Chapter 13

Empirical methods

Chapter outline

Objective

In this chapter we introduce several empirical methods that are being increasingly used in statistical computations as an alternative or as an improvement to classical statistical methods.

Stanislaw Ulam
(Source: http://scienceworld.wolfram.com/biography/Ulam.html.)

Stanislaw Ulam (1909–84) was a Polish American mathematician who came to the United States in 1936. He worked at Princeton University. He was involved with the Manhattan Project to build the first atomic bomb. Ulam solved the problem of how to initiate fusion in the hydrogen bomb. Ulam was interested in astronomy, physics, and mathematics from an early age. He obtained his PhD from the Polytechnic Institute in Lwów in 1933, where he studied under a famous mathematician named Banach. Ulam's writings included *A Collection of Mathematical Problems* (1960); *Sets, Numbers, and Universes* (1974); and *Adventures of a Mathematician* (1976). His major contribution to statistics is through the introduction of the Monte Carlo methods along with Metropolis in 1949. These methods are

Mathematical Statistics with Applications in R. https://doi.org/10.1016/B978-0-12-817815-7.00013-0

widely used in solving mathematical problems using statistical sampling. Monte Carlo methods became widely popular with the ever-increasing power of computers and the development of specialized mathematical and statistical software.

13.1 Introduction

In statistics, major efforts are made to develop and study accurate statistical models that are able to describe natural phenomena. The dilemma is whether to use the standard model that may allow closed-form solutions, or to describe the phenomenon more accurately, which would often preclude the computation of explicit answers. Obtaining methods that result in useful qualitative and quantitative understanding of realistic complex systems is difficult, and obtaining exact analytical tools is not practical either. Because of this problem, practitioners have relied on simulation-based methods. Computer simulation methods are becoming the tools of choice for problems in statistics. Most of the empirical methods discussed in this chapter had been in existence in the statistical literature as possible numerical methods for some time. Because of the difficulty of computing by hand, these methods did not gain much popularity. These numerical techniques became popular and practical with the advent of high-quality pseudo-random number generators and high-speed computers. Modern statistics is increasingly being equipped with theoretical concepts complemented with effective computational tools to handle the challenges that arise in science and technology. The methods presented in this chapter could be effectively used for Bayesian computation and for problems arising in such diverse areas as environmental modeling, epidemiology, finance, genetics, image analysis, and statistical physics.

It is important to note that the literature on these simulation methods is growing, and it is impossible to present the whole picture in a single chapter. The purpose of this chapter is only to introduce some basic and popular computational methods. There are many specialized books for further study.

13.2 The jackknife method

It was Tukey who, in 1958, gave the name "jackknife" (sometimes also known as the Quenouille−Tukey jackknife) to a general statistical method, invented by Maurice Quenouille in 1956, for testing hypotheses and finding confidence intervals where traditional methods are not applicable or not well suited. In general usage, a jackknife is a large clasp knife that has a multitude of small pull-out tools. Because this method could be used for small tasks without resorting to other tools, it was named the jackknife. The jackknife method could also be used with multivariate data. However, here we will present only the method for univariate data. The jackknife procedure is very useful when outliers are present in the data or the dispersion of the distribution is wide. In the jackknife method, we systematically recompute the statistic, leaving out one observation at a time from the observed sample. This is used to estimate the variability of a statistic from the variability of that statistic between subsamples. This avoids the parametric assumptions that we used in obtaining the sampling distribution of the statistic to calculate standard error. Thus, this can be considered a nonparametric estimate of the parameter. Initially, the jackknife method was introduced for bias reduction (thus improving a given estimator) and is a useful method for variance estimation. In this section, we study only how to compute a jackknife estimate and a confidence interval. We do not discuss how it reduces bias or any other theoretical properties. Jackknife methods predate the bootstrap method discussed in the next section.

Let $X_1, \ldots, X_n$ be a random sample from a population with finite variance. Then the sample mean is:

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

If one of the observations, say, the kth observation, is taken out (or missing), then:

$$\overline{X}_{-k} = \frac{1}{n-1}\left(\sum_{i=1}^{n} X_i - X_k\right) = \frac{1}{n-1}\sum_{k\neq i=1}^{n} X_i.$$

Now, if we know the overall sample mean $\overline{X}$ and we calculated $\overline{X}_{-k}$, then we can obtain the deleted observation X_k by using the formula:

$$X_k = n\overline{X} - (n-1)\overline{X}_{-k}.$$

In general, suppose that the population parameter θ is estimated by a function of the sample values $\widehat{\theta}$ $(X_1, \ldots, X_n)$, represented by $\widehat{\theta}$, and let $\widehat{\theta}_{-k}$ be the corresponding estimate by removing the kth observation. Note that here θ is any parameter; it need not be the population mean. Then the set of "pseudo-values" $\widehat{\theta}_k^*$, $k = 1, 2, \ldots, n$ are obtained by:

$$\widehat{\theta}_k^* = n\widehat{\theta} - (n-1)\widehat{\theta}_{-k}^*.$$

The average of these pseudo-values,

$$\widehat{\theta}^* = \frac{1}{n}\sum_{k=1}^{n}\widehat{\theta}_{-k}^*,$$

is the *jackknife estimate* of the parameter θ.

Let s^{*2} be the sample variance of these pseudo-values. Then, the variance of $\widehat{\theta}^*$ is estimated by s^{*2}/n, and a $(1-\alpha)$ 100% *jackknife confidence interval* for θ is given by:

$$\widehat{\theta}^* \pm t_{\alpha/2}\frac{s^*}{\sqrt{n}},$$

where $t_{\alpha/2}$ is evaluated with $(n-1)$ degrees of freedom.

A procedure for jackknife point and interval estimation

1. Generate a random sample $X_1, \ldots, X_n$ from a population.
2. First remove X_1 from the sample (so the new sample will be $X_2, \ldots, X_n$) and compute the estimator $\widehat{\theta}_{-1}$ (such as the sample mean); then remove X_2 (the resulting sample will be $X_1, X_3, \ldots, X_n$) and compute the estimator $\widehat{\theta}_{-2}$, and so on until the last sample is $X_1, \ldots, X_{n-1}$, with the estimator being $\widehat{\theta}_{-n}$.
3. The jackknife point estimate of θ is:

$$\widehat{\theta}^* = \frac{1}{n}\sum_{k=1}^{n}\widehat{\theta}_{-k}^*.$$

4. Calculate the sample variance of the values $\widehat{\theta}_{-i}$, $i = 1, \ldots, n$, and denote the variance as s^{*2}.
5. A $(1-\alpha)$ 100% jackknife confidence interval for θ is given by:

$$\widehat{\theta}^* \pm t_{\alpha/2}\frac{s^*}{\sqrt{n}}.$$

EXAMPLE 13.2.1

A random sample of $n = 6$ from a given population resulted in the following data:

7.2 5.7 4.9 6.2 8.5 2.8.

(a) Find a jackknife point estimate of the population mean μ.
(b) Construct a 95% jackknife confidence interval for the population mean μ.

Solution

(a) *Here* n = 6. *Table 13.1 represents the original sample and the six jackknife samples.*

TABLE 13.1 Jackknife Samples.

Original	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 6
7.2	5.7	7.2	7.2	7.2	7.2	7.2
5.7	4.9	4.9	5.7	4.9	4.9	4.9
4.9	6.2	6.2	6.2	5.7	6.2	6.2
6.2	8.5	8.5	8.5	8.5	5.7	8.5
8.5	2.8	2.8	2.8	2.8	2.8	5.7
2.8						

Using Minitab descriptive statistics, we obtained the summary of the analysis given in Table 13.2. Now, taking the mean and standard deviation of the means of the six jackknife samples, we get $\hat{\mu}^* = 5.883$, *and the standard deviation* $s^* = 0.392$. *Thus, the jackknife point estimate of* μ *is* $\hat{\mu}^* = 5.883$, *that is, the same as the mean of the original sample. However, we can see that the standard deviation resulting from the jackknife is reduced to only 0.392, compared with 1.959 for the original sample.*

TABLE 13.2 Summary Statistics for Jackknife Samples.

Variable	N	Mean	Median	TrMean	StDev	SE Mean
Original	6	5.883	5.950	5.883	1.959	0.800
Sample 1	5	5.620	5.700	5.620	2.068	0.925
Sample 2	5	5.920	6.200	5.920	2.188	0.978
Sample 3	5	6.080	6.200	6.080	2.123	0.949
Sample 4	5	5.820	5.700	5.820	2.183	0.976
Sample 5	5	5.360	5.700	5.360	1.656	0.741
Sample 6	5	6.500	6.200	6.500	1.395	0.624

In Table 13.2, TrMean is the trimmed mean, which is computed by first ordering the data values from smallest to largest, then deleting a selected number of values from each end of the ordered data, and then, averaging the remaining data. In this case, we have removed the smallest and largest 5% of the values (rounded to the nearest integer), and then calculated the mean of the remaining values.

(b) *A 95% jackknife confidence interval for* μ *is*:

$$\hat{\mu}^* \pm t_{\alpha/2}\frac{s^*}{\sqrt{n}} = 5.883 \pm 2.571\frac{0.392}{\sqrt{6}},$$

resulting in (5.471, 6.2944). Compare this with Example 5.5.7, in which we got the confidence interval as (3.827, 7.939). Thus, through the jackknife method, we get a much tighter confidence interval for μ.

The jackknife method of resampling is also known as the "leave-one-out" method because it uses all observations but one in each subsample. Here, every observation is left out exactly once. Note that in the jackknife method, sampling is done without replacement. This procedure can also be used for other statistical procedures such as hypothesis testing and regression. We use the jackknife resampling method to estimate the prior probability density function (pdf) of true parameters in the pdf of the given data, $f(x|\theta)$, to obtain empirical Bayes estimates of θ, in Chapter 10.

Exercises 13.2

13.2.1. The following data represent the total ozone levels measured in Dobson units at randomly selected locations on the Earth on a particular day:

269 246 388 354 266 303
295 259 274 249 271 254

(a) Find a jackknife point estimate of the population mean μ ozone level.
(b) Construct a 95% jackknife confidence interval for the population mean μ.
(c) Compare the confidence interval obtained in (b) with that in Example 6.3.3.

13.2.2. A drug is suspected of causing an elevated heart rate in a certain group of high-risk patients. Twenty patients from this group were given the drug. The changes in heart rates were found to be as follows:

−1 8 5 10 2 12 7 9 1 3
4 6 4 12 11 2 −1 10 2 8

Construct a 98% jackknife confidence interval for the mean change in heart rate. Interpret your answer.

13.2.3. Air pollution in large US cities is monitored to see whether it conforms to requirements set by the Environmental Protection Agency. The following data, expressed as an air pollution index, give the air quality of a city for 10 randomly selected days:

57.3 58.1 58.7 66.7 58.6 61.9 59.0 64.4 62.6 64.9

Construct a 95% jackknife confidence interval for the actual average air pollution index for this city and interpret.

13.2.4. The mileage (in thousands) for a random sample of 10 rental cars from a large rental company's fleet is listed:

7 13 5 5 11 15 7 9 13 8

Find a 95% jackknife confidence interval for the population mean mileage of the rental cars of this company.

13.2.5. The following data represent cholesterol levels (in mg/dL) of 10 randomly selected patients from a large hospital on a particular day:

360 352 294 160 146 142 318 200 142 116

Determine a 95% jackknife confidence interval for σ^2. Compare this with the confidence interval obtained in Example 6.4.2.

13.2.6. Air pollution in large US cities is monitored to see whether it conforms to requirements set by the Environmental Protection Agency. The following data, expressed as an air pollution index, give the air quality of a city for five randomly selected days:

56.23 57.12 57.7 63.92 59.40

Construct a 99% jackknife confidence interval for the actual variance of the air pollution index for this city and interpret.

13.2.7. It is known that some brands of peanut butter contain impurities within an acceptable level. A test conducted on 12 randomly selected jars of a certain brand of peanut butter resulted in the following percentages of impurities:

1.9 2.7 2.1 2.8 2.3 3.6 1.4 1.8 2.1 3.2 2.0 1.9

(a) Construct a 95% jackknife confidence interval for the average percentage of impurities in this brand of peanut butter.
(b) Give an approximate 95% jackknife confidence interval for the population variance.
(c) Interpret your results.

13.2.8. The following is a random sample taken from the data that represent the time intervals in days between earthquakes that either registered magnitudes greater than 7.5 on the Richter scale or produced more than 1000 fatalities during the time period December 1902 to March 1977.

263 1901 121 832 150 99

(a) Construct a 95% jackknife confidence interval for the average number of days between earthquakes of this type.
(b) Give an approximate 95% jackknife confidence interval for the population variance of number of days between earthquakes of this type.

13.3 An introduction to bootstrap methods

In this section, we describe some aspects of a relatively recent statistical technique known as the bootstrap method that can be used when the statistical distribution is unknown or the assumptions of normality are not satisfied and especially when the samples are small. This is a general method for estimating sampling distributions. The concept of the bootstrap was

introduced by Bradley Efron in 1979 and further developed by Efron and Tibshirani in 1993. We often try to determine the exact (sampling) distribution in an inferential procedure, such as the sampling distribution of the sample mean, the median, or the variance, to be used in computing confidence intervals and for testing hypotheses. However, as we have seen, this is often the most difficult part of the work, because the sampling distribution depends on the population distribution, which is often unknown. This is the reason asymptotic methods are quite frequently used for hypothesis testing and interval estimation. The bootstrap procedure provides us with a simple method for obtaining an approximate sampling distribution of the statistic, conditional on the observed data. However, it should be noted that the distribution thus obtained is only approximate. It is not as "good" as the exact distribution, because we have only a sample from the population. However, often, a bootstrap sampling distribution is easier to compute. Bootstrap methods are computer-intensive methods that use simulation to calculate standard errors, confidence intervals, and significance tests. The methods are applied by researchers in business, econometrics, life sciences, medical sciences, social sciences, and other areas where statistics is being utilized. The bootstrap method uses computer-generated pseudo-random numbers. So, the same situation might give similar but possibly different results. Also, it is computationally more involved to obtain results than by using the asymptotic distribution. The advantage is that the results are conditional on observed data, not based on large sample approximations. How does bootstrap help in reality? For instance, suppose we have 10 years of monthly return data on a particular stock. If we were to use these data to predict the future return, say through linear regression, we would be assuming that the future is going to behave similar to what happened in the past. We know from experience that such an assumption may not give us a good prediction and the underlying parametric assumptions may not hold. By creating bootstrap samples from these available data, what we are creating is not what happened, but rather what could have happened in the past from what did happen. For example, to see how resampling affects sample mean, a particular mutual fund had the following total return (in percentage) for the past 5 years:

Year	1	2	3	4	5
Total return	40.7	10.8	29.2	9.9	0.7

In this case, the average return for the past 5 years is 18.26%. A two-times resampling (what could have happened) resulted in the following outcomes.

Year	1	2	3	4	5
Total return	29.2	40.7	9.9	10.8	10.8

Here, the average is 20.28%. The next resampling gave the following:

Year	1	2	3	4	5
Total return	0.7	0.7	40.7	0.7	9.9

The resulting average return is 10.54%. A realistic future prediction method should depend on these possible fluctuations that could have happened in different scenarios.

