Computer Intensive Methods

Omiros Papaspiliopoulos

2005

Contents

Chapter 1

Introduction

1.1 Scope of course

The term 'computer intensive methods' means different things to different people. It is also a dynamic subject: what requires intensive computing today may be solvable with a pocket calculator tomorrow. Not so long ago, the calculation of normal probabilities to reasonable accuracy would have required considerable CPU time. An initial classification of computer intensive methods as applied to statistics is the following:

1. Computers for graphical data exploration.
2. Computers for data modelling.
3. Computers for inference.

There is obviously some overlap in these three, but in this course I intend to focus mostly on the third of the above. That is, we shall aim at understanding computer techniques which require innovative algorithms to apply standard inferences. However, especially for Bayesian inference, and its associated numerical technique, Markov chain Monte Carlo (MCMC), 2 and 3 have become so closely linked that it makes sense to consider the two together. MCMC is not covered in this course, since its importance has let to it having a course of its own, Math457. On the other hand, some techniques used for computationally intensive maximum likelihood calculations such as the bootstrap (Chapter 4) are intrinsically non-parametric, and as such the numerical techniques can be treated totally in isolation of any modelling considerations. I'll exclude material dealing explicitly with modern regression approaches, such as kernel regression, lowess, etc. (although an illustration of bootstrap on kernel regression is illustrated in Section 3.6.1).

I see two roles of this type of course. The first is to gain some understanding and knowledge of the techniques and tools which are available (so long as you've got the computing power). The second is that many of the techniques (if not all of them) are themselves clever applications or interpretations of the interplay between probability and statistics. So, understanding the principles behind the different algorithms can often lead to a better understanding of inference, probability and statistics generally. In short, the techniques are not just a means to an end. They have their own intrinsic value as statistical exercises.

This is not a course on computing itself. We won't get into the details of programming itself. Furthermore, this is not a course which will deal with specialised statistical packages often used in statistical computing. All the examples will be handled using simple R functions - far from the most efficient way of implementing the various techniques. It is important to recognise that high-dimensional complex problems do require more efficient programming (commonly in C or FORTRAN). However the emphasis of this course is to illustrate the various methods and their application on relatively simple examples. A basic familiarity with R will be assumed.

1.2 Computers as inference machines

It is something of cliché to point out that computers have revolutionized all aspects of statistics. In the context of inference there have really been two substantial impacts: the first has been the freedom to make inferences without the catalogue of arbitrary (and often blatantly inappropriate) assumptions which standard techniques necessitate in order to obtain analytic solutions — Normality, linearity, independence etc. The second is the ability to apply standard type models to situations of greater data complexity — missing data, censored data, latent variable structures.

1.3 References

The notes have been based on the following books

- *Stochastic simulation*, B. Ripley.
- *Bootstrap methods and their application*, A.C. Davinson and D.V. Hinkley
- *Tools for statistical inference*, M. Tanner.

A very interesting recent book which covers some of the issues raised in these notes, is Liu, 2001, Monte Carlo Strategies in Scientific Computing. By sticking closely to specific texts it should be easy to follow up any techniques that you want to look at in greater depth. Within this course I'll be concentrating very much on developing the techniques themselves rather than elaborating the mathematical and statistical niceties. You're recommended to do this for yourselves if interested.

1.4 Acknowledgement

Parts of these notes are based on previous course notes, written mainly by Stuart Coles and Gareth Roberts. I am indebted to Martin Sköld for many helpful suggestions.

Chapter 2

Simulation

2.1 Introduction

In this chapter we look at different techniques for simulating values from distributions. We will also describe the Monte Carlo method and its use in computing integrals and performing hypothesis tests. Unlike all subsequent chapters we won't look much at applications here, but suffice it to say the applications of simulation are as varied as the subject of statistics itself.

2.2 Motivation

Many applications of simulation are based on the idea of using samples from a distribution in order to approximate characteristics of this distribution. As a trivial example, suppose that we are interested in the mean of a real random variable X with distribution function F and probability density function f. This expectation is given by

$$\int_{-\infty}^{\infty} x f(x) dx.$$

However, it might be difficult or impossible to perform the above integration. Suppose that a random sample $x_1, \ldots, x_n$ was available where each x_i is a realisation of X_i, and the $X_1, \ldots, X_n$ are i.i.d (independent and identically distributed) random variables with distribution F. Then we could approximate the mean of X by the observed sample mean

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

This result, which is routinely used in statistical estimation, will be shown to be related with the so-called Monte Carlo integration (Section 2.7) and with the bootstrap methods (Chapter 3).

As another simple example, suppose that we are interested in the median of the distribution of X. Notice now that unless F is a symmetric distribution, the median cannot be expressed as an expectation of some function of X. Instead, the median is the 0.5 quantile of F, where the αth quantile of F is $F^{-1}(\alpha)$, where F^{-1} denotes the inverse of F and $0 \leq \alpha \leq 1$. (You have encountered such quantiles when constructing confidence intervals.) As before, analytical calculation of the median (or of any other quantile) might be infeasible. Nevertheless, we can use the sample median to approximate the median of F. This technique is again related to bootstrap methods.

The validity of using samples to estimate characteristics of F is justified by some basic results of probability theory, summarised in the next section.

2.3 Some limit theorems

The *Law of Large Numbers* is the main result which validates using sample averages to estimate population means. Before stating (a very simple version of) the result, let $X_1, \ldots, X_n$ be a sequence of random variables, and $\theta_n = \theta_n(X_1, \ldots, X_n)$ be a real function of them (e.g $\theta_n = n^{-1}\sum_{i=1}^n X_i$). We say that θ_n converges in mean square sense to a fixed value θ if

$$E(\theta_n - \theta)^2 \to 0, \text{ as } n \to \infty.$$

The proof of the following theorem is left as an exercise.

Theorem 2.3.1. (A Law of Large Numbers). *Let $X_1, \ldots, X_n$ be a sequence of independent random variables with common means $E(X_i) = \theta$ and variances $Var(X_i) = \sigma^2 < \infty$. If $\theta_n = n^{-1}\sum_{i=1}^n X_i$ then*

$$E(\theta_n - \theta)^2 = \sigma^2/n \to 0, \text{ as } n \to \infty.$$

Although this theorem suggests that sample averages will converge to the population mean (with rate $1/n$), the following theorem (which we state without proof) gives more information about the (asymptotic) error of θ_n in estimating θ.

Theorem 2.3.2. (A Law of Large Numbers). *Let $X_1, \ldots, X_n$ be a sequence of independent random variables with common means $E(X_i) = \theta$ and variances $Var(X_i) = \sigma^2$. If $\theta_n = n^{-1}\sum_{i=1}^n X_i$ then*

$$\frac{\sqrt{n}}{\sigma}(\theta_n - \theta)$$

is approximately distributed as a $N(0,1)$ random variable, as $n \to \infty$.

Therefore, we wish to have a small σ in order to obtain accurate estimates of θ by θ_n. (This issue is related with the variance reduction techniques of Section 2.7.2.)

Similar results show that sample quantiles converge to population quantiles as the sample size increases.

2.4 Issues in simulation

The results of the previous section show that it is reasonable to use samples from a distribution in order to approximate certain characteristics of that distribution. Therefore, of key importance becomes how to generate samples from a specified distribution. In particular, this chapter using basic results from probability theory, will give some answers to the following two questions:

1. How to do it; and

2. How to do it efficiently.

(Note: Course Math457 on Markov chain Monte Carlo methods is also addressing these questions.) To some extent, just doing it is the priority, since many applications are sufficiently fast for even inefficient routines to be acceptably quick. On the other hand, efficient design of simulation can add insight into the statistical model itself, in addition to CPU savings.

2.5 Raw ingredients

The raw material for any simulation exercise is uniformly distributed random numbers: $U \sim U[0,1]$. Transformation or other types of manipulation can then be applied to build simulations of more complex univariate or multivariate distributions, as we show in the rest of this chapter. But, how can uniform random numbers be generated?

The standard approach is to use a **deterministic** algorithm which however produces a sequence of numbers which do not exhibit any obvious pattern: they look random in the sense that they can pass all the statistical tests of randomness despite their deterministic derivation. The most common technique is to use a *congruential generator*, which generates a sequence of integers via the algorithm

$$X_i = aX_{i-1}(modM) \tag{2.1}$$

for suitable choices of a and M. Dividing this sequence by M gives a sequence $U_i = X_i/M$ which can be regarded as realizations from the Uniform $U[0,1]$ distribution. Ripley gives details of the number theoretic arguments which support this method, and gives illustrations of the problems which can arise by using inappropriate choices of a and M. We won't worry about this issue here, as any statistical package has its random number generator checked pretty thoroughly. The point worth remembering though is that computer generated random numbers aren't random at all, but that (hopefully) they look random enough for that not to matter.

Nowadays, however, true random numbers can be obtained from the Internet, check for example `www.foumilab.ch/hotbits`. Rather controversially, pseudo-random numbers are much more preferable than true random numbers for several reasons, for example for code-checking or variance reduction techniques. From this perspective, it is actually an advantage to be able to produce random-looking pseudo-random numbers, rather than a limitation.

In subsequent sections we will assume that we can generate a sequence of numbers $U_1, U_2, \ldots, U_n$ which may be regarded as n independent realizations from the $U[0,1]$ distribution.

2.6 Simulating from specified distributions

In this section we look at ways of simulating data from a specified **univariate** distribution F, on the basis of a simulated sample $U_1, U_2, \ldots, U_n$ from the distribution $U[0,1]$. Some of the techniques of this section can be generalized to higher dimensions, although they become less efficient with increasing dimensions. Math457 course will introduce much more powerful techniques for simulation from multivariate distributions.

Notice however, that in principle we could simulate from a **multivariate** distribution using only univariate simulations. Assume that $(Y_1, \ldots, Y_m)$ is an m-dimensional vector with joint distribution function $F_{(Y_1,\ldots,Y_m)}$. This joint distribution has a well-known representation as a product of conditional distributions:

$$F_{(Y_1,\ldots,Y_m)} = F_{Y_1} F_{Y_2|Y_1} F_{Y_3|(Y_1,Y_2)} \cdots F_{Y_m|(Y_1,\ldots,Y_{m-1})}$$

where F_{Y_1} is the marginal distribution of Y_1, $F_{Y_2|Y_1}$ is the conditional distribution of Y_2 given Y_1, and so on. Therefore, we can simulate a realization of $(Y_1, \ldots, Y_m)$ by first simulating $Y_1 \sim F_{Y_1}$, then conditionally on that value we simulate $Y_2 \sim F_{Y_2|Y_1}$, etc, and finally $Y_m \sim F_{Y_m|Y_1,\dots,Y_{m-1}}$. This sometimes is called the *composition* method.

The problem with this approach is that it is difficult to derive the conditional distributions from the marginal. Nevertheless, for small m, for example $m = 2$, it might be feasible.

As a trivial special case, this method can be used to simulate a vector of independent random variables (how?).

We now describe some of the established techniques for univariate simulation.

2.6.1 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 $F(X) \sim U[0,1]$, so by inversion if $U \sim U[0,1]$, then $F^{-1}(U) \sim F$. Thus, defining $x_i = F^{-1}(u_i)$, generates a sequence of independent realizations from F. Figure 2.1 illustrates how this works.

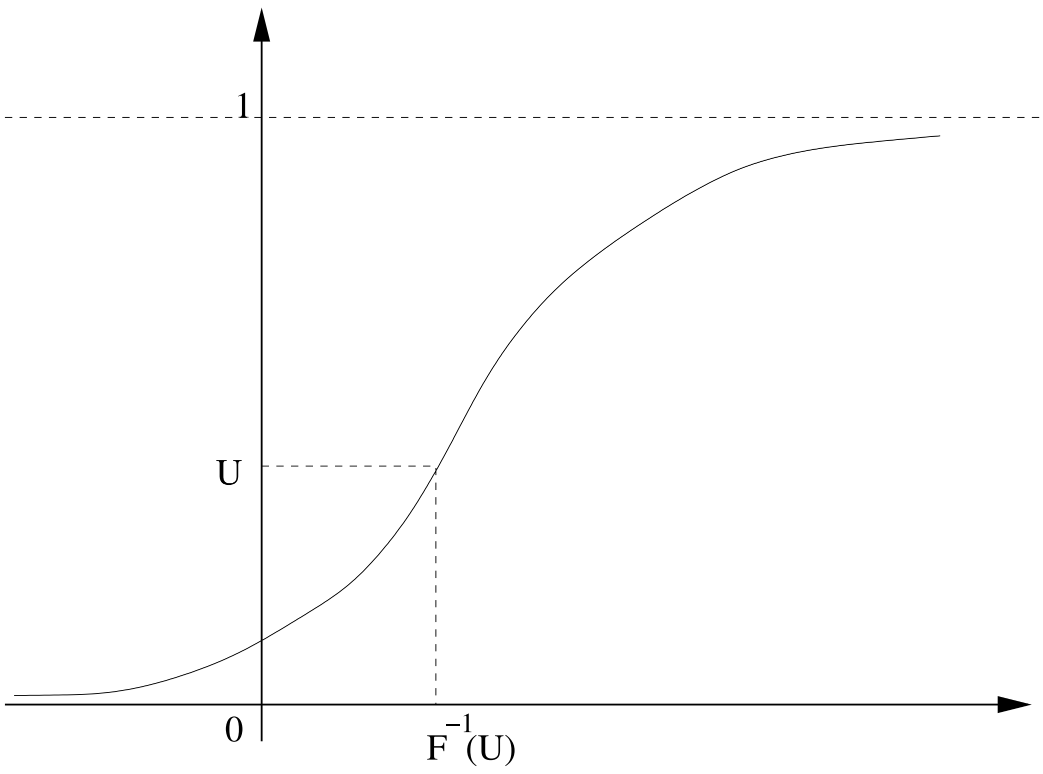

Figure 2.1: Simulation by inversion

This procedure is easily implemented in R with the following code:

```
dsim=function(n, inv.df) {
        u = runif(n)
        inv.df(u)
}
```

where `inv.df` is the user–supplied inverse distribution function F^{-1}. For example, to simulate from the exponential distribution we have $F(x) = 1 - \exp(-\lambda x)$, so $F^{-1}(u) = -\lambda^{-1}\log(1-u)$. Thus defining

```
inv.df=function(x,lam=1) -(log(1-x))/lam
```

we can simulate from the exponential distribution (unit exponential as default). Figure 2.2 shows a histogram of 1000 standard exponential variates simulated with this routine.

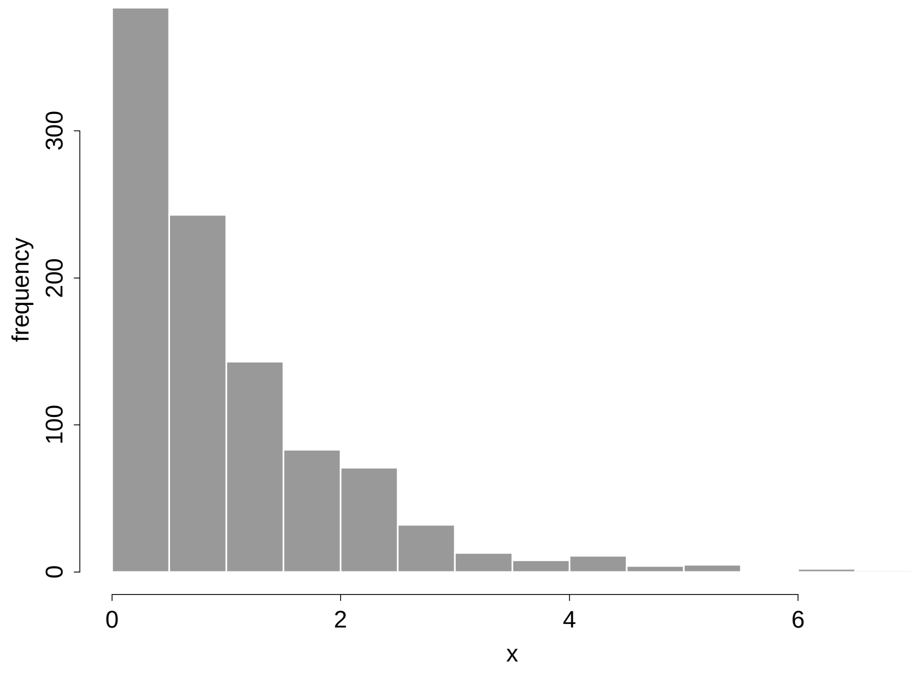

Figure 2.2: Histogram of 1000 simulated unit exponential variates

This procedure works equally well for discrete distributions, provided we interpret the inverse distribution function as

$$F^{-1}(u) = \min\{x|F(x) \geq u\} \tag{2.2}$$

The procedure then simply amounts to searching through a table of the distribution function. For example, the distribution function of the Poisson (2) distribution is

```
x     F(x)
------------
0  0.1353353
1  0.4060058
2  0.6766764
3  0.8571235
4  0.9473470
5  0.9834364
6  0.9954662
7  0.9989033
```

```
  8  0.9997626
  9  0.9999535
 10  0.9999917
```

so, we generate a sequence of standard uniforms $u_1, u_2, \dots, u_n$ and for each u_i obtain a Poisson (2) variate x where $F(x-1) < u_i \le F(x)$. So, for example, if $u_1 = 0.7352$ then $x_1 = 3$.

The limitation on the efficiency of this procedure is due to the necessity of searching through the table, and there are various schemes to optimize this aspect.

Returning to the continuous case, 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 (or both) complicated to program or excessively inefficient to run. The inversion method is only really useful if the inverse distribution function is easy to program and compute. This is not the case, for example, with the Normal distribution function for which the inverse distribution function, Φ^{-1}, is not available analytically and slow to evaluate numerically. To deal with such cases, we turn to a variety of alternative schemes.

The main limitation of the method however, is that it cannot be extended to multivariate simulation: it is by a construction a method for simulating from univariate distributions (why?).

2.6.2 Transformation of variables

The main idea is simple and is used extensively in simulation: suppose that we want to simulate a random variable $X \sim F$, and we know that it can be written as $X = t(Y)$, where $Y \sim G$ is another random variable and $t(\cdot)$ is some function. Then, defining $x_i = t(y_i)$ where y_i has been generated from G, generates a sequence of independent realizations from F. Notice that the inversion method is a special case of this technique.

