pi}\sin\theta \ d\theta \\
&= \frac{\ell}{d\pi}\Big[-\cos\theta\Big]_0^{\pi} \\
&= \frac{2\ell}{d\pi} \\
&= p.
\end{aligned}$$

Alternatively,

$$\text{(B.1)} \; = \frac{1}{d}\int_0^d \mathbb{P}(\text{touches}|x)dx,$$

leading to the same result. □

This is a Binomial (n, p) experiment, i.e. $R \sim \text{Bin}(n, p)$. Therefore $\text{Var}(R) = np(1-p)$ and hence $\text{Var}(\hat{p}) = \text{Var}(R/n) = \frac{1}{n^2}np(1-p) = p(1-p)/n$
Now, $\phi = 1/\pi$ and $\hat{\phi} = \hat{p}d/2\ell$. Therefore

$$\begin{aligned} \text{Var}(\hat{\phi}) &= \frac{d^2}{4\ell^2}\text{Var}(\hat{p}) \\ &= \frac{d^2}{4\ell^2}\frac{1}{n}\frac{2\ell}{d\pi}\left(1 - \frac{2\ell}{d\pi}\right). \end{aligned}$$

So, $\min Var(\hat{\phi})$ is given when $(1 - 2\ell/d\pi)$ is smallest (subject to $\ell \leq d$). This implies that it is minimised when ℓ is as big as possible, i.e. $\ell = d$. □

Solution 2.2: Split into three cases.

1. $U_1 + U_2 < 1$ (i.e. consider $i = 3$ w.l.o.g.). Then $U_3 = U_1 + U_2 > U_1$ and U_2. Therefore $U_3 \notin (U_1, U_2)$.

2. $U_1 + U_2 = 1$. This case is trivial, since $U_3 = 0$, so that $U_3 \notin (U_1, U_2)$.

3. $U_1 + U_2 > 1$. Now $U_1, U_2 > 1$, so $U_3 \in (U_1, U_2)$ if and only if $U_1 < U_1 + U_2 - 1 < U_2$ (since $U_1 + U_2 > 1$)

   This implies that $1 + U_1 < U_1 + U_2 < U_2 + 1$ and thus $1 < U_2 < U_2 + 1 - U_1$.

   However, $U_2 < 1$ hence a contradiction, and hence $U_3 \notin (U_1, U_2)$.

   Hence $\mathbb{P}(U_3 \in (U_1, U_2)) = 0$. □

Finally, if the U_i are truly random, then

$$\begin{aligned} \mathbb{P}(U_3 \in (U_1, U_2)) &= \int_0^1 \int_{U_1}^1 \int_{U_1}^{U_2} dU_3 dU_2 dU_1 \\ &= \int_0^1 \int_{U_1}^1 (U_2 - U_1) dU_2 dU_1 \\ &= \int_0^1 (\frac{1}{2} - U_1 + \frac{1}{2}U_1^2) dU_1 \\ &= \left[\frac{1}{2}U_1 - \frac{1}{2}U_1^2 + \frac{1}{6}U_1^3\right]_0^1 \\ &= \frac{1}{6}. \end{aligned}$$

□

Solution 2.3:

(a) For maximum cycle length, c and M must have no common factors, $(a-1)$ is a multiple of every prime that divides M, and $(a-1)$ is a multiple of 4 if M is.

Let $M = 10^k$, for $k \geq 2$. M is a multiple of 4, so

$$(a-1) = 4n \text{ for some } n \in \mathbb{Z}$$

.

The primes dividing M are 5 and 2, so

$$(a-1) = 10\ell \text{ for some } \ell \in \mathbb{Z}.$$

Thus $(a-1)$ must be a multiple of 2, 5 and 4, i.e.,

$$\begin{aligned}
(a-1) &= 20d \text{ for some } d \in \mathbb{Z} \\
\Rightarrow a &= 20d + 1.
\end{aligned}$$

□

(b) Proof by induction.

Assume true for $k = s$ and show true for $k = s+1$.

$$\begin{aligned}
X_{n+s+1} &= [aX_{n+s} + c] \bmod M \\
&= \left[a\left(a^s X_n + \left(\frac{a^s - 1}{a-1}\right) c\right) + c\right] \bmod M \\
&= \left[a^{s+1} X_n + \left(\frac{a^{s+1} - a}{a-1}\right) c + c\right] \bmod M \\
&= \left[a^{s+1} X_n + \frac{a^{s+1} - 1}{a-1} c\right] \bmod M.
\end{aligned}$$

Therefore, given it is true for $k = s$, it is also true for $k = s+1$. Since it is clearly true for $k = 0$, it is therefore true for all $k \geq 0$, by induction. □

In a multiplicative generator, $c = 0$ and hence

$$X_{n+k} = a^k X_n \pmod{M}.$$

It follows that the cycle length is given by the smallest j s.t. $a^j = 1 \pmod{M}$.

Solution 2.4: X has a geometric distribution with parameter p, therefore

$$p_X(x) = p(1-p)^{x-1} \quad x = 1, 2, \ldots$$

The cumulative sums are given by:

$$\begin{aligned}
P_X(j) &= \sum_{x=1}^{j} p_X(x) \\
&= \sum_{x=1}^{j} p(1-p)^{x-1} \\
&= p\left[\frac{1-(1-p)^j}{1-(1-p)}\right] \\
&= 1 - (1-p)^j \quad j \geq 1.
\end{aligned}$$

Thus the algorithm is:

STEP 1. SELECT $u \sim U(0,1)$.
STEP 2. SET $X = j$ IF AND ONLY IF

$$\begin{aligned} & 1-(1-p)^{j-1} < u \leq 1-(1-p)^j \\ \Leftrightarrow\quad & (1-p)^j \leq 1-u < (1-p)^{j-1} \\ \Leftrightarrow\quad & j-1 < \frac{\log(1-u)}{\log(1-p)} \leq j \\ & \qquad (\text{SINCE } \log(1-p) < 0). \end{aligned}$$

Therefore, set

$$X = 1 + \left[\frac{\log_e(1-u)}{\log_e(1-p)}\right],$$

where $[a]$ denotes the smallest integer greater than or equal to a. □

Solution 2.5: Find the distribution of $Y = [X]$, where

$$f_X(x) = \lambda e^{-\lambda x} \quad x \geq 0.$$

$$\begin{aligned} \mathbb{P}(Y=i) &= \mathbb{P}([X]=i) \\ &= \mathbb{P}(i \leq X < i+1)) \\ &= \int_i^{i+1} \lambda e^{-\lambda x} dx \\ &= \left[-e^{-\lambda x}\right]_i^{i+1} \\ &= e^{-\lambda i}(1-e^{-\lambda}) \quad i = 0,1,... \end{aligned}$$

Consider $Z = Y + 1$, then

$$\mathbb{P}(Z=i) = e^{-\lambda(i-1)}(1-e^{-\lambda}) \quad i = 1,2,...$$

and so $Z \sim$ Geo $\left[1-e^{-\lambda}\right]$.

