Chapter 5

Monte Carlo Integration and Variance Reduction

5.1 Introduction

Monte Carlo integration is a statistical method based on random sampling. Monte Carlo methods were developed in the late 1940's after World War II, but the idea of random sampling was not new. As early as 1777, Comte de Buffon used a random experiment to empirically check his probability calculation for the famous Buffon's needle experiment. Another well known example is that W. S. Gossett used random sampling to study the distribution of what are now called "Student t" statistics, publishing under the alias Student in 1908 [256]. The development of ENIAC, the first electronic computer, completed in 1946 at the University of Pennsylvania, and the seminal article by Metropolis and Ulam in 1949 [198] marked an important new era in the application of sampling methods. Teams of scientists at the Los Alamos National Laboratory and many other researchers contributed to the early development, including Ulam, Richtmyer, and von Neumann [276, 283]. For an interesting discussion of the history of the Monte Carlo method and scientific computing, see Eckhart [78] and Metropolis [195, 196].

5.2 Monte Carlo Integration

Let $g(x)$ be a function and suppose that we want to compute $\int_a^b g(x)dx$ (assuming that this integral exists). Recall that if X is a random variable with density $f(x)$, then the mathematical expectation of the random variable $Y = g(X)$ is

$$E[g(X)] = \int_{-\infty}^{\infty} g(x)f(x)dx.$$

If a random sample is available from the distribution of X, an unbiased estimator of $E[g(X)]$ is the sample mean.

5.2.1 Simple Monte Carlo estimator

Consider the problem of estimating $\theta = \int_0^1 g(x)dx$. If $X_1, \dots, X_m$ is a random Uniform(0,1) sample then

$$\hat{\theta} = \overline{g_m(X)} = \frac{1}{m}\sum_{i=1}^{m} g(X_i)$$

converges to $E[g(X)] = \theta$ with probability 1, by the Strong Law of Large Numbers. The simple Monte Carlo estimator of $\int_0^1 g(x)dx$ is $\overline{g_m(X)}$.

Example 5.1 (Simple Monte Carlo integration)

Compute a Monte Carlo estimate of

$$\theta = \int_0^1 e^{-x}\, dx$$

and compare the estimate with the exact value.

```
m <- 10000
x <- runif(m)
theta.hat <- mean(exp(-x))
print(theta.hat)
print(1 - exp(-1))

[1] 0.6355289
[1] 0.6321206
```

The estimate is $\hat{\theta} \doteq 0.6355$ and $\theta = 1 - e^{-1} \doteq 0.6321$. ◇

To compute $\int_a^b g(t)dt$, make a change of variables so that the limits of integration are from 0 to 1. The linear transformation is $y = (t-a)/(b-a)$ and $dy = (1/(b-a))dt$. Substituting,

$$\int_a^b g(t)dt = \int_0^1 g(y(b-a)+a)(b-a)dy.$$

Alternately, we can replace the Uniform(0,1) density with any other density supported on the interval between the limits of integration. For example,

$$\int_a^b g(t)dt = (b-a)\int_a^b g(t)\frac{1}{b-a}dt$$

is $b-a$ times the expected value of $g(Y)$, where Y has the uniform density on (a,b). The integral is therefore $(b-a)$ times the average value of $g(\cdot)$ over (a,b).

Example 5.2 (Simple Monte Carlo integration, cont.)

Compute a Monte Carlo estimate of

$$\theta = \int_2^4 e^{-x}\, dx$$

and compare the estimate with the exact value of the integral.

```
m <- 10000
x <- runif(m, min=2, max=4)
theta.hat <- mean(exp(-x)) * 2
print(theta.hat)
print(exp(-2) - exp(-4))

[1] 0.1172158
[1] 0.1170196
```

The estimate is $\hat{\theta} \doteq 0.1172$ and $\theta = e^{-2} - e^{-4} \doteq 0.1170$. $\diamond$

To summarize, the simple Monte Carlo estimator of the integral $\theta = \int_a^b g(x)dx$ is computed as follows.

1. Generate $X_1, \ldots, X_m$, iid from Uniform(a, b).
2. Compute $\overline{g(X)} = \frac{1}{m} g(X_i)$.
3. $\hat{\theta} = (b - a)\overline{g(X)}$.

Example 5.3 (Monte Carlo integration, unbounded interval)

Use the Monte Carlo approach to estimate the standard normal cdf

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}}\, e^{-t^2/2}\, dt.$$

First, notice that we cannot apply the algorithm above directly because the limits of integration cover an unbounded interval. However, we can break this problem into two cases: $x \geq 0$ and $x < 0$, and use the symmetry of the normal density to handle the second case. Then the problem is to estimate $\theta = \int_0^x e^{-t^2/2} dt$ for $x > 0$. This can be done by generating random Uniform$(0, x)$ numbers, but it would mean changing the parameters of the uniform distribution for each different value of the cdf required. Suppose that we prefer an algorithm that always samples from Uniform(0,1).

This can be accomplished by a change of variables. Making the substitution $y = t/x$, we have $dt = x\, dy$ and

$$\theta = \int_0^1 x e^{-(xy)^2/2} dy.$$

Thus, $\theta = E_Y[xe^{-(xY)^2/2}]$, where the random variable Y has the Uniform(0,1) distribution. Generate iid Uniform(0,1) random numbers $u_1, \ldots, u_m$, and compute

$$\hat{\theta} = \overline{g_m(u)} = \frac{1}{m}\sum_{i=1}^{m} x\, e^{-(u_i x)^2/2}.$$

The sample mean $\hat{\theta}$ converges to $E[\hat{\theta}] = \theta$ as $m \to \infty$. If $x > 0$, the estimate of $\Phi(x)$ is $0.5 + \hat{\theta}/\sqrt{2\pi}$. If $x < 0$ compute $\Phi(x) = 1 - \Phi(-x)$.

```
x <- seq(.1, 2.5, length = 10)
m <- 10000
u <- runif(m)
cdf <- numeric(length(x))
for (i in 1:length(x)) {
    g <- x[i] * exp(-(u * x[i])^2 / 2)
    cdf[i] <- mean(g) / sqrt(2 * pi) + 0.5
}
```

Now the estimates $\hat{\theta}$ for ten values of x are stored in the vector `cdf`. Compare the estimates with the value $\Phi(x)$ computed (numerically) by the `pnorm` function.

```
Phi <- pnorm(x)
print(round(rbind(x, cdf, Phi), 3))
```

Results for several values $x > 0$ are shown compared with the value of the normal cdf function `pnorm`. The Monte Carlo estimates appear to be very close to the `pnorm` values. (The estimates will be worse in the extreme upper tail of the distribution.)

```
     [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10]
x    0.10 0.367 0.633 0.900 1.167 1.433 1.700 1.967 2.233 2.500
cdf  0.54 0.643 0.737 0.816 0.879 0.925 0.957 0.978 0.990 0.997
Phi  0.54 0.643 0.737 0.816 0.878 0.924 0.955 0.975 0.987 0.994
```

Notice that it would have been simpler to generate random Uniform$(0, x)$ random variables and skip the transformation. This is left as an exercise. In fact, the integrand of the previous example is itself a density function, and we can generate random variables from this density. This provides a more direct approach to estimating the integral. ◇

Example 5.4 (Example 5.3, cont.)

Let $I(\cdot)$ be the indicator function, and $Z \sim N(0, 1)$. Then for any constant x we have $E[I(Z \leq x)] = P(Z \leq x) = \Phi(x)$, the standard normal cdf evaluated at x.

Generate a random sample $z_1, \dots, z_m$ from the standard normal distribution. Then the sample mean

$$\widehat{\Phi(x)} = \frac{1}{m} \sum_{i=1}^{m} I(z_i \le x)$$

converges with probability one to its expected value $E[I(Z \le x)] = P(Z \le x) = \Phi(x)$.

```
x <- seq(.1, 2.5, length = 10)
m <- 10000
z <- rnorm(m)
dim(x) <- length(x)
p <- apply(x, MARGIN = 1,
        FUN = function(x, z) {mean(z < x)}, z = z)
```

Now the estimates in `p` for the sequence of x values can be compared to the result of the R normal cdf function `pnorm`.

```
Phi <- pnorm(x)
print(round(rbind(x, p, Phi), 3))

     [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10]
x   0.10  0.367 0.633 0.900 1.167 1.433 1.700 1.967 2.233 2.500
p   0.546 0.652 0.741 0.818 0.876 0.925 0.954 0.976 0.988 0.993
Phi 0.54  0.643 0.737 0.816 0.878 0.924 0.955 0.975 0.987 0.994
```

In this example, compared with the results in Example 5.3, it appears that we have better agreement with `pnorm` in the upper tail, but worse agreement near the center. ◇

Summarizing, if $f(x)$ is a probability density function supported on a set A, (that is, $f(x) \ge 0$ for all $x \in \mathbb{R}$ and $\int_A f(x) = 1$), to estimate the integral

$$\theta = \int_A g(x) f(x) dx,$$

generate a random sample $x_1, \dots, x_m$ from the distribution $f(x)$, and compute the sample mean

$$\hat{\theta} = \frac{1}{m} \sum_{i=1}^{m} g(x_i).$$

Then with probability one, $\hat{\theta}$ converges to $E[\hat{\theta}] = \theta$ as $m \to \infty$.