Most of the inferential procedures we learned are based on a single sample drawn from the population. Bootstrap methods, in contrast, generate repeated subsamples from the single original sample itself and make inferences without assuming any particular functional form for the population distribution. Because this has the effect of sampling with replacement, we can create as many subsamples as we wish. These subsamples will have the same sample size and values as the original sample, except that many values in each of the subsamples will be repeated because of sampling with replacement. It should be noted that the effectiveness of a bootstrap procedure depends on the original sample being representative of the population. If the original sample is not representative, the conclusions drawn from the bootstrap methods will be completely inappropriate.

Using the jackknife method, the size of resamples is confined to $(n - 1)$, and the number of total possible samples is only n, the original sample size. The resampling strategy based on bootstrap has no such limitations in terms of the number and magnitude of replications possible. The only limitation comes from the computing resources, and these new sets of samples can be treated as a virtual population.

EXAMPLE 13.3.1

Suppose that the population distribution is a $N(1, \sigma^2)$. Estimate σ^2.

Solution

Because we know the functional form of the distribution, we could use the estimation procedures discussed in Chapter 5. There is no need for the bootstrap method. These steps are as follows:

Step 1. *If we have a random sample from* $N(1, \sigma^2)$ *of size* n, *use it. Otherwise, generate a random sample* $X_1, \ldots, X_n$ *from* $N(\mu, \sigma^2)$. *This could be done using the method described in Project 4A of Chapter 4.*

Step 2. *Estimate* σ^2 *by using the method of maximum likelihood, yielding:*

$$\widehat{\sigma}^2_{ml} = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2.$$

Note that the maximum likelihood procedure requires the knowledge of the functional form of the distribution; see the derivation in Chapter 5. Suppose the form of the population distribution is not known but we do have a random sample $X_1, \ldots, X_n$ from a distribution. Now we will describe how we can estimate σ^2 using the bootstrap method.

Let $X_1, \ldots, X_n$ be a random sample from a probability distribution F with $\mu = E(X_i)$ and $\sigma^2 = Var(X_i)$. Then the *standard error* of $\overline{X}$ is defined as σ^2/n. In general, the population distribution F is unknown. A simple estimate of F is the *empirical (or sample) cumulative distribution function* defined by:

$$\widehat{F}(x) = \frac{\#\{X_i \le x\}}{n} = \text{Proportion of } X_i's \le x.$$

This $\widehat{F}$ is a step function with the size of the jump being $1/n$ at each ordered X_i.

Summary of bootstrap method of estimating the standard error of $\overline{X}$

Step 1. Use the sample $X, \ldots, X_n$ and find $\widehat{F}$, the empirical cumulative distribution function of F.

Step 2. Generate a sample $\{X^*_{11}, X^*_{12}, \ldots, X^*_{1n}\}$ from $\widehat{F}$. From this sample, compute $\overline{X}^*_1$.

Step 3. Repeat step 2 $(N-1)$ times to obtain samples $\{X^*_{i1}, X^*_{i2}, \ldots, X^*_{in}\}$, $i = 1, 2, \ldots, N$, and find $\overline{X}^*_2, \overline{X}^*_3, \ldots, \overline{X}^*_N$. Now calculate $\overline{X}^* = \frac{1}{N}\sum_{i=1}^{N}\overline{X}^*_i$. This is the bootstrap mean.

Step 4. Then the bootstrap estimate of $Var(\overline{X})$, denoted by $\widehat{\sigma}^2_{bs}$, is given by:

$$\widehat{\sigma}^2_{bs} = \frac{1}{N-1}\sum_{i=1}^{N}\left(\overline{X}^*_i - \overline{X}^*\right)^2.$$

Observe that once we have the subsample means $\overline{X}^*_1, \ldots, \overline{X}^*_N$, the formulas for calculating the bootstrap mean and bootstrap variance are the same as those for calculating the mean and variance of a given sample.

Note that when $\widehat{F}$ is taken to be the empirical cumulative distribution function, generating a sample from $\widehat{F}$ is equivalent to generating a sample from $\{X_1, \ldots, X_n\}$ with replacement. As a result, we obtain the following algorithm.

Bootstrap algorithm for estimating the standard error of $\overline{X}$

1. Draw N random samples *with replacement* from the original sample $X_1, \ldots, X_n$, with each observation having the same probability of being drawn $(1/n)$. Let these bootstrap samples be denoted by $\{\{X^*_{i1}, X^*_{i2}, \ldots, X^*_{in}\}, i = 1, 2, \ldots, N\}$
2. Calculate the sample means of each of these bootstrap samples and the overall sample mean by:

$$\overline{X}^*_i = \frac{1}{n}\sum_{j=1}^{n}X^*_{ij} \quad \text{and} \quad \overline{X}^* = \frac{1}{N}\sum_{i=1}^{N}\overline{X}^*_{ij}.$$

3. Compute:

$$\widehat{\sigma}^2_{bs} = \frac{1}{N-1}\sum_{i=1}^{N}\left(\overline{X}^*_i - \overline{X}^*\right)^2.$$

4. Then the bootstrap estimate of $Var(\overline{X})$ is $\widehat{\sigma}^2_{bs'}$, or equivalently, the standard error of $\overline{X}$ is $\sqrt{\widehat{\sigma}^2_{bs}}$.

It is not necessary that the size of the bootstrap sample also must be n or that the samples have to be obtained with replacement. However, it is suggested that the best results are obtained when the repeated samples are the same size n as the original sample and when the samples are obtained with replacement. The number of bootstrap samples N could be in the hundreds or more, depending only on the capacity of the software that we are using to generate these samples.

EXAMPLE 13.3.2

The following data represent the total ozone levels measured in Dobson units at randomly selected locations on Earth on a particular day:

269	246	388	354	266	303
295	259	274	249	271	254

Generate $N = 6$ bootstrap samples of size 12 each and find the bootstrap mean and standard deviation (standard error).

Solution

Using Minitab (see Example 13.7.1 for the steps) we have created 200 bootstrap samples of size 12. We obtain the following summary results:

$$\overline{X}^* = 285.74$$

and

$$\widehat{\sigma}^2_{bs} = 153.02 \quad \text{and} \quad \widehat{\sigma}_{bs} = 12.37.$$

Note that the mean of the original sample is 285.7, but the standard deviation is 43.9 (see Example 5.5.9). Even though the mean of the original sample and the bootstrap means are very close, their standard deviations are substantially different.

In real applications, one of the difficulties is to estimate the standard errors of more complicated statistics. We can now generalize the bootstrap method for those situations. Let $\widehat{\theta} = \widehat{\theta}(X_1, \ldots, X_n)$ be a sample statistic that are estimates of the parameter θ of an unknown distribution F using some procedure. We wish to estimate the standard error of $\widehat{\theta}$ using the bootstrap procedure, which is summarized next.

General bootstrap procedure to estimate the standard error of $\widehat{\theta}$

1. Draw N samples *with replacement* from the original sample, $(X_1, \ldots, X_n)$. Denote these bootstrap samples as $\{X^*_{i1}, X^*_{i2}, \ldots, X^*_{in}\}$, $i = 1, 2, \ldots, N$.
2. Compute $\widehat{\theta}_1, \widehat{\theta}_2, \ldots, \widehat{\theta}_N$, where

$$\widehat{\theta}^*_i = \widehat{\theta}_i(X_{i1}, X_{i2}, \ldots, X_{in}).$$

The procedure for computing $\widehat{\theta}^*_i$ is the same procedure as that used to compute $\widehat{\theta}$ for the original sample $X_1, \ldots, X_n$. Also, compute:

$$\widehat{\theta}^* = \frac{1}{N}\sum_{i=1}^{N} \widehat{\theta}^*_i.$$

3. The bootstrap estimator of standard error of $\widehat{\theta}$ is given by:

$$\widehat{\left[BSE(\widehat{\theta})\right]} = \sqrt{\frac{\sum_{i=1}^{N}\left(\widehat{\theta}^*_i - \widehat{\theta}^*\right)^2}{N-1}}.$$

It is clear that these algorithms are considerably computer intensive and it is necessary to have suitable software to implement them. The accuracy of the bootstrap approximation depends on the accuracy of $\widehat{F}$ as an estimate of F and how large a bootstrap sample is used to estimate the standard error of $\widehat{\theta}$. We will leave the computation to Project 13A. We now give a theoretical example.

EXAMPLE 13.3.3

Let $X_1, \ldots, X_n$ be a sample from a Poisson distribution with parameter λ. Let

$$\theta = P\{X \le 1\} = e^{-\lambda}(1 + \lambda).$$

Obtain a bootstrap estimate of θ.

Solution

It can be shown that the maximum likelihood estimator (MLE) of θ *is:*

$$\widehat{\theta}_{ml} = e^{-\overline{X}(1+\overline{X})}.$$

To estimate the bias of θ*, take* N *bootstrap samples from* $\{X_1, \ldots, X_n\}$*. Let*

$$\widehat{\theta}_i = e^{-\overline{X}_i}\left(1+\overline{X}_i\right) - \frac{\{\#X_i's \le 1\}}{n}.$$

Then the bootstrap estimate of the bias of θ *is:*

$$\widehat{\theta}_{bias} = \frac{\widehat{\theta}_1 + \cdots + \widehat{\theta}_N}{N}.$$

One might now use:

$$e^{-\overline{X}_i}\left(1+\overline{X}_i\right) - \widehat{\theta}_{bias}$$

as an estimator of θ.

13.3.1 Bootstrap confidence intervals

We could use the repeated sampling method to construct bootstrap confidence intervals. We now give a procedure to obtain this.

Procedure to find the bootstrap confidence interval for the mean

1. Draw N samples (N will be in the hundreds, and if the software allows, in the thousands) from the original sample with replacement.
2. For each of the samples, find the sample mean.
3. Arrange these sample means in order of magnitude.
4. To obtain, say, a 95% confidence interval, we will find the middle 95% of the sample means. For this, find the means at the 2.5% and 97.5% percentiles. The 2.5th percentile will be at the position $(0.025)(N+1)$, and the 97.5th percentile will be at the position $(0.975)(N+1)$. If any of these numbers are not integers, round to the nearest integer. The values of these positions are the lower and upper limits of the 95% bootstrap interval for the true mean.

It should be noted that every time we do this procedure, we may get a slightly different bootstrap interval. We now give an example.

EXAMPLE 13.3.4

For the data given in Example 13.3.2, obtain a 95% bootstrap confidence interval for μ.

Solution

We took $N = 200$ *samples of size 12. Thus* $0.025 \times 201 = 5.025 \approx 5$ *and* $0.975 \times 201 = 195.975 \approx 196$*. Thus, taking the 5th and 196th values of sorted (in ascending order) sample means, we get the 95% bootstrap confidence interval for* μ *is* $(263.8,\ 311.5)$.

1. *Comparing the classical confidence interval we obtained in Example 6.3.3, which is (257.81, 313.59), the bootstrap confidence interval of Example 13.3.4 has smaller length, and thus less variability. In addition, we saw in Example 6.3.3 that the normality assumption is necessary for the confidence interval there was suspected. In the bootstrap method, we did not have any distributional assumptions.*
2. *Because the bootstrap methods are more in tune with nonparametric methods, sometimes it makes sense to obtain a confidence interval about the median rather than the mean. With a slight modification of the procedure that we have described for the bootstrap confidence interval for the mean, we can obtain the bootstrap confidence interval for the median.*

Procedure to find the bootstrap confidence interval for the median

1. Draw N samples (N will be in the hundreds, and if the software allows, in the thousands) from the original sample with replacement.
2. For each of the samples, find the sample median.
3. Arrange these sample medians in order of magnitude.
4. To obtain, say, a 95% confidence interval we will find the middle 95% of the sample medians. For this, find the medians at the 2.5% and 97.5% quartiles. The 2.5th percentile will be at the position $(0.025)(N+1)$, and the 97.5th percentile will be at the position $(0.975)(N+1)$. If any of these numbers are not integers, round to the nearest integer. The values of these positions are the lower and upper limits of the 95% bootstrap interval for the median.

In practice, how many bootstrap samples should be taken? The answer depends on two things: how much the result matters, and what type of computing power is available. In general, it is better to start with 1000 subsamples. With the computational power available now, even taking 10,000 replications is not much of a problem. There are many works in the literature on bootstrap hypothesis testing and regression. These are beyond the scope of this chapter. In Chapter 10 we use bootstrap resampling to obtain empirical Bayes estimates.

Exercises 13.3

13.3.1. Using the data of Exercise 13.2.2, generate $N = 8$ bootstrap samples of size 20 each and find the bootstrap mean and standard deviation (standard error).

13.3.2. Using the data of Exercise 13.2.5, generate $N = 12$ bootstrap samples of size 10 each and find the bootstrap mean and standard deviation (standard error).

13.3.3. Using the data of Exercise 13.3.3, obtain a 95% bootstrap confidence interval for μ.

13.3.4. Using the data of Exercise 13.2.6, (a) obtain a 95% bootstrap confidence interval for μ, and (b) obtain a 95% bootstrap confidence interval for the population median.

13.3.5. Using the data of Exercise 13.2.8, (a) obtain a 95% bootstrap confidence interval for μ, and (b) obtain a 95% bootstrap confidence interval for the population median.

13.4 The expectation maximization algorithm

In this section, we introduce an algorithm, called the *expectation maximization* (EM) algorithm, that is widely used to compute maximum likelihood estimates when some elements of the data set are missing, unobservable, or incomplete. In real-life problems, observing the complete data set is the exception rather than the rule. For example, in lifetime studies, when n items are placed on a given test, we may have the failure times of only $n_1 < n$ items, while for the rest of the $(n - n_1)$ items we know only the censored failure time, that they survived a particular failure time T (fixed beforehand). For example, we may want to know whether the lifetime of a certain brand of fluorescent light bulbs is at least 24 months. For this purpose, let us say we randomly test 100 light bulbs of this brand. In this case, our data will contain all the months within which the bulbs burned out, and some that survived for 24 months. After 24 months, we may not follow when these bulbs will burn out; all we know is that these bulbs lasted for 24 months. Such a data set is an example of censored data. We can consider the censored failure times of $(n - n_1)$ items as the unobservable data values.

Another common problem is missing data. For example, suppose we were to take a survey on some socioeconomic problems from a random sample of families from a city in 2009 and then a follow-up study on the same families in 2014. This may result in many missing values in the follow-up study, because it is possible that we may not be able to locate some of the families. Missing values can also occur if some of the respondents refuse to answer certain questions. We have seen in Section 5.3 that sometimes it is not possible to obtain closed-form solutions for the MLE. In the completely observed case, there are other algorithms, such as Newton−Raphson, that can be used to numerically obtain appropriate estimates. With missing values, those algorithms cannot be used. The name *EM algorithm* was coined by Dempster, Laird, and Rubin in 1977. This is a general iterative algorithm to obtain the MLE when the data set is incomplete. The EM algorithm is a formalization of an intuitive idea of obtaining approximate estimates of the parameters with missing data: (1) replace missing values with estimated values as true values, (2) estimate the parameters, and (3) repeat.

Let $X_1, \ldots, X_{n_1}$ be the n_1 observed data values, and let $y_1, \ldots, y_{n-n_1}$ be the $(n - n_1)$ unobserved data values. Assume that $X_i's$ are independent and identically distributed (iid) random variables with pdf $f(x|\theta)$ and $X_i's$ and $Y_i's$ are independent, that is, data are missing at random.

We denote the random vector by $\mathbf{X}$ and the corresponding data vector by $\mathbf{x}$.