What is $\mathbb{P}(Y \leq i)$?

$$\begin{aligned} \mathbb{P}(Y \leq i) &= \mathbb{P}([X] \leq i) \\ &= \mathbb{P}(X < i+1) \\ &= \int_0^{i+1} \lambda e^{-\lambda x} dx \\ &= 1 - e^{-\lambda(i+1)}. \end{aligned}$$

Therefore, the algorithm would be

STEP 1. SELECT $u \sim U(0,1)$.
STEP 2. SET $Y = i$, IF

$$\begin{aligned} & 1-e^{-\lambda i} < u \leq 1-e^{-\lambda(i+1)} \\ \Leftrightarrow\quad & i < -1/\lambda \log_e(1-u) \leq i+1. \end{aligned}$$

THUS, WE SET

$$Y = \left[\frac{-\log_e(1-u)}{\lambda}\right],$$

WHERE $[a]$ DENOTES THE INTEGER PART OF a.

So Y is an exponential random variable, rounded down to the nearest integer and thus $Z = Y + 1$ is an exponential rounded up to the nearest integer. □

Solution 2.6: We need to find $f_{\boldsymbol{X}}(x_1, x_2)$.

$$X_1 = \sqrt{-2\log U_1}\cos U_2;$$

$$X_2 = \sqrt{-2\log U_1}\sin U_2.$$

Since $U_1 \sim U(0,1)$, $U_2 \sim U(0, 2\pi)$ and $U_1,\ U_2$ are independent,

$$f_{\boldsymbol{U}}(U_1, U_2) = \begin{cases} 1 \times \frac{1}{2\pi} & 0 < U_1 < 1,\ 0 < U_2 < 2\pi \\ 0 & \text{else.} \end{cases}$$

Now,

$$|J| = \begin{vmatrix} \frac{\partial u_1}{\partial x_1} & \frac{\partial u_1}{\partial x_2} \\ \frac{\partial u_2}{\partial x_1} & \frac{\partial u_2}{\partial x_2} \end{vmatrix},$$

$$\begin{aligned} X_1^2 + X_2^2 &= (-2\ln U_1 \cos^2 U_2 - 2\ln U_1 \sin^2 U_2) \\ &= -2\ln U_1, \end{aligned}$$

and

$$\frac{X_2}{X_1} = \tan U_2.$$

Therefore,

$$U_1 = \exp\left(\frac{1}{2}(X_1^2 + X_2^2)\right)$$

and

$$U_2 = \tan^{-1}(X_2/X_1).$$

Therefore,

$$\begin{aligned} \frac{\partial u_1}{\partial x_1} &= -x_1 e^{-\frac{1}{2}(x_1^2+x_2^2)} \\ \frac{\partial u_1}{\partial x_2} &= -x_2 e^{-\frac{1}{2}(x_1^2+x_2^2)} \\ \frac{\partial u_2}{\partial x_1} &= \frac{1}{1+(x_2/x_1)^2} \times -\frac{x_2}{x_1^2} \\ \text{and } \frac{\partial u_2}{\partial x_2} &= \frac{1}{1+(x_2/x_1)^2} \times \frac{1}{x_1}. \end{aligned}$$

Therefore,

$$\begin{aligned}
|J| &= -x_1 \exp\left(-\frac{1}{2}(x_1^2+x_2^2)\right)\frac{1}{x_1}\frac{1}{1+(x_2/x_1)^2} \\
&\quad - - x_2 \exp\left(-\frac{1}{2}(x_1^2+x_2^2)\right)\frac{-x_2}{x_1^2}\frac{1}{1+(x_2/x_1)^2} \\
&= -\exp\left(-\frac{1}{2}(x_1^2+x_2^2)\right)\frac{1}{1+(x_2/x_1)^2}\left[1+\frac{x_2^2}{x_1^2}\right] \\
&= -\exp\left(-\frac{1}{2}(x_1^2+x_2^2)\right).
\end{aligned}$$

Therefore,

$$\begin{aligned}
f_{\boldsymbol{X}}(x_1,x_2) &= \frac{1}{2\pi}\exp\left(-\frac{1}{2}(x_1^2+x_2^2)\right) \\
&= \left[\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}(x_1^2)\right)\right]\left[\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}(x_2^2)\right)\right],
\end{aligned}$$

for $-\infty < x_1,\ x_2 < \infty$. Thus, the joint density is simply the product of two independent standard normals. $\square$

Solutions to Exercises of Chapter 3

Solution 3.1: From notes, see Section 3.2.1, page 29. □

Solution 3.2:

$$\begin{aligned} I &= \int_0^a \frac{1}{1+x^2}\, dx. \\ &= a\int_0^a \frac{1}{1+x^2}\frac{1}{a}\, dx \\ &= a\int_{-\infty}^{\infty} f(x)p(x)dx \\ &= a\mathbb{E}_p\left(f(x)\right), \end{aligned}$$

where

$$f(x) = \frac{1}{1+x^2},$$

and

$$p(x) = \begin{cases} \frac{1}{a} & 0 \le x \le a \\ 0 & \text{else.} \end{cases}$$

That is $p(x) \sim u[0,a]$. Therefore if $u_i \sim u[0,a]$ then we can estimate I by

$$\hat{I} = \frac{a}{n}\sum_{i=1}^{n} \frac{1}{1+u_i^2}.$$

That is $a\overline{f}(u)$ where

$$\overline{f}(u) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{1+u_i^2}.$$

Note if $u_i \sim u[0,1]$ then

$$\hat{I} = \frac{a}{n}\sum_{i=1}^{n} \frac{1}{1+(au_i)^2}.$$

□

Solution 3.3: Each time we have N choices. If N_i is the number of acceptable choices given i distinct observations already chosen, then $N_i = N - i$. Therefore,

$$\begin{aligned} \mathbb{P}(\text{no repetitions}) &= \prod_{i=1}^{n}\frac{N_{i-1}}{N} \\ &= \prod_{i=1}^{n}\frac{N-(i-1)}{N} \\ &= \prod_{i=1}^{n}\left(1-\frac{i-1}{N}\right) \\ &= \prod_{j=0}^{n-1}\left(1-\frac{j}{N}\right). \end{aligned}$$

□

Solution 3.4: Let n_i be the number of times that $x_i \in (x_1, ..., x_n)$ appears in the bootstrap sample. Then, we are interested in picking the $n_i \in \{0, 1, .., n\}$ such that $\sum_{i=1}^{n} n_i = n$.