The standard error of $\hat{\theta} = \frac{1}{m}\sum_{i=1}^{m} g(x_i)$.

The variance of $\hat{\theta}$ is σ^2/m, where $\sigma^2 = Var_f(g(X))$. When the distribution of X is unknown we substitute for F_X the empirical distribution F_m of the sample $x_1, \dots, x_m$. The variance of $\hat{\theta}$ can be estimated by

$$\frac{\hat{\sigma}^2}{m} = \frac{1}{m^2}\sum_{i=1}^{m}[g(x_i) - \overline{g(x)}]^2. \tag{5.1}$$

Note that

$$\frac{1}{m}\sum_{i=1}^{m}[g(x_i) - \overline{g(x)}]^2 \tag{5.2}$$

is the plug-in estimate of $Var(g(X))$. That is, (5.2) is the variance of U, where U is uniformly distributed on the set of replicates $\{g(x_i)\}$. The corresponding estimate of standard error of $\hat{\theta}$ is

$$\widehat{se}(\hat{\theta}) = \frac{\hat{\sigma}}{\sqrt{m}} = \frac{1}{m}\left\{\sum_{i=1}^{m}[g(x_i) - \overline{g(x)}]^2\right\}^{1/2}. \tag{5.3}$$

The Central Limit Theorem implies that

$$\frac{\hat{\theta} - E[\hat{\theta}]}{\sqrt{Var\,\hat{\theta}}}$$

converges in distribution to $N(0,1)$ as $m \to \infty$. Hence, if m is sufficiently large, $\hat{\theta}$ is approximately normal with mean θ. The large-sample, approximately normal distribution of $\hat{\theta}$ can be applied to put confidence limits or error bounds on the Monte Carlo estimate of the integral, and check for convergence.

Example 5.5 (Error bounds for MC integration)

Estimate the variance of the estimator in Example 5.4, and construct approximate 95% confidence intervals for the estimate of $\Phi(2)$ and $\Phi(2.5)$.

```
x <- 2
m <- 10000
z <- rnorm(m)
g <- (z < x)  #the indicator function
v <- mean((g - mean(g))^2) / m
cdf <- mean(g)
c(cdf, v)
c(cdf - 1.96 * sqrt(v), cdf + 1.96 * sqrt(v))

[1] 9.772000e-01 2.228016e-06
[1] 0.9742744 0.9801256
```

The probability $P(I(Z < x) = 1)$ is $\Phi(2) \approxeq 0.977$. Here $g(X)$ has the distribution of the sample proportion of 1's in $m = 10000$ Bernoulli trials with $p \doteq 0.977$, and the variance of $g(X)$ is therefore $(0.977)(1 - 0.977)/10000 =$ 2.223e-06. The MC estimate 2.228e-06 of variance is quite close to this value.

For $x = 2.5$ the output is

```
[1] 9.94700e-01 5.27191e-07
[1] 0.9932769 0.9961231
```

The probability $P(I(Z < x) = 1)$ is $\Phi(2.5) \approxeq 0.995$. The Monte Carlo estimate 5.272e-07 of variance is approximately equal to the theoretical value $(0.995)(1 - 0.995)/10000 =$ 4.975e-07. ◇

5.2.2 Variance and Efficiency

We have seen that a Monte Carlo approach to estimating the integral $\int_a^b g(x)dx$ is to represent the integral as the expected value of a function of a uniform random variable. That is, if $X \sim$ Uniform(a, b), then $f(x) = \frac{1}{b-a}$, $a < x < b$, and

$$\begin{aligned}\theta &= \int_a^b g(x)dx \\ &= (b-a)\int_a^b g(x)\frac{1}{b-a}dx = (b-a)E[g(X)].\end{aligned}$$

Recall that the sample-mean Monte Carlo estimator of the integral θ is computed as follows.

1. Generate $X_1, \ldots, X_m$, iid from Uniform(a, b).
2. Compute $\overline{g(X)} = \frac{1}{m}g(X_i)$.
3. $\hat{\theta} = (b-a)\overline{g(X)}$.

The sample mean $\overline{g(X)}$ has expected value $g(X) = \theta/(b-a)$, and

$$Var(\overline{g(X)}) = (1/m)Var(g(X)).$$

Therefore $E[\hat{\theta}] = \theta$ and

$$Var(\hat{\theta}) = (b-a)^2 Var(\overline{g(X)}) = \frac{(b-a)^2}{m} Var(g(X)). \qquad (5.4)$$

By the Central Limit Theorem, for large m, $\overline{g(X)}$ is approximately normally distributed, and therefore $\hat{\theta}$ is approximately normally distributed with mean θ and variance given by (5.4).

The "hit-or-miss" approach to Monte Carlo integration also uses a sample mean to estimate the integral, but the sample mean is taken over a different sample and therefore this estimator has a different variance than formula (5.4).

Suppose $f(x)$ is the density of a random variable X. The "hit-or-miss" approach to estimating $F(x) = \int_{-\infty}^{x} f(t)dt$ is as follows.

1. Generate a random sample $X_1, \dots, X_m$ from the distribution of X.
2. For each observation X_i, compute

$$g(X_i) = I(X_i \leq x) = \begin{cases} 1, & X_i \leq x; \\ 0, & X_i > x. \end{cases}$$

3. Compute $\widehat{F(x)} = \overline{g(X)} = \frac{1}{m} \sum_{i=1}^{m} I(X_i \leq x)$.

Note that the random variable $Y = g(X)$ has the Binomial$(1, p)$ distribution, where the success probability is $p = P(X \leq x) = F(x)$. The transformed sample $Y_1, \dots, Y_m$ are the outcomes of m independent, identically distributed Bernoulli trials. The estimator $\widehat{F(x)}$ is the sample proportion $\hat{p} = y/m$, where y is the total number of successes observed in m trials. Hence $E[\widehat{F(x)}] = p = F(x)$ and $Var(\widehat{F(x)}) = p(1-p)/m = F(x)(1-F(x))/m$.

The variance of $\widehat{F(x)}$ can be estimated by $\hat{p}(1-\hat{p})/m = \widehat{F(x)}(1-\widehat{F(x)})/m$. The maximum variance occurs when $F(x) = 1/2$, so a conservative estimate of the variance of $\widehat{F(x)}$ is $1/(4m)$.

Efficiency

If $\hat{\theta}_1$ and $\hat{\theta}_2$ are two estimators for θ, then $\hat{\theta}_1$ is more efficient (in a statistical sense) than $\hat{\theta}_2$ if

$$\frac{Var(\hat{\theta}_1)}{Var(\hat{\theta}_2)} < 1.$$

If the variances of estimators $\hat{\theta}_i$ are unknown, we can estimate efficiency by substituting a sample estimate of the variance for each estimator.

Note that variance can alway be reduced by increasing the number of replicates, so computational efficiency is also relevant.

5.3 Variance Reduction

We have seen that Monte Carlo integration can be applied to estimate functions of the type $E[g(X)]$. In this section we consider several approaches to reducing the variance in the sample mean estimator of $\theta = E[g(X)]$.

If $\hat{\theta}_1$ and $\hat{\theta}_2$ are estimators of the parameter θ, and $Var(\hat{\theta}_2) < Var(\hat{\theta}_1)$, then the percent reduction in variance achieved by using $\hat{\theta}_2$ instead of $\hat{\theta}_1$ is

$$100 \left(\frac{Var(\hat{\theta}_1) - Var(\hat{\theta}_2)}{Var(\hat{\theta}_1)} \right).$$

The Monte Carlo approach to estimating $\theta = E[g(X)]$ is to compute the sample mean $\overline{g(X)}$ for a large number m of replicates from the distribution of $g(X)$. The function $g(\cdot)$ is often a statistic; that is, an n-variate function $g(X_1, \ldots, X_n)$ of a sample. When $g(X)$ is used in that context, we have $g(\mathcal{X}) = g(X_1, \ldots, X_n)$, where $\mathcal{X}$ denotes the sample elements. Unless it is not clear in context, however, for simplicity we use $g(X)$.

Let

$$X^{(j)} = \{X_1^{(j)}, \ldots, X_n^{(j)}\}, \qquad j = 1, \ldots, m$$

be iid from the distribution of X, and compute the corresponding replicates

$$Y_j = g(X_1^{(j)}, \ldots, X_n^{(j)}), \qquad j = 1, \ldots, m. \tag{5.5}$$

Then $Y_1, \ldots, Y_m$ are independent and identically distributed with distribution of $Y = g(\mathcal{X})$, and

$$E[\overline{Y}] = E\left[\frac{1}{m}\sum_{j=1}^{m} Y_j\right] = \theta.$$

Thus, the Monte Carlo estimator $\hat{\theta} = \overline{Y}$ is unbiased for $\theta = E[Y]$. The variance of the Monte Carlo estimator is

$$Var(\hat{\theta}) = Var\overline{Y} = \frac{Var_f g(X)}{m}.$$

Increasing the number of replicates m clearly reduces the variance of the Monte Carlo estimator. However, a large increase in m is needed to get even a small improvement in standard error. To reduce the standard error from 0.01 to 0.0001, we would need approximately 10000 times the number of replicates. In general, if standard error should be at most e and $Var_f(g(X)) = \sigma^2$, then $m \geq \lceil \sigma^2/e^2 \rceil$ replicates are required.