As an example, suppose that $X \sim Exp(\theta)$, that is X has the exponential distribution with mean $1/\theta > 0$, then $X = \theta^{-1}Y$, where $Y \sim Exp(1)$. Therefore, we can simulate X by first simulating a standard exponential random variable, for example using the inversion method, and then dividing the drawn value by θ.

$t(\cdot)$ can be a many-to-one function. For example, suppose that $X \sim Gamma(n, \theta)$, that is X has the gamma distribution with mean n/θ and variance n/θ^2, where n is some integer. It is known that $X = t(Y_1, \dots, Y_n) = \sum_{i=1}^{n} Y_i$, where the Y_is are independent and identically distributed $Exp(\theta)$ random variables. Thus, X can be simulated by simulating n independent Y_is and then summing them up.

2.6.3 Rejection sampling

The idea in rejection sampling is to simulate from one distribution which is easy to simulate from, but then to only accept that simulated value with some probability p. By choosing p correctly, we can ensure that the sequence of accepted simulated values are from the desired distribution. Rejection sampling has a very intuitive geometrical interpretation, illustrated in Figure 2.3. It is not surprising (and it can be proved) that points can be simulated uniformly inside a set C (e.g the star-shaped set in Figure 2.3) by simulating uniformly points on a superset $C \subset B$ (e.g the unit square in Figure 2.3) and rejecting those which fall outside C. The hope is that it is easier to simulate points uniformly inside B than inside C. In our example, it is trivial to simulate a point uniformly inside B, simply take (U, V), where U, V are independent $U[0, 1]$ variables. Notice that if B is much larger than C then most of the simulated points will be rejected, thus it is desirable to find B so that to match closely C.

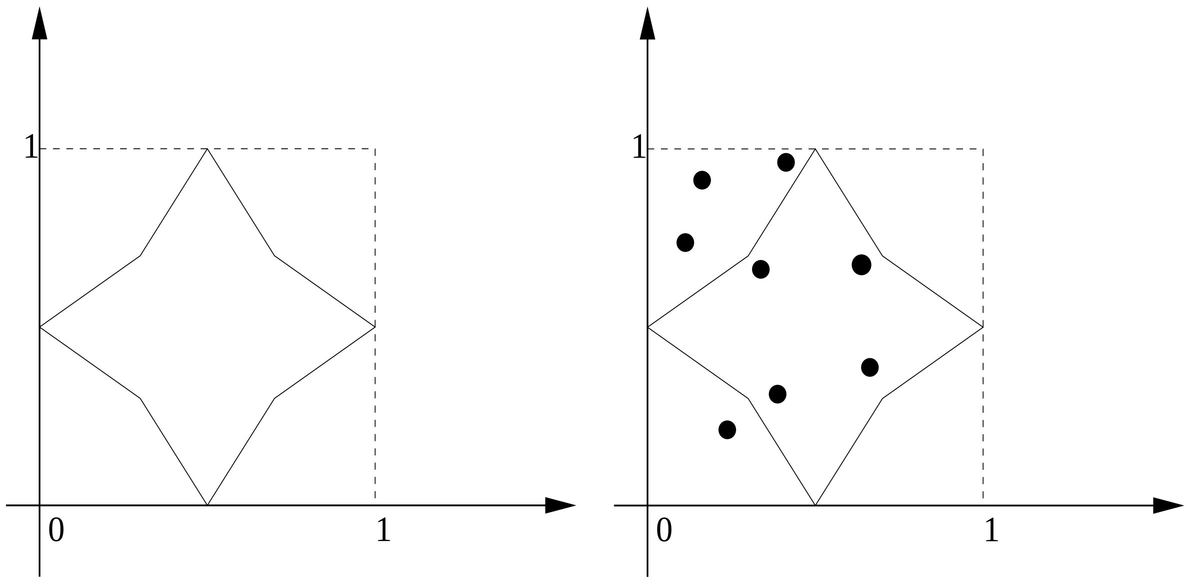

Figure 2.3: Simulation of uniformly distributed points inside the star-shaped set (left) by simulating uniformly points inside the unit square (formed by the axes and the dashed lines) and rejecting those who fall outside the star-shaped set (right)

We now turn to the problem of simulating from a given density f, and link this problem with the rejection idea described above. Suppose that f is a probability density function on the real line. Moreover, let (X, Y) be a uniformly distributed point under the density f (the grey-shaded area in Figure 2.4). Therefore, if $h_{(X,Y)}(x, y)$ denotes the probability density function of (X, Y), then

$$h_{(X,Y)}(x, y) = 1, \ -\infty < x < \infty, 0 < y < f(x).$$

It is easy to show (check!) that the marginal density of X is exactly f. The rejection sampling idea described earlier can be used to generate samples uniformly under f. Suppose that $g(x)$ is another density function, which is easy to simulate from, and K a constant such that

$$f(x) \leq Kg(x), \text{ for all } x,$$

therefore the set of points under f lies entirely within the set of points under Kg (see Figure 2.5). A point (X^*, Y^*) can be simulated uniformly under Kg, by sampling X^* from g, and then simulate Y^* given X^* from a uniform distribution $U[0, Kg(X^*)]$ (composition method). Applying the rejection idea described earlier, if we keep all such points that fall under f as well, i.e all those (X^*, Y^*) such that $Y^* < f(X^*)$ we will obtain a sample of points uniformly distributed under f. The X^*s of this sample are an independent sample from f.

Thus, the procedure described above is equivalent to the following algorithm:

Algorithm 2.1.

1. *Simulate x^* from $g(x)$.*
2. *Simulate y^* from $U(0, Kg(x^*))$.*
3. *Accept x^* if $y^* \leq f(x^*)$.*

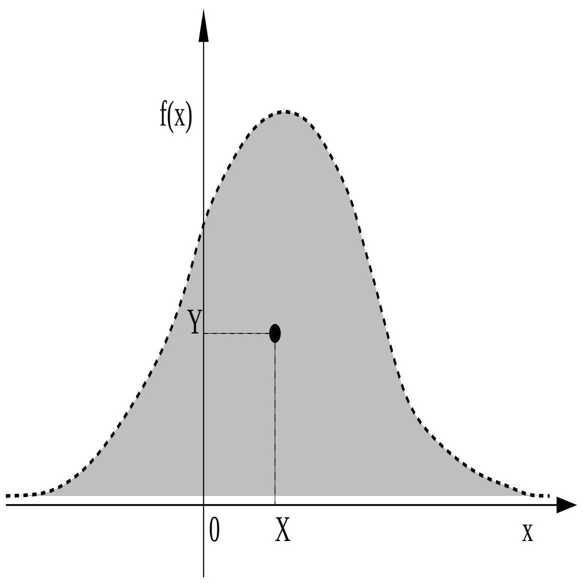

Figure 2.4: The density f (thick dashed line) and a point (X, Y) uniformly distributed under the area defined by f and the x-axis.

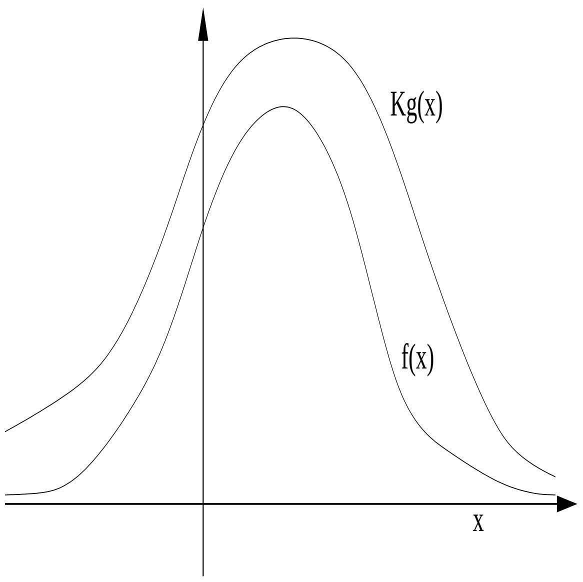

Figure 2.5: The density f and the envelope function $Kg(x)$.

4. Continue.

Furthermore, $\Pr(X \text{ accepted}) = 1/K$.

Formal Proof Here give the formal proof that rejection sampling works. Generalizing slightly, let f be the density we want to simulate up to a proportionality constant, i.e $\int f(x)dx = c_1$, and g be the proposal density also up to a proportionality constant, $\int g(x)dx = c_2$, where both c_1 and c_2 might be unknown. Let K be such that $f(x) \leq Kg(x)$, for all x, and I be an indicator variable, such that $\Pr(I = 1 \mid X^* = x^*) = f(x^*)/(Kg(x^*))$, and $\Pr(I = 0 \mid X^* = x^*) = 1 - f(x^*)/(Kg(x^*))$. The rejection sampling algorithm we have presented, can be implemented using the indicator variables, as:

1. Simulate x^* from $g(x)$.
2. Conditionally on $X^* = x^*$, simulate the indicator I, so that $\Pr(I = 1 \mid X^* = x^*) = f(x^*)/(Kg(x^*))$.
3. Accept x^* if $I = 1$.
4. Continue.

Therefore, all those x^*s which are paired with $I = 1$ are retained as sample from f. Since x^* is simulated from g, we have that:

$$P(I = 1) = \int P(I = 1 \mid X^* = x^*)\frac{g(x^*)}{c_2}dx^* = \frac{c_1}{c_2 K},$$

which gives the overall acceptance probability of the algorithm (which equals $1/K$ when f and g are proper densities). To show that rejection sampling ends up with samples from f we need to show that the conditional density of x^* given that $I = 1$ is proportional to f. Let $k(x^* \mid I = 1)$ denote this conditional density. Then, by Bayes theorem:

$$k(x^* \mid I = 1) = \frac{\Pr(I = 1 \mid X^* = x^*)g(x^*)/c_2}{\Pr(I = 1)} = f(x^*)/c_1,$$

which completes the proof.

□

Note that we only need to know f up to a normalizing constant in order for this technique to work (which is obvious geometrically). This is a big advantage of the method since in many statistical applications we need to simulate from densities for which we cannot compute the normalizing constant (typically, maximization is easier than integration). The efficiency of the procedure depends on the quality of the agreement between f and the bounding envelope Kg since if a large value of K is necessary, then the acceptance probability is low, so that large numbers of simulations are needed to achieve a required sample size.

As an example, consider the distribution with density

$$f(x) \propto x^2 e^{-x}; \quad 0 \leq x \leq 1 \tag{2.3}$$

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. Figure 2.6 shows both $f(x)$ and $g(x)$. Clearly in this case the envelope is very poor so the routine is highly inefficient (though statistically correct).

Applying this to generate a sample of 100 data using the following code

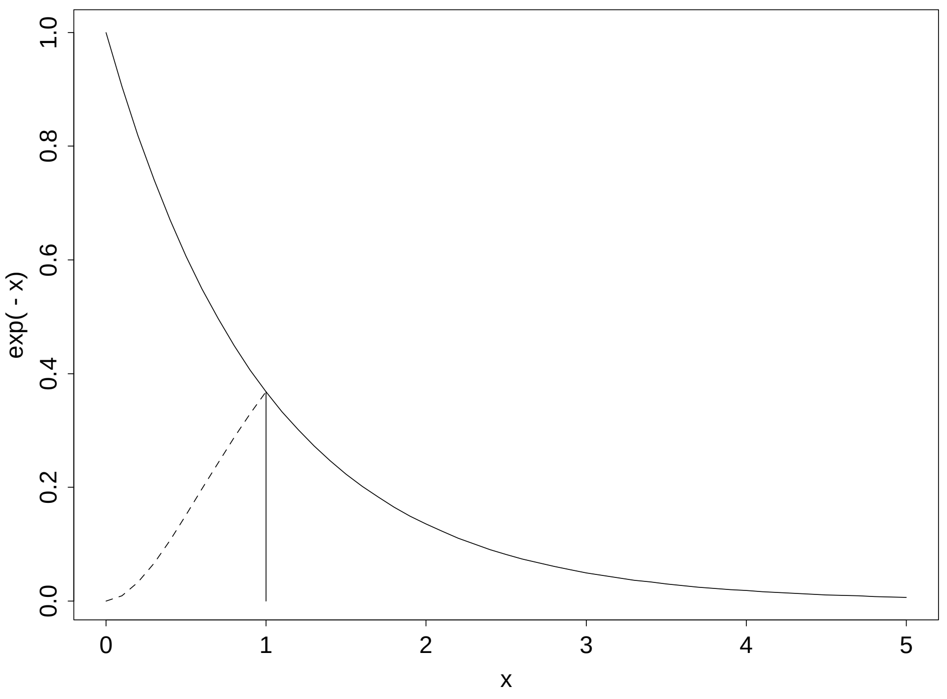

Figure 2.6: Scaled density and envelope

```
 rej.sim=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
}
```

gave the histogram in Figure 2.7.

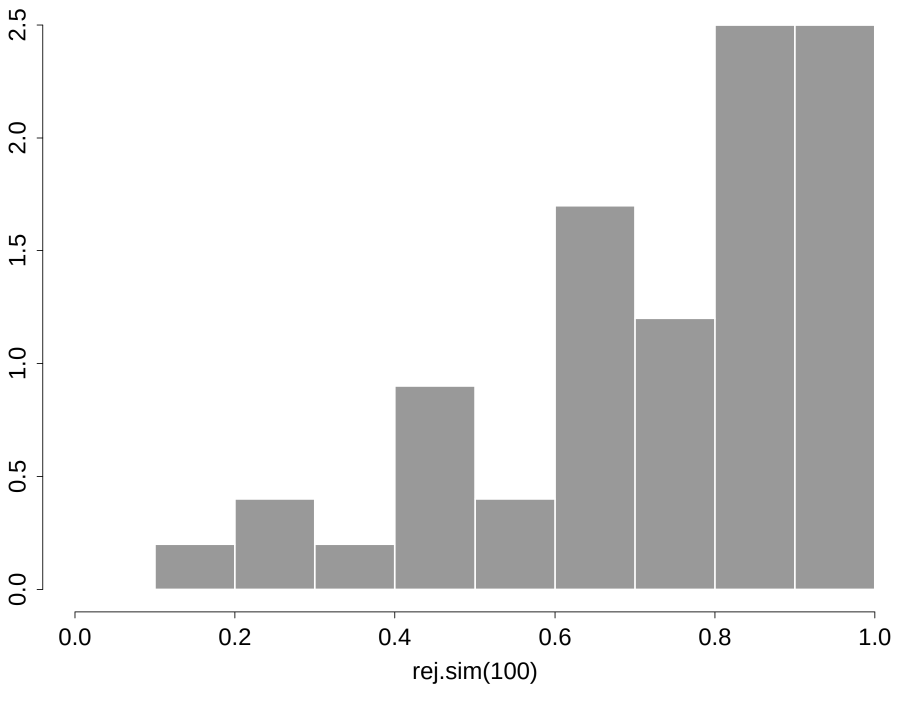

Figure 2.7: Histogram of simulated data

2.6.4 Ratio of uniforms (NOT COVERED IN THE COURSE)

An adaptation of the rejection algorithm which works well for many distributions is the ratio of uniforms method. Here a pair of independent uniforms are simulated and the ratio accepted as a simulant from the required distribution according to a rejection scheme.

The basis of the technique is the following argument. Suppose h is a non–negative function such that $\int h < \infty$ and we let $C_h = \{(u, v) : 0 \leq u \leq \sqrt{h(v/u)}\}$. Then if (U, V) is uniformly distributed over C_h then $X = V/U$ has pdf $h/\int h$.

So, to simulate from a density proportional to h, we simulate uniformly over the region C_h, and take ratios of coordinates. In practice, C_h may be complicated in form, so the only practical solution is to bound it with a rectangle (if possible), simulate within the rectangle (by a pair of uniforms), and apply rejection: hence, *ratio of uniforms.*

The reason this works is as follows. Let Δ_h be the area of C_h. Then on changing variables $(u, v) \rightarrow (u, x)$, where $x = v/u$,

$$\Delta_h = \int\int_{C_h} dudv \quad = \int\int_0^{\sqrt{h(x)}} u \; dudx \quad = \int \frac{1}{2} h(x) dx \tag{2.4}$$

Because of the uniformity of (U, V) over C_h, (U, V) has pdf $1/\Delta_h$ so that on transformation, (U, X) has pdf u/Δ_h, and integrating out U gives the marginal pdf of X as:

$$\Delta_h^{-1} \int_0^{\sqrt{h(x)}} u \; du \quad = \quad h(x)/\{2\Delta_h\} \quad = \quad h(x)/\int h(x) dx \tag{2.5}$$

Thus V/U has pdf proportional to h. Again, a key property of this method is that h is only required to be specified up to proportionality.

As discussed above, this is only useful if we can generate uniformly over C_h, which is most likely to be achieved by simulating uniformly within a rectangle $[0, a] \times [b_-, b_+]$ which contains C_h (provided such a rectangle exists). If it does, we have the following algorithm.

Algorithm 2.2.

1. *Simulate independent $U \sim U[0, a]$, $V \sim U[b_-, b_+]$.*
2. *If $(U, V) \in C_h$, accept $X = V/U$, otherwise repeat.*
3. *Continue.*

As an example, consider the Cauchy distribution with density

$$h(x) \propto \frac{1}{1 + x^2} \tag{2.6}$$

Then $C_h = \{(u, v) : 0 \leq u \leq \sqrt{h(v/u)}\} = \{(u, v) : 0 \leq u, u^2 + v^2 \leq 1\}$, a semicircle. Hence we can take $[0, a] \times [b_-, b_+] = [0, 1] \times [-1, 1]$ and get the algorithm

Algorithm 2.3.

1. *Simulate independent $U \sim U[0, 1]$, $V \sim U[-1, 1]$.*
2. *If $u^2 + v^2 \leq 1$ accept $x = v/u$, otherwise repeat.*
3. *Continue.*

This can be implemented with the R code

```
ru.sim=function(n) {
        r = NULL
        for(i in 1:n) {
                t = 1
                while(t > 0) {
                        u = runif(1, 0, 1)
                        v = runif(1, -1, 1)
                        t = u^2 + v^2 - 1
                }
                r[i] = u/v
        }
        r
}
```

and a histogram of 1000 simulated values is given in Figure 2.8 (note the unusual scale (!) because the heaviness of the Cauchy tail causes one or two simulated values to be extreme relative to the rest of the data).