We could consider the case where r of the n_i are zero thereby splitting the problem up or we could set $n_i^{'} = n_i + 1$. In the latter case, $n_i^{'} \in \{1, .., n+1\}$ and

$$\sum_{i=1}^{n} n_i^{'} = \sum_{i=1}^{n} n_i + n = 2n.$$

Now picking the $n_i^{'}$ is equivalent to placing $(n-1)$ dividers between $2n$ objects and setting $n_i^{'}$ to be the number of objects between the i^{th} and $(i-1)^{th}$ divider. The number of ways of doing this is $\binom{2n-1}{n-1}$, arranging $n-1$ objects in $2n-1$ positions. However, $\binom{2n-1}{n-1} = \binom{2n-1}{n}$, hence the result.
If $n = 15$, $\begin{pmatrix} 2n-1 \\ n \end{pmatrix} = 77,558,760.$ □

Solution 3.5: The following could be used in each case.

a) $\bar{x}$.

b) $\frac{1}{n}\sum_{i=1}^{n} I_{(10,25)}(x_i)$ where $I_{(a,b)}(x) = \begin{cases} 1 & x \in (a,b) \\ 0 & \text{else.} \end{cases}$

c) $\displaystyle \frac{1}{n}\sum_{i=1}^{n} \sin(x_i)$.

d) $\displaystyle \frac{\bar{x}}{\hat{\sigma}^2}$ where $\hat{\sigma}^2 = \frac{1}{n-1}\sum_{i-1}^{n}(x_i - \bar{x})^2$. □

Solution 3.6:
i) If $\boldsymbol{x}^* = (x_1^*, ..., x_7^*)$ then $x_i^* \in (x_{(1)}^*, ..., x_{(7)}^*)$ since we are using the bootstrap. Similarly since $s(\boldsymbol{x}^*) = x_{(4)}^*$ and $x_{(4)}^* = x_i^*$ for some $i = 1, ..., 7$ then the result follows. □

ii)

$$\begin{aligned}
P(i) &= \mathbb{P}(s(\boldsymbol{x}^*) = x_{(i)}) \\
&= \mathbb{P}(x_{(4)}^* = x_{(i)}^*). \\
\text{Now } \ \mathbb{P}(x_{(k)}^* > x_{(i)}) &= \mathbb{P}((k-1) \text{ or fewer } x_i^* \leq x_{(i)}) \\
&= \sum_{r=0}^{k-1} \binom{n}{r} (i/n)^r \ (1 - i/n)^{n-r} \\
\text{and } \ \mathbb{P}(x_{(4)}^* = x_{(i)}) &= \mathbb{P}((x_{(4)}^* > x_{(i-1)}) - \mathbb{P}(x_{(4)}^* > x_{(i)}) \\
&= \sum_{r=0}^{3} \binom{n}{r} \left(\frac{i-1}{n}\right)^r \left(1 - \frac{i-1}{n}\right)^{n-r} \\
&\quad - \sum_{r=0}^{3} \binom{n}{r} (i/n)^r \ (1 - i/n)^{n-r} \\
&= \sum_{r=0}^{3} \{\text{Bin } (r : n, (i-1)/n) - \text{ Bin } (r : n, i/n).\}
\end{aligned}$$

The $p(i)$ values are

i	1	2	3	4	5	6	7
$p(i)$	0.0102	0.0981	0.2386	0.3062	0.2386	0.0981	0.0102

□

Solution 3.7:

$$\widehat{SE}_{\text{jack}}(\hat{\theta}) = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}_{(i)}) - \bar{\theta}_{(.)})^2}.$$

Here, $\hat{\theta} = \overline{x}$ and $\hat{\theta}_{(i)} = \frac{1}{n-1}\sum_{j\neq i} x_i$ with $\bar{\theta}_{(.)} = \bar{x}$. Therefore

$$\begin{aligned}
\widehat{SE}_{\text{jack}}(\hat{\theta}) &= \left(\frac{n-1}{n}\sum_{i=1}^{n}\left[\frac{1}{n-1}\sum_{j\neq i} x_j - \bar{x}\right]^2\right)^{\frac{1}{2}} \\
&= \left(\frac{n-1}{n}\sum_{i=1}^{n}\left[n\sum_{j\neq i}^{n} x_j - (n-1)\sum_{j=i} x_j\right]^2 \frac{1}{(n(n-1))^2}\right)^{\frac{1}{2}} \\
&= \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{n} x_j - nx_i\right)^2 . \frac{1}{n^2(n-1)^2}} \\
&= \sqrt{\frac{1}{n(n-1)}\sum_{i=1}^{n}(\bar{x} - x_i)^2}, \quad \text{as required .}
\end{aligned}$$

Note: we could also show that the bias is zero.

□

Solution 3.8:

$$
\begin{aligned}
\widehat{SE}_{\text{boot}}(\hat{\theta}) &= \sqrt{\operatorname{Var}_{\hat{F}}(\hat{\theta}(\boldsymbol{x}^*))} \\
&= \sqrt{\operatorname{Var}_{\hat{F}}\left(\frac{1}{n}\sum_{i=1}^{n}(x_i^*)\right)} \\
&= \sqrt{\frac{1}{n}\operatorname{Var}_{\hat{F}}(x_i^*)} \\
&= \sqrt{\frac{1}{n}\mathbb{E}_{\hat{F}}(x_i^* - \bar{x})^2} \\
&\quad \text{since } \mathbb{E}_{\hat{F}}(x_i^*) = \bar{x} \\
&= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\mathbb{P}(x_i^* = x_i)} \\
&= \sqrt{\frac{1}{n^2}\sum_{i=1}^{n}(x_i - \bar{x})^2} \\
\text{Therefore } \widehat{SE}_{\text{boot}}(\hat{\theta}) &= \sqrt{\frac{n-1}{n}}\widehat{SE}_{\text{jack}}(\hat{\theta}).
\end{aligned}
$$

Clearly as $n \to \infty$, $\widehat{SE}_{\text{boot}} \simeq \widehat{SE}_{\text{jack}} \simeq SE_{\hat{\theta}}$.