The joint pdf of $X_1, \ldots, X_{n_1}$ is represented by $f(\mathbf{x}|\theta)$, where θ is the parameter vector with values in $\Theta \subset R^p$, a p-dimensional Euclidean space. Let $g(\mathbf{x}, \mathbf{y}|\theta)$ denote the pdf of the complete data set $\mathbf{x}$ and $\mathbf{y}$; that is, the vector $(\mathbf{x}, \mathbf{y})$ represents the conceptualized complete data set. Let $h(\mathbf{y}|\theta, \mathbf{x})$ be the conditional pdf of the unobserved data $\mathbf{y}$ given θ and the observed data $\mathbf{x}$. The likelihood function for the observed data $\mathbf{x}$ is, by definition,

$$L(\theta; \mathbf{x}) = f(\mathbf{x}|\theta).$$

The likelihood function for the combined data $(\mathbf{x}, \mathbf{y})$ is, again by definition, given by:

$$L_c(\theta;\ \mathbf{x},\ \mathbf{y}) = g(\mathbf{x},\ \mathbf{y}|\theta).$$

The problem is to find the MLE that maximizes the likelihood function $L(\theta, \mathbf{x})$, at the same time using $L_c(\theta; \mathbf{x}, \mathbf{y})$.

From the foregoing definitions, we know that:

$$g(\boldsymbol{x}, \boldsymbol{y}|\theta) = f(\boldsymbol{x}|\theta)h(\boldsymbol{y}|\theta, \boldsymbol{x}).$$

Thus, we have the conditional pdf of the missing (or unobserved) data $\mathbf{y}$, given $\mathbf{x}$:

$$h(\mathbf{y}|\theta, \mathbf{x}) = \frac{g(\mathbf{x}, \mathbf{y}|\theta)}{f(\mathbf{x}|\theta)},$$

or equivalently,

$$f(\mathbf{x}|\theta) = \frac{g(\mathbf{x}, \mathbf{y}|\theta)}{h(\mathbf{y}|\theta, \mathbf{x})}. \tag{13.1}$$

Let $\theta_0 \in \Theta$ be a given θ value. Because $h(\mathbf{y}|\theta_0, \mathbf{x})$ is a pdf, we have:

$$\int h(\mathbf{y}|\theta_0,\ \mathbf{x})d\mathbf{y} = 1.$$

Thus, the ln of the observed likelihood,

$$\begin{aligned} \ln\ L(\theta; \mathbf{x}) &= \ln\ L(\theta; \mathbf{x}) \int h(\mathbf{y}|\theta_0, \mathbf{x})d\mathbf{y} \\ &= \int \ln L(\theta; \mathbf{x})h(\mathbf{y}|\theta_0, \mathbf{x})d\mathbf{y}\ \ (\text{as}\ \ln L(\theta; \mathbf{x}) \text{is independent of } \mathbf{y}). \end{aligned}$$

Because $L(\theta, \mathbf{x}) = f(\mathbf{x}|\theta)$, we have:

$$\begin{aligned} \ln L(\theta; \mathbf{x}) &= \int \ln f(\mathbf{x}|\theta)h(\mathbf{y}|\theta_0, \mathbf{x})d\mathbf{y} \\ &= [\ln g(\mathbf{x}, \mathbf{y}|\theta) - \ln h(\mathbf{y}|\theta, \mathbf{x})]h(\mathbf{y}|\theta_0, \mathbf{x})d\mathbf{y}(\text{from}(1)) \\ &= \int \ln g(\mathbf{x}, \mathbf{y}|\theta)h(\mathbf{y}|\theta_0, \mathbf{x})d\mathbf{y} - \int \ln h(\mathbf{y}|\theta, \mathbf{x})h(\mathbf{y}|\theta_0, \mathbf{x})d\mathbf{y} \\ &= E_{\theta_0}[\ln g(\mathbf{x}, \mathbf{y}|\theta)] - E_{\theta_0}[\ln h(\mathbf{y}|\theta, \mathbf{x})], \end{aligned} \tag{13.2}$$

where the expectation is taken with respect to the conditional distribution of $\mathbf{y}$ given θ_0 and $\mathbf{x}$. Let us now consider maximizing this with respect to θ. This maximization is the maximization step (M step) in the EM algorithm. It is important to note that EM-based estimates are approximate estimates.

Let θ_0 be an initial estimate of θ. The choice of this initial value θ_0 could be made randomly or heuristically based on any prior knowledge about the optimal value of the parameter. For instance, suppose we have to estimate mean and variance of a normal distribution. One good starting point could be to take the sample mean $\bar{x}$ and sample variance s^2 based on a subset of the data containing no missing values.

Let

$$\begin{aligned} Q(\theta|\theta_0, \mathbf{x}) &= E_{\theta_0}[\ln L_c(\theta; \mathbf{x}, \mathbf{y})] \\ &= E_{\theta_0}[\ln g(\mathbf{x}, \mathbf{y}|\theta)]. \end{aligned}$$

Here, θ_0 is used only to compute the expectation; we should not substitute for θ in the complete data log-likelihood. Let $\widehat{\theta}_{(1)}$ be the maximizer that maximizes $Q(\theta|\theta_0, \mathbf{x})$ with respect to θ. That is, $Q\left(\widehat{\theta}_{(1)}|\theta_0, \mathbf{x}\right) \geq Q\left(\theta|\theta_0, \mathbf{x}\right)$ for all $\theta_0 \in \Theta$. Then $\widehat{\theta}_{(1)}$ is the first-step estimator of θ. Continuing the procedure we obtain a sequence of approximate estimators $\widehat{\theta}_{(m)}$, which under appropriate conditions converges to the maximum likelihood estimate with likelihood $L_c(\theta; \mathbf{x}, \mathbf{y})$.

Steps for using the expectation maximization algorithm

1. $\widehat{\theta}_{(n)}$ is the estimate of the parameter θ on the nth step.
2. Expectation step (E step). Compute:

$$Q\left(\theta|\widehat{\theta}_{(n)}, \mathbf{x}\right) = E_{\widehat{\theta}_{(n)}}[\ln g(\mathbf{x}; \mathbf{y}|\theta)],$$

where the expectation is with respect to the conditional pdf of $\mathbf{y}$ given $\widehat{\theta}_{(n)}$ and $\mathbf{x}$ $\left(\text{i.e., with respect to } h\left(\mathbf{y}|\widehat{\theta}_{(n)}, \mathbf{x}\right)\right)$

3. M step. Find $\widehat{\theta}_{(n+1)} \in \Theta$ such that:

$$\widehat{\theta}_{(n+1)} = \max_{\theta} Q\left(\theta|\widehat{\theta}_{(n)}, \mathbf{x}\right).$$

4. Repeat until specified convergence criteria are met.

Thus, in the EM algorithm, each iteration involves two steps: the E step, followed by the M step. In the E step, we find the conditional expectation of the unobserved or missing data given the observed data and the current estimated parameters. Thus, in E step, missing data are estimated given the observed data and the present estimate of the model parameters. That is, the E step constitutes the calculation of:

$$\begin{aligned} Q\left(\theta|\widehat{\theta}_{(n)}, \mathbf{x}\right) &= E_{\widehat{\theta}_{(n)}}\left[\ln g(\mathbf{x}, \mathbf{y}|\theta)\right] \\ &= \int \ln g(\mathbf{x}, \mathbf{y}|\theta) h(\mathbf{y}|\theta_{(n)}, \mathbf{x}) d\mathbf{y}, \end{aligned}$$

(which is the sum if discrete), where the integration is over the range of values that $\mathbf{y}$ can assume. The M step constitutes maximization of $Q\left(\theta\middle|\widehat{\theta}_{(n)}, \mathrm{x}\right)$ with respect to θ. Thus, in the M step, the likelihood function is maximized under the assumption that the missing or hidden data are known. The estimates of the missing data from the E step are used in place of the actual missing data. This procedure improves the log-likelihood at every iteration; that is, the log-likelihood is nondecreasing for every iteration. Thus, for the sequence $\left(\widehat{\theta}_{(n)}\right)$ obtained through the EM algorithm, we have $L\left(\widehat{\theta}_{(n+1)}; \mathrm{x}\right) \geq L\left(\widehat{\theta}_{(n)}; \mathrm{x}\right)$ with equality holding if and only if $Q\left(\widehat{\theta}_{(n+1)}\middle|\widehat{\theta}_{(n)}, \mathbf{x}\right) = Q\left(\widehat{\theta}_{(n)}\middle|\widehat{\theta}_{(n)}, \mathbf{x}\right)$. When we have filled the completed data set, the parameter θ can be estimated by maximizing the log-likelihood estimating procedure (M step). It can be shown that under some conditions (such as that $\ln f(\mathbf{x}|\theta)$ is bounded or that $Q(\theta|\theta_0, \boldsymbol{x})$ is continuous in both θ and θ_0), $\widehat{\theta}_{(n)}$ converges in probability as $n \to \infty$ to the maximum likelihood estimate based on the complete likelihood $L_c(\theta; \boldsymbol{x}, \boldsymbol{y})$.

For computational convergence purposes, the E and M steps are alternated repeatedly until the difference $L\left(\widehat{\theta}_{(n+1)}, \mathbf{x}\right) - L\left(\widehat{\theta}_{(n)}, \mathbf{x}\right)$ is less than δ, a small but specified quantity. Another possible convergence criterion is to stop the iteration when the distance between $\widehat{\theta}_{(n+1)}$ and $\widehat{\theta}_n$ becomes arbitrarily small. In practice, it may be necessary to run the EM algorithm a number of times with different (random) starting points to ensure that the global maximum is obtained.

In general, the E and M steps could be complex. Even though the EM algorithm is applicable to any model, it is particularly effective if the data come from an exponential family. It turns out that, in this case, the log-likelihood is linear in the sufficient statistic for θ. For the E step, simply compute the expectation of the complete data sufficient statistic given the observed data. By substituting the conditional expectations of the sufficient statistics computed in the E step for the sufficient statistics that occur in the expression obtained for the complete data MLEs of θ, we can obtain the next iterate in the M step. Thus, when the complete data set is from an exponential family, both the E and M steps are simplified.

Let $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ be the complete observation vector. A particular case in which $g(\mathbf{x}, \mathbf{y}|\theta) = g(\mathbf{z}, \theta)$ is from an exponential family:

$$g(\mathbf{z}, \theta) = a(\mathbf{x}) \exp\{\mathbf{k}'(\theta)\mathbf{t}(\mathbf{x})\}/c(\theta),$$

where $\mathbf{t}(\mathbf{x})$ is a vector of sufficient statistics with complete data, $\mathbf{k}'(\theta)$ is a vector function of the parameter vector θ, and $a(\mathbf{x})$ and $c(\theta)$ are scalar functions. Recall that the members of the exponential family include many popular distributions, such as the normal, multivariate normal, Poisson, and multinomial distributions. In this case, the E step can be written as:

$$Q\Big(\theta|\theta_{(n)},\ \mathbf{x}\Big) = E_{\theta_{(n)}}[\ln a(\mathbf{x})|\mathbf{x}\Big] + \mathbf{k}'(\theta)\mathbf{t}_{(n)} - \ln c(\theta)$$

where $\mathbf{t}_{(n)} = E_{\theta(n)}[\mathbf{t(Z)}|\mathbf{x}]$ is an estimator of the sufficient statistic. The M step maximizes the Q function with respect to θ. Because $E_{\theta(n)}[\ln\ a(\mathbf{x})|\mathbf{x}]$ does not depend on θ, we can rewrite the steps as follows:

E step: Compute $\mathbf{t}(n) = E_{\theta(n)}[\mathbf{t(Z)}|\mathbf{x}]$.

M step: Find $\widehat{\theta}_{(n+1)} \in \Theta$, such that,

$$\widehat{\theta}_{(n+1)} = \max_{\theta}\big[\mathbf{k}'(\theta)\mathbf{t}_{(n)} - \ln c(\theta)\big].$$

The following example gives an EM algorithm for a special case of censored survival times. In the following example, the survival function is defined as the probability that an individual survives beyond time y, that is, $S(y) = P(Y > y)$.

EXAMPLE 13.4.1

Let $\mathbf{x} = (x_1, \ldots, x_{n_1})$ be the observed data and the censored observations at T are $\mathbf{y} = (y_1, \ldots, y_{n_2})$ (that is, the survival time is at least T). Let the mean survival time be θ, and the probability density be given by:

$$f(x|\theta) = \theta^{-1}\exp(-x/\theta), \quad x > 0.$$

(a) Obtain the MLE, $\widehat{\theta}_{ML}$.
(b) Obtain an EM algorithm.
(c) Consider the following censored data, which represent the number of years 20 patients survived after a major surgery, where a + symbol represents that we know only that this patient survived for 4 years and have no further information. That is,

4+	12	12	1	4+	3	3	5	2	0
5	1	4+	0	3	13	13	1	0	4

Using the algorithm developed in (b), run for 50 iterations with the initial value of θ_0 being the observed sample mean, $\overline{x}$, and with $\theta_0 = 0$. Comment on the results.

Solution

The joint pdf of the uncensored observation, $\mathbf{x}$, *is:*

$$f(\mathbf{x}|\theta) = \frac{1}{\theta^n}\exp\left(-\sum_{i=1}^{n_1} x_i/\theta\right).$$

For the right censored (at T*) observations* y_i, $i = 1, \ldots, n_2$, *the pdf can be calculated as follows:*

$$K\int_T^{\infty}\frac{1}{\theta}e^{-y/\theta}dy = 1,$$

which implies that $K = e^{T/\theta}$. *Thus, the pdf of* y_i *is given by:*

$$h(y|\theta, x) = \frac{e^{T/\theta}}{\theta}e^{-y/\theta} = \frac{1}{\theta}e^{\frac{1}{\theta}(T-y)}, y \geq T.$$

(a) *The likelihood,* $L_c(\theta, x, y)$, *can also be written in the form:*

$$L_c(\theta, x, y) = \frac{1}{\theta^{n_1}}e^{-\sum_{i=1}^{n_1}(x_i/\theta)}[1 - F(T)]^{n_2}$$

$$= \frac{1}{\theta^{n_1}}e^{-\sum_{i=1}^{n_1}(x_i/\theta)}e^{-\frac{n_2 T}{\theta}}.$$

Thus,

$$\ln L_c(\theta, x, y) = -n_1 \ln \theta - \frac{\sum_{i=1}^{n_1} x_i}{\theta} - \frac{n_2 T}{\theta}.$$

Differentiating with respect to θ*, and equating to zero, we have:*

$$\frac{\partial}{\partial \theta} \ln L_c(\theta, x, y) = -\frac{n_1}{\theta} + \frac{\sum_{i=1}^{n_1} x_i}{\theta^2} + \frac{n_2 T}{\theta^2} = 0.$$

This implies that:

$$n_1 \theta = \sum_{i=1}^{n_1} x_i + n_2 T$$

or

$$\widehat{\theta} = \frac{1}{n_1} \sum_{i=1}^{n_1} x_i + \frac{n_2}{n_1} T = \overline{x} + \frac{n_2}{n_1} T.$$

Hence, the MLE of θ *is:*

$$\widehat{\theta}_{ML} = \overline{X} + \frac{n_2}{n_1} T.$$

(b) *Because* g(X, Y|θ) *denote the pdf of the complete data, and we assumed that the pdf of all the data (censored or not) follows exponential distribution, we have:*

$$g(\mathbf{x}, \mathbf{y}|\theta) = \frac{1}{\theta^{n_1}} e^{-\sum_{i=1}^{n_1}(x_i/\theta)} \frac{1}{\theta^{n_2}} e^{-\sum_{i=1}^{n_2} y_i/\theta},$$

and we get:

$$\ln g(\mathbf{x}, \mathbf{y}|\theta) = -n_1 \ln \theta - \sum_{i=1}^{n_1} \frac{x_i}{\theta} - n_2 \ln \theta - \sum_{i=1}^{n_2} \frac{y_i}{\theta}.$$