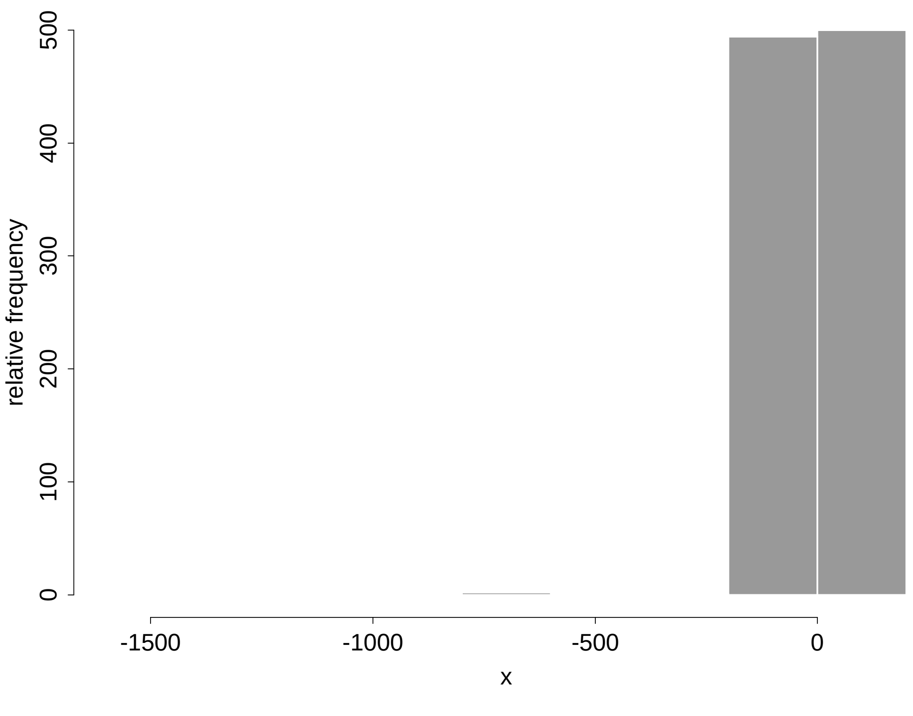

Figure 2.8: Histogram of simulated Cauchy

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

Another method for improving the efficiency is by a process known as ‘squeezing’ or ‘pre–testing’. This applies to both the rejection and ratio of uniform methods. The point is that, in the ratio of uniforms method for example, the slowest part of the algorithm can be the check of whether $(u, v) \in C_h$ or not. However, there may be simpler 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 it lies inside C_h or not.

2.7 Monte–Carlo integration

In one form or another, the reason for simulation can often be evaluating an integral. Section 2.3 showed that we can use sample means of the form

$$\hat{\theta}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \phi(x_i), \tag{2.7}$$

where the x_is are a sample from the distribution of X with density f, in order to approximate expectations

$$\theta = \int \phi(x)f(x)dx = E_f(\phi(X)), \tag{2.8}$$

where $E_f(\phi(X))$ denotes the expectation of $\phi(X)$ with respect to the density f. The notation $\hat{\theta}_n(f)$ reflects that it is an estimate of θ (thus the "hat" notation) which depends both on the sample size n and the fact that the x_is have been simulated from f. The idea behind Monte Carlo integration is to express an integral we wish to compute as an expectation of a random variable and then simulate samples from that random variable to approximate the integral by the appropriate sample mean. This approach is remarkably easy to use, even in high dimensions. The cost for this simplicity is that the variance of the sample mean estimators might be high.

(The main advantage of Monte Carlo integration over numerical analysis methods for approximating integrals - such as Simpson's rule - is that the latter suffer in high dimensions. However, for small-dimensional integrals (say up to dimension 8) numerical analysis methods with the same computational cost as Monte Carlo can achieve a much better approximation.)

As an example of this, suppose we wish to calculate $P(X < 1, Y < 1)$ where (X, Y) are bivariate Standard Normal with correlation 0.5. This can be written as

$$\int I_A(x, y)f(x, y)dxdy \tag{2.9}$$

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

$$n^{-1}\sum_{i=1}^{n} I_A(x_i, y_i) \tag{2.10}$$

which is simply the proportion of simulated points falling in A. There are various approaches to simulating bivariate Normals. In my experiment, on the basis of 1000 simulations I got 0.763 as an estimate of the probability.

2.7.1 Importance sampling

There are several applications (Bayesian inference and filtering are two classic examples) where, although we want to estimate (2.8), we cannot obtain samples from f by any of the available methods (for example inverse CDF, rejection sampling, etc) due to the complicated and/or high-dimensional form of the density. Moreover, we might be interested in estimating the expectation of several different functions with respect to f. Suppose, however, that instead we can sample from a density g defined on the same space as f, which dominates f, in the sense $g(x) = 0 \implies f(x) = 0$. For a given function ϕ, we re-write (2.8) as,

$$\theta = \int \phi(x) \frac{f(x)}{g(x)} g(x) dx = E_g(\phi(X)w(X)), \quad \text{where } w(x) = \frac{f(x)}{g(x)};$$

$w(X)$ is known as the *importance weight* associated to the sampled point (*particle*) X, and g is called the *importance density.* The above expression suggests two different so-called *importance sampling* estimators:

$$\hat{\theta}_n(g) = \frac{1}{n} \left\{ \sum_{i=1}^{n} \phi(x_i)w(x_i) \right\},$$

and

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

where the x_is are iid draws from g. As an exercise, you can show that $\hat{\theta}_n(g)$ is an unbiased estimator of θ. Moreover, you can also show that $E_g(w(X)) = 1$, therefore $\hat{\theta}_n^b(g)$ is a consistent estimator of θ (appealing to a stronger version of Theorem 2.3.1). Generally $\hat{\theta}_n^b(g)$ is a biased estimator of θ (thus the superscript "b"), but in many cases it might have a smaller mean square error than $\hat{\theta}_n(g)$. However, the main advantage of the biased estimator over the unbiased is that the former does not require knowledge of the normalizing constants of f and g in order to be computed.

A necessary requirement for the implementation of the method is that the importance density g is easy to sample from. An important question however, is which is the "optimal" choice of g among the class of densities which we can sample from and which dominate f. When interest lies in estimating the expectation of a specific function ϕ, the next section provides an answer (where it will also become clear why the method is called *importance sampling*). However, when the expectation of several different functions are to be estimated we can informally recommend that g should be chosen to be as "close" to f as possible. More formally, for importance sampling to be efficient it is crucial to ensure that the variance of the importance weights is finite:

$$Var(w(X)) = \int (f(x)/g(x) - 1)^2 g(x) dx < \infty.$$

It can be seen from this expression that this variance depends on the discrepancy between f and g and it would be 0 if we could indeed use f as the importance density. Check that when $f(x)/g(x) \leq K < \infty$, the variance of the importance weights will be finite. How does this relate to rejection sampling?

2.7.2 Variance Reduction

A quote from Brian Ripley's book:

Variance reduction obscures the essential simplicity of simulation.

Indeed, although Monte Carlo methods (such as Monte Carlo integration) are very intuitive, the methods employed to reduce the variance of the resulting estimates are often rather involved. Nevertheless, in many modern applications (such as Monte Carlo maximum likelihood estimation, particle filters etc) variance reduction techniques have a substantial role to play, since they improve the precision of the Monte Carlo estimates by many orders of magnitude. In this section we describe some of these methods. However, the applications given here are rather basic and do not represent the most exciting problems where variance reduction is relevant.

Importance sampling

Suppose that we want to estimate

$$\theta = \int \phi(x) f(x) dx, \tag{2.11}$$

for a specific function ϕ. It will be useful for the context of this section to take ϕ to be positive. The previous section introduced the importance sampling as a method for Monte Carlo integration, especially when it is infeasible to simulate from f. However, even if simulation from f is possible, the importance sampling estimator $\hat{\theta}_n(g)$ (or $\hat{\theta}^b_n(g)$) can give a much more efficient estimator than the plain-vanilla estimator based on samples from f, $\hat{\theta}_n(f)$, if g is appropriately chosen.

The variance of $\hat{\theta}_n(f)$ is large when $\phi(x)$ varies a lot in areas where f is very small. The example in Figure 2.9, where $\phi(x) = x^8$ and $f(x) = e^{-x^2/2}/\sqrt{2\pi}$ is revealing as to why $\hat{\theta}_n(f)$ might give poor estimates of θ.

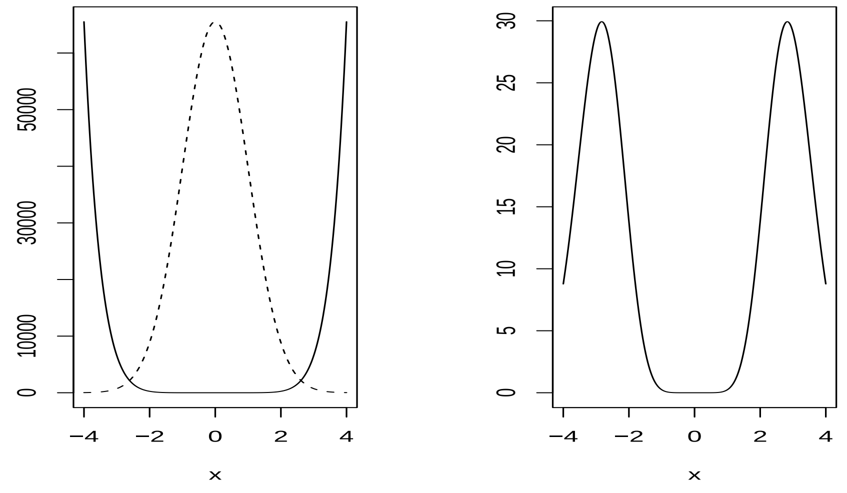

Figure 2.9: Left: $\phi(x) = x^8$ (solid line) and $f(x) \propto e^{-x^2/2}$ (dashed line). f has been scaled to have the same maximum as ϕ for visual purposes. Right: The quantity to be estimated is the area under the plotted curve $f(x)\phi(x)$.

Instead, it is preferable to simulate draws from the *important* areas, i.e the areas where ϕ varies a lot. More formally, the variance of $\hat{\theta}_n(g)$ is

$$Var(\hat{\theta}_n(g)) = n^{-1} \int \left\{ \frac{\phi(x)f(x)}{g(x)} - \theta \right\}^2 g(x) dx \tag{2.12}$$

This variance can be much lower than the variance of $\hat{\theta}_n(f)$, if g can be chosen so as to make the ratio $\phi(x)f(x)/g(x)$ nearly constant in x. Essentially what is happening is that the simulations are being concentrated in the areas where there is greatest variation in the integrand, so that the informativeness of each simulated value is greatest.

This example taken from Ripley illustrates the idea. Suppose we want to estimate the probability $P(X > 2)$, where X follows a Cauchy distribution with density function

$$f(x) = \frac{1}{\pi(1+x^2)} \tag{2.13}$$

so we require the integral

$$\theta = \int I_A(x) f(x) dx \tag{2.14}$$

where $A = \{x : x > 2\}$. The most intuitive and straightforward Monte Carlo method would be to simulate values from the Cauchy distribution directly and apply (2.7), that is approximate $P(X > 2)$ by the proportion of the simulated values which are bigger than 2. Nevertheless, the variance of this estimator is substantial, since it is the empirical proportion of draws which exceed 2, but due to the importance density (f in this case) draws rarely exceed 2, so the variance is large compared to its mean.

The previously suggested method, although inefficient it still is very intuitive and beautifully simple, since it approximates a probability by a sample frequency. On the other hand, the method we will now describe is not intuitive but it can be very efficient! The key in understanding this method is to think of θ not as a probability, but as an area under a function. Specifically, in our example θ is the area under $f(x)$, for $x \geq 2$ (see Figure 2.10a),

$$\theta = \int_2^\infty f(x) dx.$$

Observe that for large x, $f(x)$ is approximately equal to $1/(\pi x^2)$, since the term πx^2 dominates

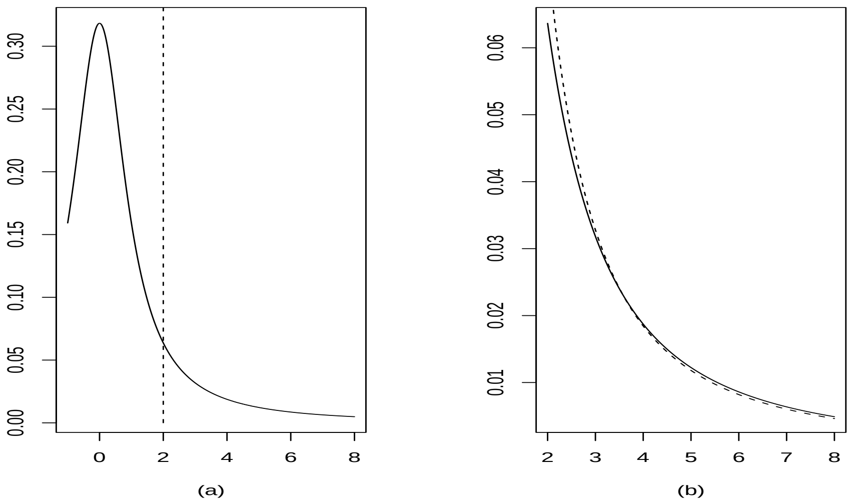

Figure 2.10: (a): The Cauchy density, (b) the Cauchy density $f(x), x \geq 2$ with solid line and the function $0.1476g(x)$, $g(x) = 2/x^2$ with a dashed line superimposed.

over π in the denominator of f. If we normalise this function so that it is a density function with support on $[2, \infty)$ we get $g(x) = 2/x^2, x \geq 2$, and we re-write θ as

$$\theta = \int_2^\infty \frac{f(x)}{g(x)} g(x) dx.$$

It is easy to simulate from g using the inversion method: take $x_i = 2/u_i$ where $u_i \sim U[0,1]$. Thus, our estimator becomes:

$$\hat{\theta}_n(g) = n^{-1} \sum_{i=1}^{n} \frac{x_i^2}{2\pi(1+x_i^2)} \tag{2.15}$$

where $x_i = 2/u_i$. Notice that $f(x)/g(x)$ is close to being a constant (see Figure 2.10b) therefore $\hat{\theta}_n(g)$ has much smaller variance than $\hat{\theta}_n(f)$. Implementing this with the R function

```
i.s=function(n) {
        x = 2/runif(n)
        psi = x^2/(2 * pi * (1 + x^2))
        mean(psi)
}
```

gave the estimate $\hat{\theta} = .1478$. The exact value is $.5 - \pi^{-1} \tan 2 = .1476$.

Figure 2.11 compares the estimators $\hat{\theta}_n(f)$ and $\hat{\theta}_n(g)$ using a simulation. We plot the convergence of the sample means $\hat{\theta}_n(f)$ (left), and $\hat{\theta}_n(g)$ (right) for $n = 1, \ldots, 1000$, and we have repeated this experiment 10 different times. Notice that $\hat{\theta}_n(g)$ converges very quickly to the true value of θ for all 10 independent realizations of the experiment. On the other hand, some of the paths corresponding to $\hat{\theta}_n(f)$ are far from the true value of θ even for $n = 1000$. The spread of the values $\hat{\theta}_n(f)$ for any n among the 10 different simulated paths, shows the high variance of the estimator. However, the average values of the estimator is θ, since the estimator is unbiased.

Control variates

In general, the idea of control variates is to modify an estimator according to a correlated variable whose mean is known. To fix ideas, suppose we wish to estimate $\theta = E(Z)$ where $Z = \phi(X)$. Then $\hat{\theta} = Z$ is an unbiased estimator of θ. Suppose now that W is another random variable which however has known mean $E(W)$. Then, we can construct an alternative unbiased estimator

$$\hat{\theta} = Z - W + E(W). \tag{2.16}$$

In this context W is called a *control variate.* The variance of the estimator is given by

$$\text{Var}(\hat{\theta}) = \text{Var}(Z) - 2\ \text{Cov}(W,Z) + \text{Var}(W). \tag{2.17}$$

Therefore it is desirable to choose as control variate a random variable W which we believe is correlated with Z so that Cov (W,Z) is large and the variance of the above estimator is smaller than the variance of the basic estimator $\hat{\theta} = Z$. Notice that

$$\hat{\theta} = Z - \beta(W + E(W)) \tag{2.18}$$

is also an unbiased estimator of θ, for any β, where we now have the added degree of freedom to choose β in order to minimize the variance of the estimator. Generally, we could come up with several variables, say $W^{(1)}, \ldots, W^{(p)}$, which could serve as candidate control variates and construct the following unbiased estimator:

$$\hat{\theta} = Z - \beta_1(W^{(1)} + E(W^{(1)})) - \cdots \beta_p(W^{(p)} - E(W^{(p)})), \tag{2.19}$$

where the coefficients $\beta = (\beta_1, \ldots \beta_p)$ are chosen to minimize the variance of the estimator. Linear regression theory suggests to choose the coefficients by regressing observations of Z on

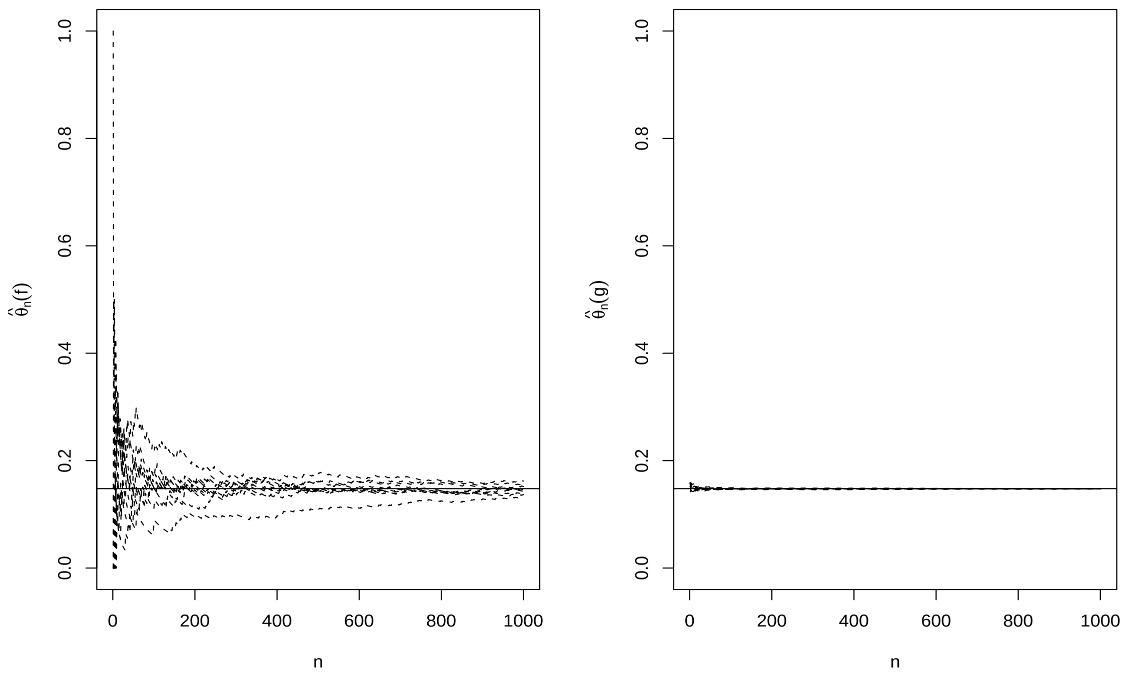

Figure 2.11: Convergence of the standard $\hat{\theta}_n(f)$ (left) and the importance $\hat{\theta}_n(g)$ (right) sampled mean.

observations of the control variates. To avoid biases, one way to implement this idea is to run a small preliminary simulation experiment, where independent realizations from Z and from $W^{(j)}$s are simulated. Based on these realizations the coefficients are estimated using linear regression yielding an estimate $\hat{\beta}$ say. Subsequently n independent realizations of Z and $W^{(j)}$s are simulated and θ is estimated by

$$\frac{1}{n}\sum_{i=1}^{n}\left\{Z_i - \hat{\beta}_1(W_i^{(1)} + E(W^{(1)})) - \cdots \hat{\beta}_p(W_i^{(p)} - E(W^{(p)}))\right\}. \tag{2.20}$$

It turns out (and it is quite easy to see) that the variance of the estimator becomes approximately $n^{-2}RSS$ where RSS is the residual sum of squares of the regression fit.