Thus, although variance can always be reduced by increasing the number of Monte Carlo replicates, the computational cost is high. Other methods for reducing the variance can be applied that are less computationally expensive than simply increasing the number of replicates.

In the following sections some approaches to reducing the variance of this type of estimator are introduced. Several approaches have been covered in the literature. Readers are referred to [69, 112, 113, 121, 228, 233, 238] for reference and more examples.

5.4 Antithetic Variables

Consider the mean of two identically distributed random variables U_1 and U_2. If U_1 and U_2 are independent, then

$$Var\left(\frac{U_1+U_2}{2}\right) = \frac{1}{4}(Var(U_1)+Var(U_2)),$$

but in general we have

$$Var\left(\frac{U_1+U_2}{2}\right) = \frac{1}{4}(Var(U_1)+Var(U_2)+2Cov(U_1,U_2)),$$

so the variance of $(U_1+U_2)/2$ is smaller if U_1 and U_2 are negatively correlated than when the variables are independent. This fact leads us to consider negatively correlated variables as a possible method for reducing variance.

For example, suppose that $X_1,\dots,X_n$ are simulated via the inverse transform method. For each of the m replicates we have generated $U_j \sim$ Uniform(0,1), and computed $X^{(j)} = F_X^{-1}(U_j)$, $j = 1,\dots,n$. Note that if U is uniformly distributed on $(0,1)$ then $1-U$ has the same distribution as U, but U and $1-U$ are negatively correlated. Then in (5.5)

$$Y_j = g(F_X^{-1}(U_1^{(j)}),\dots,F_X^{-1}(U_n^{(j)}))$$

has the same distribution as

$$Y_j' = g(F_X^{-1}(1-U_1^{(j)}),\dots,F_X^{-1}(1-U_n^{(j)})).$$

Under what conditions are Y_j and Y_j' negatively correlated? Below it is shown that if the function g is *monotone*, the variables Y_j and Y_j' are negatively correlated.

Define $(x_1,\dots,x_n) \le (y_1,\dots,y_n)$ if $x_j \le y_j$, $j = 1,\dots,n$. An n-variate function $g = g(X_1,\dots,X_n)$ is *increasing* if it is increasing in its coordinates. That is, g is increasing if $g(x_1,\dots,x_n) \le g(y_1,\dots,y_n)$ whenever $(x_1,\dots,x_n) \le (y_1,\dots,y_n)$. Similarly g is *decreasing* if it is decreasing in its coordinates. Then g is *monotone* if it is increasing or decreasing.

PROPOSITION 5.1 *If $X_1,\dots,X_n$ are independent, and f and g are increasing functions, then*

$$E[f(X)g(X)] \ge E[f(X)]E[g(X)]. \tag{5.6}$$

Proof. Assume that f and g are increasing functions. The proof is by induction on n. Suppose $n = 1$. Then $(f(x)-f(y))(g(x)-g(y)) \ge 0$ for all $x, y \in \mathbb{R}$. Hence $E[(f(X)-f(Y))(g(X)-g(Y))] \ge 0$, and

$$E[f(X)g(X)] + E[f(Y)g(Y)] \ge E[f(X)g(Y)] + E[f(Y)g(X)].$$

Here X and Y are iid, so

$$\begin{aligned}2E[f(X)g(X)] &= E[f(X)g(X)] + E[f(Y)g(Y)] \\ &\geq E[f(X)g(Y)] + E[f(Y)g(X)] = 2E[f(X)]E[g(X)],\end{aligned}$$

so the statement is true for $n = 1$. Suppose that the statement (5.6) is true for $X \in \mathbb{R}^{n-1}$. Condition on X_n and apply the induction hypothesis to obtain

$$\begin{aligned}E[f(X)g(X)|X_n = x_n] &\geq E[f(X_1, \dots, X_{n-1}, x_n)]E[g(X_1, \dots, X_{n-1}, x_n)] \\ &= E[f((X)|X_n = x_n]E[g((X)|X_n = x_n)],\end{aligned}$$

or

$$E[f(X)g(X)|X_n] \geq E[f(X)|X_n]E[g(X)|X_n)].$$

Now $E[f(X)|X_n]$ and $E[g(X)|X_n)]$ are each increasing functions of X_n, so applying the result for $n = 1$ and taking the expected values of both sides

$$E[f(X)g(X)] \geq E[E[f(X)|X_n]\, E[g(X)|X_n)]] \geq E[f(X)]E[g(X)].$$

□

COROLLARY 5.1 *If $g = g(X_1, \dots, X_n)$ is monotone, then*

$$Y = g(F_X^{-1}(U_1), \dots, F_X^{-1}(U_n))$$

and

$$Y' = g(F_X^{-1}(1 - U_1), \dots, F_X^{-1}(1 - U_n)).$$

are negatively correlated.

Proof. Without loss of generality we can suppose that g is increasing. Then

$$Y = g(F_X^{-1}(U_1), \dots, F_X^{-1}(U_n))$$

and

$$-Y' = f = -g(F_X^{-1}(1 - U_1), \dots, F_X^{-1}(1 - U_n))$$

are both increasing functions. Therefore $E[g(U)f(U)] \geq E[g(U)]E[f(U)]$ and $E[YY'] \leq E[Y]E[Y']$, which implies that

$$Cov(Y, Y') = E[YY'] - E[Y]E[Y'] \leq 0,$$

so Y and Y' are negatively correlated. □

The antithetic variable approach is easy to apply. If m Monte Carlo replicates are required, generate $m/2$ replicates

$$Y_j = g(F_X^{-1}(U_1^{(j)}), \dots, F_X^{-1}(U_n^{(j)})) \tag{5.7}$$

and the remaining $m/2$ replicates

$$Y_j' = g(F_X^{-1}(1-U_1^{(j)}), \dots, F_X^{-1}(1-U_n^{(j)})), \tag{5.8}$$

where $U_i^{(j)}$ are iid Uniform(0,1) variables, $i = 1, \dots, n$, $j = 1, \dots, m/2$. Then the antithetic estimator is

$$\begin{aligned}\hat{\theta} &= \frac{1}{m}\{Y_1 + Y_1' + Y_2 + Y_2' + \cdots + Y_{m/2} + Y_{m/2}'\} \\ &= \frac{2}{m}\sum_{j=1}^{m/2}\left(\frac{Y_j + Y_j'}{2}\right).\end{aligned}$$

Thus $nm/2$ rather than nm uniform variates are required, and the variance of the Monte Carlo estimator is reduced by using antithetic variables.

Example 5.6 (Antithetic variables)

Refer to Example 5.3, illustrating Monte Carlo integration applied to estimate the standard normal cdf

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}}\, e^{-t^2/2}\, dt.$$

Repeat the estimation using antithetic variables, and find the approximate reduction in standard error. In this example (after change of variables) the target parameter is $\theta = E_U[xe^{-(xU)^2/2}]$, where U has the Uniform(0,1) distribution.

By restricting the simulation to the upper tail (see Example 5.3) the function $g(\cdot)$ is monotone, so the hypothesis of Corollary 5.1 is satisfied. Generate random numbers $u_1, \dots, u_{m/2} \sim \text{Uniform}(0,1)$ and compute half of the replicates using

$$Y_j = g^{(j)}(u) = x\, e^{-(u_j x)^2/2}, \qquad j = 1, \dots, m/2$$

as before, but compute the remaining half of the replicates using

$$Y_j' = x\, e^{-((1-u_j)x)^2/2}, \qquad j = 1, \dots, m/2.$$

The sample mean

$$\begin{aligned}\hat{\theta} = \overline{g_m(u)} &= \frac{1}{m}\sum_{j=1}^{m/2}\left(x\, e^{-(u_j x)^2/2} + x\, e^{-((1-u_j)x)^2/2}\right) \\ &= \frac{1}{m/2}\sum_{j=1}^{m/2}\left(\frac{x\, e^{-(u_j x)^2/2} + x\, e^{-((1-u_j)x)^2/2}}{2}\right)\end{aligned}$$

converges to $E[\hat{\theta}] = \theta$ as $m \to \infty$. If $x > 0$, the estimate of $\Phi(x)$ is $0.5+\hat{\theta}/\sqrt{2\pi}$. If $x < 0$ compute $\Phi(x) = 1 - \Phi(-x)$. The Monte Carlo estimation of the integral $\Phi(x)$ is implemented in the function `MC.Phi` below. Optionally `MC.Phi` will compute the estimate with or without antithetic sampling. The `MC.Phi` function could be made more general if an argument naming a function, the integrand, is added (see `integrate` for an example of this type of argument to a function).

```
MC.Phi <- function(x, R = 10000, antithetic = TRUE) {
    u <- runif(R/2)
    if (!antithetic) v <- runif(R/2) else
        v <- 1 - u
    u <- c(u, v)
    cdf <- numeric(length(x))
    for (i in 1:length(x)) {
        g <- x[i] * exp(-(u * x[i])^2 / 2)
        cdf[i] <- mean(g) / sqrt(2 * pi) + 0.5
    }
    cdf
}
```