For the E step of the EM algorithm, we first compute:

$$E_{\theta_0} Y = e^{T/\theta_0} \int_T^\infty y \frac{1}{\theta_0} e^{-y/\theta_0} dy$$

$$= T + \theta_0 \text{ (using the integration by parts).}$$

Thus, we get:

$$\begin{aligned} Q(\theta|\theta_0, \mathbf{x}) &= E_{\theta_0}[g(\mathbf{x}, \mathbf{y}|\theta)] \\ &= E_{\theta_0}\left[-n_1 \ln \theta - \sum_{i=1}^{n_1} \frac{x_i}{\theta} - n_2 \ln \theta - \sum_{i=1}^{n_2} \frac{y_i}{\theta} \right] \\ &= -n_1 \ln \theta - \sum_{i=1}^{n_1} \frac{x_i}{\theta} - n_2 \ln \theta - \frac{1}{\theta} \sum_{i=1}^{n_2} E_{\theta_0}(y_i) \\ &= -n_1 \ln \theta - \sum_{i=1}^{n_1} \frac{x_i}{\theta} - n_2 \ln \theta - \frac{1}{\theta} n_2 (T + \theta_0) \\ &= -n_1 \ln \theta - \sum_{i=1}^{n_1} \frac{x_i}{\theta} - n_2 \ln \theta - \frac{n_2 T + n_2 \theta_0}{\theta}. \end{aligned}$$

For the M step, we differentiate $Q(\theta|\theta_0, \mathrm{x})$ *with respect to* θ *and set it equal to zero,*

$$\frac{\partial}{\partial\theta}Q(\theta|\theta_0,x) = \frac{\partial}{\partial\theta}\left[-n_1\ln\theta - \sum_{i=1}^{n_1}\frac{x_i}{\theta} - n_2\ln\theta - \frac{n_2T+n_2\theta_0}{\theta}\right]$$

$$= -\frac{n_1}{\theta} + \frac{\sum_{n_1}^{i=1}x_i}{\theta^2} - \frac{n_2}{\theta} + \frac{n_2T+n_2\theta_0}{\theta^2} = 0$$

$$[n_1+n_2]\theta = \sum_{i=1}^{n_1}x_i + n_2T + n_2\theta_0$$

and

$$\widehat{\theta}_1 = \frac{1}{[n_1+n_2]}\sum_{i=1}^{n_1}x_i + \frac{n_2T}{[n_1+n_2]} + \frac{n_2}{[n_1+n_2]}\theta_0$$

$$= \frac{n_1}{[n_1+n_2]}\overline{x} + \frac{n_2T}{[n_1+n_2]} + \frac{n_2}{[n_1+n_2]}\theta_0.$$

Thus, for the general n, *the algorithm is:*

$$\widehat{\theta}_{(n+1)} = \frac{n_1}{[n_1+n_2]}\overline{x} + \frac{n_2T}{[n_1+n_2]} + \frac{n_2}{[n_1+n_2]}\widehat{\theta}_{(n)}.$$

Now putting $\theta_{(k+1)} = \theta_{(k)} = \theta^*$ *in the previous equation and solving for* θ^*, *we have the EM sequence* $\{\theta_{(k)}\}$, *which has the MLE* $\widehat{\theta}_{ML}$ *as its unique limit point, as* $k \to \infty$. *That is,* $\theta^* = \widehat{\theta}_{ML}$.

(c) *We used the following MATLAB code to run the algorithm with starting value* θ_0 *as the sample mean, that is, 4.5. Here* $T = 4$. *We run it for 50 iterations.*

```
A(1)=4.5
for n=2: 50
A(n)=4.41*(17./20)+3*4/20+(3./20)*A(n-1)
end
```

The following is the output:

4.5000	5.0235	5.1020	5.1138	5.1156	5.1158	5.1159
5.1159	5.1159	5.1159	5.1159	5.1159	5.1159	5.1159
5.1159	5.1159	5.1159	5.1159	5.1159	5.1159	5.1159
5.1159	5.1159	5.1159	5.1159	5.1159	5.1159	5.1159
5.1159	5.1159	5.1159	5.1159	5.1159	5.1159	5.1159
5.1159	5.1159	5.1159	5.1159	5.1159	5.1159	5.1159
5.1159	5.1159	5.1159	5.1159	5.1159	5.1159	5.1159
		5.1159				

Thus, $\widehat{\theta} = 5.1159$.

To run with $\theta_0 = 0$, *in the previous code, just change A(1) = 0. We get the following output:*

0.0000	4.3485	5.0008	5.0986	5.1133	5.1155
5.1158	5.1159	5.1159	5.1159	5.1159	5.1159
5.1159	5.1159	5.1159	5.1159	5.1159	5.1159
5.1159	5.1159	5.1159	5.1159	5.1159	5.1159
5.1159	5.1159	5.1159	5.1159	5.1159	5.1159
5.1159	5.1159	5.1159	5.1159	5.1159	5.1159
5.1159	5.1159	5.1159	5.1159	5.1159	5.1159
5.1159	5.1159	5.1159	5.1159	5.1159	5.1159
		5.1159	5.1159		

With $\theta_0 = \overline{x} = 4.5$, *it took six iteration steps to converge, whereas with* $\theta_0 = 0$, *it took seven steps to converge. Note that in both cases,* $\widehat{\theta} = 5.1159 = \widehat{\theta}_{ML}$.

Example 13.4.1 is a simple case, where there is no need for iterative computation of $\widehat{\theta}_{ML}$. However, this demonstrates how the EM algorithm would work. These types of problems are abundant in the medical field. For example, we may be

interested in the survival times of n patients after a treatment. For practical reasons, we may be observing only for a fixed duration, such as 10 years. In Example 13.4.1, the vector $\mathbf{x}$ will represent the time of death for the n_1 individuals. For the remaining $n_2 = n - n_1$ individuals, the only data we have state that they survived for more than 4 years. Thus, the value of T is 4. There is a possibility that during these experimental times, we may lose contact with some individuals, perhaps because they moved to some other place or they simply refused to participate in this experiment. In those cases, we will know only that the individual survived until we lost contact. This generalization of Example 13.4.1 to where the survival time data are different for each observation is given in Exercise 13.4.5.

We now give a similar example with a normal sample.

EXAMPLE 13.4.2

Let $\mathbf{x} = (x_1, \ldots, x_{n_1})$ be observed data from a normal population with mean θ and variance 1. Let the censored observations at T be $y = (y_1, \ldots, y_{n_2})$ (that is, the survival time is at least T) from the same population. Assume that the two sets of observations $\{x_i\}$ and $\{y_i\}$ are independent. Write down an EM algorithm to estimate θ.

Solution

For the uncensored observed sample $x_1, \ldots, x_{n_1}$*, the likelihood function is:*

$$L(\theta|\mathbf{x}) = f_x(\mathbf{x}|\theta) = \frac{1}{\left(\sqrt{2\pi}\right)^{n_1}} e^{-\frac{1}{2}\sum_{i=1}^{n_1}(x_i-\theta)^2}.$$

Furthermore, the complete likelihood for both samples is:

$$L(\theta|\mathbf{x}, \mathbf{y}) = \frac{1}{\left(\sqrt{2\pi}\right)^{n_1}} e^{-\frac{1}{2}\sum_{i=1}^{n_1}(x_i-\theta)^2} \frac{1}{\left(\sqrt{2\pi}\right)^{n_2}} e^{-\frac{1}{2}\sum_{i=1}^{n_2}(y_i-\theta)^2}. \tag{13.3}$$

From the definition of $Q(\theta|\theta_0, \mathbf{x})$*, we obtain:*

$$Q(\theta|\theta_0, \mathbf{x}) = E_{\theta_0}[\ln L_c(\theta|\mathbf{x}, \mathbf{y})], \tag{13.4}$$

where the expectation is taken with respect to the conditional pdf:

$$\begin{aligned} h(y|\theta_0, x, T) &= \frac{1}{\sqrt{2\pi}} e^{-(y-\theta_0)^2/2} \frac{1}{1 - F_Y(T, \theta_0)} \\ &= \frac{1}{\sqrt{2\pi}} e^{-(y-\theta_0)^2/2} \frac{1}{1 - \Phi(T - \theta_0)}, \end{aligned}$$

where:

$$F_Y(T, \theta_0) = \int_{-\infty}^{T} \frac{1}{\sqrt{2\pi}} e^{-(y-\theta_0)^2/2} dy = \int_{-\infty}^{T-\theta_0} \frac{1}{\sqrt{2\pi}} e^{-u^2/2} du = \Phi(T - \theta_0).$$

Thus, from Eqs. (13.4) and (13.5),

$$\begin{aligned} Q(\theta|\theta_0, \mathbf{x}) &= E_{\theta_0} \sum_{i=1}^{n_1} \ln\left[\frac{1}{\sqrt{2\pi}} e^{-\frac{(x_i-\theta)^2}{2}}\right] + E_{\theta_0} \ln\left[\frac{1}{\sqrt{2\pi}^{n_2}} e^{-\frac{(y_i-\theta)^2}{2}}\right] \\ &= -\frac{n_1}{2}\ln(2\pi) - \sum_{i=1}^{n_1} \frac{(x_i - \theta)^2}{2} \\ &\quad + n_2 \int_T^{\infty} \ln\left[\frac{1}{\left(\sqrt{2\pi}\right)^{n_2}} e^{-\frac{(y_i-\theta)^2}{2}}\right] \times \frac{1}{\sqrt{2\pi}} e^{-(y-\theta_0)^2/2} \frac{1}{1 - \Phi(T - \theta_0)} dy. \end{aligned}$$

Now taking the derivative with respect to θ*, we have:*

$$\frac{\partial Q}{\partial \theta} = \sum_{i=1}^{n_1} (x_i - \theta)^2 + \frac{n_2}{\sqrt{2\pi}} \int_T^{\infty} (y - \theta) \frac{e^{-(y-\theta_0)^2/2}}{1 - \Phi(T - \theta_0)} dy$$

$$= \sum_{i=1}^{n_1} x_i - n_1\theta + \frac{n_2}{[1 - \Phi(T - \theta_0)]} \Phi(T - \theta_0) - n_2(\theta - \theta_0).$$

Solving $\frac{\partial Q}{\partial \theta} = 0$, *and letting* $n = n_1 + n_2$, *we obtain:*

$$\theta = \frac{\sum_{i=1}^{n_1} x_i}{n} + \frac{n_2}{n}\theta_0 + \frac{n_2 \Phi(T - \theta_0)}{1 - \Phi(T - \theta_0)}. \tag{13.5}$$

From Eq. (13.5), we obtain the EM algorithm as:

$$\widehat{\theta}_{m+1} = \frac{\sum_{i=1}^{n_1} x_i}{n} + \frac{n_2}{n}\widehat{\theta}_m + \frac{n_2 \Phi\left(T - \widehat{\theta}_m\right)}{1 - \Phi\left(T - \widehat{\theta}_m\right)},$$

where Φ *is the cumulative distribution function of a standard normal random variable.*

We have seen that incomplete data could occur as a result of missing data, or the complete data may contain variables that are not observable (hidden). The following is an example of the latter situation.

EXAMPLE 13.4.3

Suppose that in a set of n twin pairs of children, n_1 are male twin pairs, n_2 are female twin pairs, and $n_3 = n - (n_1 + n_2)$ are opposite-sex twin pairs. Let p be the probability that a twin pair is identical and q be the probability that a child is male. It is not known which pairs of same-sex twins are identical. Obtain an EM sequence for $\theta = (p, q)$.

Solution

We have $n = (n_1 + n_2 + n_3)$, *and* $\theta = (p, q)$ *is the parameter vector. Let* $x = (n_1, n_2, n_3)$ *be the observed data. Because we do not know which pairs of the same sex are identical, postulate the complete data set as* $z = (n_{11}, n_{12}, n_{21}, n_{22}, n_3)$, *where* n_{11} *is the number of male identical pairs,* n_{21} *is the number of female identical pairs, and* n_{12} *and* n_{22} *are the nonidentical pairs for males and females, respectively. Here, the complete data, z, have a multinomial distribution with the likelihood given by:*

$$\begin{aligned} L(\mathbf{z}, \theta) &= f(\mathbf{z}|\theta) \\ &= \binom{n}{n_{11}, n_{12}, n_{21}, n_{22}, n_3} (pq)^{n_{11}} \left[(1-p)q^2\right]^{n_{12}} [p(1-q)]^{n_{21}} \\ &\quad \times \left[(1-p)(1-q)^2\right]^{n_{22}} [2(1-p)(1-q)q]^{n_3} \end{aligned}$$

where the identical twins involve one choice of sex and the nonidentical twins involve two choices of sex. The log-likelihood for the complete data is:

$$\begin{aligned} \ln f(x|\theta) &= (n_{11} + n_{21}) \ln p + (n_{12} + n_{22} + n_3) \ln(1-p) \\ &\quad + (n_{11} + 2n_{12} + n_3) \ln q + (n_{21} + 2n_{22} + n_3) \\ &\quad \times \ln(1-q) + \text{constant}. \end{aligned}$$

For the E step, use Bayes' rule to obtain the following:

$$n_{11}^{(k)} = E\left(n_{11}|x,\theta_{(k)}\right) = n_1\frac{p_{(k)}q_{(k)}}{p_{(k)}q_{(k)}+\left(1-p_{(k)}\right)\left(q_{(k)}\right)^2}$$

$$n_{12}^{(k)} = E\left(n_{12}|x,\theta_{(k)}\right) = n_1\frac{\left(1-p_{(k)}\right)\left(q_{(k)}\right)^2}{p_{(k)}q_{(k)}+\left(1-p_{(k)}\right)\left(q_{(k)}\right)^2}$$

$$n_{21}^{(k)} = E\left(n_{21}|x,\theta_{(k)}\right) = n_2\frac{p_{(k)}\left(1-q_{(k)}\right)}{p_{(k)}\left(1-q_{(k)}\right)+\left(1-p_{(k)}\right)\left(1-q_{(k)}\right)^2}$$

and

$$n_{22}^{(k)} = E\left(n_{22}|x,\theta_{(k)}\right) = n_2\frac{\left(1-p_{(k)}\right)\left(1-q_{(k)}\right)^2}{p_{(k)}\left(1-q_{(k)}\right)+\left(1-p_{(k)}\right)\left(1-q_{(k)}\right)^2}.$$

Thus, the **Q** *function is given by:*

$$\begin{aligned}Q\left(\theta,\theta_{(k)}\right) = &\left(n_{11}^{(k)}+n_{21}^{(k)}\right)\ln p+\left(n_{12}^{(k)}+n_{22}^{(k)}+n_3\right)\ln(1-p)\\ &+\left(n_{11}^{(k)}+2n_{21}^{(k)}+n_3\right)\ln q+\left(n_{21}^{(k)}+2n_{22}^{(k)}+n_3\right)\\ &\times\ \ln(1-q)+\text{constant}.\end{aligned}$$

It can be verified that the M step gives the following:

$$p_{(k+1)} = \frac{n_{11}^{(k)}+n_{21}^{(k)}}{n}$$

and

$$q_{(k+1)} = \frac{n_{11}^{(k)}+2n_{12}^{(k)}+n_3}{n+n_{12}^{(k)}+n_{22}^{(k)}}.$$

Substituting for the log-likelihoods with log-posteriors, the EM algorithm can also be used for computations related to Bayesian analysis to find the posterior mode of θ. In the context of incomplete data coming from mixtures of parametric families, the EM algorithm provides a very powerful numerical technique. In this book, we will not go into the mixture models. The steps necessary to compute the required quantities depend on the particular application, and thus, in general, how to code the EM algorithm is not clear. There are special cases available in some software packages such as SAS using PROC MI with EM option when the data come from a multivariate normal distribution. It is desirable to search the literature on the particular software you are using to find the availability of "EM codes" to suit the particular application in which you are interested. Also, another difficulty with implementation of the EM algorithm is that in each E step, we require computation of the conditional expectation. To overcome this difficulty, Wei and Tanner in 1990 proposed an algorithm called MCEM (Monte Carlo EM) based on the Monte Carlo approach explained in Section 13.7. This basically involves simulating m variables, $Y_1, \ldots, Y_m$, from the conditional distribution $h(\mathbf{y}|\theta_{(n)}, \mathbf{x})$ and then maximizing the approximate complete data likelihood, that is,

$$\widehat{Q}\left(\theta\middle|\widehat{\theta}_{(n)},\mathbf{x}\right) = \frac{1}{m}\sum_{i=1}^{m}[\ln g(\mathbf{x},\mathbf{y}|\theta)].$$

We will not go into the details of these methods. The student may refer to Wei and Tanner's paper for further details.