This is easily applied in the context of Monte–Carlo integration. Again developing Ripley's Cauchy example, we require an estimate of

$$\theta = \frac{1}{2} - \int_0^2 f(x)dx \tag{2.21}$$

where $f(x) = \frac{1}{\pi(1+x^2)}$. We can estimate this as

$$\hat{\theta} = \frac{1}{2} - 2 \times \frac{1}{n}\sum_{i=1}^{n} f(x_i), \tag{2.22}$$

where the $x_i \sim U[0,2]$.

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 over $[0,2]$ 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. To minimize variance however, we need to choose the β's so as to optimize agreement between $f(x)$ and $\beta_1 x^2 + \beta_2 x^4$. This is achieved through simple regression. I chose to simulate 10 observations uniformly over $[0,2]$, leading to the regression equation

$$W(x) = .288 - .143x^2 + .024x^4 \tag{2.23}$$

and hence,

$$\hat{\theta} = \frac{1}{2} - 2\left[\frac{1}{n}\sum_{i=1}^{n}\{f(X_i) - W(x_i)\} + .288 \times 2 - .143 \times 8/6 + .024 \times 32/10\right] \tag{2.24}$$

With $n = 1000$ I got an estimate of $\theta = .1474$.

Antithetic variates

Antithetic variates are almost the converse of control variates: we obtain a variate Z^* which has the same distribution as Z, but is negatively correlated with Z. Then,

$$\hat{\theta} = \frac{1}{2}(Z + Z^*) \tag{2.25}$$

is an unbiased estimator of θ, with variance:

$$\mathrm{Var}(\hat{\theta}) = \frac{1}{2}\mathrm{Var}(Z)\{1 + \mathrm{corr}(Z, Z^*)] \tag{2.26}$$

which constitutes a reduction in variance provided the correlation is indeed negative. For simple problems, antithetic variates are easily achieved by inversion, since if $Z = F^{-1}(U)$ then $Z^* = F^{-1}(1-U)$ has the same distribution as Z and can be shown to be negatively correlated with Z for all choices of F. Applying this to the estimation of $\theta = \frac{1}{2} - \int_0^2 f(x)dx$ in the Cauchy example leads to the estimator

$$\frac{1}{2} - \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{1}{\pi(1+u_i^2)} + \frac{1}{\pi(1+(2-u_i)^2)}\right\} \tag{2.27}$$

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

2.8 Monte Carlo Tests

Monte Carlo methods can be used in various ways in order to investigate, either formally or informally, whether the observed data are consistent with a certain statistical hypothesis. To fix ideas, let H_0 be a simple null-hypothesis, $x_1, \dots, x_n$ be the data set, $T = t(X_1, \dots, X_n)$ be a test-statistic, and $t = t(x_1, \dots, x_n)$ be its observed value. (Simple hypotheses totally specify the distribution of the data.)

When we test a hypothesis we might want to either accept or reject it. The result of this procedure is a yes/no answer (actually a maybe/no answer, since formally we don't accept hypotheses). Alternatively, more insight into the scientific question posed by H_0, can be gained by computing the P-value. This is the probability of observing a value for T at least as extreme as t, given that the null-hypothesis is true:

$$p = \Pr(t(X_1, \dots, X_n) \text{ is more extreme than } t(x_1, \dots, x_n) \mid H_0). \tag{2.28}$$

For this calculation, we need to know the distribution of the test statistic under the null hypothesis, but this distribution is often unknown.

Monte Carlo methods can be useful in both of the approaches above, but they can be used to learn even more about our data. Although P-values are more informative than a simple yes/no answer, they have weaknesses as measures of support that the data provide for H_0 (for an accessible presentation of this issue, see Schervish, *The American Statistician, 1996*). Example 2.1 in this section shows how we can use graphical and Monte Carlo methods in order to investigate if the data are consistent with H_0, and if not which aspect of the model is inconsistent with the data. This approach can reveal characteristics of the data which should be modelled using a more elaborate model than that suggested by H_0.

Exact Monte Carlo tests

We can use the computer in order to construct exact tests for simple hypotheses. For illustration we consider one-sided tests only. Let T be our chosen test statistic and t_1 the value of T for the observed data. We simulate $s-1 > 0$ data sets independently according to H_0 and $t_2, \dots, t_s$ are the values of T for each of the simulated data sets. Let $t_{(j)}$ denote the jth largest among the $t_i, i = 1, \dots, s$. According to H_0, the t_is are i.i.d, and (since they are exchangeable)

$$\Pr(t_1 = t_{(j)}) = s^{-1},\ j = 1, \dots, s,$$

thus rejection of H_0 on the basis that t_1 ranks kth largest or higher gives an exact, one-sided test of size k/s. It shouldn't come as a surprise: if H_0 is true, the probability of rejecting H_0 given it is true, is the probability of t_1 ranking kth largest or higher, which is exactly $1 - k/s$. Thus, if we want a test of size 95%, we can take $s = 20, k = 1$, but we could also take $s = 100, k = 5$ etc. Notice that if we repeat the simulation, we might get a different answer, i.e we might accept H_0 although we previously rejected it! The Monte Carlo test is based on the ranks of the t_is, rather than their actual values. On the other hand, the test based on the distribution of T under H_0 takes into account the exact value of t_1.

The value of s doesn't have to be very large, but it is recommended (for good reasons!) to increase s as the size of the test increases (see p.9 of Diggle, *Spatial Point Patterns*, for more details). We can also compute the P-value of the Monte Carlo test. In the example above, it is the proportion of t_is with higher ranks than t_1.

Monte Carlo estimates of P-values

In many situations, the distribution of the test-statistic T under the null-hypothesis is complicated and the P-value difficult or impossible to calculate. On the other hand, the P-value can be expressed as an expectation, and as long as simulation of data under H_0 is feasible, Monte Carlo integration can be used to approximate it. Thus, we simulate $s - 1$ independent datasets under the null-hypothesis and for each simulated dataset we evaluate the test-statistic. Subsequently, we can approximate the P-value by the proportion of times the simulated test-statistics were more extreme than the one calculated from the observed data.

The following example illustrates how Monte Carlo simulation can be used for more informal and exploratory investigation of the agrement of the data with H_0. The example is from the area of Spatial Statistics. Monte Carlo tests are very popular in the analysis of point patterns, since the starting point is usually a so-called test of complete spatial randomness (further examples will be given in the course of Environmental Epidemiology, Math465).

Example 2.1. *Completely random scattering of points.*

Figure 2.12 shows a point pattern on $[0, 1] \times [0, 1]$, consisting of 25 points. The data are stored in the R data frame `point.pattern.dat`. These kind of data appear very frequently in the context of Spatial Statistics and a scientific question of interest is whether the points are randomly scattered. Alternatively, the points might exhibit clustering, or mutual repulsion. Therefore, the null-hypothesis takes the form

H_0: the points are randomly uniformly scattered,

which in our example means that the observed 25 points are independent draws from the uniform distribution on $[0, 1] \times [0, 1]$. Let X_i denote the location of the point i on $[0, 1] \times [0, 1]$. We take as our test statistic

$$T = t(X_1, \ldots, X_n) = \sum_{i=1}^{n} \tau_i,$$

where for each point i, τ_i is the distance to its nearest neighbour. Very small values (compared to draws from the null-distribution) of T suggest clustering of points, whereas very large values suggest repulsion of points, i.e the presence of a point at a location x, makes less likely other points to be near x.

There are R-libraries which contain functions which can be used to compute such test-statistics. In particular, we can use the function `nndistG` of the SPLANCS R-library. The following R-function carries out the Monte Carlo test.

```
inter.dist = function(n,s)
```

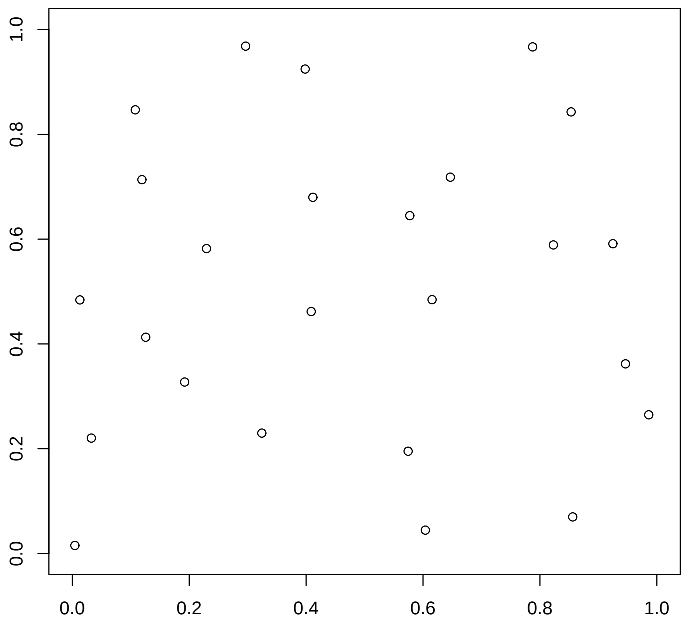

Figure 2.12: A spatial point pattern on the observation window $[0,1] \times [0,1]$.

```
{
  t = rep(0,s)
  for (i in 1:s)
    {
      y.coord = runif(2*n)
      y = matrix(y.coord,n,2)
      t[i] = sum(nndistG(y)\$dists)
    }
  t
}
```

Figure 2.13 shows the histogram of the test-statistic for 1000 different simulated data under the null-hypothesis. It can be seen that the observed value is very large compared to the null-distribution (only 3 data sets gave higher values for T). This suggests strong evidence against H_0 and indicates that the points show mutual repulsion. This characteristic could be incorporated in a more complex model, which allows for interaction between the points. That interaction could be represented by a parameter in the model and be estimated from the data.

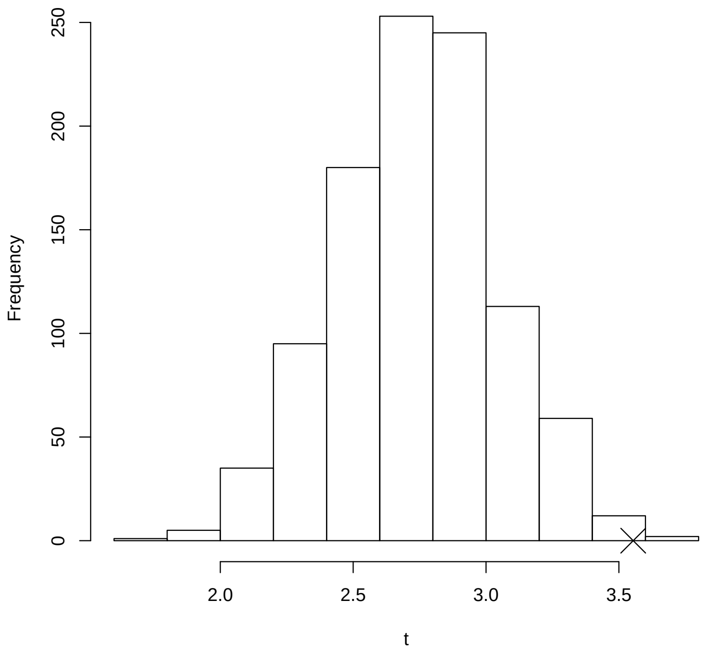

Figure 2.13: Histogram of 1000 values of the test-statistic under the null-hypothesis of random scattering. The "x" denotes the observed value of the statistic.

Chapter 3

The Bootstrap

3.1 Notation

This section sets up some basic notation and definitions. F will denote the distribution function of a real random variable X. Expectations of scalar functions $g(X)$ with respect to F are denoted by

$$E(g(X); F) = \int g(x)dF(x). \tag{3.1}$$

When F admits a probability density function $f(x)$, (3.1) simplifies to

$$\int g(x)f(x)dx,$$

whereas when F is a discrete probability distribution, such that $\Pr[X = x_j] = p_j$, (3.1) becomes

$$\sum_j g(x_j)p_j.$$

Given a set of observations $x_1, \ldots, x_n$, we denote the ordered sample as $x_{(1)} \leq \ldots \leq x_{(n)}$. We define the Empirical Distribution Function (EDF) through

$$\hat{F}(x) = \frac{1}{n}\sum_{i=1}^{n} 1[x_{(i)} \leq x] \tag{3.2}$$

where $1[x \leq y]$ is 1 if $x \leq y$ and 0 otherwise. Notice that the values of the EDF are fixed $\{0, 1/n, 2/n, \ldots, n/n\}$ so the EDF is characterized by its points of increase, the ordered values $x_{(1)} < \cdots x_{(n)}$ of the data. The EDF defines a discrete probability measure with support points on the observed data x_i. Therefore, when we write that $X \sim \hat{F}$ we mean that $\Pr(X = x_i) = 1/n, i = 1, \ldots, n$.

If the observations x_i are independent realizations of a random variable X with distribution F, then it is reasonable and it can be shown that $\hat{F}$ approaches F as the sample size n increases. This is illustrated in Figure 3.1, where we assume that the x_is come from a standard normal distribution. On the left, the EDF based on 10 data points is a rather poor approximation to the true F, whereas on the right the EDF based 1000 data points is almost identical to F.

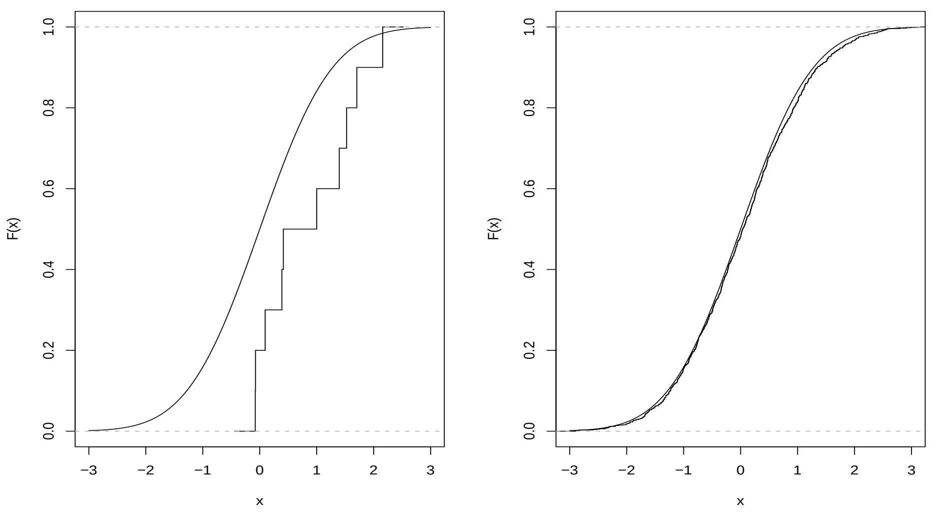

Figure 3.1: Comparison of true and empirical distribution functions, based on $n = 10$ (left) and $n = 1000$ (right) observations from the $N(0, 1)$ distribution.

3.2 Bootstrap estimation

Assume that we have a set of i.i.d observations $x_1, \ldots, x_n$ from some distribution F. In parametric modelling F is assumed to belong to some parametric family and it is characterized by few parameters ψ, therefore $F = F_\psi$. For example we might assume that F is a $N(\mu, \sigma^2)$ distribution, thus $\psi = (\mu, \sigma^2)$. The fitting of the model to the data is done by estimating (by maximum likelihood for example) ψ. Consequently, our estimate of F is $F_{\hat{\psi}}$, where $\hat{\psi}$ is the estimate of ψ.

In non-parametric statistics, we wish to make minimal assumptions about the unknown distribution F of the data. In this context, the EDF introduced earlier, plays the role of fitted model when no mathematical form has been assumed for F.

We are often interested in estimating characteristics of the population that the data come from. We will denote such characteristics by θ. Typically θ can be seen as a function of the (unknown) distribution F, and we denote that by writing $\theta = c(F)$, where c is the function of interest. For example,

$$\theta = \int x dF(x)$$

is the population mean, and

$$\theta = F^{-1}(0.5)$$

is the population median. Since F is unknown, θ is unknown as well and it is estimated using a statistic T, which is called an estimator. A statistic is a function $T = t(X_1, \ldots, X_n)$ and its observed value $t = t(x_1, \ldots, x_n)$ for a given sample is called the estimate of θ. Typically, the function $t(\cdot)$ depends on $x_1, \ldots, x_n$ only through their ordered values, which essentially means that $t(\cdot)$ depends on the sample only through the EDF $\hat{F}$, therefore we write $t = t(\hat{F})$. It is also very often the case that the same function $t(\cdot)$ applied to F gives θ, that is $\theta = t(F)$. We now give an example.