A comparison of estimates obtained from a single Monte Carlo experiment is below.

```
x <- seq(.1, 2.5, length=5)
Phi <- pnorm(x)
set.seed(123)
MC1 <- MC.Phi(x, anti = FALSE)
set.seed(123)
MC2 <- MC.Phi(x)
print(round(rbind(x, MC1, MC2, Phi), 5))

       [,1]    [,2]    [,3]    [,4]    [,5]
x   0.10000 0.70000 1.30000 1.90000 2.50000
MC1 0.53983 0.75825 0.90418 0.97311 0.99594
MC2 0.53983 0.75805 0.90325 0.97132 0.99370
Phi 0.53983 0.75804 0.90320 0.97128 0.99379
```

The approximate reduction in variance can be estimated for given x by a simulation under both methods, the simple Monte Carlo integration approach and the antithetic variable approach.

```
m <- 1000
MC1 <- MC2 <- numeric(m)
x <- 1.95
for (i in 1:m) {
    MC1[i] <- MC.Phi(x, R = 1000, anti = FALSE)
    MC2[i] <- MC.Phi(x, R = 1000)
}

> print(sd(MC1))
[1] 0.007008661
> print(sd(MC2))
[1] 0.000470819
> print((var(MC1) - var(MC2))/var(MC1))
[1] 0.9954873
```

The antithetic variable approach achieved approximately 99.5% reduction in variance at $x = 1.95$. ◇

5.5 Control Variates

Another approach to reduce the variance in a Monte Carlo estimator of $\theta = E[g(X)]$ is the use of control variates. Suppose that there is a function f, such that $\mu = E[f(X)]$ is known, and $f(X)$ is correlated with $g(X)$.

Then for any constant c, it is easy to check that $\hat{\theta}_c = g(X) + c(f(Y) - \mu)$ is an unbiased estimator of θ.

The variance

$$Var(\hat{\theta}_c) = Var(g(X)) + c^2 Var(f(X)) + 2c\, Cov(g(X), f(X)) \tag{5.9}$$

is a quadratic function of c. It is minimized at $c = c^*$, where

$$c^* = -\frac{Cov(g(X), f(X))}{Var(f(X))}$$

and minimum variance is

$$Var(\hat{\theta}_{c^*}) = Var(g(X)) - \frac{[Cov(g(X), f(X))]^2}{Var(f(X))}. \tag{5.10}$$

The random variable $f(X)$ is called a *control variate* for the estimator $g(X)$. In (5.10) we see that $Var(g(X))$ is reduced by

$$\frac{[Cov(g(X), f(X))]^2}{Var(f(X))},$$

hence the percent reduction in variance is

$$100\frac{[Cov(g(X),f(X))]^2}{Var(g(X))\,Var(f(X))} = 100[Cor(g(X),f(X))]^2.$$

Thus, it is advantageous if $f(X)$ and $g(X)$ are strongly correlated. No reduction of variance is possible in case $f(X)$ and $g(Y)$ are uncorrelated.

To compute the constant c^*, we need $Cov(g(X),f(X))$ and $Var(f(X))$, but these parameters can be estimated if necessary, from a preliminary Monte Carlo experiment.

Example 5.7 (Control variate)

Apply the control variate approach to compute

$$\theta = E[e^U] = \int_0^1 e^u du,$$

where $U \sim$ Uniform(0,1). In this example, we do not need simulation because $\theta = e - 1 = 1.718282$ by integration, but this provides an example where we can verify that the control variate approach is correctly implemented. If the simple Monte Carlo approach is applied with m replicates, the variance of the estimator is $Var(g(U))/m$, where

$$Var(g(U)) = Var(e^U) = E[e^{2U}] - \theta^2 = \frac{e^2-1}{2} - (e-1)^2 \doteq 0.2420351.$$

A natural choice for a control variate is $U \sim$ Uniform(0,1). Then $E[U] = 1/2$, $Var(U) = 1/12$, and $Cov(e^U,U) = 1 - (1/2)(e-1) \doteq 0.1408591$. Hence

$$c^* = \frac{-Cov(e^U,U)}{Var(U)} = -12 + 6(e-1) \doteq -1.690309.$$

Our controlled estimator is $\hat{\theta}_{c^*} = e^U - 1.690309(U - 0.5)$. For m replicates, $mVar(\hat{\theta}_{c^*})$ is

$$\begin{aligned} Var(e^U) - \frac{[Cov(e^U,U)]^2}{Var(U)} &= \frac{e^2-1}{2} - (e-1)^2 - 12\left(1 - \frac{e-1}{2}\right) \\ &\doteq 0.2420356 - 12(0.1408591)^2 \\ &= 0.003940175. \end{aligned}$$

The percent reduction in variance using the control variate compared with the simple Monte Carlo estimate is 100(1-0.003940175/0.2429355) = 98.3781%.

Now we implement the control variate method for this problem and compute empirically the percent reduction in variance achieved in the simulation. Comparing the simple Monte Carlo estimate with the control variate approach

```
m <- 10000
a <- - 12 + 6 * (exp(1) - 1)
U <- runif(m)
T1 <- exp(U)                  #simple MC
T2 <- exp(U) + a * (U - 1/2)  #controlled
```

gives the following results

```
> mean(T1)
[1] 1.717834
> mean(T2)
[1] 1.718229
> (var(T1) - var(T2)) / var(T1)
[1] 0.9838606
```

illustrating that the percent reduction 98.3781% in variance derived above is approximately achieved in this simulation. ◇

Example 5.8 (MC integration using control variates)

Use the method of control variates to estimate

$$\int_0^1 \frac{e^{-x}}{1+x^2}dx.$$

(A version of this problem appears in [64, p. 734].) The parameter of interest is $\theta = E[g(X)]$ and $g(X) = e^{-x}/(1+x^2)$, where X is uniformly distributed on (0,1). We seek a function 'close' to $g(x)$ with known expected value, such that $g(X)$ and $f(X)$ are strongly correlated. For example, the function $f(x) = e^{-.5}(1+x^2)^{-1}$ is 'close' to $g(x)$ on (0,1) and we can compute its expectation. If U is uniformly distributed on (0,1), then

$$E[f(U)] = e^{-.5}\int_0^1 \frac{1}{1+u^2}\,du = e^{-.5}\arctan(1) = e^{-.5}\frac{\pi}{4}.$$

Setting up a preliminary simulation to obtain an estimate of the constant c^*, we also obtain an estimate of $Cor(g(U), f(U) \approxeq 0.974$.

```
f <- function(u)
    exp(-.5)/(1+u^2)

g <- function(u)
    exp(-u)/(1+u^2)

set.seed(510) #needed later
u <- runif(10000)
B <- f(u)
A <- g(u)
```

Estimates of c^* and $Cor(f(U), g(U))$ are

```
> cor(A, B)
[1] 0.9740585
a <- -cov(A,B) / var(B)    #est of c*
> a
[1] -2.436228
```

Simulation results with and without the control variate follow.

```
m <- 100000
u <- runif(m)
T1 <- g(u)
T2 <- T1 + a * (f(u) - exp(-.5)*pi/4)

> c(mean(T1), mean(T2))
[1] 0.5253543 0.5250021
> c(var(T1), var(T2))
[1] 0.060231423 0.003124814
> (var(T1) - var(T2)) / var(T1)
[1] 0.9481199
```

Here the approximate reduction in variance of $g(X)$ compared with $g(X) + \hat{c}^*(f(X)-\mu)$ is 95%. We will return to this problem to apply another approach to variance reduction, the method of importance sampling. $\diamond$

5.5.1 Antithetic variate as control variate.

The antithetic variate estimator of the previous section is actually a special case of the control variate estimator. First notice that the control variate estimator is a linear combination of unbiased estimators of θ. In general, if $\hat{\theta}_1$ and $\hat{\theta}_2$ are any two unbiased estimators of θ, then for every constant c,

$$\hat{\theta}_c = c\hat{\theta}_1 + (1-c)\hat{\theta}_2$$

is also unbiased for θ. The variance of $c\hat{\theta}_1 + (1-c)\hat{\theta}_2$ is

$$Var(\hat{\theta}_2) + c^2 Var(\hat{\theta}_1 - \hat{\theta}_2) + 2c\, Cov(\hat{\theta}_2, \hat{\theta}_1 - \hat{\theta}_2). \tag{5.11}$$

In the special case of antithetic variates in (5.7) and (5.8), $\hat{\theta}_1$ and $\hat{\theta}_2$ are identically distributed and $Cor(\hat{\theta}_1, \hat{\theta}_2) = -1$. Then $Cov(\hat{\theta}_1, \hat{\theta}_2) = -Var(\hat{\theta}_1)$, and the variance in (5.11) is

$$Var\hat{\theta}_c = 4c^2 Var(\hat{\theta}_1) - 4cVar(\hat{\theta}_1) + Var(\hat{\theta}_1) = (4c^2 - 4c + 1)Var(\hat{\theta}_1),$$

and the optimal constant is $c^* = 1/2$. The control variate estimator in this case is

$$\hat{\theta}_{c^*} = \frac{\hat{\theta}_1 + \hat{\theta}_2}{2},$$

which (for this particular choice of $\hat{\theta}_1$ and $\hat{\theta}_2$) is the antithetic variable estimator of θ.