Exercises 13.4

13.4.1. Suppose that Y is a noise-corrupted observation of a signal S. That is, $Y = S + N$, where S is independent of N. Assume that for a known σ, $N \sim N(0, \sigma^2)$ and $S \sim N(0, \theta^2)$, where θ is unknown. Given the observation $Y = y$:

(a) Obtain the MLE, $\widehat{\theta}_{ML}$.

(b) Obtain an EM algorithm.

13.4.2. Let $X_1, \ldots, X_n$ be an observed random sample and $X_{(n_1+1)}, \ldots, X_n$ be the missing (at random) observations. Assume that X_i are iid from an $N(\mu, \sigma^2)$ distribution.

(a) Show that $\left(\Sigma_{i=1}^{n} X_i,\ \Sigma_{i=1}^{n} X_i^2\right)$ are sufficient statistics for $\theta = (\mu, \sigma^2)$.

(b) Obtain the EM sequence for $\theta = (\mu, \sigma^2)$.

(c) Consider a censored normal sample with $n = 10$, with the largest three being censored:

$$1.613 \quad 1.644 \quad 1.663 \quad 1.732 \quad 1.740 \quad 1.763 \quad 1.778$$

Using the results of (a), obtain an EM estimate of $\theta = (\mu, \sigma^2)$ with an arbitrary starting point.

13.4.3. In Example 13.4.3, suppose that q is the probability that a child is a female. Obtain an EM sequence for $\theta = (p, q)$.

13.4.4. Let $\mathbf{x} = (x_1, \ldots, x_{n_1})$ and the censored observations be $(x_{n_1}+1, \ldots, x_n)$ (that is, in the ith experiment, if $i > n_1$, the survival time is at least y_i). Let the new complete censored data y_i be such that:

$$y_i = \begin{cases} x_i, & i \leq n_1 \\ y_i, & i > n_1. \end{cases}$$

Let the mean survival time be θ and the probability density of y be:

$$f(y|\theta) = \theta^{-1} \exp(-y/\theta), \quad y > 0$$

and let the survival function be defined as the probability that an individual survives beyond time y, that is, $S(y) = P(Y > y)$. Thus,

$$S(y) = \exp(-y/\theta), \quad y > 0.$$

(a) Obtain the MLE, $\widehat{\theta}_{ML}$.

(b) Obtain an EM algorithm.

13.4.5. Let $\mathbf{x} = (x_1, \ldots, x_{n_1})$ be observed data and the censored observations be $\mathbf{y} = \left(y_1, \ldots, y_{n_2}\right)$ (that is, in the ith experiment, if $i > n_1$, the survival time is at least y_i). Let the mean survival time be 9, and the probability density be given by:

$$f(x|\theta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(x-\theta)^2\right).$$

(a) Obtain the MLE, $\widehat{\theta}_{ML}$.

(b) Obtain an EM algorithm.

13.5 Introduction to Markov chain Monte Carlo

In this section, we give a brief introduction to Markov chain Monte Carlo (MCMC) methods. Among the computational simulation methods, MCMC is enormously useful for realistic statistical modeling. MCMC methods were initially developed and used in physics. These methods have had a profound influence in statistics since the turn of the century, especially in Bayesian inference. MCMC is a computer-driven sampling method that allows us to characterize a distribution without the knowledge of the distribution's mathematical properties. MCMC methods are used to solve problems in many diverse areas such as archaeology, biology, biophysics, computational chemistry, computer graphics, finance, nuclear medicine, transport theory, and zoology. These methods have enabled researchers to exploit a degree of complexity and realism in modeling and analysis of problems in these areas that were previously beyond reach. The name *Monte Carlo method* was coined by Stan Ulam and John von Neumann, who introduced this method to solve neutron shielding and other related problems at Los Alamos in the early 1940s. MCMC originated with the now classic paper of Metropolis et al., in 1953, where it was used to simulate the distribution of states for a system of idealized molecules.

The popular MCMC procedures make use of two standard algorithms: the Metropolis algorithm and the Gibbs sampler. In the Metropolis approach, all the parameters are varied at once. In the Gibbs method, each variable of the target pdf is changed one at a time. An improvement on Metropolis, called the Metropolis–Hastings (M–H) algorithm, was introduced by Hastings in 1970. There are other efficient hybrid methods, such as the Hamiltonian Monte Carlo method, which

alternates between Gibbs and Metropolis procedures. In our present study, we will explain only the first three methods, namely, the Metropolis algorithm, the M−H algorithm, and the Gibbs sampler.

The objective of MCMC techniques is to generate random variables having certain distributions called target distributions with pdf $\pi(x)$. The simulation of standard distributions is readily available in many statistical software packages, such as Minitab. In cases where the functional form of $\pi(x)$ is not known, MCMC techniques become very useful. The basic idea of MCMC methods is to find a Markov chain with a stationary distribution that is the same as the desired probability distribution $\pi(x)$; this is the target distribution. Run the Markov chain for a long time (say, K iterations) and observe in which state the chain is after these K iterations. The probability that the chain is in state x will be approximately the same as the probability that the discrete random variable equals x.

In Bayesian analysis, whether we are finding a posterior distribution or a Bayesian estimate (usually, the posterior mean), integration is involved. We know from calculus that obtaining closed-form solutions for integrations becomes almost impossible (too difficult) for all but some simple functions. A standard approach to numerical integration of a function $f(x)$ is to first divide the range of integration R into n segments $x_1, \ldots, x_n$, calculate the value of $f(x)$ at each of these points $f(x_1), \ldots, f(x_n)$, multiply the values by the length of each segment, and sum these rectangles to approximate the integral, which is the area under the curve. The error in this approximation is reduced by increasing the number of segments n.

In *Monte Carlo integration*, instead of taking $x_1, \ldots, x_n$ as fixed deterministic numbers, we proceed to draw a random sample from a uniform distribution over the range of integration R, then evaluate $f(x_i)$ for each x_i, and take the average. This assumes that the range R is bounded. If R is not bounded, then $f(x)$ can be integrated when it can be written as the product of another function $h(x)$ and a distribution function $\pi(x)$ from which we can draw values of x (that is, $x_1, \ldots, x_n$ is drawn from the distribution $\pi(x)$). That is,

$$\int f(x)dx = \int h(x)\pi(x)dx,$$

where integration is over the range R. Then, the integral can be approximated by averaging the $f(x_i)$ values, that is,

$$\int f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} h(x_i),$$

where we assume that x_i values are a random sample from $\pi(x)$ and in the range R. When $\pi(x)$ is a standard distribution, many statistical software packages, such as Minitab, can generate random samples from this distribution. In these cases, a general coding to evaluate this integral can be written as:

```
sum ≐ 0
   For i = 1 to n
         {Draw x_i from π(x)
         sum ≐ sum + h(x_i)}
   return sum/n
```

In the preceding coding, by multiplying $h(x_i)$ by the indicator function of R (that is, $I_R(x_i) = 1$, if $x_i \in R$, and 0 otherwise), we can avoid the assumption that x_i values are in the range R. For instance, let $X_1, \ldots, X_n$ be a random sample generated from a target pdf, $\pi(x)$. Then the expectation of any function $f(X)$ can be estimated using the Monte Carlo method by:

$$E_\pi f(X) = \int f(x)\pi(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i) = \bar{f},$$

where E_π denotes the expectation with respect to the pdf $\pi(x)$. By the law of large numbers, it follows that:

$$\frac{1}{n}\sum_{i=1}^{n} f(X_i) \to E_\pi[f(X)] \quad \text{as} \quad n \to \infty,$$

provided $X_1, \ldots, X_n$ are independent. We can verify that $\overline{f}$ is an unbiased estimate of $E_\pi f$. In addition, the sampling distribution of $\overline{f}$ is approximately normal, with variance σ^2/n, where σ^2 is estimated by:

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}\left(f(x_i) - \overline{f}\right)^2.$$

For example, in a Bayesian setting, an estimate of the posterior mean can be obtained by taking $f(x) = x$, and the variance can be obtained by taking $f(x) = (x - \overline{x})^2$, if $\pi(x)$ is the posterior distribution (recall that in Chapter 10, we used the notation $\pi(\theta|x)$ for the posterior distribution). Using the sampling distribution of $\overline{f}$, we can also construct point and interval estimates for $E_\pi f$.

Observe that the heart of the Monte Carlo method is to obtain random samples from the target distribution $\pi(x)$. One of the problems encountered using this approach is that, while it is easy to generate samples from standard distributions using popular statistical software packages, it is very difficult (sometimes not feasible) to do so from any distribution that is not standard (see Project 4A for a method of generating random samples from a given distribution). For these reasons, the ordinary Monte Carlo method can be implemented in only a very few cases for Bayesian inference. That is where the MCMC method plays a crucial role. MCMC methods allow the data analyst to build and analyze more realistic statistical models that may be more complex than standard formulations.

Using the MCMC methods, we will construct a Markov chain $\{X_n\}$ with a limiting distribution as the target distribution, $\pi(x)$. Let us first introduce the concept of Markov chains. For a brief description of Markov chains, refer to Appendix A2. We call a sequence of random variables $\{X_n\}$ a *Markov chain* with state space S if:

$$P(X_n = x_n|X_{n-1} = x_{n-1}, \ldots, X_1 = x_1) = P(X_n = x_n|X_{n-1} = x_{n-1}).$$

That is, the probability distribution of future states of a Markov chain depends only on the present state and not on the past states. However, it is important to note that a Markov chain $\{X_n\}$ is a dependent sequence of random variables; thus, the independence assumption inherent in a random sample cannot be used. The *transition probability function* of a discrete parameter Markov chain is defined as:

$$p_{m,n}(x, y) = P(X_n = y|X_m = x), \quad x, y \text{ in } S.$$

We denote this transition probability simply as $p(x, y)$. When the number of elements in the state space S is finite, we can form a matrix P with the (x, y)th element being $p(x, y)$. This matrix is called a one-step *transition probability matrix.* $\pi(x)$ is called an *invariant (limiting) distribution* if it satisfies the equation:

$$\pi(x) = \sum_{y \in S}\pi(y)p(y, x).$$

We say that the chain satisfies the *reversibility or detailed balanced condition* if $\pi(x)p(x, y) = \pi(y)p(y, x)$ holds for some $\pi(.)$. It can be shown that such a $\pi(x)$ that satisfies the reversibility condition is invariant. Basically, if a Markov chain is reversible and its limiting distribution exists, then the limiting distribution is the invariant distribution.

The results explained for discrete Markov chains can be extended to continuous time defined in a continuous state space. The stationary or the equilibrium distribution $\pi(x)$ of a continuous Markov chain satisfies:

$$\pi(x) = \int p(y, x)\pi(y)dy.$$

Assume that the samples are generated from a Markov chain whose equilibrium distribution is the target distribution, $\pi(x)$. We know by the law of large numbers that:

$$\frac{1}{n}\sum_{i=1}^{n}f(X_i) \to E_\pi[f(X)] \quad \text{as} \quad n \to \infty$$

provided $X_1, \ldots, X_n$ are independent. It turns out that, if we generate a Markov chain $X_1, \ldots, X_n$ from the target distribution $\pi(x)$, the result:

$$\frac{1}{n}\sum_{i=1}^{n}f(X_i) \to E_\pi[f(X)] \quad \text{as} \quad n \to \infty$$

still holds. In this sense, the chain $\{X_i\}$ resulting from an MCMC algorithm with stationary distribution π is similar to the use of a random sample from π. The analytical details are beyond the scope of this book. Instead, we focus on the question, How do we construct a Markov chain whose stationary distribution is our target distribution, $\pi(x)$? The answer is given by the M−H algorithm, and the two special cases: the Metropolis algorithm and the Gibbs sampler. An MCMC method for simulating a distribution π can be defined as any method that produces an ergodic (thus, forgets the initial starting point x_0) Markov chain $\{X_i\}$ whose stationary distribution is π. We start with the Metropolis algorithm. Subsequently, we will explain both the M−H algorithm and the Gibbs sampler. MCMC methods are increasingly being used for simulation of complex probability models, for computation of integrals, and optimization.

13.5.1 Metropolis algorithm

One of the simplest algorithms in MCMC calculations is the Metropolis algorithm, introduced by the Greek American mathematician Nicholas Constantine Metropolis and his colleagues in 1953. This work was mentioned in *Computing in Science & Engineering* as being among the top 10 algorithms having the "greatest influence on the development and practice of science and engineering in the 20th century." In this case, we make a trial perturbation from the current position in a parameter space by randomly selecting a trial step from a symmetric probability distribution called *candidate-generating density* or *proposal density* $q(x, y)$ (in the discrete case, it is a symmetric matrix called the *nominating matrix* $A = (a_{ij})$, with $a_{ij} = a_{ji}$, where $i, j \in S$, the state space of the Markov chain). The $q(x, y)$ depends only on the current state x and the new proposed state y (that is, $q(x, y) = q_x(y)$ is a function of the next proposed state y that is allowed to depend on the current state x). Thus, starting at x, $q(x, y)$ can be regarded as the conditional density of landing at y in one transition step. The trial step is either accepted or rejected on the basis of the probability of the new position relative to the previous one. The Metropolis algorithm is formulated as an instance of the rejection method used for generating steps in a Markov chain. The idea of the rejection algorithm is that if we want to sample from a specific distribution, simply sample from any distribution that is convenient, but keep only the good samples.

We now give the Metropolis algorithm for a discrete distribution. We want to obtain a sample from a distribution $\{\pi_j\}$, where $\pi(j) = P(X_{k+1} = j)$, and we have a symmetric nominating matrix A; then we can write the Metropolis algorithm in four steps as follows.

Metropolis algorithm (discrete case)

For $k = 0$, start with an arbitrary point, $x_k = i$.

1. Generate j from the probability distribution $\{a_{ij},\ j = 1, 2, \ldots\}$.
2. Set
$$r = \frac{\pi(j)}{\pi(i)}.$$
3. If $r \geq 1$, set $x_{k+1} = j$ (acceptance),
otherwise generate u from Uniform (0, 1);
if $u < r$, set $x_{k+1} = j$ (acceptance),
else $x_{k+1} = x_k$ (rejection) (note that the value of x_{k+1} becomes the next state).
4. Set $k = k + 1$, go to step 1.