Example 3.1. *Non-parametric estimation of a population mean*

Let θ be the mean of a population described by the (unknown) distribution function F. Then, by definition

$$\theta = t(F) = \int x dF(x)$$

where we define $t(F) = \int x dF(x)$. For a sample $x_1, \ldots, x_n$ from this population, the sample mean

$$\overline{x} = \sum_{i=1}^{n} x_i \frac{1}{n}$$

is an estimate of θ. We will now show that

$$\overline{x} = \int x d\hat{F}(x) = t(\hat{F}).$$

We said earlier that $\hat{F}$ corresponds to a discrete probability measure which assigns probability $1/n$ to $x_i, i = 1, \ldots, n$. Therefore if $X \sim \hat{F}$ then

$$E(X; \hat{F}) = \sum_{i=1}^{n} x_i \Pr(X = x_i) = \sum_{i=1}^{n} x_i \frac{1}{n} = \overline{x}$$

but by definition

$$E(X; \hat{F}) = \int x d\hat{F}(x).$$

Therefore $\overline{x} = t(\hat{F})$.

Example 3.2. *Estimation of the mean of a normally distributed population*

We now make the assumption that F is known to belong to the normal family with unknown mean θ and known variance 1, $F = F_\theta \equiv N(\theta, 1)$. By definition,

$$\theta = \int x dF_\theta(x) = t(F).$$

On the other hand, the MLE of θ is $\overline{x}$ and the estimate of F is $F_{\overline{x}} \equiv N(\overline{x}, 1)$. Again by definition,

$$\overline{x} = \int x dF_{\overline{x}}(x) = t(F_{\overline{x}}).$$

Bootstrap is a method of estimating a function of the unknown distribution F. The bootstrap estimate of $c(F)$, where $c(\cdot)$ is a function of F (e.g. the mean or the median), is $c(\hat{F})$, where $\hat{F}$ is either a parametric ($F_{\hat{\psi}}$) or a non-parametric (the EDF) estimate of F, based on data $x_1, \ldots, x_n$. When $\hat{F}$ is a parametric estimate, we use the term parametric bootstrap, whereas when the EDF is used we term the method non-parametric bootstrap. We have already seen a bootstrap estimate: the sample mean is the bootstrap estimate of the population mean.

3.3 Moment, distribution and quantile estimates of estimators

Bootstrap methods are very useful in approximating distributional characteristics of estimators. Assume that $T = t(X_1, \ldots, X_n)$ is an estimator of θ, where the X_is are i.i.d from F, and

$t = t(x_1, \ldots, x_n)$ is the estimate based on a sample $x_1, \ldots, x_n$. Clearly, the distribution of T depends on F, since T is a function of the X_is. We are often interested in properties of the estimator, such as its bias in estimating θ

$$b(F) = E(T; F) - \theta = \int t(y_1, \ldots, y_n) dF(y_1) \cdots dF(y_n) - \theta$$

and its variance

$$v(F) = \mathrm{Var}(T; F) = \int (t(y_1, \ldots, y_n) - \theta - b(F))^2 dF(y_1) \cdots dF(y_n).$$

Knowledge of this quantities can be used to construct confidence intervals for θ. In many problems, the above quantities are unknown. The bootstrap estimates of bias and variance are obtained by replacing F by $\hat{F}$ in the above expressions, therefore the bias estimate is

$$b(\hat{F}) = E(T; \hat{F}) - t = \int t(y_1, \ldots, y_n) d\hat{F}(y_1) \cdots d\hat{F}(y_n) - t$$

and the variance estimate is

$$v(\hat{F}) = \mathrm{Var}(T; \hat{F}) = \int (t(y_1, \ldots, y_n) - t - b(\hat{F}))^2 d\hat{F}(y_1) \cdots d\hat{F}(y_n).$$

Similarly, bootstrap can be used to estimate quantiles of the distribution of T and probabilities such as $\Pr(T \leq u)$ for any value u.

Example 3.3. *Revisit Example 3.1: The bootstrap estimate of the bias of the non-parametric estimator of the population mean.*

Let $X_i, i = 1, \ldots, n$ be i.i.d variables with common distribution F and mean θ. In example 3.1 we showed that the bootstrap estimator of θ is $T = \sum_{i=1}^n X_i/n$. In this case we can show analytically that the bias of T is 0, since $E(X_i) = \theta$. Nevertheless, as an illustration we also find the bootstrap estimate of the bias of T:

$$\begin{aligned} b(\hat{F}) &= E(T; \hat{F}) - t \\ &= E(\sum_{i=1}^n X_i/n; \hat{F}) - \overline{x} \\ &= (1/n) \sum_{i=1}^n E(X_i; \hat{F}) - \overline{x} \qquad \text{(by exploiting the linearity of expectation)} \\ &= (1/n) \sum_{i=1}^n \left\{ (1/n) \sum_{j=1}^n x_j \right\} - \overline{x} \qquad \text{(by definition of } E(X_i; \hat{F})\text{)} \\ &= (1/n) \sum_{j=1}^n x_j - \overline{x} \\ &= 0. \end{aligned}$$

Example 3.4. *Revisit Example 3.2: The bootstrap estimate of the bias of the parametric estimator of the population mean.*

The same calculations as in the previous example show that the bootstrap estimator of θ, for data coming from a $N(\theta, 1)$ distribution, is unbiased. In this case we use the fact that $E(X_i; \hat{F}) = E(X_i; F_{\overline{x}}) = \overline{x}$.

3.4 Linking bootstrap with Monte Carlo methods

The bootstrap estimate of an expectation $E(g(T); F)$, where $T = t(X_1, \ldots, X_n)$ is an estimator and $g(\cdot)$ some function, is given by

$$E(g(T); \hat{F}) = \int g(t(y_1, \ldots, y_n)) d\hat{F}(y_1) \cdots d\hat{F}(dy_n). \tag{3.3}$$

In Examples 3.3 and 3.4 we showed that this expectation can be easily calculated analytically for some choices of $t(\cdot)$ and $g(\cdot)$ (especially when $t(\cdot)$ is a linear function of the X_is). However, in many other cases this calculation would be very difficult or impossible. In parametric bootstrap, i.e when $F = F_{\hat{\psi}}$, the calculation is likely to be impossible, since it involves an n-dimensional integration. In non-parametric bootstrap, however, the integration in (3.3) is in fact a sum: recall that $\hat{F}$ corresponds to a discrete probability measure with support on $\{x_1, \ldots, x_n\}$. Specifically,

$$E(g(T); \hat{F}) = \frac{1}{N} \sum_{j=1}^{N} g\left(t(\mathbf{y}^{*\mathbf{j}})\right) \tag{3.4}$$

where $\mathbf{y}^{*\mathbf{j}}, j = 1, \ldots, N$, are all possible different n-tuples with elements from the set $\{x_1, \ldots, x_n\}$. There are $N = n^n$ such different n-tuples, since each of the position in the tuple could be filled with any of the n elements of $\{x_1, \ldots, x_n\}$. We give an example.

Example 3.5.

Suppose that $n = 2$ and the observed sample is x_1, x_2 $(x_1 \neq x_2)$. There are $N = 2^2 = 4$ different 2-tuples with elements in the set $\{x_1, x_2\}$: $(x_1, x_1), (x_1, x_2), (x_2, x_1), (x_2, x_2)$. If we sample with replacement from $\{x_1, x_2\}$ each of the above 2-tuples occurs with probability 1/4. Therefore,

$$E(t(X_1, X_2); \hat{F}) = \frac{1}{4}[t(x_1, x_1) + t(x_1, x_2) + t(x_2, x_1) + t(x_2, x_2)]$$

(check for example that for $t(X_1, X_2) = (X_1 + X_2)/2$, the above calculation gives you $(x_1 + x_2)/2$ which is the answer according to Example 3.3).

Therefore, in principle we could calculate the non-parametric bootstrap expectation, although it is a very computationally demanding calculation. For example, even for a small number of n, say $n = 50$, $N > 8 \times 10^{84}$, therefore enumeration of all possible $\mathbf{y}^{*\mathbf{j}}$s is infeasible.

In summary, calculating bootstrap expectations often involves intractable integrations over the distribution of n i.i.d random variables. However, in Section 2.7 we encountered exactly this situation and we used Monte Carlo integration to approximate the integrals. Therefore, we approximate (3.3) by

$$E(g(T); \hat{F}) \approx \frac{1}{m} \sum_{j=1}^{m} g\left(t(\mathbf{y}^{*\mathbf{j}})\right) \tag{3.5}$$

where $\mathbf{y}^{*\mathbf{1}}, \mathbf{y}^{*\mathbf{2}}, \ldots, \mathbf{y}^{*\mathbf{m}}$ are m samples each consisting of n independent realizations from $\hat{F}$. Each of the $\mathbf{y}^{*\mathbf{j}}$s is called a bootstrap sample. The following algorithm shows how to obtained a bootstrap sample of size n:

```
For i=1:n
   simulate $y_i^* \sim \hat{F}$
Set $\mathbf{y}^* = \{y_1^*, \ldots, y_n^*\}$
```

Sampling from the distribution of the estimator

For every bootstrap sample $\mathbf{y}^{*\mathbf{j}}$, we can calculate $t^*_j = t(\mathbf{y}^{*\mathbf{j}})$. Applying the argument of Section 2.6.2, it is easy to see that $t^*_1, \ldots, t^*_m$ is a sample from the distribution of the estimator T under $\hat{F}$. We are primarily interested in the distribution of T under F, but as long as $\hat{F}$ is a good approximation of F, the sample of the t^*_js can give us an idea about the distribution of the estimator.

Monte Carlo methods can also be used to estimate quantiles of the bootstrap estimate of the distribution of T. Suppose for example that we are interested in the qth quantile of T. The bootstrap estimate of this quantile is obtained by replacing F by $\hat{F}$, but still the calculation might be very difficult since the distribution of T under $\hat{F}$ is likely to be intractable. Nevertheless, the bootstrap estimate can be very easily approximated using Monte Carlo: estimate the qth quantile by $t^*_{(m\times q)}$, where $t^*_{(m\times q)}$ is the $(m \times q)$th ordered value among the simulated values $t^*_1, \ldots, t^*_m$ (assuming that $m \times q$ is an integer).

As we saw, we need to simulate variates from $\hat{F}$ in order to obtain a bootstrap sample. In parametric bootstrap, $\hat{F} = F_{\hat{\psi}}$ is some parametric distribution and the techniques of Chapter 2 can be employed to simulate from $\hat{F}$ (actually most statistical packages contain built-in functions for simulations from all common parametric distributions). In non-parametric bootstrap, $\hat{F}$ is the EDF, which corresponds to a discrete probability measure which assigns probability $1/n$ to each of the observed data $x_i, i = 1, \ldots, n$ (see Section 3.1). Therefore, a draw from the EDF is obtained simply by uniform sampling from the discrete set $\{x_1, \ldots, x_n\}$.

We can increase the accuracy of the Monte Carlo estimates as in (3.5) by taking large values of m, and as $m \to \infty$ the Monte Carlo error is eliminated. Nevertheless, (3.3) is still an estimate of $E(g(T); F)$ and the statistical error of estimating F by $\hat{F}$ can only be reduced by obtaining more data, thus increasing the sample size n.

The Monte Carlo implementation of the bootstrap is really very simple and the following sections give several examples. The following R-function `bootstrap` implements the non-parametric bootstrap. It takes as input the observed sample (`x`), the number m of required bootstrap samples (`nboot`) and the function $t(\cdot)$ (`theta`) we wish to estimate. The R-function returns the bootstrap estimate $t(\hat{F})$ (`z$est`), the Monte Carlo estimate of the bootstrap estimate of bias (`z$bias`), the Monte Carlo estimate of the bootstrap estimate of the standard error of the estimator (`z$se`) and the sample of the t^*_js (`z$distn`):

```
bootstrap=function(x, nboot, theta, ...) {
        z = list()
        data = matrix(sample(x, size = length(x) * nboot, replace = T), nrow
                 = nboot)
        bd = apply(data, 1, theta, ...)
        est = theta(x, ...)
        z$est = est
        z$distn = bd
        z$bias = mean(bd) - est
        z$se = sqrt(var(bd))
        z
}
```

Remark!!! Since most of the times the bootstrap is implemented using Monte Carlo, instead of saying “the Monte Carlo estimate of the bootstrap estimate” we will simply say “the bootstrap estimate”.

We now look at an example.

Example 3.6. *MINITAB gives a data set representing the observed daily activity time, in hours, of a group of salamanders. These data are stored in the vector* `sal.dat`*. A histogram of the data is shown in Figure 3.2.*

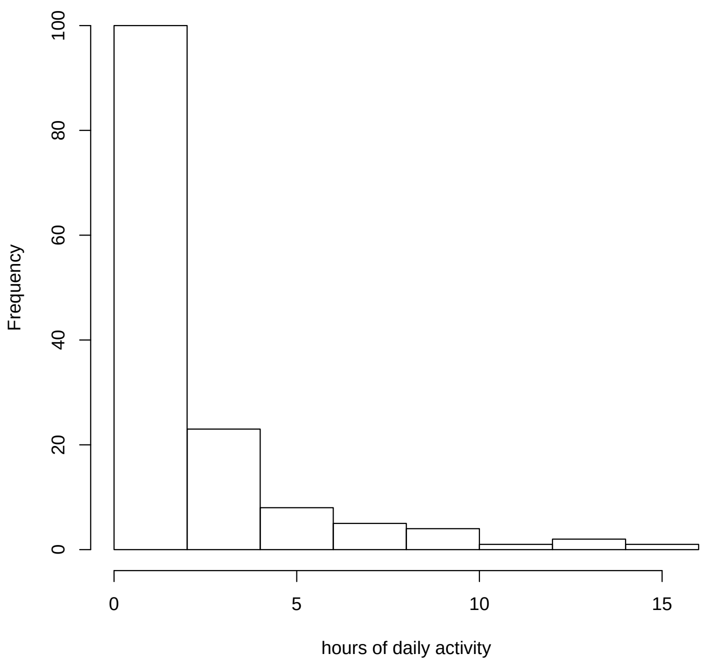

Figure 3.2: Histogram of salamander data

A distinctive characteristic of the data is that most of the salamanders have zero or very few hours of activity per day, but there are a few outlying ones which remain active for many hours per day. We are interested in the median activity time of the population, which we denote by θ, and we assume that the observed sample is a random sample from this population. The R-command

```
bootstrap(sal.dat,200,median)
```

gives the bootstrap estimate of θ, which is the sample median, $\hat{\theta} = 0.8$. Moreover, based on 200 bootstrap samples, the bootstrap estimate of the bias is 0.0065, and the bootstrap estimate of the standard error of the estimator is 0.2414. A histogram of the bootstrap realisations of the sample median for the salamander data is given in Figure 3.3.

The bootstrap technique can be applied almost as easily to slightly more complicated data sets. For example, F might be a multivariate distribution, and each of the x_is a vector of observations.

Example 3.7. *We consider here a set of data (given by Efron) relating two score tests, LSAT and GPA, at a sample of 15 American law schools. Of interest is the correlation θ between these measurements.*

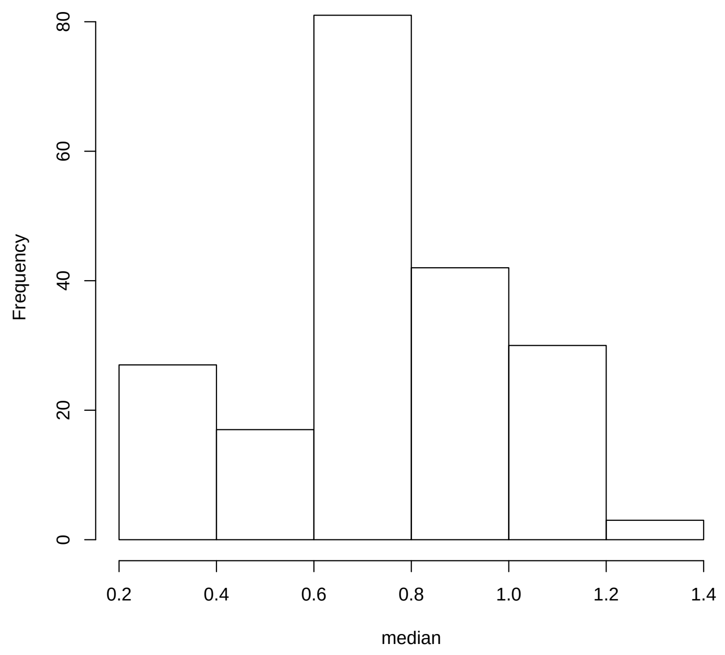

Figure 3.3: Histogram of bootstrap sample medians

The data are given as

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

and are plotted in Figure 3.4. They are stored in R as the matrix `law.dat`.