5.5.2 Several control variates.

The idea of combining unbiased estimators of the target parameter θ to reduce variance can be extended to several control variables. In general, if $E[\hat{\theta}_i] = \theta$, $i = 1, 2, \dots k$ and $c = (c_1, \dots, c_k)$ such that $\sum_{i=1}^k c_i = 1$, then

$$\sum_{i=1}^k c_i \hat{\theta}_i$$

is also unbiased for θ. The corresponding control variate estimator is

$$\hat{\theta}_c = g(X) + \sum_{i=1}^k c_i^* (f_i(X) - \mu_i))$$

where $\mu_i = E[f_i(X)]$, $i = 1, \dots, k$, and

$$E[\hat{\theta}_c] = E[g(X)] + \sum_{i=1}^k c_i^* E[f_i(X) - \mu_i] = \theta.$$

The controlled estimate $\hat{\theta}_{\hat{c}^*}$, and estimates for the optimal constants c_i^*, can be obtained by fitting a linear regression model. The details are discussed in section 5.5.3.

5.5.3 Control variates and regression.

In this section we will discuss the duality between the control variate approach and simple linear regression. This provides more insight into how the control variate reduces the variance in Monte Carlo integration. In addition, we have a convenient method for estimating the optimal constant c^*, the target parameter, the percent reduction in variance, and the standard error of the estimator, all by fitting a simple linear regression model.

Suppose that $(X_1, Y_1), \dots, (X_n, Y_n)$ is a random sample from a bivariate distribution with mean (μ_X, μ_Y) and variances (σ_X^2, σ_Y^2). Let us compare the least squares estimators for regression of X on Y with the control variate estimator.

If there is a linear relation $X = \beta_1 Y + \beta_0 + \varepsilon$, and $E[\varepsilon] = 0$, then

$$E[X] = E[E[X|Y]] = E[\beta_0 + \beta_1 Y + \varepsilon] = \beta_0 + \beta_1 \mu_Y.$$

Here β_0 and β_1 are constant parameters and ε is a random error variable.

Let us consider the bivariate sample $(g(X_1), f(X_1)), \dots, (g(X_n), f(X_n))$. Now if $g(X)$ replaces X and $f(X)$ replaces Y, we have $g(X) = \beta_0 + \beta_1 f(X) + \varepsilon$, and

$$E[g(X)] = \beta_0 + \beta_1 E[f(X)].$$

The least squares estimator of the slope is

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (X_i - \overline{X})(Y_i - \overline{Y})}{\sum_{i=1}^n (Y_i - \overline{Y})^2} = \frac{\widehat{Cov}(X,Y)}{\widehat{Var}(Y)} = \frac{\widehat{Cov}(g(X), f(X))}{\widehat{Var}(f(X))} = -\hat{c}^*.$$

This shows that a convenient way to estimate c^* is to use the estimated slope from the fitted simple linear regression model of $g(X)$ on $f(X)$:

```
    L <- lm(gx ~ fx)
    c.star <- -L$coeff[2]
```

The least squares estimator of the intercept is $\hat{\beta}_0 = \overline{g(X)} - (-\hat{c}^*)\overline{f(X)}$, so that the predicted response at $\mu = E[f(X)]$ is

$$\begin{aligned}\hat{\beta}_0 + \hat{\beta}_1 \mu &= \overline{g(X)} + \hat{c}^*(\overline{f(X)} - \hat{c}^* \mu) \\ &= \overline{g(X)} + \hat{c}^*(\overline{f(X)} - \mu) = \hat{\theta}_{\hat{c}^*}.\end{aligned}$$

Thus, the control variate estimate $\hat{\theta}_{\hat{c}^*}$ is the predicted value of the response variable $(g(X))$ at the point $\mu = E[f(X)]$.

The estimate of the error variance in the regression of X on Y is

$$\begin{aligned}\hat{\sigma}_\varepsilon^2 &= \widehat{Var}(X - \hat{X}) = \widehat{Var}(X - (\hat{\beta}_0 + \hat{\beta}_1 Y)) \\ &= \widehat{Var}(X - \hat{\beta}_1 Y) = \widehat{Var}(X + \hat{c}^* Y),\end{aligned}$$

the residual mean squared error (MSE). The estimate of variance of the control variate estimator is

$$\begin{aligned}\widehat{Var}(\overline{g(X)} + \hat{c}^*(\overline{f(X)} - \mu)) &= \frac{\widehat{Var}(g(X) + \hat{c}^*(f(X) - \mu))}{n} \\ &= \frac{\widehat{Var}(g(X) + \hat{c}^* f(X))}{n} = \frac{\hat{\sigma}_\varepsilon^2}{n}.\end{aligned}$$

Thus, the estimated standard error of the control variate estimate is easily computed using R by applying the `summary` method to the `lm` object from the fitted regression model, for example using

```
    se.hat <- summary(L)$sigma
```

to extract the value of $\hat{\sigma}_\varepsilon = \sqrt{MSE}$.

Finally, recall that the proportion of reduction in variance for the control variate is $[Cor(g(X), f(X))]^2$. In the simple linear regression model, the coefficient of determination is same number (R^2), which is the proportion of total variation in $g(X)$ about its mean explained by $f(X)$.

Example 5.9 (Control variate and regression)

Returning to Example 5.8, let us repeat the estimation by fitting a regression model. In this problem,

$$g(x) = \int_0^1 \frac{e^{-x}}{1+x^2} dx$$

and the control variate is

$$f(x) = e^{-.5}(1+x^2)^{-1}, \qquad 0 < x < 1,$$

with $\mu = E[f(X)] = e^{-.5}\pi/4$. To estimate the constant c^*,

```
set.seed(510)
u <- runif(10000)
f <- exp(-.5)/(1+u^2)
g <- exp(-u)/(1+u^2)
c.star <-  - lm(g ~ f)$coeff[2]   # beta[1]
mu <- exp(-.5)*pi/4

> c.star
       f
-2.436228
```

We used the same random number seed as in Example 5.8 and obtained the same estimate for c^*. Now $\hat{\theta}_{\hat{c}^*}$ is the predicted response at the point $\mu = 0.4763681$, so

```
u <- runif(10000)
f <- exp(-.5)/(1+u^2)
g <- exp(-u)/(1+u^2)
L <- lm(g ~ f)
theta.hat <- sum(L$coeff * c(1, mu))  #pred. value at mu
```

The estimate $\hat{\theta}$, residual mean squared error and the proportion of reduction in variance (R-squared) agree with the estimates obtained in Example 5.8.

```
> theta.hat
[1] 0.5253113
> summary(L)$sigma^2
[1] 0.003117644
> summary(L)$r.squared
[1] 0.9484514
```

$\diamond$

In case several control variates are used, similarly one can estimate a linear model

$$X = \beta_0 + \sum_{i=1}^{k} \beta_i Y_i + \varepsilon$$

to estimate the optimal constants $c^* = (c_1^*, \dots, c_k^*)$. Then $-\hat{c}^* = (\hat{\beta}_1, \dots, \hat{\beta}_k)$ and the estimate is the predicted response $\hat{X}$ at the point $\mu = (\mu_1, \dots, \mu_k)$ (see section 5.5.2). The estimated variance of the controlled estimator is again $\hat{\sigma}_\varepsilon^2/n = MSE/n$, where n is the sample size (the number of replicates, in this case).

5.6 Importance Sampling

The average value of a function $g(x)$ over an interval (a, b) is usually defined (in calculus) by

$$\frac{1}{b-a}\int_a^b g(x)dx.$$

Here a uniform weight function is applied over the entire interval (a, b). If X is a random variable uniformly distributed on (a, b), then

$$E[g(X)] = \int_a^b g(x)\frac{1}{b-a}\,dx = \frac{1}{b-a}\int_a^b g(x)dx, \tag{5.12}$$

which is simply the average value of the function $g(x)$ over the interval (a, b) with respect to a uniform weight function. The simple Monte Carlo method generates a large number of replicates $X_1, \dots, X_m$ uniformly distributed on $[a, b]$ and estimates $\int_a^b g(x)dx$ by the sample mean

$$\frac{b-a}{m}\sum_{i=1}^m g(X_i),$$

which converges to $\int_a^b g(x)dx$ with probability 1 by the strong law of large numbers. One limitation of this method is that it does not apply to unbounded intervals. Another drawback is that it can be inefficient to draw samples uniformly across the interval if the function $g(x)$ is not very uniform.