Each of the accepted points is considered to be a sample value from the target distribution $\{\pi_j\}$.
The continuous case of the Metropolis algorithm is given next.

Metropolis algorithm (continuous case)

1. Start with an arbitrary point, x_0.
2. Select a new position $x^* = \Delta_k + \Delta_x$, where Δx is randomly chosen from a symmetric distribution.
3. Calculate the ratio:
$$r = \frac{\pi(x^*)}{\pi(x_k)},$$
where $\pi(x)$ is the target distribution.
4. Accept the trial position, that is, set
$$x_{k+1} = x^*, \quad \text{if } r \geq 1.$$
Otherwise generate u from Uniform (0, 1).
If $u < r$, set $x_{k+1} = x^*$,
else set $x_{k+1} = x_k$.
5. Set $k = k + 1$, go to step 2.

If the proposal step size is dx, we could use the proposal distribution as $U(-dx, dx)$; for example, if the step size is 1, then randomly choose $\Delta x \sim U(-1, 1)$. For further discussion on selection of the proposal distribution, read Subsection 13.8.4. The Metropolis algorithm generates a set of states that is a Markov chain because each state x_{k+1} depends only on the previous state x_k. Using Markov chain techniques, it can be shown that the equilibrium distribution of the chain constructed by the Metropolis algorithm is indeed $\pi(x^*)$. Note that in the Metropolis algorithm, it is not necessary to have the pdf; instead, all that is necessary is to know the ratio $\pi(x^*)/\pi(x_k)$. Thus, none of the multiplicative constants in the pdf π plays a role in the algorithm.

This algorithm works well in most applications. Following is a simple example to show how the Metropolis algorithm works.

EXAMPLE 13.5.1

Using the Metropolis algorithm, generate a random sample from a Poisson distribution with mean λ. For the nominating matrix, use the symmetric matrix with elements:

$$a_{00} = 1/2, a_{ij} = \begin{cases} 1/2, & j = i-1 \\ 1/2, & j = i+1 \\ 0, & \text{otherwise.} \end{cases}$$

Solution

The nominating probability matrix is a one-step transition matrix (see Appendix A2),

$$A = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & \dots \\ 1/2 & 0 & 1/2 & 0 & 0 & \dots \\ 0 & 1/2 & 0 & 1/2 & 0 & \dots \\ 0 & 0 & 1/2 & 0 & 1/2 & \dots \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \end{bmatrix}.$$

Now we apply the Metropolis algorithm for generating samples from Poisson (λ) in the following steps.

Step 1. *Start with* $x_{n-1} = i$.

Step 2. *Generate* j *from* $A = \{a_{ij}\}$. *How do we do it? We can do this using the following procedure:*

For $i \neq 0$,

Generate u_1 *from* $U(0, 1)$.

If $u_1 \geq \frac{1}{2}$, *set* $j = i + 1$, *else set* $j = i - 1$.

For $i = 0$,

if $u_1 < \frac{1}{2}$, *set* $j = 0$,

else set $j = 1$.

Step 3. ***Set***

$$r = \frac{\pi(i)}{\pi(i)} = \frac{e^{-\lambda}\lambda^j/j!}{e^{-\lambda}\lambda^i/i!} = \frac{i!\lambda^j}{j!\lambda^i} = \frac{i!\lambda^{j-i}}{j!}.$$

Set

$$r = \begin{cases} 1, & \text{if } i = 0, j = 0 \\ \dfrac{\lambda}{j}, & \text{if } j = i+1 \\ \dfrac{i}{\lambda}, & \text{if } j = i-1. \end{cases}$$

Step 4. *Acceptance/rejection:*

If $r \geq 1$, *set* $x_n = j$ *(i.e., accept the new state j).*

Otherwise, generate u_2 *from* $U(0, 1)$;

if $u_2 < r$, *set* $x_n = j$ *(i.e., accept the new state* j*),*

else set $x_n = x_{n-1}$ *(i.e., reject the new state j and keep the current state* i*).*

Step 5. *Set* $n = n + 1$, *go to step 2.*

In Example 13.5.1, let us say that we want to generate a random sample from Poisson with $\lambda = 2$ and we are at state $i = 3$ in the iteration step $(n - 1)$. If our proposed new state is $j = 4$, then $r = 2/4 = 1/2$. Suppose we obtained the value of u_2 as 0.672772. Because this value is larger than 1/2, we reject the proposed new state 4 and stay at state 3 for the iteration step n (if you generate a new u_2, your decision might be different). Instead, suppose our proposed step was $j = 2$; then $r = i/\lambda = 3/2 > 1$, and we will immediately accept our new state as $j = 2$ (no need to generate a uniform random number; if you did, it would have been smaller than 3/2 anyway) for the iteration step n.

EXAMPLE 13.5.2

Let $\pi(x) = c\exp(-f(x))$ be the form of the target distribution function. Write a general Metropolis algorithm to generate a sample from π.

Solution

Let q(x, y) *be any symmetric distribution. Starting from an arbitrary* $x_{(0)}$*, we can write the Metropolis algorithm through the following steps.*

Step 1. *Let* $x_{(t)}$ *be the current state.*

Step 2. *Generate* y *from the distribution* q(x, y).

Because,

$$r = \frac{\pi(y)}{\pi(x_{(t)})} = \frac{c\exp(-f(y))}{c\exp(-f(x_{(t)}))} = \exp(-f(y) - f(x_{(t)})),$$

calculate the change in f, $\Delta f = f(y) - f(x_{(t)})$.

Step 3. *Generate a random number from the uniform distribution,* U(0, 1). *If* $u \le \exp(-\Delta f)$*, set* $x_{(t+1)} = y$ *(accept the proposed new state), otherwise set* $x_{(t+1)} = x_{(t)}$ *(reject the proposed new state).*

Step 4. *Continue (i.e., go to step 1).*

Note that in the previous example, the normalizing constant in $\pi(x)$ is not important, because it cancels in the ratio. In fact this is true in all Metropolis and M−H algorithms. In the special case where $q(x, y) = q(|y - x|)$, the Metropolis algorithm is also called the *random-walk Metropolis*. Another special choice is $q(x, y) = q(y)$; this is called the *independence sampler*. In all of these cases, it is important to observe that, whereas the target distribution is independent of the positions, the proposal functions depend on where we are. For example, let $\pi(x)$ be standard normal density, and let the proposal density be of the form:

$$q(x, y) \propto \exp\left(-\frac{(y - x)^2}{2(.25)^2}\right).$$

Fig. 13.1 gives a representation of the target distribution and some representative proposals. For each point x of the target distribution, we generate a y from the corresponding proposal distribution. Then, according to the accept/reject rule that we specified earlier, we will make a decision whether to treat this new value y as being from the target distribution.

13.5.2 The Metropolis−Hastings algorithm

The M−H algorithm is a generalization of the Metropolis algorithm, in which we need not assume symmetry of the nominating matrix A (or for proposal density $q(x, y)$). The acceptance probability is given by:

$$\alpha(i, j) = \min\left\{\frac{\pi(j)a_{ji}}{\pi(i)a_{ij}}, 1\right\}.$$

This algorithm is the basic building block of MCMC methods. The M−H algorithm is widely used in applied statistics and is very useful for sampling from complicated, high-dimensional probability distributions. Now we present the steps involved in the M−H algorithm in the discrete case.

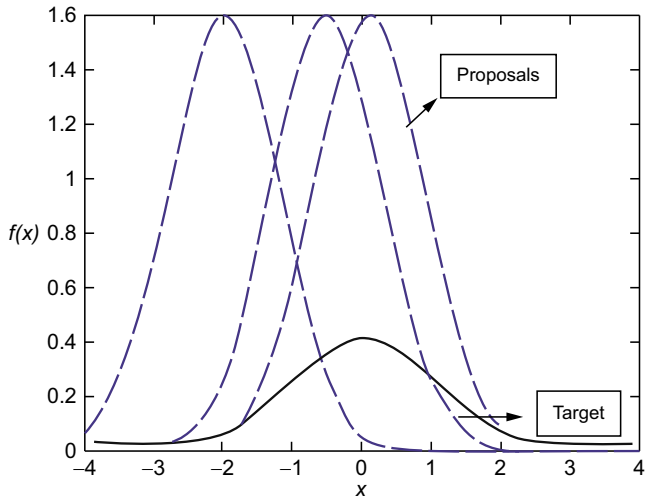


FIGURE 13.1 Target and proposal densities.

Metropolis–Hastings algorithm (discrete case)

For $k = 0$, start with an arbitrary point, $x_k = i$.

1. Generate j from the nominating distribution $\{a_{ij},\ j = 1,\ 2,\ \ldots\}$.
2. Set
$$r = \frac{\pi(j)a_{ji}}{\pi(i)a_{ij}}.$$
3. If $r \geq 1$, set $x_{k+1} = j$.
Otherwise generate u from $U(0,\ 1)$;
if $u < r$, set $x_{k+1} = j$,
else set $x_n = x_{n-1}$.
4. Set $k = k + 1$, go to step 1.

In the preceding algorithm, if we calculate $\alpha(i, j) = \min\{r, 1\}$, basically, we accept the proposed new step j if $u < \alpha(i, j)$; otherwise we stay at the current step i. The resulting Markov chain from both Metropolis and M–H algorithms would have the transition probability matrices defined by:

$$p(i,j) = a_{ij}\alpha(i,j) \text{ for } i \neq j$$

and

$$p(i,i) = 1 - \sum_{j \neq i} a_{ij}\alpha(i,j).$$

In the continuous case, for any given $\pi(x)$, the M–H algorithm takes the following form. To start the algorithm, we choose an arbitrary proposal distribution $q(x, y)$ so that it is easy to obtain a sample from this distribution. Define the acceptance/rejection function as:

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

If both $\pi(x)$ and $\pi(y)$ are zero, set $\alpha(x,\ y) = 0$.

Metropolis–Hastings algorithm (continuous case)

Step 1. Start with an arbitrary point, x_0.
Step 2. Given a current state $x^{(t)}$, draw y from the proposal distribution $q(x, y)$.
Step 3. Draw u from $U[0, 1]$.
Step 4. If $u < \alpha(x^{(t)}, y)$, set $x^{(t+1)} = y$, otherwise set $x^{(t+1)} = x^{(t)}$.
Step 5. Set $t = t + 1$, go to step 2.

Note that if the $q(x, y)$ is symmetric (i.e., $q(x, y) = q(y, x)$), then the M–H algorithm reduces to the Metropolis algorithm. In practice, there are other forms of acceptance/rejection functions suggested. Observe that in the M–H algorithm, as in the Metropolis algorithm, it is not necessary to have the pdf; instead, all that is necessary is to know the ratio $\pi(y)/\pi(x)$. Thus, none of the multiplicative constants in the pdf, $\pi(x)$, plays a role in the algorithm.

Because of the versatility of this method, there are many generalizations of the M–H algorithm in the literature. It is also necessary to impose some conditions both on $\pi(x)$ and on the proposal distribution $q(x, y)$ for $\pi(x)$ to be the limiting distribution of the Markov chain $\{X^{(t)}\}$ produced by the M–H algorithm. We do not want a large ratio of the proposed new values to be rejected. Discussion of these issues is beyond the scope of this book.

EXAMPLE 13.5.3

Using the M–H algorithm, generate a sample from the following distribution. Let $\Omega = \{2, 3, \ldots, 11, 12\}$, which represents the sum of the up faces of two balanced dice, and let the distribution be given by:

Sum *i*	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
$\pi(i)$	1/36	2/36	3/36	4/36	5/36	6/36	5/36	4/36	3/36	2/36	1/36

Using the nominating matrix:

$$a_{22} = a_{(12)(12)} = 1/2,\ a_{ij} = \begin{cases} 1/2, & j = i-1 \\ 1/2, & j = i+1,\ i, j \in \Omega \\ 0, & \text{otherwise,} \end{cases}$$

write the M–H algorithm to generate samples from the distribution $\pi(x)$.

Solution

Suppose that we start with state $i \in \Omega$*, say at 5 (starting at any other state is OK).*

Step 1. *Generate* j *from the nominating distribution* $\{a_{ij}, j = 1, 2, \ldots\}$*. Thus,* $j = i - 1$ *or* $i + 1$*, and in this case* j *has to be 4 or 6. We can follow the same procedure as in Example 13.5.1 to choose between* $i - 1$ *and* $i + 1$*. Let us say we got* $j = i + 1$*, here 6.*
Step 2. *Set* $r = \frac{\pi(j)a_{ji}}{\pi(j)a_{ij}}$*. In this case,* $r = \frac{\pi(6)}{\pi(5)} = \frac{5/36}{4/36} = \frac{5}{4}$*. (If we had chosen 4, then,* $r = \frac{\pi(4)}{\pi(5)} = \frac{3}{4}$*.)*
Step 3. *If* $r \geq 1$*, set* $x_n = j$*. Here,* $r > 1$*; hence, we accept the new state,* $x_n = 6$*. Otherwise generate* u *from* U(0, 1)*, if* $u < r$*, set* $x_n = j$*, else set* $x_n = x_{n-1}$*.*
Step 4. *Set* $n = n + 1$*, and go to step 1.*

EXAMPLE 13.5.4

Write an M–H algorithm to generate samples from $N(0, 1)$ based on the proposal $U[-1, 1]$.

Solution

Note that for y *to be generated based on* $U[-1, 1]$*, we need* $y - x^{(t)} \sim U[-1, 1]$*. Thus,* $y \sim U[x^{(t)} - 1, x^{(t)} + 1]$*. Fig. 13.2 shows the target distribution as the standard normal and the representative proposals that are uniform at points* $x^{(t)} = -2$ *and* 2.

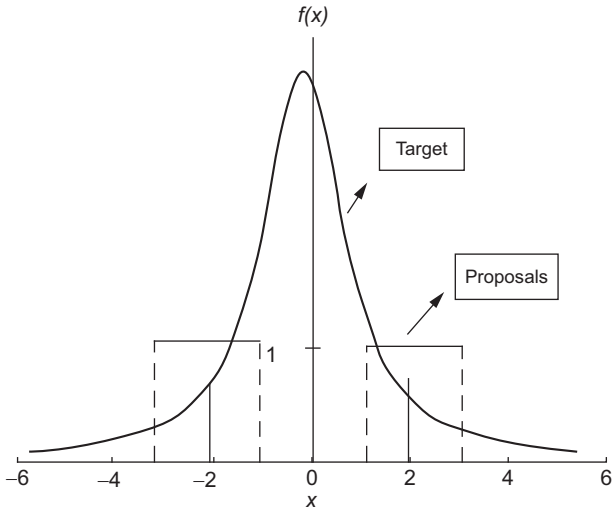


FIGURE 13.2 Normal target and uniform proposal distributions.

Now, the M–H algorithm can be obtained in the following way.