In this example, F is a bivariate distribution and each observation x_i consists of the values for the two tests, $x_i = (z_{i1}, z_{i2})$. The assumption is that the x_is are i.i.d realizations of the random variable $(Z_1, Z_2) \sim F$. In this case θ is the correlation between Z_1 and Z_2. The bootstrap estimate of θ is the sample correlation, $\hat{\theta} = 0.776$. Non-parametric bootstrap samples

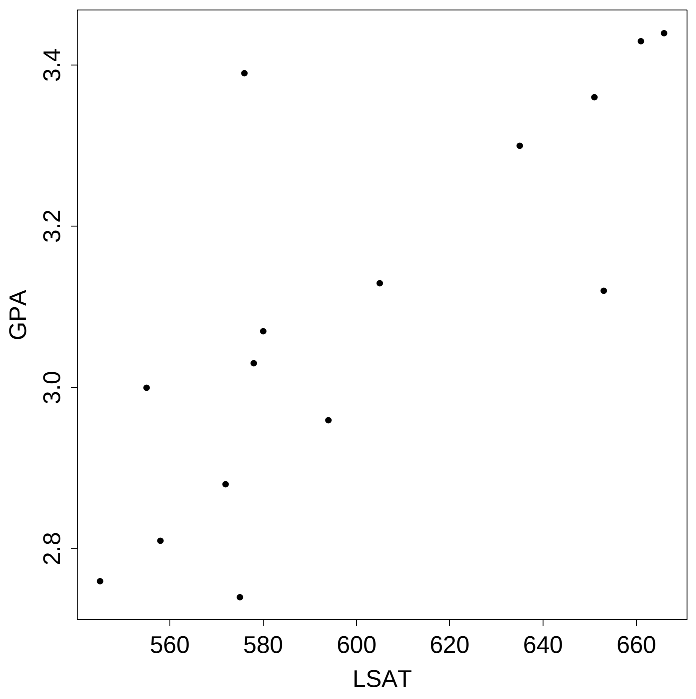

Figure 3.4: Scatterplot of law school data

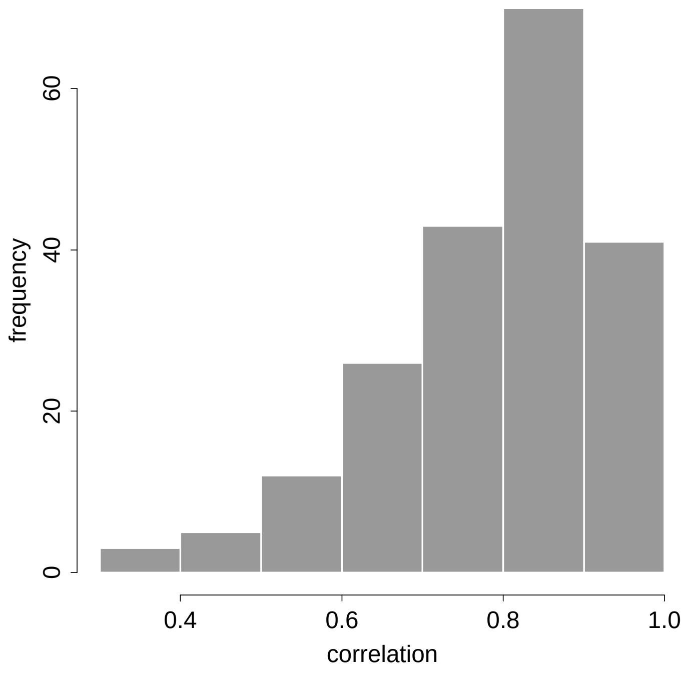

Figure 3.5: Histogram of bootstrap correlations

are obtained as before, by sampling n i.i.d draws from $\hat{F}$, which in this example means sampling from the rows of the data matrix. This requires care in the use of our `bootstrap` R function, since there it was anticipated that `x` would be a vector. We get round this problem by setting `x` to be the vector `1:n`, in this case, `1:15`. Then the data matrix is passed to the function `theta`, which in this case evaluates the correlation coefficient, as an additional argument. Thus, we need

```
bootstrap(1:15,nboot,theta1,law.dat)
```

where

```
theta1=function(x,xdata){ cor(xdata[x,1],xdata[x,2]) }
```

A histogram of 200 bootstrap simulations of $\hat{\theta}$ is given in Figure 3.5 and the sample standard error of these values is found to be 0.135. Note in particular the skewness of the bootstrap distribution.

3.5 Bootstrap confidence intervals

Generally care is needed in constructing a bootstrap confidence interval (CI) for a parameter θ, since there are several different methods. A note of warning should be given: the same procedures appear with different names in the literature, and different procedures are given the same name by different people! This section sketches some of the options but you are certainly advised to look at the relevant suggested literature for more details.

Normal/Student-t type of intervals

Many statistical estimators $T = t(X_1, \ldots, X_n)$ obey a central limit theorem (Section 2.3) which suggests that for large n, we can assume that

$$\frac{T(X_1, \ldots, X_n) - \theta - b(F)}{v(F)^{1/2}} \tag{3.6}$$

is approximately distributed as $N(0, 1)$, where $b(F), v(F)$ are the bias and the variance respectively of the estimator (Section 3.3). Therefore, if z_α denotes the α-quantile of the normal distribution,

$$(t - b(F) - z_{1-\alpha} v(F)^{1/2}, t - b(F) - z_\alpha v(F)^{1/2}) \tag{3.7}$$

will be for large n an approximate $(1 - 2\alpha)$ CI for θ. Typically, the bias or the variance or both will be unknown, but bootstrap estimates can be readily obtained, $b(\hat{F}), v(\hat{F})$ say, and a bootstrap CI can be found by plugging these estimates into (3.7). When $v(F)$ is unknown and a bootstrap estimate is used, the normal quantiles can/should be replaced by Student-t quantiles with $n - 1$ degrees of freedom.

Using quantiles of the pivotal distribution

Bootstrap can be used to by-pass the asymptotic normality arguments in order to obtain CIs. Recall that the usual method for constructing a CI for a parameter θ, estimated by an estimator T, is to find some function of T and θ which is independent of θ and has a known distribution. This function, denoted here by $\Delta = h(T, \theta)$, is known as a *pivot*. For example, in a parametric framework, if we know that the data are normally distributed then the expression (3.6) gives the appropriate function. Bootstrap samples from the distribution of Δ can be easily obtained:

```
Construct $m$ bootstrap samples $\mathbf{y}^{*\mathbf{1}}, \ldots, \mathbf{y}^{*\mathbf{m}}$
Compute $t_j^*, j = 1, \ldots, m$, where $t_j^* = t(\mathbf{y}^{*\mathbf{j}})$
Compute $\Delta_1^* = h(t_1^*, t), \ldots, \Delta_m^* = h(t_m^*, t)$, where $t$ is the bootstrap estimate of $\theta$
$\Delta_1^*, \ldots, \Delta_m^*$ is a bootstrap sample from the distribution of $\Delta$.
```

The quantiles of the distribution of Δ can be used to construct CIs. For example, suppose that $\Delta = T - \theta$. Using bootstrap, we approximate that αth and $(1-\alpha)$th quantiles of the distribution of Δ by $\hat{f}_\alpha = \Delta^*_{(m\alpha)}$ and $\hat{f}_{1-\alpha} = \Delta^*_{(m(1-\alpha))}$ respectively. Then, it is approximately true that $\Pr(\hat{f}_\alpha < \Delta < \hat{f}_{1-\alpha}) = 1 - 2\alpha$, thus an approximate equisided $1 - 2\alpha$ CI for θ is

$$[t - \hat{f}_{1-\alpha}, t - \hat{f}_\alpha] = [t - (t^*_{(m(1-\alpha))} - t), t - (t^*_{(m\alpha)} - t)].$$

The success of this method depends on whether the distribution of Δ is actually independent of θ. Davinson and Hinkley give recommendations for how to find pivots and we refer to that book for further details.

Example 3.8. *Revisit Example 3.7: 95% CIs for the law school data.*

We now try the different techniques in order to find a CI for the correlation θ between the law exam scores. In this example, $n = 15$, $t = \hat{\theta} = 0.776$, $b(\hat{F}) = -0.0065, v(\hat{F}) = 0.133^2$ (based on $m = 10000$ simulations). The CI based on the normal approximation is $[0.52, 1.04]$ and the corresponding Student-t is $[0.5, 1.07]$. The approximate nature of the CIs manifests itself in the intervals containing values larger than 1, which is impossible for θ. If we take $\Delta = T - \theta$, the bootstrap CI is $[0.59, 1.09]$.

We now look at bootstrap methods applied on specific problems.

3.6 Regression models

The example of this section has some similarities with the law data Example 3.7, where F was a bivariate distribution. A regression–type model has the structure

$$y_i = f(x_i; \beta) + \epsilon_i \tag{3.8}$$

where f is a specified function acting on the covariates, x_i, with parameters β, and ϵ_i is a realization from a specified error distribution. The aim is to fit the model (3.8) and obtain standard errors for the parameters without making specific distributional assumptions about the error distribution.

As a working example we will use the Australian timber data contained in the file `wood.dat`. These data relate the air-dried density of timbers with the modulus of rigidity (hardness); a plot of the data is shown in Figure 3.6. The model we will fit is a simple linear regression model

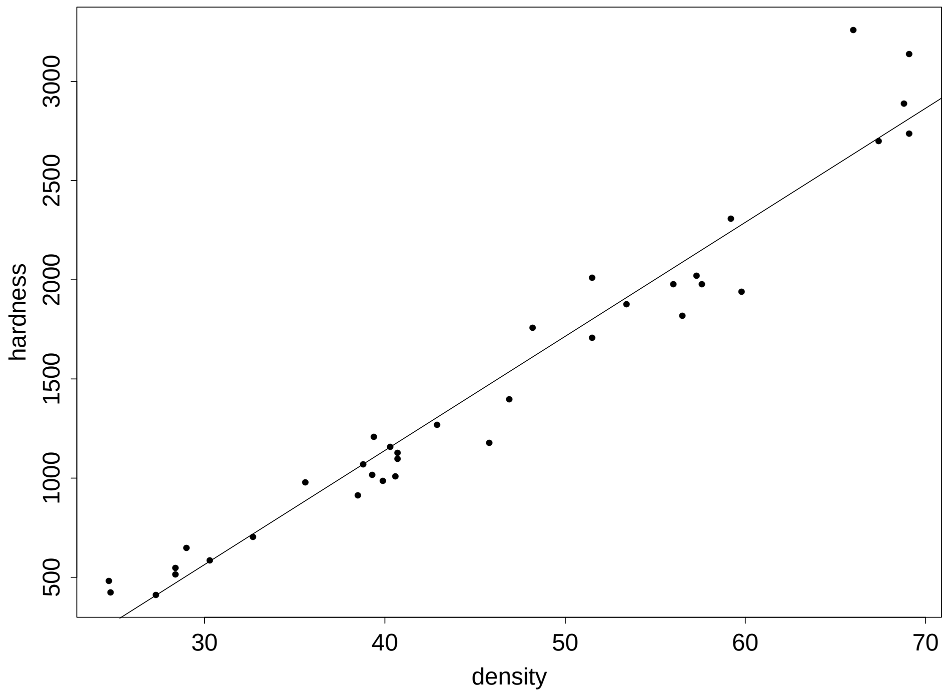

Figure 3.6: Hardness versus density of Australian timbers

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \tag{3.9}$$

but without making any distributional assumptions about the ϵ_i. It is easy to obtain estimates about the regression parameters. Recall that the least squares method is a non-parametric fitting technique. The least squares fit to these data gives $\hat{\beta}_0 = -1160.5$ and $\hat{\beta}_1 = 57.51$ and the fitted line is plotted in Figure 3.6.

Standard errors and CIs for the parameters, however, are typically obtained based on asymptotic normality arguments. On the other hand, bootstrap methods can be used for this purpose as a non-parametric alternative. Within this model framework there are two alternative ways to use bootstrap:

1. Re-sample from the rows of the data matrix, i.e re-sample pairs (x_i, y_i), re–fit the model to each bootstrap sample, form the bootstrap distribution of the estimator $\hat{\beta}$, and from this derive standard errors and CIs as shown in the previous sections.
2. Fit the regression model, form the empirical distribution function of the residuals, generate bootstrap replications of the data by substitution into (3.8), re–fit the model to each realization to obtain bootstrap distribution of $\hat{\beta}$.

The first approach is similar to what we did for the law data in Example 3.7: we treat each (x_i, y_i) as a single observation from a bivariate unknown distribution F. This is not very sensible when the covariates x_i have been pre-fixed to specified values at the study design stage.

The second approach exploits the specific model structure, but it makes more assumptions. Specifically, it assumes that the variance of the residuals does not depend on x (this is sometimes called the homoscedasticity assumption). The fact that we assume more in the second approach has the usual statistical pros and cons: if the assumptions are correct we have a much more efficient method, if the assumptions are wrong we can get much more biased results.

For our example, to apply the first approach, we use the `bootstrap` function with `theta5` defined as:

```
theta5=function(x, xdata) {
        ls = lsfit(xdata[x, 1], xdata[x, 2])
        abline(ls)
        ls$coef
}
```

A graph of 10 bootstrap realisations of the regression line is given in Figure 3.7.

The bootstrap distribution of the two regression parameters is given in Figure 3.8. The means and standard deviations of these empirical distributions are $(-1157.5, 57.3)$ and $(115.7, 2.91)$ respectively.

To apply the second method, we call the `bootstrap` function as

```
bootstrap(resids,10,theta6,xdata=wood.dat)
```

where `resids` contains the residuals from the original fit, with `theta6` defined as

```
theta6=function(x, xdata) {
        y = x + xdata[, 2]
        ls = lsfit(xdata[, 1], y)
```

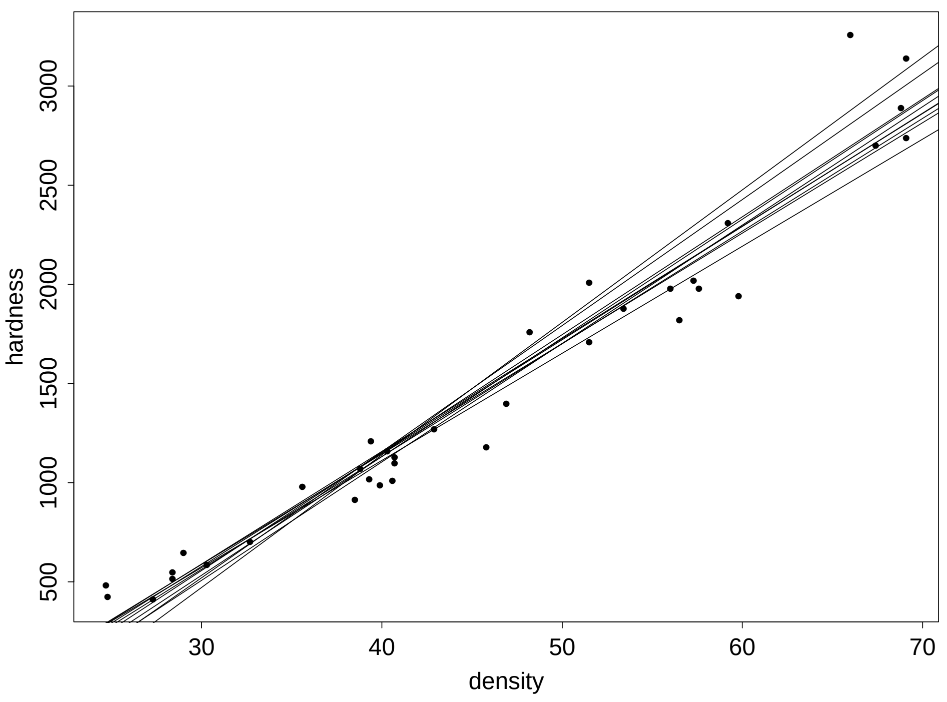

Figure 3.7: Bootstrap regression lines

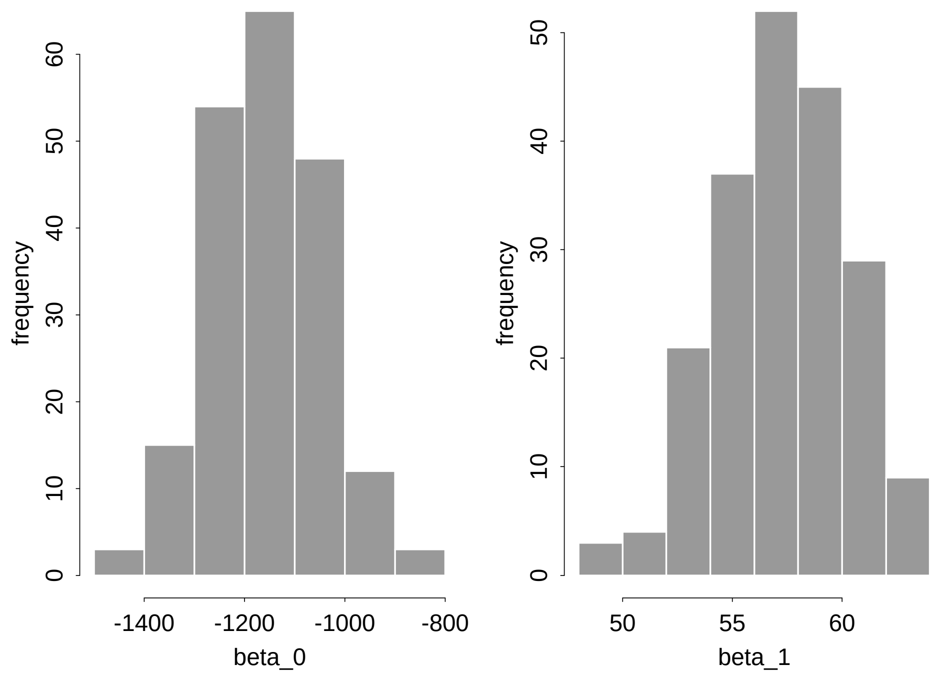

Figure 3.8: Bootstrap distributions of regression parameters

```
        abline(ls)
        ls$coef
}
```

Plots corresponding to Figures 3.7 and 3.8 are given in Figures 3.9 and 3.10. In this case the respective means and standard deviations are $(-1158.8, 57.5)$ and $(96.1, 2.04)$.

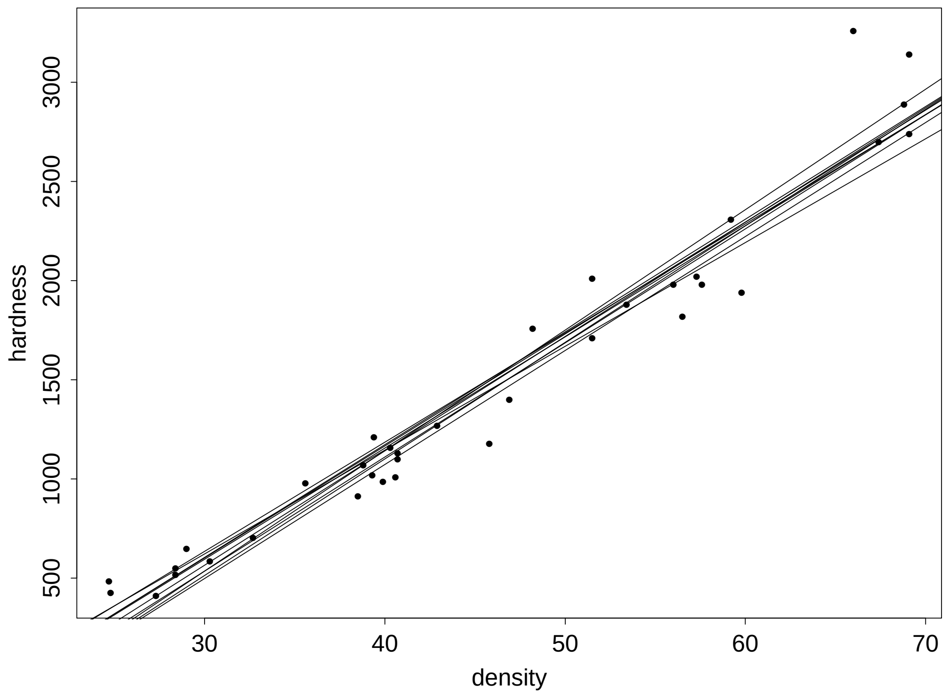

Figure 3.9: Bootstrap regression lines using second approach

3.6.1 Non–parametric regression

This section is mainly included for an illustration of bootstrap to even more complicated settings. Do not worry if you don't understand all the details.