□

Solution 3.9:

$$
\begin{aligned}
\mathbb{P}(T \leq x) &= \mathbb{P}\left(n(\theta - x_{(n)})/\theta \leq x\right) \\
&= \mathbb{P}\left(\theta - x_{(n)} \leq \frac{\theta x}{n}\right) \\
&= \mathbb{P}\left(x_{(n)} \geq \theta(1 - x/n)\right) \\
&= 1 - \mathbb{P}\left(\frac{x_{(n)}}{\theta} < (1 - x/n)\right). \qquad \text{(B.2)}
\end{aligned}
$$

Now $X_1, ..., X_n \sim$ *iid* $U[0,\theta]$ therefore

$$Z_i = \frac{X_i}{\theta} \sim U[0,1],$$

and

$$Z_{(n)} = \frac{X_{(n)}}{\theta}.$$

Therefore

$$
\begin{aligned}
\text{(B.2)} &= 1 - \mathbb{P}(z_{(n)} < 1 - x/n) \\
&= 1 - (\mathbb{P}(z_i < 1 - x/n))^n, \text{ since the } z_i \text{ are } iid\ U(0,1) \\
&= 1 - (1 - x/n)^n) \\
&\to 1 - e^{-x} \text{ as } n \to \infty.
\end{aligned}
$$

Therefore $\quad T \sim \exp(1)$ for large n.

□

The plug-in principle says that $\hat{\theta} = X_{(n)}$. Therefore,

$$\begin{aligned} \theta &\longrightarrow X_{(n)} \\ X_n &\longrightarrow Y_{(n)} \end{aligned}$$

and

$$T(Y, \hat{\theta}) = n\, \frac{X_{(n)} - Y_n}{X_{(n)}}.$$

$$\begin{aligned} \mathbb{P}(T(Y,\hat{\theta}) = 0) &= \mathbb{P}(X_{(n)} = Y_{(n)}) \\ &= \mathbb{P}(\text{at least one } Y_i = X_n) \\ &= 1 - \mathbb{P}(\text{No } Y_i's = X_{(n)}) \\ &= 1 - (1 - 1/n)^n \\ &\longrightarrow 1 - e^{-1} \text{ as } n \to \infty. \end{aligned}$$

□

The bootstrap fails to work for this functional, since the bootstrap distribution does not converge to the true distribution, i.e., the plug-in principle does not hold.
Note: If we bootstrap by drawing samples of size $m < n$, the method works again!

Solutions to Exercises of Chapter 4

Solution 4.1: $p(\theta_1) = \pi_1$.

$$
\begin{aligned}
\mathbb{P}(X_0 = \theta_1 | X_n = \theta_1) &\propto \pi_1 \ \mathbb{P}(X_n = \theta_1 | X_0 = \theta_1). \\
&= \frac{\pi_1 \ \mathbb{P}(X_n = \theta_1 | X_0 = \theta_1)}{\pi_1 \ \mathbb{P}(X_n = \theta_1 | X_0 = \theta_1) + (1-\pi_1)\mathbb{P}(X_n = \theta_1 | X_0 = \theta_2)}.
\end{aligned}
$$

Now

$$
\mathbb{P}(X_n = \theta_i | X_0 = \theta_j) = \begin{pmatrix} a_0 & a_1 \\ a_0 & a_1 \end{pmatrix} + \frac{(\alpha+\beta-1)^n}{2-\alpha-\beta} \begin{pmatrix} 1-\alpha & \alpha-1 \\ \beta-1 & 1-\beta \end{pmatrix}.
$$

where $a_0 = \frac{1-\beta}{2-\alpha-\beta}$ and $a_1 = \frac{1-\alpha}{2-\alpha-\beta}$.

Note that $\boldsymbol{a}$ satisfies $\boldsymbol{a}P = \boldsymbol{a}$ and $\alpha+\beta-1$ is the second largest eigenvalue of P, see Markov Chain notes!

So,

$$
\begin{aligned}
\mathbb{P}(X_n = \theta_1 | X_0 = \theta_1) &= \frac{1-\beta}{2-\alpha-\beta} + \frac{(\alpha+\beta-1)^n}{2-\alpha-\beta}(1-\alpha). \\
\text{and} \quad \mathbb{P}(X_n = \theta_1 | X_0 = \theta_2) &= \frac{1-\beta}{2-\alpha-\beta} + \frac{(\alpha+\beta-1)^n}{2-\alpha-\beta}(1-\beta).
\end{aligned}
$$

So the constant of proportionality is given by

$$
C = [\pi_1 \, (1-\beta+(\alpha+\beta-1)^n(1-\alpha)) + (1-\pi_1)(1-\beta+(\alpha+\beta-1)^n)] \ / \ (2-\alpha-\beta)
$$

and

$$
\mathbb{P}(X_0 = \theta_1 | X_n = \theta_1) = \frac{\pi_1(1-\beta+(1-\alpha)(\alpha+\beta-1)^n)}{\pi_1(1-\beta+(\alpha+\beta-1)^n(1-\alpha)) + (1-\pi_1)(1-\beta+(\alpha+\beta-1)^n)}
$$

Finally, $\alpha, \ \beta < 1$. Therefore $\alpha+\beta-1 < 1$ and hence $(\alpha+\beta-1)^n \to 0$ as $n \to \infty$. Hence, as $n \to \infty$

$$
\begin{aligned}
\mathbb{P}(X_0 = \theta_1 | X_n = \theta_1) &\to \frac{\pi_1(1-\beta)}{\pi_1(1-\beta) + (1-\pi_1)(1-\beta)} \\
&= \pi_1.
\end{aligned}
$$

□

Solution 4.2: ¿From Exercise 4.1 $a_0 \equiv \pi_1$ $\quad a_1 \equiv \pi_2$ $\quad$ Therefore

$$
\begin{aligned}
\pi_1 &= \frac{1-\beta}{2-\alpha-\beta} = \frac{0.6}{1.4} = \frac{3}{7} \\
\pi_2 &= \frac{1-\alpha}{2-\alpha-\beta} = \frac{0.8}{1.4} = \frac{4}{7}.
\end{aligned}
$$

□

Solution 4.3:

$$
\alpha(x,y) = \min\left(1, \ \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}\right).
$$

Here

$$\pi(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\ e^{-\frac{1}{2}x^2},$$

and

$$q(x,y) = \frac{1}{\sqrt{2\pi\tau^2}}\ \exp\left(-\frac{1}{2\tau^2}(y-ax)^2\right).$$

Therefore

$$\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)} = \exp\left(-\frac{1}{2}(y^2-x^2)\left(\frac{1}{\sigma^2}+\frac{a^2-1}{\tau^2}\right)\right)$$

This sampler never rejects if $\alpha(x,y) \equiv 1\ \forall x,y$ and $\alpha(x,y) \equiv 1$ iff $\frac{1}{\sigma^2}+\frac{a^2-1}{\tau^2} = 0$, i.e. $\tau^2 = \sigma^2(1-a^2)$. □

Solution 4.4:

i) $\mathcal{K}_H(x,y) = q(x,y)\alpha(x,y)$. Therefore the chain is reversible if

$$\pi(x)q(x,y)\alpha(x,y) = \pi(y)q(y,x)\alpha(y,x).$$

Take two cases.