Set

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

$$= \min\left\{\left(\exp\left\{x^{(t)^2} - y^2/2\right\}\right)\frac{x+1}{y+1}, 1\right\}.$$

Generate $u \sim U[0, 1]$.
If $u < \alpha(x^{(t)}, y)$, *set* $x^{(t+1)} = y$, *otherwise set* $x^{(t+1)} = x^{(t)}$. *Continue.*

Observe that to generate normal random variables, it is not necessary to use M–H algorithms. Most of the statistical software packages will give us a random sample from the normal distribution. Example 6.5.2 (originally suggested by Hastings in 1970) is given for demonstration of the M–H algorithm. The algorithm is effective in general cases, for instance, to generate a sample from a gamma distribution. In *Gamma*(α, β), if α is an integer, we can use the method of Project 4A to generate a random sample. However, if α is not an integer, we could use *Gamma*([α], β) (here [α] denotes the integer part of α) as the proposal distribution, and follow the steps of the M–H algorithm to generate a sample from *Gamma*(α, β) (see Exercise 13.5.3).

13.5.3 Gibbs algorithm

The name *Gibbs algorithm* (or *Gibbs sampler*) was coined by the brothers Stuart Geman and Donald Geman in 1984 and refers to Gibbs distributions in statistical physics. This is very useful in obtaining a sequence of observations from a specified multivariate probability distribution, when direct sampling is hard or the joint distribution is not known explicitly. A Gibbs sampler can be used in those situations when the conditional distribution of each variable is known and is relatively easier to sample from. In the Gibbs sampler, only one parameter is varied at a time, while all others are held fixed. The parameter then is randomly drawn from a conditional pdf, the probability distribution of one parameter, given all other parameters, $\pi(x_i|x_{-i})$, where x_{-i} is the full set of parameters excluding only the single component x_i. Let $x = (x_1, \ldots, x_k)$ be k ($\geq$2)-dimensional. Recall from Chapter 3 that these conditional densities can be obtained as follows:

$$\pi(x_i|x_{-i}) = \pi(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)$$

$$= \frac{\pi(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_k)}{\int \pi(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_k)dx_i}.$$

The basic assumption under which the Gibbs algorithm works is that we could easily draw a random sample from these conditional pdfs. Thus, the Gibbs algorithm is a particular case of M−H algorithms. For example, at the ith step, y_i is generated from the nominating density $q_i(x_i, y_i)$ where q_i depends on the current state x_i. The candidate y_i is accepted with probability:

$$\alpha_i(x_i, y_i) = \min\left\{\frac{\pi_i(y_i)q_i(y_i, x_i)}{\pi_i(x_i)q_i(x_i, y_i)}, 1\right\}.$$

If y_i is accepted, we will set the ith component of x_n, $x_{n,\,i} = y_i$; otherwise set $x_{n,\,i} = x_{n\,i}$. The remaining components of $\mathbf{x}_n$ are not changed in step i. This is repeated for each i, at the end of which the entire vector $\boldsymbol{x}_n$ would have been updated. Thus, if we are in state $\mathbf{x}$ at time t, at time $t + 1$ we either remain at $\mathbf{x}$ or go to $\mathbf{y}$ by modifying only one component of $\mathbf{x}$. It is important to use the most recent values of updated components to update the next component. That is, given $\mathbf{x}^{(t)} = \left(x_1^{(t)}, \ldots, x_k^{(t)}\right)$ at time t, generate:

$$x_1^{(t+1)} \sim \pi\left(x_1 \middle| x_2^{(t)}, x_3^{(t)}, \ldots, x_k^{(t)}\right)$$

$$x_2^{(t+1)} \sim \pi\left(x_2 \middle| x_1^{(t+1)}, x_3^{(t)}, \ldots, x_k^{(t)}\right)$$

$$x_3^{(t+1)} \sim \pi\left(x_3 \middle| x_1^{(t+1)}, x_2^{(t+1)}, x_4^{(t)}, \ldots, x_k^{(t)}\right)$$

.

.

.

$$x_k^{(t+1)} \sim \pi\left(x_k \middle| x_1^{(t+1)}, x_2^{(t+1)}, \ldots, x_{k-1}^{(t+1)}\right).$$

For instance, let $k = 2$. The Gibbs sampler updates in the following manner. Start at $\mathbf{x}^{(0)} = \left(x_1^{(0)}, x_2^{(0)}\right)$; first update $x_1^{(0)}$ to $x_1^{(1)}$, using this updated value $x_1^{(1)}$ and$x_2^{(0)}$, update $x_2^{(0)}$ to$x_2^{(1)}$, resulting in the updated vector $\boldsymbol{x}^{(1)}$. Repeat this procedure to obtain $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$, …. Fig. 13.3 depicts this updating procedure.

The conditional densities $f_1, \ldots, f_k$ are called the *full conditionals*. In the Gibbs sampler, only these conditional densities are needed for simulation. Thus, this procedure becomes very efficient when the vector $\mathbf{x}$ is large, because all of the simulations can be done as univariate.

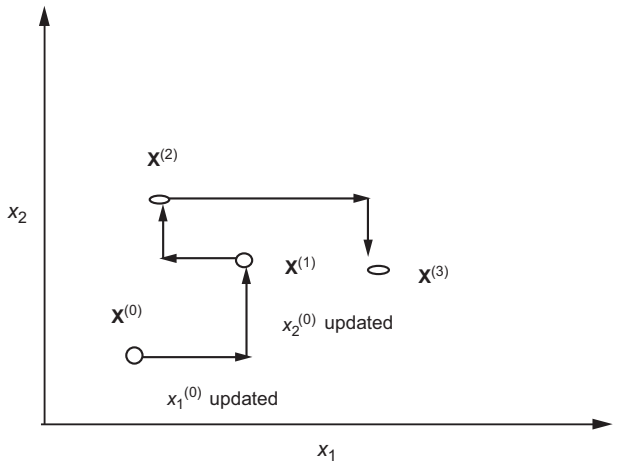


FIGURE 13.3 Gibbs updating procedure.

The following example of bivariate density is popularly used in the literature to illustrate the Gibbs sampler. It is the case where the joint density is complex, because one variable (x) is discrete, while the other variable (y) is continuous. However, the conditional densities are simple known distributions, binomial and beta distributions, respectively. It is then easier to simulate these distributions, thus, demonstrating the power of the Gibbs sampler.

EXAMPLE 13.5.5

(a) Write a Gibbs sampler for generating samples from the following bivariate density:

$$f(x,y) = \binom{n}{x} y^{x+\alpha-1}(1-y)^{n-x+\beta-1}, \text{ for } x = 0,1,\ldots,n$$

and $0 \le y \le 1$.

(b) Starting with $y_0 = 1/4$, $n = 15$, and $\alpha = 1$, $\beta = 2$, obtain the first three realizations of the Gibbs sequence.

Solution

(a) *From Exercise 3.3.14, we know that:*

$$f(x|y) \propto \binom{n}{x} y^x (1-y)^{n-x}.$$

That is, the conditional distribution of x *(treating* y *as a constant) is binomial with parameters* n *and* y, $0 \le y \le 1$. *Also,*

$$f(x|y) \propto y^{x+\alpha-1}(1-y)^{n-x+\beta-1}.$$

Thus, the conditional distribution of y *given* x *is a beta distribution with parameters* $x + \alpha$ *and* $n - x + \beta$. *The Gibbs sampler for generating bivariate samples from* f(x, y) *is then given as follows: For* $i = 1, \ldots, n$, *repeat:*

1. *Generate* y_i *from* $f_{Y|X}\left(\left|x^{(i-1)}\right.\right)$, *that is, from* ***Beta***$(x_{i-1} + \alpha,\ n - x_{i-1} + \beta)$
2. *Generate* x_i *from* $f_{X|Y}\left(\left|y^{(i)}\right.\right)$, *that is, from* ***binomial*** **(n, y_i).**
3. *Return* (x_i, y_i).

(b) *We proceed with the following steps.*

(i) *For* $y_0 = 1/4$, x_0 *is obtained from generating a random variable from binomial with* $n = 15$, $y_0 = 1/4$, *that is, from* B(15, 1/4), *resulting in a value of 4 (generated using Minitab; you may get a different value when you do it). Thus,* $x_0 = 4$.

(ii) *Generate* y_1 *randomly from:*

$$Beta(x_0 + \alpha, n - x_0 + \beta) = Beta(4 + 1, 15 - 4 + 2)$$
$$= Beta(5, 13),$$

resulting in $y_1 = 0.53$ *(approximated to second digit). Now* $x_1 \sim$ B(15, 0.53), *resulting in* $x_1 = 6$.

(iii) *Generate* y_2 *randomly from:*

$$Beta(x_1 + \alpha, n - x_1 + \beta) = Beta(7, 11),$$

resulting in $y_2 = 0.30$. *Now,* $x_2 \sim$ B(15, 0.30), *resulting in* $x_2 = 3$.

Thus, a particular realization of the Gibbs sampler for the first three iterations is (4, 0.25), (6, 0.53), and (3, 0.30).

From Exercise 13.5.8, it can be observed that, at the beginning, the values of the chain are highly dependent on the choice of the initial value y_0. In practice, it is necessary to run a sufficient number of iterations to remove the effect of the starting values. Even though the Gibbs sampler is a special case of the M−H algorithm, it is important to observe that, unlike the M−H algorithm, every sample generated by the Gibbs algorithm is accepted. Also, we should have at least a two-dimensional problem for the Gibbs sampler to be used. Since Gibbs sampling (like other MCMC sampling) generates a Markov chain of samples, each sample is correlated with neighboring samples; to obtain a random sample, one needs to perform *thinning* of the resulting chain by taking only every kth value (like taking every 50th value). There are some pros and cons to the practice of thinning; for that and some nice applications of the Gibbs method, we refer the reader to specialized books.

From the previous discussions, we can see that a general description of an MCMC method can be summarized in the following algorithm.

Initialize X_0
For $i = 1; \ldots; N$ repeat
$x = X_{i-1}$;
Generate Y from a nominating density, $q(x; y)$;
Calculate the acceptance rate, $\alpha(x; y)$;
Generate U from the uniform $U(0; 1)$;
If $(U < \alpha(x; y))$ set $X_{(i)} = y$,
Else set $X_{(i)} = x$;
End;

If we choose a nominating density $q(x, y)$ and an acceptance rate $\alpha(x, y)$, such that, the reversibility condition:

$$\pi(x)\alpha(x, y)q(x, y) = \pi(y)\alpha(y, x)q(y, x),$$

is satisfied, then the foregoing procedure generates a Markov chain with limiting distribution $\pi(x)$. To use Gibbs sampling for Bayesian analysis, we must have an explicit analytical posterior conditional distribution.

13.5.4 Markov chain Monte Carlo issues

Two major issues in MCMC are convergence and burn-in. Because in all three MCMC algorithms we start the sequence from an arbitrary point, any particular sequence may take some time to pass through the transient stage, and the effect of the starting value is very small and can be ignored—that is, it attains convergence. In practice, we will have to run the algorithm for a few thousand iterations so that the effect of this initial state is negligible. The samples obtained during this burn-in period should be discarded for the subsequent analysis as they do not represent the target pdf. By monitoring the sequence itself, we can determine whether the sequence has reached the convergence. A simple way to decide how much burn-in is necessary is to create scatterplots of X_i versus X_j, $i \neq j$. When the wild variations stop, then it is safe to assume that the chain has reached stationarity.

Another major issue in the implementation of MCMC algorithms is the choice of proposal density. In the continuous case, popular choices among others are the multivariate normal density and the multivariate t with specified parameters. Even in these cases, there is the question of appropriate size of the spread, or scale of the proposal density. The size of the acceptance ratio is another issue. If the ratio is too small, the samples will get stuck (because almost all proposed new states will be rejected), and if the ratio is too high, the samples will show tracking. A general rule of thumb is that the acceptance ratio should be within 30%−60%. If not, adjust the step size (for a small ratio, decrease the step size, and for a high ratio, increase the step size). There are many publications devoted to these issues.

For the Bayesian computation, MCMC allows us to sample from any posterior. Because of the availability of specialized software packages, such as BUGS, it is practical to code up for a particular problem.

There are many references, including books, on MCMC methods; some of these are listed in the references at the end of this book. For a good discussion including some technical details, refer to http://mpdc.mae.cornell.edu/Courses/MAE714/Papers/wp9.pdf.

Exercises 13.5

13.5.1. For Example 13.5.1, let $\lambda = 3$. Starting with initial state $x_0 = 6$, compute relevant quantities performing 10 iterations of the algorithm.

13.5.2. Using the M−H algorithm, generate a random sample from a geometric distribution with mean θ. Use the nominating distribution $\{a_{ij}, j = 1, 2, \ldots\}$, such that,

$$a_{ij} = \begin{cases} \frac{1}{2} & j = i-1, i+1, \text{ and } i = 1, 2, 3, \ldots \\ \frac{1}{2} & j = 0, 1 \text{ and } i = 0 \\ 0 & \text{otherwise.} \end{cases}$$

(Recall that if X is geometric with parameter θ, then $P(X = x) = (1 - \theta)^x\theta$, for $x = 0, 1, 2, \ldots$.)

13.5.3. Write down the M−H algorithm to generate a sample from *Gamma*(α, β) using the proposal density as *Gamma*([α], [α]/α).

13.5.4. Write down the M−H algorithm for simulating a Markov chain with stationary distribution $\pi = (1/6, 2/3, 1/6)$, using the "proposal" transition matrix:

$$Q = \begin{pmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1/2 & 1/2 \end{pmatrix}.$$

13.5.5. In tossing three fair coins, let the random variable X be defined as $X =$ number of tails. Then the distribution of X is given by:

x	0	1	2	3
$\pi(x)$	1/8	3/8	3/8	1/8

Write down the Metropolis or M−H algorithm for simulating a Markov chain with stationary distribution $\pi(x)$. Use any nominating matrix.

13.5.6. Write a Metropolis algorithm to generate samples from a target distribution, $\pi(x) \propto \exp\left(-\frac{x^2}{2}\right)$, based on the proposal density:

$$q_x(y) = \exp\left(-\frac{(y-x)^2}{2(0.4)^2}\right).$$

13.5.7. Write a general Metropolis or M−H algorithm to generate a sample from a target distribution π, where π is an exponential random variable with parameter θ.

13.5.8. Write a general Metropolis or M−H algorithm to generate a sample from a target distribution π, where $\pi(x)$ α $x^{34}(1-x)^{38}(2+x)^{125}$. Use the proposal density as $q(x, y) = 1$ on the interval [0, 1].

13.5.9. For the bivariate density given in Example 13.5.5, starting with three different values of y_0, say, 1/3, 1/2, and 2/3; $n = 15$; and $\alpha = 1$, $\beta = 2$, obtain the first three realizations of the Gibbs sequence. Comment on the influence of the initial values.

13.5.10. Consider a problem of sampling bivariate random variables with joint density given by:

$$f(x, y) = \begin{cases} ce^{-(x+y+4xy)}, & x \geq 0, y \geq 0 \\ 0, & \text{otherwise.} \end{cases}$$

(a) Find $f(x|y)$ and $f(y|x)$.
(b) Write a Gibbs procedure to generate samples from this distribution. Discuss why it is easier to use the Gibbs sampler for this case.
(c) Starting from an arbitrary point, obtain the first three sample points.

13.5.11. Suppose the target distribution is:

$$(X, Y) \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right).$$

Then write the Gibbs sampler to generate a sample from this distribution. In particular, say, we start with $(X, Y) = (12, 12)$ and $\rho = 0.7$. What is the Gibbs procedure to generate a sample from a binormal distribution? The pdf of a bivariate normal distribution with:

$$\boldsymbol{x} = \begin{pmatrix} x \\ y \end{pmatrix} \quad \boldsymbol{\mu} = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2 \end{pmatrix}$$

is given by:

$$f(\boldsymbol{x}) = \frac{1}{2\pi}(\det\boldsymbol{\Sigma})^{-1/2} \exp\left[-\frac{1}{2}(x-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu}) \right]$$

where ' denotes the vector transpose.