The purpose is to fit regression models like (3.8) but where, not only the error distribution, but also the function f remain un-specified. Therefore, we assume that the pairs (x_i, y_i) are mutually independent, and there is some (typically smooth) function f, such that $E(Y \mid X) = f(X)$. This setting is known as non-parametric regression, and there is a vast literature on this topic (see for example Silverman, 1985).

We consider a well–studied data set of Silverman (1985), giving measurements of acceleration against time for a simulated motorcycle accident. The data are shown in Figure 3.11 and is contained in the file `mc.dat`. Clearly the relationship is non–linear, and has structure that will not easily be modelled parametrically. R provides a number of different routines for non–parametric regression — here we will use just one: `loess`. (To use this package you need to first of all load the library modreg using the command `library(modreg)`). The details don't matter

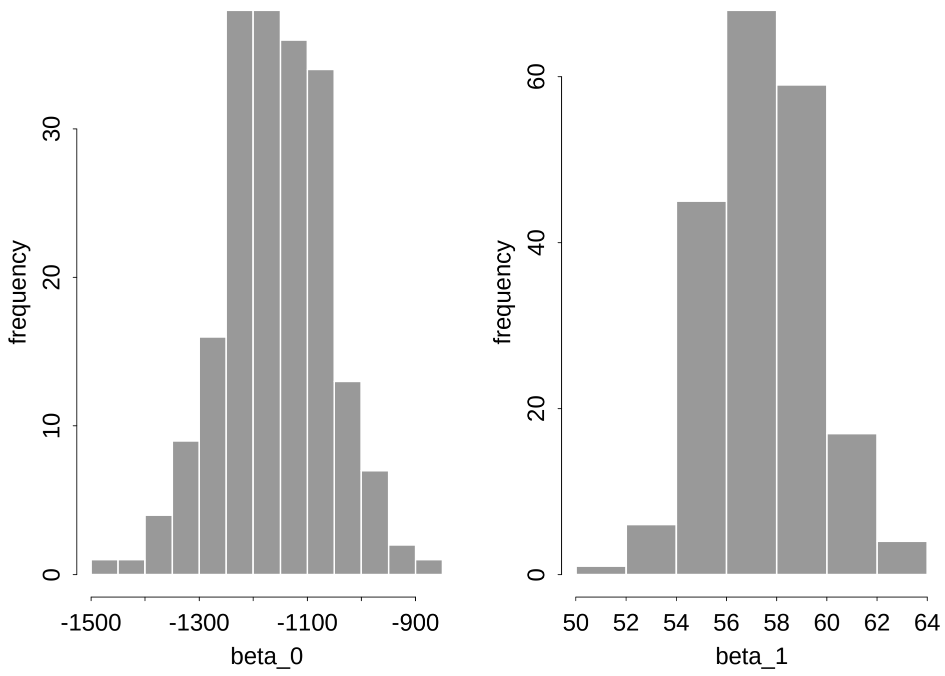

Figure 3.10: Bootstrap distributions of regression parameters using second approach

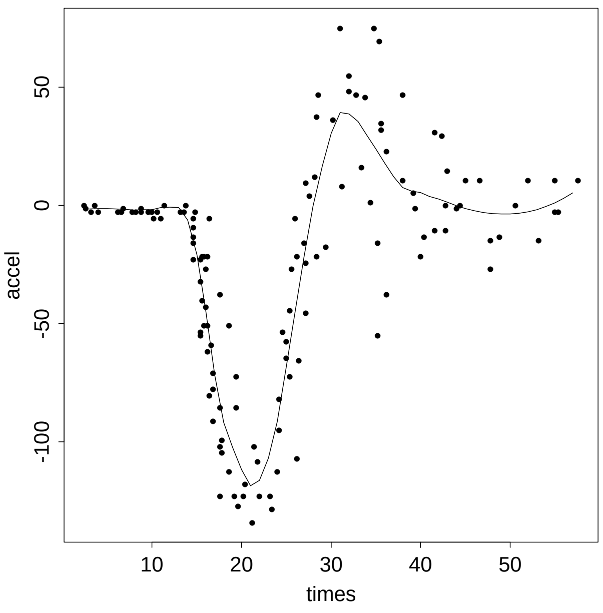

Figure 3.11: Scatterplot of motorcycle data

too much, but in outline the fit is a locally weighted least–squares regression. The smoothing is determined by an argument `span`, which determines the proportion of data to be included in the moving window which selects the points to be regressed on. With `span = 1/3`, the `loess` fit to the motorcycle data is included in Figure 3.11.

Because of the non–parametric structure of the model, classical approaches to assessment of parameter precision are not available. The challenge again in bootstrapping is to create bootstrap realizations of the data which have similar stochastic structure to the original series.

This is achieved exactly as in the simpler regression problem considered earlier. We prefer the first method, since from Figure 3.11 it is evident that the variance of the residuals varies with x. Thus, we simply bootstrap the pairs (x, y) in the original plot and fitting `loess` curves to each simulated bootstrap series. In R we simply use

```
bootstrap(1:133,20,theta4,mc.dat)
```

where `mc.data` contains the data, and

```
theta4=function(x, xdata) {
        mc.lo = loess(xdata[x, 2] ~ xdata[x, 1], span = 1/3)
        y = predict.loess(mc.lo, 1:60)
        lines(1:60, y)
}
```

Applying this to the motorcycle data gave the `loess` curves in Figure 3.12. In this way we can

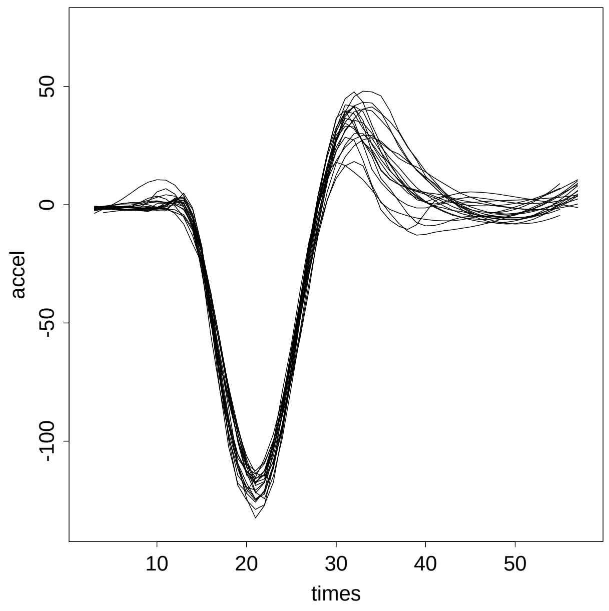

Figure 3.12: Bootstrap realizations of loess fits to motorcycle data

assess both pointwise and globally the variability in the original `loess` fit.

3.7 Dependent data

In all the examples we have seen so far, we have assumed that the data are i.i.d replications from some (potentially multivariate) distribution F. Bootstrap can be used when this assumption is relaxed, but the problem is not straightforward anymore. If $x_1, \ldots, x_n$ is a realization from an n-dimensional distribution F, then we have sample of size 1 from F. Therefore, the techniques described earlier cannot apply anymore. But if we are prepared to make some assumptions about the dependence structure among the x_is then bootstrapping can be used.

3.8 The Jackknife

We mention briefly that bootstrapping isn't the only scheme for estimating the statistical properties of estimators by re-sampling. A slightly older idea is that of jackknifing. The idea is similar to bootstrapping, but instead of re-sampling from the original sample, we apply the estimator systematically to all samples obtained from the original sample with just one sample value deleted.

Chapter 4

The EM algorithm

4.1 Introduction

In its basic form, the EM algorithm is a *deterministic* algorithm designed to find maximum likelihood estimates (or posterior modes) in situations where there are missing/incomplete data.

The algorithm is guaranteed to increase the likelihood at every iteration, thus it converges to a local maximum of the likelihood function under very mild conditions. Moreover, in many situations it is very easy to implement the EM algorithm. These two characteristics often make the EM algorithm a more attractive option to other optimization methods (such as the Fisher scoring or the Newton-Raphson). Hence, although there might not be any sort of missing or incomplete data, it is common to try to express a problem as a missing data problem in order to be able to use the EM. We will give such an example from the area of genetics.

The EM algorithm is closely connected to the *data augmentation* algorithm, which is a *stochastic* algorithm and a special case of Markov chain Monte Carlo methods, covered in Math457.

I have tried to keep consistency with the notation used in the literature.

4.2 Missing/incomplete data problems

Quite generally, a missing data problem can be described as follows: a statistical model is specified for X by the density $f(X \mid \theta)$, where θ are the parameters of interest. However, we only observe $Y = h(X)$, where $h(\cdot)$ is a many-to-one function, therefore we only have incomplete information about X. We denote by $\mathcal{X}(Y)$ the set of all X compatible with the observed Y, in the sense that $h(X) = Y$ for all $X \in \mathcal{X}(Y)$. The density of the observed data is denoted by $f(Y \mid \theta)$ and the term *observed likelihood* is used for

$$L(\theta \mid Y) = f(Y \mid \theta),$$

whereas the term *complete-data likelihood* is used for

$$L(\theta \mid X) = f(X \mid \theta).$$

Slightly more specifically, in many applications we can decompose X as $X = (Y, Z)$ and assume that only Y is observed, therefore Z represents the missing variables. Then,

$$L(\theta \mid Y) = f(Y \mid \theta) = \int f(Y, Z \mid \theta) dZ = \int L(\theta \mid X) dZ. \tag{4.1}$$

The MLE of θ is derived by maximizing $L(\theta \mid Y)$. However, there might be two problems with this approach: i) although $L(\theta \mid X)$ might have a simple form (for example in the exponential family), $L(\theta \mid Y)$ might be difficult or impossible to derive due to the integration involved in (4.1), ii) although $L(\theta \mid Y)$ is possible to evaluate, it is still hard to maximize. The EM algorithm can be used when either of this problems arise.

Example 4.1. *Censored data*

Suppose $X = (x_1, \ldots, x_n)$ are i.i.d from $f(x \mid \theta)$, however there is right censoring at the value a. Therefore, we have a sample $Y = (y_1, \ldots, y_n)$, where $y_i = \min(x_i, a)$. It is common to denote censored values superscripted with an asterisk. For convenience we re-arrange the sample as $Y = (y_1, \ldots, y_m, y^*_{m+1}, \ldots, y^*_n)$, where $y^*_j = a$. In this case, $\mathcal{X}(y^*_i) = \{x : x \geq y^*_i\}$. The likelihood function can be written down

$$L(\theta \mid Y) = f(Y \mid \theta) = \prod_{i=1}^{m} f(y_i \mid \theta)[1 - F_\theta(a)]^{n-m}, \tag{4.2}$$

where F_θ is the CDF corresponding to f. Nevertheless, it is typically difficult to maximize $L(\theta \mid Y)$ because of the presence of the non-standard term $[1 - F_\theta(a)]^{n-m}$. Let $Z = (z_{m+1}, \ldots, z_n)$ represent the true but unobserved values of $y^*_m, \ldots, y^*_n$. Then, the pair (Y, Z) corresponds to complete information and

$$f(Y, Z \mid \theta) = f(y_1, \ldots, y_m \mid \theta) f(y^*_{m+1}, \ldots, y^*_n, z_{m+1}, \ldots, z_n \mid \theta)$$

since the observations are i.i.d. It is left as an exercise to show how to obtain (4.2) from (4.1).

Example 4.2. *Random effect models*

Suppose that the data $Y = (y_1, \ldots, y_n)$ are counts and that for each data point y_i a vector of covariates d_i is available. A simple Generalized Linear Model (GLM) is to assume $y_i \sim \mathit{Poisson}\left(\exp\{d_i^T \theta\}\right)$, where θ are the regression parameters on the log-scale and d_i^T is the transpose of d_i. However, count data often exhibit over-dispersion, a feature which can be modelled using random effect models. For example, we can assume that for each data point there are relevant unmeasured covariates Z_i, and we re-write the model as

$$y_i \mid (Z_i, \theta) \sim \mathit{Poisson}\left(\exp\{d_i^T \theta + Z_i\}\right),$$

where we take, for instance, $Z_i, i = 1, \ldots, n$ to be i.i.d $N(0, \nu^2)$.

Complete information corresponds to observing (Y, Z), where $Z = (Z_1, \ldots, Z_n)$. Where Z observed inference for θ would be a trivial exercise in Poisson regression, but Z is missing. The observed likelihood is

$$\begin{aligned} L(\theta \mid Y) &= \int f(Y, Z \mid \theta) dZ \\ &= \int f(Y \mid Z, \theta) f(Z) dZ \\ &= \int \prod_{i=1}^{n} \{f(y_i \mid Z_i, \theta) f(Z_i)\} \, dZ \end{aligned}$$

where $f(Z_i)$ is the density of the $N(0, \nu^2)$ distribution. In this case it is difficult to obtain a closed form expression for $L(\theta \mid Y)$, due to the integration above. Notice that in this example, inference for the missing data Z_i might be of interest as well.

4.3 The Algorithm

We first describe the EM algorithm and how to implement it, and in the next section we show why it works.

Suppose that the complete data are X, for which the model $f(X \mid \theta)$ is specified. X is decomposed as $X = (Y, Z)$ but only Y is observed, thus we are faced with a missing data problem. The density of the observed data is $f(Y \mid \theta)$, and the conditional density of the missing data Z, given the observed data Y, and the parameters θ, is $f(Z \mid Y, \theta)$. The complete-data likelihood is $L(\theta \mid Y, Z)$ and the observed data likelihood is $L(\theta \mid Y)$. For any pair (θ, θ') we define the function

$$Q(\theta, \theta') = \int \log(L(\theta|Y,Z)) f(Z|Y,\theta') dZ \tag{4.3}$$

that is

$$Q(\theta, \theta') = E(\log(L(\theta|Y,Z)) \tag{4.4}$$

where expectation is with respect to the distribution of Z given the θ' and Y.

The EM algorithm takes its name by the iteration of two main steps: the **The E–Step**, where an **E**xpectation is calculated, and the **The M–Step**, where a **M**aximization is performed. The algorithm is formally defined as:

Algorithm 4.1.

1. *Choose an initial value $\theta^{(0)}$, set $p = 0$.*
2. **The E–Step***: Calculate $Q(\theta, \theta^{(p)})$;*
   (at this step $\theta^{(p)}$ is fixed, and $Q(\theta, \theta^{(p)})$ is a function of θ)
3. **The M–Step***: Find $\theta^{(p+1)}$ which maximizes $Q(\theta, \theta^{(p)})$ as a function of θ;*
4. *Set p=p+1, goto 2.*

Each iteration of the EM algorithm defines an updating $\theta^{(p)} \to \theta^{(p+1)}$. The main feature of the algorithm is that

$$L(\theta^{(p)} \mid Y) \leq L(\theta^{(p+1)} \mid Y),$$

i.e the likelihood function is increased at each iteration of the algorithm and consequently under mild conditions the sequence $\theta^{(0)}, \theta^{(1)}, \theta^{(2)}, \ldots$, converges to the location of a local maximum of the likelihood. Actually, another property of the EM algorithm, is that if $\theta^{(p)} = \theta^*$, for some $p > 0$, where θ^* is the location of a local maximum, then $\theta^{(p+1)} = \theta^*$.

Essentially, each iteration of the EM algorithm updates $\theta^{(p)} \to \theta^{(p+1)}$, by solving the following equation:

$$\frac{\partial}{\partial \theta} E(\log(L(\theta|Y,Z))) = 0 \tag{4.5}$$

where $\theta^{(p)}$ is used to find $E(\log(L(\theta|Y,Z)))$, and $\theta^{(p+1)}$ is the solution to (4.5). The way the algorithm is described above, and as suggested in the original paper of Dempster, Laird and Rubin, 1977, JRSSB, first the expectation is found and then its derivative with respect to θ is set to 0.

4.3.1 Some insight into the algorithm

By definition,

$$f(Z \mid Y, \theta) = \frac{f(Z, Y \mid \theta)}{f(Y \mid \theta)}, \tag{4.6}$$

from which follows the fundamental equality of the EM,

$$\log L(\theta \mid Y) = \log L(\theta \mid Y, Z) - \log f(Z \mid Y, \theta), \tag{4.7}$$

which is true for any Z, and notice that only the right-hand side of the equality depends on Z. In fact, identity (4.7), which is the basic identity for the EM algorithm, shows that finding $f(Z \mid Y, \theta)$ in closed form is a problem of the same difficulty as performing the integration in (4.1) (since if we know $f(Z \mid Y, \theta)$ we can substitute in the identity and evaluate the observed log-likelihood for any value of θ).

We now take expectations with respect to $f(Z \mid Y, \theta')$ on both sides of (4.7), where θ' is some known value (in fact the current estimate). Defining,

$$H(\theta, \theta') = E(\log(f(Z|Y, \theta)) = \int \log f(Z|Y, \theta) f(Z|Y, \theta') dZ,$$

(4.7) becomes

$$\log L(\theta \mid Y) = Q(\theta, \theta') - H(\theta, \theta') \tag{4.8}$$

which holds for any θ'. Roughly speaking, in many problems $Q(\theta, \theta')$ is easy to compute and maximize, whereas $H(\theta, \theta')$ is intractable. The strength of the EM algorithm is based on the observation that the term $H(\theta, \theta')$ can be ignored. The key observation is that,

$$H(\theta, \theta') \leq H(\theta', \theta'), \quad \text{for any } \theta \neq \theta',$$

which we now show:

$$\begin{aligned} H(\theta, \theta') - H(\theta', \theta') &= \int \log \left\{ \frac{f(Z|Y, \theta)}{f(Z|Y, \theta')} \right\} f(Z|Y, \theta') dZ \\ &\leq \log \int \frac{f(Z|Y, \theta)}{f(Z|Y, \theta')} f(Z|Y, \theta') dZ \\ &= 0, \end{aligned}$$

where the inequality holds due to Jensen's inequality. If θ' represents the current estimate of the MLE, and θ'' is chosen so that $Q(\theta'', \theta') > Q(\theta', \theta')$, then $L(\theta'' \mid Y) > L(\theta' \mid Y)$. Therefore every iteration of the EM increases the likelihood.