However, once we view the integration problem as an expected value problem (5.12), it seems reasonable to consider other weight functions (other densities) than uniform. This leads us to a general method called *importance sampling.*

Suppose X is a random variable with density function $f(x)$, such that $f(x) > 0$ on the set $\{x : g(x) > 0\}$. Let Y be the random variable $g(X)/f(X)$. Then

$$\int g(x)dx = \int \frac{g(x)}{f(x)}f(x)dx = E[Y].$$

Estimate $E[Y]$ by simple Monte Carlo integration. That is, compute the average

$$\frac{1}{m}\sum_{i=1}^m Y_i = \frac{1}{m}\sum_{i=1}^m \frac{g(X_i)}{f(X_i)},$$

where the random variables $X_1, \dots, X_m$ are generated from the distribution with density $f(x)$. The density $f(x)$ is called the *importance function.*

In an importance sampling method, the variance of the estimator based on $Y = g(X)/f(X)$ is $Var(Y)/m$, so the variance of Y should be small. The variance of Y is small if Y is nearly constant, so the density $f(\cdot)$ should be 'close' to $g(x)$. Also, the variable with density $f(\cdot)$ should be reasonably easy to simulate.

In Example 5.5, random normals are generated to compute the Monte Carlo estimate of the standard normal cdf, $\Phi(2) = P(X \leq 2)$. In the naive Monte Carlo approach, estimates in the tails of the distribution are less precise. Intuitively, we might expect a more precise estimate for a given sample size if the simulated distribution is not uniform. In this case, the average must be a weighted average rather than the unweighted sample mean, to correct for this bias. This method is called *importance sampling* (see e.g. Robert and Casella [228, Sec. 3.3]). The advantage of importance sampling is that the importance sampling distribution can be chosen so that variance of the Monte Carlo estimator is reduced.

Suppose that $f(x)$ is a density supported on a set A. If $\phi(x) > 0$ on A, then the the integral

$$\theta = \int_A g(x) f(x) dx,$$

can be written

$$\theta = \int_A g(x) \frac{f(x)}{\phi(x)} \phi(x) dx.$$

If $\phi(x)$ is a density on A, then an estimator of $\theta = E_\phi[g(x)f(x)/\phi(x)]$ is

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} g(X_i) \frac{f(X_i)}{\phi(X_i)},$$

where $X_1, \ldots, X_n$ is a random sample from density $\phi(x)$. The function $\phi(\cdot)$ is called the *envelope* or the *importance sampling function.* There are many densities $\phi(x)$ that are convenient to simulate. Typically one should choose $\phi(x)$ so that $\phi(x) \approxeq |g(x)|f(x)$ on A (and $\phi(x)$ has finite variance).

Example 5.10 (Choice of the importance function)

In this example (from [64, p. 728]) several possible choices of importance functions to estimate

$$\int_0^1 \frac{e^{-x}}{1+x^2}\, dx$$

by importance sampling method are compared. The candidates for the importance functions are

$$\begin{aligned}
f_0(x) &= 1, \qquad 0 < x < 1, \\
f_1(x) &= e^{-x}, \qquad 0 < x < \infty, \\
f_2(x) &= (1+x^2)^{-1}/\pi, \qquad -\infty < x < \infty, \\
f_3(x) &= e^{-x}/(1-e^{-1}), \qquad 0 < x < 1, \\
f_4(x) &= 4(1+x^2)^{-1}/\pi, \qquad 0 < x < 1.
\end{aligned}$$

The integrand is

$$g(x) = \begin{cases} e^{-x}/(1+x^2), & \text{if } (0 < x < 1); \\ 0, & \text{otherwise.} \end{cases}$$

While all five of the possible importance functions are positive on the set $0 < x < 1$ where $g(x) > 0$, f_1 and f_2 have larger ranges and many of the simulated values will contribute zeros to the sum, which is inefficient. All of these distributions are easy to simulate; f_2 is standard Cauchy or $t(\nu = 1)$. The densities are plotted on (0,1) for comparison with $g(x)$ in Figure 5.1(a). The function that corresponds to the most nearly constant ratio $g(x)/f(x)$ appears to be f_3, which can be seen more clearly in Figure 5.1(b). From the graphs, we might prefer f_3 for the smallest variance.

```
m <- 10000
theta.hat <- se <- numeric(5)
g <- function(x) {
    exp(-x - log(1+x^2)) * (x > 0) * (x < 1)
    }

x <- runif(m)     #using f0
fg <- g(x)
theta.hat[1] <- mean(fg)
se[1] <- sd(fg)

x <- rexp(m, 1)   #using f1
fg <- g(x) / exp(-x)
theta.hat[2] <- mean(fg)
se[2] <- sd(fg)

x <- rcauchy(m)   #using f2
i <- c(which(x > 1), which(x < 0))
x[i] <- 2  #to catch overflow errors in g(x)
fg <- g(x) / dcauchy(x)
theta.hat[3] <- mean(fg)
se[3] <- sd(fg)

u <- runif(m)     #f3, inverse transform method
x <- - log(1 - u * (1 - exp(-1)))
fg <- g(x) / (exp(-x) / (1 - exp(-1)))
theta.hat[4] <- mean(fg)
se[4] <- sd(fg)

u <- runif(m)    #f4, inverse transform method
x <- tan(pi * u / 4)
fg <- g(x) / (4 / ((1 + x^2) * pi))
theta.hat[5] <- mean(fg)
se[5] <- sd(fg)
```

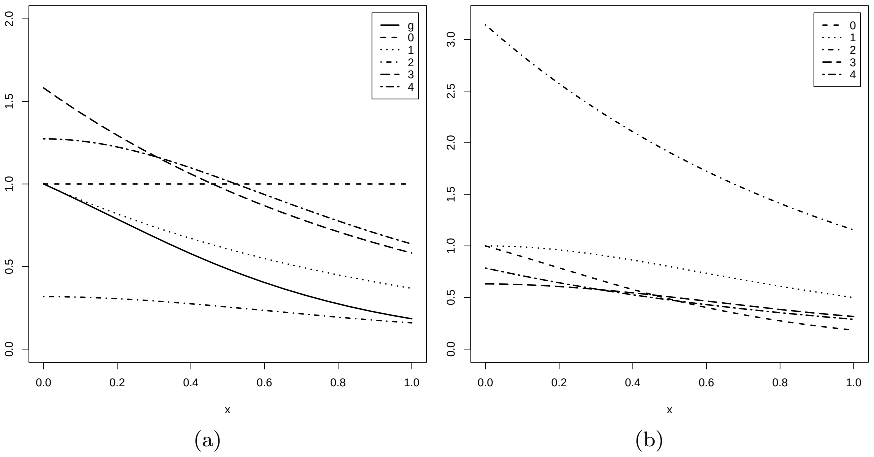


FIGURE 5.1: Importance functions in Example 5.10: $f_0, \ldots, f_4$ (lines 0:4) with $g(x)$ in (a) and the ratios $g(x)/f(x)$ in (b).

Code to display Figures 5.1(a) and 5.1(b) is given on page 152.

The estimates (labeled `theta.hat`) of $\int_0^1 g(x)dx$ and the corresponding standard errors `se` for the simulation using each of the importance functions are

```
> rbind(theta.hat, se)
                [,1]      [,2]      [,3]       [,4]      [,5]
theta.hat 0.5241140 0.5313584 0.5461507 0.52506988 0.5260492
se        0.2436559 0.4181264 0.9661300 0.09658794 0.1427685
```

so the simulation indicates that f_3 and possibly f_4 produce smallest variance among these five importance functions, while f_2 produces the highest variance. The standard Monte Carlo estimate without importance sampling has $\widehat{se} \doteq 0.244$ ($f_0 = 1$). The importance functions f_1 and f_2 do not reduce error, but f_3 and f_4 each reduce the standard error in estimating θ.

The Cauchy density f_2 is supported on the entire real line, while the integrand $g(x)$ is evaluated on (0,1). There are a very large number of zeros (about 75%) produced in the ratio $g(x)/f(x)$ in this case, and all other values far from 0, resulting in a large variance. The following summary statistics for the ratio $g(x)/f_2(x)$ confirm this.

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  0.0000  0.5173  0.0000  3.1380
```

For f_1 there is a similar inefficiency, as f_1 is supported on $(0, \infty)$, which also generates many zeros in the sum of $g(x)/f(x)$ for the values outside of (0,1). The inefficiency for f_1 is not as bad as f_2 (about 37% zeros), however, because the tail of the distribution is lighter. The following summary statistics for the ratio $g(x)/f_1(x)$ also confirm this.

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.0000  0.6891  0.5314  0.9267  1.0000 
```

◇

Example 5.10 illustrates that care must be taken to select an importance function that results in small variance of $Y = g(X)/f(X)$. The importance function should be an f that is supported on exactly the set where $g(x) > 0$, and such that the ratio $g(x)/f(x)$ is nearly constant.