13.5.12. Suppose the target distribution is:

$$(X, Y) \sim N\left(\begin{pmatrix} \mu \\ \mu \end{pmatrix}, \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \right).$$

Then write the Gibbs sampler to generate a sample from this distribution.

13.6 Chapter summary

In this chapter, we introduced some empirical methods that are becoming increasingly popular in modern statistical analysis. The methods presented must be viewed as introductory in nature and by no means most efficient or general. Because of ever-evolving applications and advancements in technology, most of the methods presented here also evolve. Also, based on the situation, it is necessary to write computer codes to run the algorithms introduced in this chapter. Our hope is that students will explore these topics in more detail by referring to specialized books and publications.

In this chapter, we also learned the following important concepts and procedures:

- The jackknife method
- General bootstrap procedure to estimate the standard error of $\widehat{\theta}$
- Bootstrap confidence intervals
- EM algorithm
- MCMC methods
- Metropolis algorithm
- M−H algorithm
- Gibbs sampler

13.7 Computer examples

Most of the procedures described in this chapter could be implemented using Minitab, SAS, or SPSS. There are other specialized programs that will do a good job of implementing the methods discussed in this chapter. BUGS (Bayesian Inference Using Gibbs Sampling) is free software that has proven to be effective in MCMC computations, and the details are at the website: http://www.mrc-bsu.cam.ac.uk/bugs/. Most of the procedures discussed in this chapter can also be implemented in R, which is also free software that can be downloaded from http://www.rproject.org/. A few examples in R are given. We will also present an example in Minitab. However, we will not discuss SAS or SPSS examples.

13.7.1 Examples using R

EXAMPLE 13.7.1 Bootstrap

Using the following data, perform a bootstrap point and interval estimate for the median. Generate six replications or bootstrap samples of size 12 each.

Sample (x): 269 246 388 354 266 303 295 259 274 249 271 254

R-code:

```
library('boot');
mystatfun = function(data,index) {
return(median(data[index]));
}
mybs=boot(x,mystatfun,R=6);
print(mean(mybs$t));
print(sd(mybs$t));
boot.ci(mybs,type="basic");
```

Load the boot library

Create a function returning your parameter to be estimated. Notice this requires an index.

mybs$t contains the values generated by your function from each bootstrap replication

Output:

```
269.6667
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 6 bootstrap replicates

CALL:
boot.ci(boot.out = mybs, type = "basic")

Intervals :
Level Basic
95% (241, 286)
Calculations and Intervals on Original Scale
Warning: Basic Intervals used Extreme Quantiles
Some basic intervals may be unstable
```

EXAMPLE 13.7.2 Jackknife

Using the data from the previous example, perform a jackknife point estimate for the mean and standard deviation. Notice the jackknife computation is not simulated like the bootstrap and will have one answer.

R-code:

```
tmp=c();
for(i in 1:12) {
tmp=c(tmp,mean(x[-i]));
}
mean(tmp);
sd(tmp);
```

Output:

```
285.6667

3.988777
```

Mean

Standard deviation

EXAMPLE 13.7.3 Markov chain Monte Carlo

MCMC is used to simulate random variables from distributions we cannot sample from. In this example our **target** distribution is chisq(4) and our **proposal** distribution is normal(i,1). Notice we can use the rchisq() function to do this and obtain a better result; however we are going to compare the results of rchisq() with MCMC for learning purposes. Another important note, our proposal distribution's mean is the previous value in the Markov chain.

The chain variable may be treated as a generated random sample from our target distribution. Notice that we can evaluate the target distribution but perhaps we cannot sample or integrate the target.

Your means will be unique for both your rchisq() and your chain but they should be close. Observe the density curves over the histogram (Fig. 13.4).

R-code:

```
i=10; #Step 1.
chain=c();
for(c in 1:100,000) {
  j=rnorm(1,i,1); #Step 2
  u=runif(1,0,1); #Step 3
  r=(dchisq(j,df=4)*dnorm(j,i,1))/(dchisq(i,df=4)*dnorm(i,j,1));
  a=min(c(r,1),na.rm=TRUE);
  if(u<a) {
    chain=c(chain,j);
    i=j;
  } else {
    chain=c(chain,i);
  }
}
mean(chain);
mean(rchisq(3000,df=4));
plot(density(chain),col="blue",type="l");
lines(density(rchisq(3000,df=4)),col="red");
lines(seq(0,25,by=0.1),dchisq(seq(0,25,by=0.1),df=4),col="black");
```

Step 4

Step 5

Output:

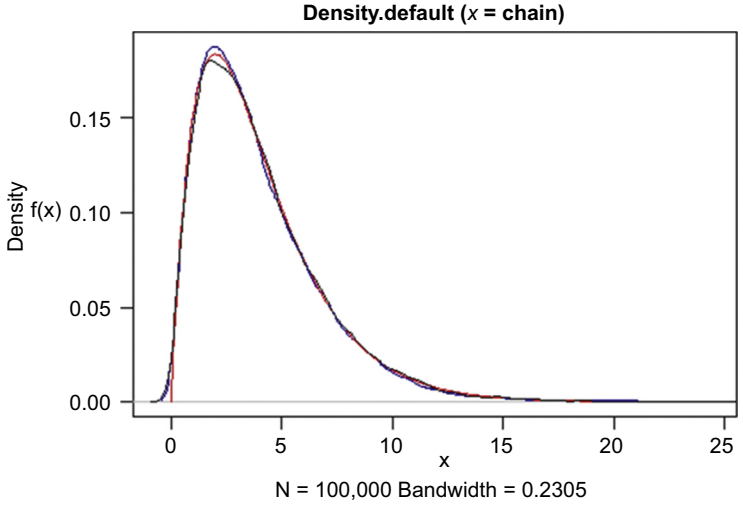


FIGURE 13.4 Simulated densities.

EXAMPLE 13.7.4 (EM algorithm)

Using the data of Exercise 13.4.2 (c) give the R-code.

Solution

We take arbitrary initial values for the parameters μ *and* σ.

```
#Change the values in (*) and (**) and put any arbitrary values as follows:
em.norm <- function(Y){
Yobs <- Y[!is.na(Y)]
Ymis <- Y[is.na(Y)]
n <- length(c(Yobs, Ymis))
r <- length(Yobs)
# initial values.
mut <- 1 # (*)put arbitrary value for μ
sit <- 0.1 # (**)put arbitrary value for σ
# Define log-likelihood function
ll <- function(y, mu, sigma2, n){
-.5*n*log(2*pi*sigma2)-.5*sum((y-mu)^2)/sigma2
}
# Compute the log-likelihood for the initial values, and ignoring the missing data mechanism
lltm1 <- ll(Yobs, mut, sit, n)
repeat{
# E-step
EY <- sum(Yobs) + (n-r)*mut
EY2 <- sum(Yobs^2) + (n-r)*(mut^2 + sit)
# M-step
mut1 <- EY / n
sit1 <- EY2 / n - mut1^2
# Update parameter values
mut <- mut1
sit <- sit1
# compute log-likelihood using current estimates, and ignoring the missing data mechanism
llt <- ll(Yobs, mut, sit, n)
# Print current parameter values and likelihood
cat(mut, sit, llt, "\n")
# Stop if converged
if ( abs(lltm1 - llt) < 0.001) break
lltm1 <- llt
}
# fill in missing values with new mu.
return(mut,sit)
}
```

EXAMPLE 13.7.5 (MCMC) Write an MCMC algorithm for Example 13.5.4.

Solution

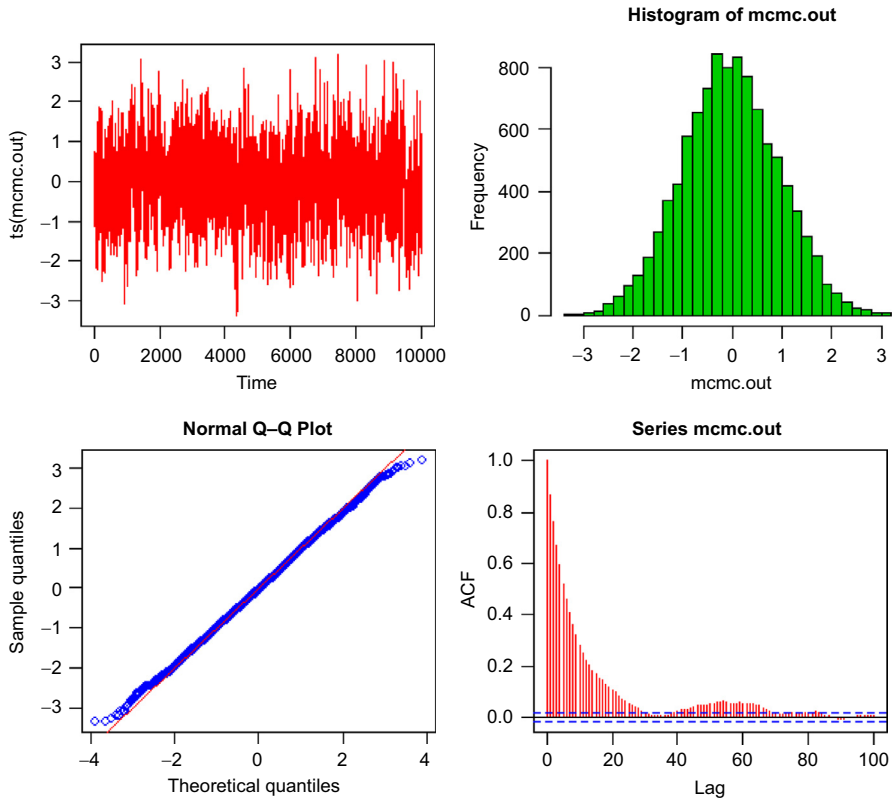
```
metrop3=function(n=1000,eps=0.5)
{
     vec=vector("numeric", n)
     x=0
     oldll=dnorm(x,log=TRUE)
     vec[1]=x
     for (i in 2:n) {
       can=x+runif(1,-eps,eps)
       loglik=dnorm(can,log=TRUE)
       loga=loglik-oldll
       if (log(runif(1)) < loga) {
         x=can
         oldll=loglik
```

```
      }
    vec[i]=x
  }
  vec
}
```

In addition, if we want to plot the results, use the following code:

```
plot.mcmc<-function(mcmc.out)
{
    op=par(mfrow=c(2,2))
    plot(ts(mcmc.out),col=2)
    hist(mcmc.out,30,col=3)
    qqnorm(mcmc.out,col=4)
    abline(0,1,col=2)
    acf(mcmc.out,col=2,lag.max=100)
    par(op)
}
metrop.out<-metrop3(10000,1)
plot.mcmc(metrop.out)
```

With the plot, we get the following output.



EXAMPLE 13.7.6 (Gibbs sampler) Write an R-code for Example 13.5.5 (b).

Solution

```
#R program for Gibbs sampling
>
> n=15
```

```
> y0=1/4
> p=y0
> x0=rbinom(1,n,p)
>
> a=1
> b=2
> A=x0+a
> B=n-x0+b
> X=matrix(x0,3);Y=matrix(y0,3)
>
> for(i in 2:3){#sample from f(y/x)
+ Y[i]=rbeta(1,A,B)
+ #sample from f(x/y)
+ X[i]=rbinom(1,n,Y[i])
+ }
> print(matrix(c(X,Y),3,2))
```

Output:

```
[,1] [,2]
[1,] 5 0.2500000
[2,] 5 0.4011747
[3,] 4 0.2047587
```

It should be noted that each time we run the code, we may get different output.

13.7.2 Examples with Minitab

EXAMPLE 13.7.7

Using the data of Example 13.3.2, give the Minitab steps.

Solution

Enter the data in ***C1****. Enter 0.08 (≈ 1/12) 12 times in* ***C2****. Then.*

Calc > ***Random Data*** > ***Discrete*** … > ***Generate*** [enter ***200***] ***rows of data*** > ***Store in column(s)****:* enter ***C3-C14*** > ***values in:*** enter ***C1*** > ***Probabilities in****:* enter ***C2*** > click ***OK***.

*We will get 200 rows of data stored in 12 columns. Because the data are generated randomly from the original data with replacement, we will consider the row data (****C3−C14****) as the sample size and the 200 columns as the number of samples. Thus* $N = 200$, *and* $n = 12$. *Now for each row we can find the mean,* $\overline{X}_i^*$ *by doing the following:*

Calc > ***Row Statistics*** … > click ***Mean*** > in ***Input variables:*** enter ***C3-C14*** > ***store results in****:* enter ***C15*** > click ***OK***.
We will get 200 values representing the sample means. To get the bootstrap mean:

Stat > ***Basic Statistics*** > ***Display Descriptive Statistics*** … > ***Variables:*** enter ***C15*** > click ***OK***.

The value in the mean is the bootstrap mean, and the value in the standard deviation is the bootstrap standard deviation. If we want to get, say, a 95% bootstrap confidence interval, first sort the sample means in ascending order:

Manip > ***Sort*** … > ***Sort column(s)****:* enter ***C15*** > ***store sorted column(s) in:*** enter ***C16*** > ***sorted by column****:* enter ***C15*** > click ***OK***.

Calculate the values of $0.025 \times (N + 1) = 0.025 \times 201 = 5.025$ *and* $0.975 \times (N + 1) = 0.975 \times 201 = 195.975$. *Approximating these values to the nearest integer, we get 5 and 196, respectively. The lower confidence limit will be the fifth entry in the sorted means, and the upper confidence limit will be the 196th value in the sorted means.*

If we want to obtain a confidence interval for the median, we follow very much the same steps as before, but instead of using the mean in the procedure, we substitute the median. For example:

Calc > ***Row Statistics*** … > click ***Median*** > in ***Input variables****:* enter ***C3-C14*** > ***store results in:*** enter ***C15*** > click ***OK***.

The rest of the steps are similar.

13.7.3 SAS examples

There are %JACK and %BOOT macros available to do jackknife and bootstrap computations. A good site with example programs from SAS Institute is http://ftp.sas.com/techsup/download/stat/jackboot.html. Sometimes, PROC IML could also be used to bootstrap. In the case of multivariate normal data, PROC MI with the EM option will perform the EM algorithm in SAS. Refer to http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#mcmc_toc.htm for the options available for the MCMC procedure. Example SAS codes could be obtained from a simple search of the web for almost all the procedures explained in this chapter.

Project for Chapter 13

13A Bootstrap computation

Use any statistical computer program to generate random numbers. By specifying a particular distribution, such as normal with mean 0 and variance 1 or other similar distributions, we can then generate numbers that follow this distribution. (This can be done either directly, if your software allows, or by the method described in Project 4A.)

(a) Use such a package to generate 200 numbers from an $N(0, 1)$ distribution. Then calculate the sample mean and sample variance. (They will be slightly off from the actual mean and variance. From this, we can draw the conclusion that the estimates of data parameters that are computed using the data set are not necessarily the true parameters, but often are reasonable guesses.) Using these values, calculate an estimate of the standard error.

(b) Now for the same data, pretend that we are not really sure what the distribution is. Then, we could consider letting the observed data specify what the distribution is. This is the essence of bootstrapping. In particular, sample, with replacement from a distribution that we have observed (the empirical distribution of the data), to study the possible estimates that might have resulted from a similar sample (same data observations, but in possibly different quantities). Using the bootstrap algorithm described in Section 13.4, obtain a bootstrap estimate of the standard error and compare this with the estimate obtained in (a).