4.3.2 Convergence

Every iteration of an EM algorithm results in a value of θ 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. The rate of convergence depends on what is known as the fraction of missing information. Informally, this fraction measures the amount of information about θ which is lost by failing to observed Z. Thus, if the complete data $X = (Y, Z)$ is much more informative about θ than the observed data Y, the fraction of missing information is large and the EM algorithm converges slowly. The last 20 years, several methods have been proposed in order to accelerate the convergence of EM. Many of them are based on incorporating information about the gradient of the likelihood into 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 4.1: Censored regression data

algorithm, trying to estimate the convergence rate in order to "enlarge" the steps of the EM, or changing the augmentation scheme so that to reduce the fraction of missing information.

It is clear from proof we gave earlier however, that it is not necessary to maximize the Q function at every step, it is sufficient to increase its value. Any step from $\theta' \to \theta''$ such that $Q(\theta'', \theta') > Q(\theta', \theta')$, will lead to an increase in the likelihood function, and the corresponding algorithm will (under the usual conditions) converge to a local maximum. This type of algorithms, are called Generalized EM algorithms.

To make sure that the algorithm has found a local than a global maximum, it is recommended to use a set of different starting values scattered around the state space of θ.

4.4 Censored regression example

Here we consider a regression problem involving censored data. Recall from section 4.2 that censored data problems belong to the family of missing/incomplete data problems.

Consider the data in Table 4.1. 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 $v_i = 1000/(\text{temperature } + 273.2)$:

$$t_i = \beta_0 + \beta_1 v_i + \epsilon_i \tag{4.9}$$

where $\epsilon_i \sim N(0, \sigma^2)$ On this scale a plot of t_i against v_i is given in Figure 4.1 in which censored data are plotted as open circles. The raw data are contained in `motor.dat`, and the transformed data are in `motor2.dat`. 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 0 otherwise. There are 23 censored values, and we have ordered the data so that $(t_1, v_1), \dots, (t_{17}, v_{17})$ correspond to un-censored observations, and $(t^*_{18}, v_{18}), \dots, (t^*_{40}, v_{40})$ to censored observations. We also define $T = (t_1, \dots, t_{40})$ and $\theta = (\beta_0, \beta_1, \sigma)$.

Now, in this situation, if the data were uncensored we would have a simple regression problem and parameters would be easily estimated by the least squares method. The censoring complicates the estimation of the regression parameters, thus we use EM algorithm. Let m be the number of uncensored values (in this case $m = 17$), and let $Z_{18}, \dots, Z_{40}$ denote the unobserved un-censored

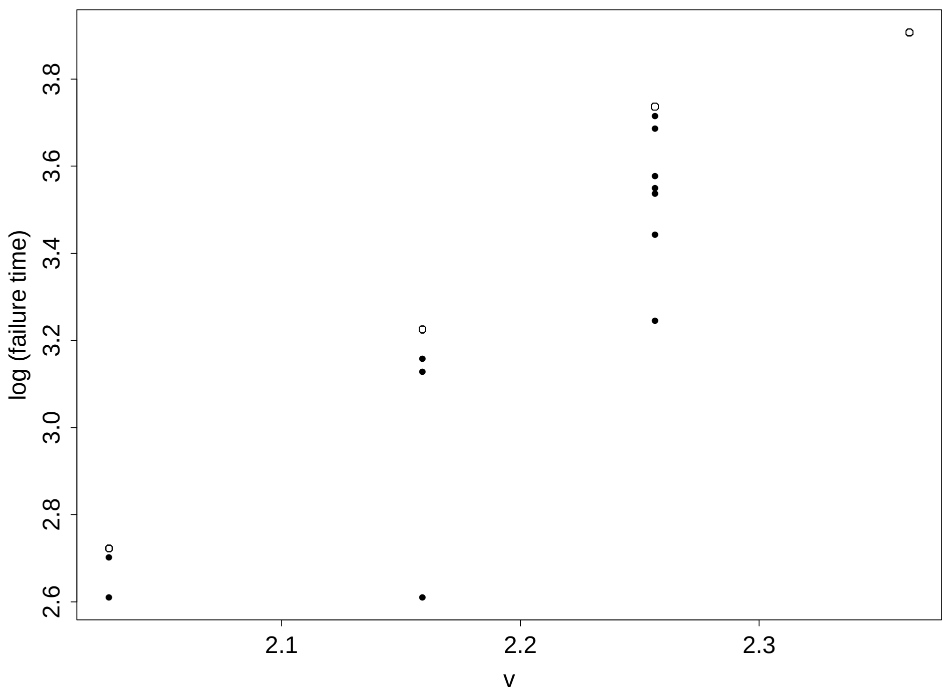

Figure 4.1: Censored regression data

measurements. Thus, complete information corresponds to having (T, Z), and the complete-data log-likelihood is:

$$\log(L(\theta|T,Z)) = -n\log\sigma - \sum_{i=1}^{m}(t_i - \beta_0 - \beta_1 v_i)^2/2\sigma^2 - \sum_{i=m+1}^{n}(Z_i - \beta_0 - \beta_1 v_i)^2/2\sigma^2 \qquad (4.10)$$

The E–step requires us to find expectations with respect to $f(Z_i|t_i^*, \theta')$, where $\theta' = (\beta_0', \beta_1', \sigma')$. Unconditionally, Z_i is distributed as a Normal with mean $\beta_0' + \beta_1' v_i$. Thus, $f(Z_i|t_i^*, \theta')$ is a normal density, conditioned on $Z_i > t_i^*$. Notice that only the third term on the right-hand side of (4.10) involves Z_i. Taking the expectation of that term with respect to Z_i, yields:

$$E\left\{\sum_{i=m+1}^{n}(Z_i - \beta_0 - \beta_1 v_i)^2/2\sigma^2\right\} = \frac{1}{2\sigma^2}\sum_{i=m+1}^{n}E\{(Z_i - \beta_0 - \beta_1 v_i)^2\}.$$

Evaluation of this expectation requires knowledge of the first two moments of the conditional distribution of the Z_i,

$$E[Z_i \mid t_i^*, \theta'] \quad \text{and} \quad E[Z_i^2 \mid t_i^*, \theta'].$$

It can be shown that in our problem,

$$\begin{aligned} E(Z_i|Z_i > t_i^*, \theta') &= \mu_i' + \sigma' H\left(\frac{t_i^* - \mu_i'}{\sigma'}\right) \\ E(Z_i^2|Z_i > t_i^*, \theta') &= (\mu_i')^2 + (\sigma')^2 + \sigma'(t_i^* + \mu_i')H\left(\frac{t_i^* - \mu_i'}{\sigma'}\right) \end{aligned}$$

where

$$\mu_i' = \beta_0' + \beta_1' v_i, \text{ and } H(x) = \phi(x)/\{1 - \Phi(x)\},$$

and $\phi(x), \Phi(x)$ are the pdf and CDF of the standard normal distribution. Combining all the above results,

$$\begin{aligned} Q(\theta,\theta') &= -n\log\sigma - \frac{1}{2\sigma^2}\sum_{i=1}^{m}(t_i - \beta_0 - \beta_1 v_i)^2/2\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=m+1}^{n}(\beta_0 + \beta_1 v_i)^2 \\ &- \frac{1}{2\sigma^2}\sum_{i=m+1}^{n}E(Z_i^2|Z_i > t_i^*, \theta') + \frac{1}{\sigma^2}\sum_{i=m+1}^{n}E(Z_i|Z_i > t_i^*, \theta')(\beta_0 + \beta_1 v_i). \end{aligned}$$

Notice now the major strength of the EM algorithm: both $E(Z_i|Z_i > t_i^*, \theta')$ and $E(Z_i^2|Z_i > t_i^*, \theta')$ are very complicated functions of θ', however they are, of course, both constants with respect to θ. That is, for fixed θ', $E(Z_i^2|Z_i > t_i^*, \theta')$ and $E(Z_i|Z_i > t_i^*, \theta')$ (as well as any other expectation of Z_i) are just fixed numbers. Thus, $Q(\theta, \theta')$ is a rather simple function of θ and it is easy to maximize it with respect to θ.

We will not go into details on how to maximize $Q(\theta, \theta')$, instead we simply note that it is based on a direct extension of the standard least squares technique. Therefore, the EM in this context consists in following the program described below:

1. Fit the regression line by least squares;
2. Estimate by maximum likelihood the error variance (allowing for uncertainty in censored observations);
3. Update the estimates of censored data on basis of fitted model;

4. Iterate these steps.

This is easily implemented in R with the following code:

```
em.reg=function(x, n) {
        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)
                ez[1:17] = x[1:17, 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))
}
```

This produced the output (each line corresponds to an EM-iteration, and the columns correspond to β_0, β_1, σ, respectively):

```
-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 4.2. 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.

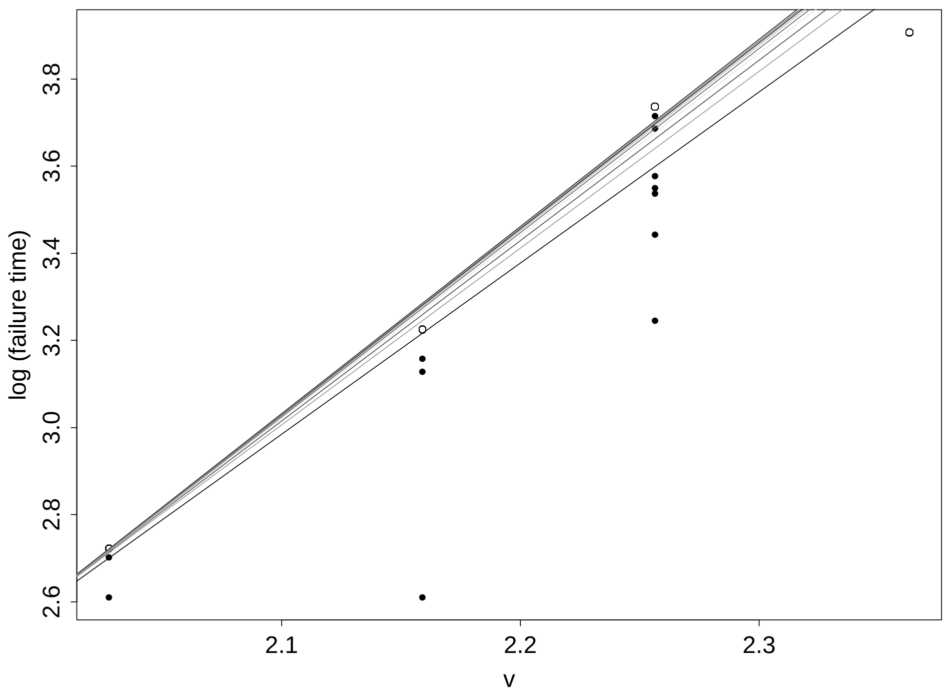

Figure 4.2: EM iterates to regression fit of censored data

4.5 A genetic example

We now present an example where a model is, possibly artificially, put into a missing-data framework in order to facilitate the use of the EM algorithm to obtain MLEs. The data set we consider contains information concerning the genetic linkage of 197 animals, in which the animals are distributed into 4 categories:

$$Y = (y_1, y_2, y_3, y_4) = (125, 18, 20, 34) \tag{4.11}$$

with cell probabilities

$$(\frac{1}{2} + \frac{\theta}{4}, \frac{1}{4}(1-\theta), \frac{1}{4}(1-\theta), \frac{\theta}{4}) \tag{4.12}$$

This is a multinomial model with observed likelihood

$$L(\theta|Y) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4} \tag{4.13}$$

and consequently

$$\log L(\theta|Y) = \text{ constant } + (y-1)\log(2+\theta) + (y_2+y_3)\log(1-\theta) + y_4\log(\theta). \tag{4.14}$$

It is difficult to maximize this multinomial likelihood directly, since it involves solving a third-order polynomial equation. Instead we illustrate how the EM algorithm brings about a substantial simplification in the calculations.

We split the first cell into two, with respective cell probabilities $\frac{1}{2}$ and $\frac{\theta}{4}$, and we assume that the observed data are an incomplete version of the data $X = (x_1, x_2, x_3, x_4, x_5)$, on a 5-cell table, where we have only observed the total of the first two cells. Therefore, $x_1 + x_2 = y_1$, and $x_3 = y_2, x_4 = y_3, x_5 = y_4$. The complete-data log-likelihood is

$$\log L(\theta|X) = \text{ constant } + \log(\theta)(x_2+x_5) + \log(1-\theta)(x_3+x_4), \tag{4.15}$$

and the MLE is obtained by solving a second-order polynomial equation, which is easier than solving the third-order equation that the observed likelihood requires. Working as before,

$$Q(\theta,\theta') = \text{ constant } + (E(x_2|Y,\theta') + x_5)\log(\theta) + (x_3+x_4)\log(1-\theta) \tag{4.16}$$

where $x_2|(Y,\theta') \sim Bin(125, \frac{\theta'}{\theta'+2})$. Thus

$$Q(\theta,\theta^i) = \left[\frac{125\theta^i}{\theta^i+2} + 34\right]\log(\theta) + 38\log(1-\theta) \tag{4.17}$$

This is easily maximized, and the transition $\theta_i \to \theta_{i+1}$ in each iteration of the EM is:

$$\theta^{i+1} = \frac{E(X_2|\theta^i,Y) + x_5}{E(X_2|\theta^i,Y) + x_3 + x_4 + x_5} \tag{4.18}$$

where

$$E(X_2|\theta^i,Y) = \frac{125\theta^i}{\theta^i+2}\ .$$

Starting with $\theta^1 = 0.5$ we obtain the sequence in Table 4.2. Hence the maximum likelihood estimate 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 4.2: Sequence of EM iterates

4.6 Standard Errors

As usual, calculation of standard errors requires calculation of the inverse Hessian matrix:

$$\left[-\frac{d^2 \log L(\theta|Y)}{d\theta_i d\theta_j}\right]^{-1} \tag{4.19}$$

evaluated at the MLE θ^*. 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:

$$-\frac{d^2 \log L(\theta|Y)}{d\theta_i d\theta_j} = \left[-\frac{d^2 Q(\theta,\phi)}{d\theta_i d\theta_j}\right]_{\phi=\theta} - \left[-\frac{d^2 H(\theta,\phi)}{d\theta_i d\theta_j}\right]_{\phi=\theta} \tag{4.20}$$

where both $Q(\theta,\phi)$ and $H(\theta,\phi)$ have been defined in Section 4.3.1.

Without the missing data, the first term on the right hand–side of (4.20) would be the Hessian; the second term compensates for the missing information. The proof of (4.20) is directly from (4.7). Application of the missing information principle is simplified by using the result:

$$-\frac{d^2 H(\theta,\phi)}{d\theta_i d\theta_j} = Var\left[\frac{d \log L(\theta|Y,Z)}{d\theta}\right] \tag{4.21}$$

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

$$\begin{aligned} -\left[\frac{d^2 Q(\theta,\phi)}{d\theta^2}\right]_{\phi=\theta^*} &= \frac{E(X_2|Y,\theta^*)+x_5}{\theta^{*2}} + \frac{x_3+x_4}{(1-\theta^*)^2} \\ &= \frac{29.83}{0.6268^2} + \frac{38}{(1-0.6268)^2} \\ &= 435.3 \end{aligned}$$

We also have:

$$\frac{d \log L(\theta|Y,Z)}{d\theta} = \frac{x_2+x_5}{\theta} - \frac{x_3+x_4}{1-\theta} \tag{4.22}$$

so that

$$\begin{aligned} Var\left[\frac{d\log L(\theta|Y,Z)}{d\theta}\right] &= \frac{Var(X_2|Y,\theta^*)}{\theta^{*2}} \\ &= 125\left(\frac{\theta^*}{2+\theta^*}\right)\left(\frac{2}{2+\theta^*}\right)\frac{1}{\theta^{*2}} \\ &= 22.71/0.6268^2 = 57.8 \end{aligned}$$

Thus

$$-\frac{d^2\log L(\theta|Y)}{d\theta^2} = 435.3 - 57.8 = 377.5 \tag{4.23}$$

and the standard error of θ^* is $\sqrt{(1/377.5)} = 0.05$.

4.7 Monte–Carlo EM algorithm

In some situations, direct evaluation of the Q function is difficult, due to the expectation involved; this is for example the case in Example 4.2. In such situations we can resort to a Monte Carlo estimate of the expectation, thus, the E–Step of the EM algorithm is replaced with

1. Simulate $z_1, z_2, \ldots, z_m$ from $f(Z|Y, \theta')$
2. Let $Q^{(m)}(\theta, \theta') = \frac{1}{m}\sum_{j=1}^{m} \log L(\theta|z_j, Y)$

$Q^{(m)}$ denotes the approximation of Q based on m Monte Carlo samples. 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 in later iterations to minimize variability due to the error in the Monte Carlo integration (see the paper by Wei and Tanner, 1990, JASA).

For illustration, we apply the Monte Carlo EM algorithm to the genetic linkage example. There we had

$$Q(\theta, \theta^i) = [E(X_2|Y, \theta^i) + x_5]\log(\theta) + (x_3 + x_4)\log(1-\theta) \tag{4.24}$$

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 (4.24). Then the M–Step continues as before with

$$\theta^{i+1} = \frac{\bar{z} + x_5}{\bar{z} + x_3 + x_4 + x_5} \tag{4.25}$$

This is implemented in R 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. Actually, one of the key advantages of the EM algorithm, the fact that every iteration increases the likelihood, is lost in the Monte Carlo EM. The Monte Carlo EM has close links with the data augmentation algorithm, described in Math457.

4.8 EM algorithm and the exponential family

The iterative steps of the EM algorithm become particularly intuitive when $f(X \mid \theta)$, i.e the model for the complete data, belongs to the exponential family. Actually, this is the case in many applied problems, and in particular in the examples we have encountered so far. As is well known, this family admits finite-dimensional sufficient statistics and the complete-data likelihood, $L(\theta \mid X)$, depends on the data X only through these statistics.

Then it can be shown that at the E-step, the algorithm replaces the unknown sufficient statistics with the current "best guess" (i.e with the conditional expectation of the statistics given the data and the current parameter estimate). Therefore, in this case the EM algorithm acts as an iterative imputation method: the missing data are replaced with some estimates, then the parameters are estimated on the basis of the *completed* data set, then the missing data are re-estimated, etc, until convergence is achieved.

Generally, implementation of the EM algorithm when $f(x \mid \theta)$ is in the exponential family, is much simpler than in the general case.