Case 1. $y \neq x$.

$$\begin{aligned}
\pi(x)\mathcal{K}_H(x,y) &= \pi(x)q(x,y)\min\left(1,\ \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}\right) \\
&= \min\left(\pi(x),\ q(x,y),\ \pi(y)\ q(y,x)\right) \\
&= \min\left(\frac{\pi(x)q(x,y)}{\pi(y)q(y,x)},1\right)\pi(y)q(y,x) \\
&= \pi(y)\mathcal{K}_H(y,x).
\end{aligned}$$

Case 2. $y = x$. Trivial. □

ii)

$$\begin{aligned}
\int \pi(x)\mathcal{K}(x,y)dx &= \int \pi(y)\mathcal{K}_H(y,x)dx \quad \text{since reversible.} \\
&= \pi(y)\int K_H(y,x)dx \\
&= \pi(y).
\end{aligned}$$

□

Solution 4.5:

$$\begin{aligned}
\pi(x_1|x_2) &\propto \pi(x_1,x_2) \\
&\propto \exp\left(-\frac{1}{2}(x_1-1)^2(x_2-2)^2\right) \\
&\sim N\left(1,\ \frac{1}{(x_2-2)^2}\right).
\end{aligned}$$

Similarly,

$$\pi(x_2|x_1) \;\sim\; N\left(2,\; \frac{1}{(x_1-1)^2}\right).$$

So both full conditionals are normal and any method from Chapter 1 can be used to sample from them. Note, $\pi(x_1, x_2)$ is improper, i.e.

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \pi(x_1,x_2)dx_1dx_2 = \infty.$$

Thus, π is not a density and so using the Gibbs sampler makes no sense for this problem. □

Solution 4.6: See Section 4.5.3.

Solution 4.7: Using Bayes' theorem, the posterior distribution of the parameters is given by,

$$\begin{aligned}
\pi(\boldsymbol{a},\sigma^2|\boldsymbol{x}) &\propto L(\boldsymbol{x}|\boldsymbol{a},\sigma^2)p(\boldsymbol{a})p(\sigma^2)\\
&\propto \frac{1}{\sigma^{N-k}}\exp\left(-\frac{1}{2}(\boldsymbol{x}_k - X_k\boldsymbol{a})^T(\sigma^2 I)^{-1}(\boldsymbol{x}_k - X_k\boldsymbol{a})\right)\exp\left(-\frac{1}{2}(\boldsymbol{a}-\boldsymbol{\mu}_a)^T\Sigma_a^{-1}(\boldsymbol{a}-\boldsymbol{\mu}_a)\right)\\
&\quad \times(\sigma^2)^{-(\alpha+1)}\exp(-\beta\sigma^{-2}).
\end{aligned}$$

Then, we have the posterior conditional distribution of σ^2 to be of the form,

$$\begin{aligned}
\pi(\sigma^2|\boldsymbol{x},\boldsymbol{a}) &\propto (\sigma^2)^{-(\alpha+1)}\exp(-\beta\sigma^{-2})(\sigma^2)^{-\frac{N}{2}}\exp\left(-\frac{\sigma^{-2}}{2}(\boldsymbol{x}_k - X_k\boldsymbol{a})^T(\boldsymbol{x}_k - X_k\boldsymbol{a})\right)\\
&= (\sigma^2)^{-(\alpha+N/2+1)}\exp\left(-\sigma^{-2}\left(\beta+\frac{1}{2}(\boldsymbol{x}_k - X_k\boldsymbol{a})^T(\boldsymbol{x}_k - X_k\boldsymbol{a})\right)\right).
\end{aligned}$$

Thus, the posterior conditional distribution is of standard form, namely,

$$\sigma^2|\boldsymbol{x},\boldsymbol{a} \sim \Gamma^{-1}\left(\alpha+\frac{(N-k)}{2}, \beta+\frac{1}{2}(\boldsymbol{x}_k - X_k\boldsymbol{a})^T(\boldsymbol{x}_k - X_k\boldsymbol{a})\right).$$

Similarly,

$$\begin{aligned}
\pi(\boldsymbol{a}|\boldsymbol{x},\sigma^2) &\propto \exp\left(-\frac{1}{2}(\boldsymbol{x}_k - X_k\boldsymbol{a})^T(\sigma^2 I)^{-1}(\boldsymbol{x}_k - X_k\boldsymbol{a})\right)\exp\left(-\frac{1}{2}(\boldsymbol{a}-\boldsymbol{\mu}_a)^T\Sigma_a^{-1}(\boldsymbol{a}-\boldsymbol{\mu}_a)\right)\\
&\propto \exp\left(-\frac{1}{2}(\sigma^{-2}(-\boldsymbol{x}_k^T X_k\boldsymbol{a} - \boldsymbol{a}^T X_k^T\boldsymbol{x}_k + \boldsymbol{a}^T X_k^T X_k\boldsymbol{a}) + (\boldsymbol{a}^T\Sigma_a^{-1}\boldsymbol{a} - \boldsymbol{a}^T\Sigma_a^{-1}\boldsymbol{\mu}_a - \boldsymbol{\mu}_a^T\Sigma_a^{-1}\boldsymbol{a})\right)\\
&= \exp\left(-\frac{1}{2}(\boldsymbol{a}^T(\sigma^{-2}X_k^T X_k + \Sigma_a^{-1})\boldsymbol{a} - \boldsymbol{a}^T(\sigma^{-2}X_k\boldsymbol{x}_k^T + \Sigma_a^{-1}\boldsymbol{\mu}_a) - (\sigma^{-2}\boldsymbol{x}_k^T X_k + \boldsymbol{\mu}_a^T\Sigma_a^{-1})\boldsymbol{a})\right)\\
&\propto \exp\left(-\frac{1}{2}(\boldsymbol{a}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{a}-\boldsymbol{\mu})\right),
\end{aligned}$$

where,

$$\Sigma^{-1} = \sigma^{-2}X_k^T X_k + \Sigma_a^{-1}; \quad \boldsymbol{\mu} = \sigma^{-2}\Sigma X_k^T\boldsymbol{x}_k + \Sigma\Sigma_a^{-1}\boldsymbol{\mu}_a.$$