Variance in Importance Sampling

If $\phi(x)$ is the importance sampling distribution (envelope), $f(x) = 1$ on A, and X has pdf $\phi(x)$ supported on A, then

$$\theta = \int_A g(x)dx = \int_A \frac{g(x)}{\phi(x)}\,\phi(x)dx = E\left[\frac{g(X)}{\phi(X)}\right].$$

If $X_1, \dots, X_n$ is a random sample from the distribution of X, the estimator is again the sample-mean

$$\hat{\theta} = \overline{g(X)} = \frac{1}{n}\sum_{i=1}^{n} \frac{g(X_i)}{\phi(X_i)}.$$

Thus, the importance sampling method is a sample-mean method, and

$$Var(\hat{\theta}) = E[\hat{\theta}^2] - (E[\hat{\theta}])^2 = \int_A \frac{g^2(x)}{\phi(x)}\,ds - \theta^2.$$

The distribution of X can be chosen to reduce the variance of the sample-mean estimator. The minimum variance

$$\left(\int_A |g(x)|dx\right)^2 - \theta^2$$

is obtained when

$$\phi(x) = \frac{|g(x)|}{\int_A |g(x)|dx}.$$

Unfortunately, the problem is to estimate $\int_A g(x)dx$, so it is unlikely that the value of $\int_A |g(x)|dx$ in the denominator of $\phi(x)$ is available. Although it may be difficult to choose $\phi(x)$ to attain minimum variance, variance may be "close to" optimal if $\phi(x)$ is chosen so that the shape of the density $\phi(x)$ is "close to" $|g(x)|$ on A.

For general $f(x)$, choose $\phi(x)$ so that $\phi(x) \approxeq |g(x)|f(x)$ on A. If the ratio of the function being integrated to the importance function is bounded, then the importance sampling estimator will have finite variance. Considering the relative computational efficiency of estimators, one should also choose $\phi(x)$ so that the cost (time) to generate the Monte Carlo replicates is small.

5.7 Stratified Sampling

Another approach to variance reduction is stratified sampling, which aims to reduce the variance of the estimator by dividing the interval into strata and estimating the integral on each of the stratum with smaller variance. Linearity of the integral operator and the strong law of large numbers imply that the sum of these estimates converges to $\int g(x)dx$ with probability 1. In stratified sampling, the number of replicates m and number of replicates m_j to be drawn from each of k strata are fixed so that $m = m_1 + \cdots + m_k$, with the goal that

$$Var(\hat{\theta}_k(m_1, \ldots, m_k)) < Var(\hat{\theta}),$$

where $\hat{\theta}_k(m_1, \ldots, m_k)$ is the stratified estimator and $\hat{\theta}$ is the standard Monte Carlo estimator based on $m = m_1 + \cdots + m_k$ replicates.

To see how this might work, let us first see a numerical example.

Example 5.11 (Example 5.10, cont.)

In Figure 5.1(a) it is clear that our integrand $g(x)$ is not constant on (0,1). Divide the interval into, say, four subintervals, and compute a Monte Carlo estimate of the integral on each subinterval using 1/4 of the total number of replicates. Then combine these four estimates to obtain the estimate of $\int_0^1 e^{-x}(1+x^2)^{-1}\ dx$. Does it appear that the variance of the estimator is reduced, compared with the variance of the standard Monte Carlo estimator?

The results are shown on the next page. Although 10 runs are not really enough to get good estimates of the standard errors, in this simulation it appears that stratification has improved variance by a factor of about 10. $\diamond$

Intuitively, there can be more reduction in variance using stratification when the means of the strata are widely dispersed, as in Example 5.11, than if the means of the strata are approximately equal. For integrands that are monotone functions, stratification similar to Example 5.11 should be an effective way to reduce variance.

```
M <- 20   #number of replicates
T2 <- numeric(4)
estimates <- matrix(0, 10, 2)

g <- function(x) {
    exp(-x - log(1+x^2)) * (x > 0) * (x < 1) }

for (i in 1:10) {
    estimates[i, 1] <- mean(g(runif(M)))
    T2[1] <- mean(g(runif(M/4, 0, .25)))
    T2[2] <- mean(g(runif(M/4, .25, .5)))
    T2[3] <- mean(g(runif(M/4, .5, .75)))
    T2[4] <- mean(g(runif(M/4, .75, 1)))
    estimates[i, 2] <- mean(T2)
}
> estimates
          [,1]      [,2]
[1,] 0.6281555 0.5191537
[2,] 0.5105975 0.5265614
[3,] 0.4625555 0.5448566
[4,] 0.4999053 0.5151490
[5,] 0.4984972 0.5249923
[6,] 0.4886690 0.5179625
[7,] 0.5151231 0.5246307
[8,] 0.5503624 0.5171037
[9,] 0.5586109 0.5463568
[10,] 0.4831167 0.5548007

> apply(estimates, 2, mean)
[1] 0.5195593 0.5291568
> apply(estimates, 2, var)
[1] 0.0023031762 0.0002012629
```

PROPOSITION 5.2 *Denote the standard Monte Carlo estimator with M replicates by $\hat{\theta}^M$, and let*

$$\hat{\theta}^S = \frac{1}{k}\sum_{j=1}^{k}\hat{\theta}_j$$

denote the stratified estimator with equal size $m = M/k$ strata. Denote the mean and variance of $g(U)$ on stratum j by θ_j and σ_j^2, respectively. Then $Var(\hat{\theta}^M) \geq Var(\hat{\theta}^S)$.

Proof. By independence of $\hat{\theta}_j$ $'s$,

$$Var(\hat{\theta}^S) = Var\left(\frac{1}{k}\sum_{j=1}^{k}\hat{\theta}_j\right) = \frac{1}{k^2}\sum_{j=1}^{k}\frac{\sigma_j^2}{m} = \frac{1}{Mk}\sum_{j=1}^{k}\sigma_j^2.$$

Now, if J is the randomly selected stratum, it is selected with uniform probability $1/k$, and applying the conditional variance formula

$$\begin{aligned}
Var(\hat{\theta}^M) = \frac{Var(g(U))}{M} &= \frac{1}{M}(Var(E[g(U|J)]) + E[Var(g(U|J)]) \\
&= \frac{1}{M}\left(Var(\theta_J) + E\left[\sigma_J^2\right]\right) \\
&= \frac{1}{M}\left(Var(\theta_J) + \frac{1}{k}\sum_{j=1}^{k}\sigma_j^2\right) \\
&= \frac{1}{M}Var(\theta_J) + Var(\hat{\theta}^S) \geq Var(\hat{\theta}^S).
\end{aligned}$$

The inequality is strict except in the case where all the strata have identical means. □

From the above inequality it is clear that the reduction in variance is larger when the means of the strata are widely dispersed.

A similar proof can be applied in the general case when the strata have unequal probabilities. See Fishman [94, Sec. 4.3] for a proof of the general case.

Example 5.12 (Examples 5.10–5.11, cont., stratified sampling)

Stratified sampling is implemented in a more general way, for the Monte Carlo estimate of $\int_0^1 e^{-x}(1+x^2)^{-1}dx$. The standard Monte Carlo estimate is also obtained for comparison.

```
M <- 10000  #number of replicates
k <- 10     #number of strata
r <- M / k  #replicates per stratum
N <- 50     #number of times to repeat the estimation
T2 <- numeric(k)
estimates <- matrix(0, N, 2)

g <- function(x) {
    exp(-x - log(1+x^2)) * (x > 0) * (x < 1)
    }

for (i in 1:N) {
    estimates[i, 1] <- mean(g(runif(M)))
    for (j in 1:k)
        T2[j] <- mean(g(runif(M/k, (j-1)/k, j/k)))
    estimates[i, 2] <- mean(T2)
}
```

The result of this simulation produces the following estimates.

```
> apply(estimates, 2, mean)
[1] 0.5251321 0.5247715
> apply(estimates, 2, var)
[1] 6.188117e-06 6.504485e-08
```

This represents a more than 98% reduction in variance. ◇

5.8 Stratified Importance Sampling

A modification to the importance sampling method of estimating $\theta = \int g(x)dx$ is stratified importance sampling.

Choose a suitable importance function f. Suppose that X is generated with density f and cdf F using the probability integral transformation. If M replicates are generated, the importance sampling estimate of θ has variance σ^2/M, where $\sigma^2 = Var(g(X)/f(X))$.