So that,

$$\boldsymbol{a}|\boldsymbol{x},\sigma^2 \;\sim\; N(\boldsymbol{\mu},\Sigma),$$

for $\boldsymbol{\mu}$ and Σ given above. □

Solution 4.8: Suppose that we propose to update the parameters in the model, within the reversible jump setup. We can do this by proposing to move from $(\boldsymbol{\theta}, \boldsymbol{\theta}')$ to $(\boldsymbol{\theta}', \boldsymbol{\theta})$, i.e. $g(\boldsymbol{\theta}, \boldsymbol{\theta}') = (\boldsymbol{\theta}', \boldsymbol{\theta})$. Then, using standard notation, the acceptance probability is $\min(1, A)$, where

$$A = \frac{\pi(\boldsymbol{\theta}', m'|\boldsymbol{x})P(m|m')q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta}, m|\boldsymbol{x})P(m'|m)q(\boldsymbol{\theta}')}\left|\frac{\partial(g(\boldsymbol{\theta}, \boldsymbol{\theta}'))}{\partial(\boldsymbol{\theta}, \boldsymbol{\theta}')}\right|.$$

The proposed model, m', is the same as the current model (i.e. $m' = m$), since we are only updating the parameters. Thus, we can simplify the expression, with,

$$P(m|m') = P(m'|m);$$

and

$$\pi(\boldsymbol{\theta}', m'|\boldsymbol{x}) \propto f_m(\boldsymbol{x}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')p(m),$$

since $m' = m$, and similarly for $\pi(\boldsymbol{\theta}, m|\boldsymbol{x})$. Substituting these expressions into the acceptance probability above, and following simplification, we obtain,

$$A = \frac{\pi(\boldsymbol{\theta}'|\boldsymbol{x})q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta}|\boldsymbol{x})q(\boldsymbol{\theta}')}\left|\frac{\partial g(\boldsymbol{\theta}, \boldsymbol{\theta}')}{\partial(\boldsymbol{\theta}, \boldsymbol{\theta}')}\right|,$$

removing the dependence on the model m, since the updating of the parameters is conditional on the model. So, comparing this expression with the acceptance probability for the Metropolis-Hastings algorithm, it is clear that we only need show that the Jacobian term is equal to unity. Suppose that we propose to move from $\boldsymbol{\theta}$ to $\boldsymbol{\theta}'$. Then, we can model this in the reversible jump procedure of moving from $(\boldsymbol{\theta}, \boldsymbol{\theta}')$ to $(\boldsymbol{\theta}', \boldsymbol{\theta})$, i.e. $g(\boldsymbol{\theta}, \boldsymbol{\theta}') = (\boldsymbol{\theta}', \boldsymbol{\theta})$. Then, the corresponding Jacobian term is equal to,

$$\begin{vmatrix} \frac{\partial \boldsymbol{\theta}'}{\partial \boldsymbol{\theta}} & \frac{\partial \boldsymbol{\theta}'}{\partial \boldsymbol{\theta}'} \\ \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\theta}} & \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\theta}'} \end{vmatrix} = \begin{vmatrix} 0 & I \\ I & 0 \end{vmatrix} = 1.$$

□

Solution 4.9: Since there are only two possible models within the reversible jump step we always propose to move to the alternative model. Suppose that we propose to move from the $Bin(n, p)$ distribution (model m) to the $Beta - Bin(n, \alpha, \beta)$ (model m'). Then, we propose the parameters,

$$\alpha = u; \quad \beta = \frac{u(1-p)}{p},$$

where,

$$u \sim Exp(\lambda),$$

for some λ. Then, the mean value of the current and proposed distributions (or models) are equal. The Jacobian term of such a move is equal to,

$$\begin{aligned} |J| = \begin{vmatrix} \frac{\partial \alpha}{\partial p} & \frac{\partial \alpha}{\partial u} \\ \frac{\partial \beta}{\partial p} & \frac{\partial \beta}{\partial u} \end{vmatrix} &= \begin{vmatrix} 0 & 1 \\ -\frac{u}{p^2} & \frac{1-p}{p} \end{vmatrix} \\ &= \frac{u}{p^2}. \end{aligned}$$

The corresponding acceptance probability is then $\min(1, A)$, where,

$$\begin{aligned} A &= \frac{L_{m'}(\boldsymbol{x}|\alpha, \beta)p(\alpha)p(\beta)p(m')}{L_m(\boldsymbol{x}|p)p(p)p(m)q(u)} \times |J| \\ &= \frac{\left(\prod_{i=1}^N \frac{\Gamma(\alpha+x_i)\Gamma(n+\beta-x_i)}{\Gamma(\alpha+\beta+n)} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\right) p(\alpha)p(\beta)p(m')}{\prod_{i=1}^N p^{x_i}(1-p)^{n-x_i} p(p)\lambda \exp(-\lambda u)} \times \frac{u}{p^2}. \end{aligned}$$

Similarly, in the reverse move, we set

$$p = \frac{\alpha}{\alpha + \beta},$$

and accept the move with probability $\min(1, A^{-1})$, where A is given above. $\square$

Solution 4.10: (a) We can calculate the posterior distribution of the parameters, using Bayes' theorem. We obtain,

$$\pi(\boldsymbol{\theta}|\boldsymbol{x}) \propto \prod_{i=1}^{2}\prod_{j=1}^{2} p_{ij}^{n_{ij}} \exp\left(-\frac{(\theta^{\phi}-\mu^{\phi})^2 + (\theta_1^A - \mu^A)^2 + (\theta_1^B - \mu^B)^2}{2\sigma^2}\right),$$

where, n_{ij} denotes the number of individuals observed in cell (i,j) and,

$$p_{ij} = \frac{\tilde{p}_{ij}}{\sum_{ij}\tilde{p}_{ij}}.$$

Then, the posterior conditional distribution for θ_1^A is equal to,

$$\begin{aligned} \pi(\theta_1^A|\boldsymbol{x}, \theta^{\phi}, \theta_1^B) &\propto \prod_{ij} p_{ij}^{n_{ij}} \exp\left(-\frac{(\theta_1^A - \mu^A)^2}{2\sigma^2}\right) \\ &\propto \left(\frac{\exp(\theta_1^A)}{\sum_{ij}\tilde{p}_{ij}}\right)^{n_{11}+n_{12}-n_{21}-n_{22}} \exp\left(-\frac{(\theta_1^A - \mu^A)^2}{2\sigma^2}\right). \end{aligned}$$

Similar results hold for the other log-linear parameters. Then, we can simulate observations from the posterior distribution, by using a Metropolis-Hastings algorithm, and estimate posterior summary statistics of interest. $\square$

(b) We are able to construct a Markov chain, with stationary distribution equal to the joint posterior distribution over parameter by updating the parameters within the model, using a Metropolis-Hastings algorithm, and then the model, using the reversible jump procedure, within each iteration of the chain. Given the model, we update the parameters using the procedure outlined above in (a), with the analogous procedure used to update the parameter θ_{11}^{AB}, when it is in the model. Then, to update the model itself, we use the following procedure.

Suppose that we are in the independent model, m_{ind} which does not contain the interaction term between sources A and B, and that we propose to move to the model, m_{AB} with the additional interaction term between A and B. Then, we simulate,

$$\theta_{11}^{'AB} \sim N(\mu_{AB}, \sigma_{AB}^2),$$

and set,

$$\theta^{'\phi} = \theta^{\phi}; \quad \theta_1^{'A} = \theta_1^A; \quad \theta_1^{'B} = \theta_1^B.$$

The corresponding Jacobian term for such a move is simply equal to unity. Then, the acceptance probability for such a move is $\min(1, A)$, where,

$$\begin{aligned} A &= \frac{f_{m_{AB}}(\boldsymbol{x}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')p(m_{AB})}{f_{m_{ind}}(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})p(m_{ind})q(\theta_{11}^{'AB})} \\ &= \frac{\prod_{i=1}^2\prod_{j=1}^2 p_{ij}^{'n_{ij}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\theta_{11}^{'AB}-\mu)^2}{2\sigma^2}\right) p(m_{AB})}{\prod_{i=1}^2\prod_{j=1}^2 p_{ij}^{n_{ij}} p(m_{ind}) \frac{1}{\sqrt{2\pi\sigma_{AB}^2}} \exp\left(-\frac{(\theta_{11}^{'AB}-\mu_{AB})^2}{2\sigma_{AB}^2}\right)}, \end{aligned}$$

where p'_{ij} are the cell probabilities, given the proposed set of parameters $\boldsymbol{\theta}'$; and the terms $P(m_{ind}|m_{AB})$ and $P(m_{AB}|m_{ind})$, are equal and cancel in the expression, since we always propose to move to the alternative model.

Then, in the reverse move, we propose to remove the log-linear term relating to the interaction between sources A and B, and once more set,

$$\theta'^{\phi} = \theta^{\phi}; \quad \theta_1'^{A} = \theta_1^{A}; \quad \theta_1'^{B} = \theta_1^{B}.$$

This move is accepted with probability $\min(1, A^{-1})$, where A is given above. □

Solution 4.11: The posterior distribution of λ is,

$$\begin{aligned} \pi(\lambda|\boldsymbol{x}) &\propto \prod_{i=1}^{N} \lambda \exp(-\lambda x_i) \times \lambda^{a-1} \exp(-b\lambda) \\ &= \lambda^{N+a-1} \exp\left(-\lambda \left(\sum_{i=1}^{N} x_i + b\right)\right). \end{aligned}$$

So that,

$$\lambda|\boldsymbol{x} \sim \Gamma\left(N + a, \sum_{i=1}^{N} x_i + b\right).$$

For the $Pareto(\alpha, \beta)$ distribution, we obtain the posterior distribution of the parameters to be,

$$\begin{aligned} \pi(\alpha, \beta|\boldsymbol{x}) &\propto \prod_{i=1}^{N} \beta\alpha^{\beta} x_i^{-(\beta+1)} \times m \exp(-\alpha m) n \exp(-\beta n) \\ &\propto \beta^N \alpha^{N\beta} \prod_{i=1}^{N} x_i^{-(\beta+1)} \times \exp(-\alpha m - \beta n). \end{aligned}$$

Thus, we have that,

$$\begin{aligned} \pi(\alpha|\boldsymbol{x}, \beta) &\propto \alpha^{N\beta} \exp(-\alpha), && \text{such that } \alpha \leq \min x_i; \\ \pi(\beta|\boldsymbol{x}, \alpha) &\propto \beta^N \exp(-\beta) \prod_{i=1}^{N} x_i^{-(\beta+1)}. \end{aligned}$$

Since the posterior condition distribution of the parameters are of non-standard form, we use a Metropolis-Hastings step to update the parameters within each iteration of the Markov chain. Suppose that we propose to update α. Then, we could use the procedure of proposing a new value,

$$\alpha' = \alpha + u,$$

where, for some fixed σ^2,

$$u \sim N(0, \sigma^2).$$

Since we need to retain $0 < \alpha \leq x_i$, for each x_i, we automatically reject the proposal if the proposed parameter $\alpha' < 0$ or $\alpha' > \min(\boldsymbol{x})$, where $\min(\boldsymbol{x}) = \min(x_1, ..., x_N)$. Assuming that the proposed parameter is not automatically rejected, we accept the proposed parameter with probability, $\min(1, A)$, where

$$\begin{aligned} A &= \frac{\pi(\alpha'|\beta, \boldsymbol{x})}{\pi(\alpha|\beta, \boldsymbol{x})} && \text{since the proposal is symmetrical} \\ &= \frac{\alpha'^{N\beta} \exp(-\alpha' m)}{\alpha^{N\beta} \exp(-\alpha m)}. \end{aligned}$$

else the parameter remains at the same value. We use an analogous procedure for updating β.

We can construct a Markov chain, with this stationary distribution, using the following procedure. Within each step of the Markov chain we update each of the parameters using the Gibbs sampler, if we are in the $Exp(\lambda)$ model, else we use a Metropolis-Hastings step for the parameters of the $Pareto(\alpha, \beta)$ distribution. Then, once the chain has been run long enough, realisations of the chain can be regarded as a sample from the posterior distribution of interest.

To calculate the posterior probability of the two models, we construct a Markov chain that can move between different models, with the stationary distribution equal to the joint posterior distribution over the parameters and models. Then, we can estimate the posterior model probabilities as the proportion of times that the chain is in the corresponding models. Thus, we need to construct a Markov chain with the correct stationary distribution. Within each step of the chain we propose to update the parameters in the model using the MCMC procedures detailed above. We then propose to update the model, using a reversible jump algorithm, since this involves a change in dimension.

Suppose that we are in the $Exp(\lambda)$ model, and that we propose to move to the $Pareto(\alpha, \beta)$ model. Then we simulate,

$$u \sim Exp(\mu),$$

for some μ, and set,

$$\alpha = \frac{u-1}{\lambda u}; \quad \beta = u.$$

Thus, the expected value under the proposed $Pareto(\alpha, \beta)$ model is equal to the expected value under the current $Exp(\lambda)$ model.

The corresponding Jacobian term for such a move is equal to,

$$|J| = \begin{vmatrix} -\frac{\lambda}{(u-1)^2} & 1 \\ \frac{u}{u-1} & 0 \end{vmatrix} = \frac{u}{u-1}.$$

To retain the condition that $\alpha < x_i$ for each i, we automatically reject the move if $\alpha > \min(\boldsymbol{x})$. Else, if this condition is satisfied, the acceptance probability of such a move is equal to $\min(1, A)$, where,

$$A = \frac{\prod_{i=1}^N \beta \alpha^\beta x_i^{-(\beta+1)} p(\alpha) p(\beta)}{\prod_{i=1}^N \lambda \exp(-\lambda x_i) p(\lambda) q(u)} \times \frac{u}{u-1}.$$

If the move is not accepted the chain remains in the same model in the iteration of the Markov chain. Conversely, in the reverse move, we set,

$$\lambda = \frac{\beta - 1}{\beta \alpha},$$

and accept the move with probability $\min(1, A^{-1})$, where A is given above. □

Solutions to Exercises of Chapter 5

Solution 5.1: Bookwork. Conditional distributions are derived by ignoring all terms in b_T which don't involve the relevant parameter. By equating the result to the standard form of the relevant distribution, the results should be obtained. □

Solution 5.2:

$$f(x,y) = x^2 + 2y^3 - 0.3\cos(3\pi x) - 0.4\cos(4\pi y) + 0.7.$$

a) Update x first and then y. We can simulate $x' \sim U[-1,1]$ and $u \sim U[0,1]$. We then accept $x_{t+1} = x'$ if
$$u \le \left(\frac{f(x',y)}{f(x,y)}\right)^{-1/T},$$
else $x_{t+1} = x_t$. Similarly for y. (NB. Here, we are minimising $\log f$ rather than f, for computational simplicity.)

b) As before, update x first and then y. We can simulate $x' \sim U[\max(-1, x-\epsilon), \min(1, x+\epsilon)]$ and $u \sim U[0,1]$. We then accept $x_{t+1} = x'$ if
$$u \le \frac{f(x',y)^{-1/T}[\min(1,x'+\epsilon) - \max(-1,x'-\epsilon)]}{f(x,y)^{-1/T}[\min(1,x+\epsilon) - \max(-1,x-\epsilon)]},$$
else $x_{t+1} = x_t$. Similarly for y.

The random walk update is most likely to be better as it produces more local moves, this is likely to be most important for lower temperatures. However, the value of ϵ will require some tuning. Note also that if $\epsilon = 1$, then the two algorithms are the same. The Gibbs sampler would be difficult to implement here, the conditionals far from standard! Of course, we could use rejection sampling, taking $g(x) = U[-1,1]$ for example. □

Solution 5.3: The MLE's are standard bookwork, differentiate the likelihood and set to zero to obtain

$$\begin{aligned}
\widehat{\lambda} &= \frac{1}{n}\sum x_t = \bar{x} \\
\widehat{\mu} &= \bar{x} \\
\widehat{\sigma}^2 &= \frac{1}{n}\sum (x_t - \bar{x})^2.
\end{aligned}$$

The likelihood under the Normal model is given by

$$L(\mu, \sigma^2|\boldsymbol{x}) \propto \sigma^{-n} \exp\left(-\frac{1}{2\sigma^2}\sum (x_t - \mu)^2\right).$$

If $b_T(\mu, \sigma^2|\boldsymbol{x}) \propto \exp(\log L(\mu, \sigma^2|\boldsymbol{x})/T)$, then

$$\begin{aligned}
b_T(\mu|\sigma^2, \boldsymbol{x}) &= N(\bar{x}, \sigma^2 T/n) \\
\text{and } b_T(\sigma^2|\mu, \boldsymbol{x}) &= \Gamma^{-1}(1 + n/2T, \textstyle\sum (x_t - \mu)^2/2T).
\end{aligned}$$

These conditionals would be used to update the parameters. Clearly, as $T \to 0$, the means of these conditionals tend to the corresponding MLE's and the variances tend to zero.

The AIC is defined as $-2\log L_k + 2n_k$. Therefore, under the Poisson model, we have an AIC of

$$2n\lambda - 2\sum x_t \log\lambda - \log\prod(1/x_t!) + 2$$

and for the Normal model, we have an AIC of

$$n\log(2\pi\sigma) - \frac{1}{2}\sum(x_t - \mu)^2 + 4.$$

For the reversible jump move, we consider moving from the Poisson to the Normal model. Since λ is the mean (and also the variance) under the Poisson model, we could set $\mu = \lambda$ and generate a completely new value for σ^2 from some distribution q. Alternatively, we could set $\sigma^2 = \lambda$ and generate a new value of μ from q. Perhaps the latter is easiest and we could perhaps take q to be a normal with mean $\bar{x}$ and variance equal to 1, say. Note it is permissible to have proposals depend on the data. Note also that the proposal variance may require tuning. Using this map, the Jacobian term is simply 1 and, since there are only two models, we always propose moving to the other one, so that these terms cancel out disappear from the acceptance ratio which simplifies to:

$$A = \frac{b_T(\mu, \sigma^2)}{b_T(\lambda)q(\mu)}.$$

$\square$

REFERENCES

Barnard, G. (1963), Contribution to the Discussion of Bartlett's paper. *Journal of the Royal Statistical Society, Series B.* **25**, 294

Efron, B. and R. Tibshirani (1986), Bootstrap Methods for Standard Errors, Confidence Intervals, and other Measures of Statistical Accuracy. (With discussion). *Statistical Science* **1**, 54–77

Fieller, E. C. and H. O. Hartley (1954), Sampling with Control Variables. *Biometrika* **12**, 494–501

Jockel, K. H. (1986), Finite Sample Properties and Asymptotic Efficiency of Monte Carlo Tests. *Annals of Statistics* **14**, 336–347

Marriott, F. H. C. (1979), Barnard's Monte Carlo Tests: How Many Simulations? *Applied Statistics* **28**, 75–77

INDEX