For the stratified importance sampling estimate, divide the real line into k intervals $I_j = \{x : a_{j-1} \leq x < a_j\}$ with endpoints $a_0 = -\infty$, $a_j = F^{-1}(j/k)$, $j = 1, \ldots, k-1$, and $a_k = \infty$. (The real line is divided into intervals corresponding to equal areas $1/k$ under the density $f(x)$. The interior endpoints are the percentiles or quantiles.) On each subinterval define $g_j(x) = g(x)$ if $x \in I_j$ and $g_j(x) = 0$ otherwise. We now have k parameters to estimate,

$$\theta_j = \int_{a_{j-1}}^{a_j} g_j(x)dx, \qquad j = 1, \ldots, k$$

and $\theta = \theta_1 + \cdots + \theta_k$. The conditional densities provide the importance functions on each subinterval. That is, on each subinterval I_j, the conditional density f_j of X is defined by

$$\begin{aligned} f_j(x) = f_{X|I_j}(x|I_j) &= \frac{f(x, a_{j-1} \leq x < a_j)}{P(a_{j-1} \leq x < a_j)} \\ &= \frac{f(x)}{1/k} = kf(x), \quad a_{j-1} \leq x < a_j. \end{aligned}$$

Let $\sigma_j^2 = Var(g_j(X)/f_j(X))$. For each $j = 1, \ldots, k$ we simulate an importance sample size m, compute the importance sampling estimator $\hat{\theta}_j$ of θ_j on the j^{th} subinterval, and compute $\hat{\theta}^{SI} = \frac{1}{k}\sum_{j=1}^k \hat{\theta}_j$. Then by independence of $\hat{\theta}_1, \ldots, \hat{\theta}_k$,

$$Var(\hat{\theta}^{SI}) = Var\left(\sum_{j=1}^k \hat{\theta}_j\right) = \sum_{j=1}^k \frac{\sigma_j^2}{m} = \frac{1}{m}\sum_{j=1}^k \sigma_j^2.$$

Denote the importance sampling estimator by $\hat{\theta}^I$. In order to determine whether $\hat{\theta}^{SI}$ is a better estimator of θ than $\hat{\theta}^I$, we need to check that $Var(\hat{\theta}^{SI})$ is smaller than the variance without stratification. The variance is reduced by stratification if

$$\frac{\sigma^2}{M} > \frac{1}{m}\sum_{j=1}^{k}\sigma_j^2 = \frac{k}{M}\sum_{j=1}^{k}\sigma_j^2 \Rightarrow \sigma^2 - k\sum_{j=1}^{k}\sigma_j^2 > 0.$$

Thus, we need to prove the following.

PROPOSITION 5.3 *Suppose $M = mk$ is the number of replicates for an importance sampling estimator $\hat{\theta}^I$, and $\hat{\theta}^{SI}$ is a stratified importance sampling estimator, with estimates $\hat{\theta}_j$ for θ_j on the individual strata, each with m replicates. If $Var(\hat{\theta}^I) = \sigma^2/M$ and $Var(\hat{\theta}_j) = \sigma_j^2/m$, $j = 1, \ldots, k$, then*

$$\sigma^2 - k\sum_{j=1}^{k}\sigma_j^2 \geq 0, \tag{5.13}$$

with equality if and only if $\theta_1 = \cdots = \theta_k$. Hence stratification never increases the variance, and there exists a stratification that reduces the variance except when $g(x)$ is constant.

Proof. To determine when the inequality (5.13) holds, we need to consider the relation between the random variables with densities f_j and the random variable X with density f.

Consider a two-stage experiment. First a number J is drawn at random from the integers 1 to k. After observing $J = j$, a random variable X^* is generated from the density f_j and

$$Y^* = \frac{g_j(X)}{f_j(X)} = \frac{g_j(X^*)}{kf(X^*)}.$$

To compute the variance of Y^* we apply the conditional variance formula

$$Var(Y^*) = E[Var(Y^*|J)] + Var(E[Y^*|J]). \tag{5.14}$$

Here

$$E[Var(Y^*|J)] = \sum_{j=1}^{k}\sigma_j^2 P(J = j) = \frac{1}{k}\sum_{j=1}^{k}\sigma_j^2$$

and $Var(E[Y^*|J]) = Var(\theta_J)$. Thus in (5.14) we have

$$Var(Y^*) = \frac{1}{k}\sum_{j=1}^{k}\sigma_j^2 + Var(\theta_J).$$

On the other hand,

$$k^2 Var(Y^*) = k^2 E[Var(Y^*|J)] + k^2 Var(E[Y^*|J]).$$

and

$$\sigma^2 = Var(Y) = Var(kY^*) = k^2 Var(Y^*)$$

which imply that

$$\sigma^2 = k^2 Var(Y^*) = k^2 \left(\frac{1}{k} \sum_{j=1}^{k} \sigma_j^2 + Var(\theta_J) \right) = k \sum_{j=1}^{k} \sigma_j^2 + k^2 Var(\theta_J).$$

Therefore

$$\sigma^2 - k \sum_{j=1}^{k} \sigma_j^2 = k^2 Var(\theta_J) \geq 0,$$

and equality holds if and only if $\theta_1 = \cdots = \theta_k$. □

Example 5.13 (Example 5.10, cont.)

In Example 5.10 our best result was obtained with importance function $f_3(x) = e^{-x}/(1-e^{-1})$, $0 < x < 1$. From 10000 replicates we obtained the estimate $\hat{\theta} = 0.5257801$ and an estimated standard error 0.0970314. Now divide the interval (0,1) into five subintervals, $(j/5, (j+1)/5)$, $j = 0, 1, \ldots, 4$.

Then on the j^{th} subinterval variables are generated from the density

$$\frac{5e^{-x}}{1-e^{-1}}, \qquad \frac{j-1}{5} < x < \frac{j}{5}.$$

The implementation is left as an exercise. ◇

Exercises

5.1 Compute a Monte Carlo estimate of

$$\int_0^{\pi/3} sin\, t\, dt$$

and compare your estimate with the exact value of the integral.

5.2 Refer to Example 5.3. Compute a Monte Carlo estimate of the standard normal cdf, by generating from the Uniform(0,x) distribution. Compare your estimates with the normal cdf function `pnorm`. Compute an estimate of the variance of your Monte Carlo estimate of $\Phi(2)$, and a 95% confidence interval for $\Phi(2)$.

5.3 Compute a Monte Carlo estimate $\hat{\theta}$ of

$$\theta = \int_0^{0.5} e^{-x}\, dx$$

by sampling from Uniform(0, 0.5), and estimate the variance of $\hat{\theta}$. Find another Monte Carlo estimator θ^* by sampling from the exponential distribution. Which of the variances (of $\hat{\theta}$ and $\hat{\theta}^*$) is smaller, and why?

5.4 Write a function to compute a Monte Carlo estimate of the Beta(3, 3) cdf, and use the function to estimate $F(x)$ for $x = 0.1, 0.2, \ldots, 0.9$. Compare the estimates with the values returned by the `pbeta` function in R.

5.5 Compute (empirically) the efficiency of the sample mean Monte Carlo method of estimation of the definite integral in Example 5.3 relative to the "hit or miss" method in Example 5.4.

5.6 In Example 5.7 the control variate approach was illustrated for Monte Carlo integration of

$$\theta = \int_0^1 e^x dx.$$

Now consider the antithetic variate approach. Compute $Cov(e^U, e^{1-U})$ and $Var(e^U + e^{1-U})$, where $U \sim$ Uniform(0,1). What is the percent reduction in variance of $\hat{\theta}$ that can be achieved using antithetic variates (compared with simple MC)?

5.7 Refer to Exercise 5.6. Use a Monte Carlo simulation to estimate θ by the antithetic variate approach and by the simple Monte Carlo method. Compute an empirical estimate of the percent reduction in variance using the antithetic variate. Compare the result with the theoretical value from Exercise 5.6.

5.8 Let $U \sim$ Uniform(0,1), $X = aU$, and $X' = a(1-U)$, where a is a constant. Show that $\rho(X, X') = -1$. Is $\rho(X, X') = -1$ if U is a symmetric beta random variable?

5.9 The Rayleigh density [156, (18.76)] is

$$f(x) = \frac{x}{\sigma^2} e^{-x^2/(2\sigma^2)}, \qquad x \geq 0,\, \sigma > 0.$$

Implement a function to generate samples from a Rayleigh(σ) distribution, using antithetic variables. What is the percent reduction in variance of $\frac{X+X'}{2}$ compared with $\frac{X_1+X_2}{2}$ for independent X_1, X_2?

5.10 Use Monte Carlo integration with antithetic variables to estimate

$$\int_0^1 \frac{e^{-x}}{1+x^2} dx,$$

and find the approximate reduction in variance as a percentage of the variance without variance reduction.

5.11 If $\hat{\theta}_1$ and $\hat{\theta}_2$ are unbiased estimators of θ, and $\hat{\theta}_1$ and $\hat{\theta}_2$ are antithetic, we derived that $c^* = 1/2$ is the optimal constant that minimizes the variance of $\hat{\theta}_c = c\hat{\theta}_2 + (1-c)\hat{\theta}_2$. Derive c^* for the general case. That is, if $\hat{\theta}_1$ and $\hat{\theta}_2$ are any two unbiased estimators of θ, find the value c^* that minimizes the variance of the estimator $\hat{\theta}_c = c\hat{\theta}_2 + (1-c)\hat{\theta}_2$ in equation (5.11). (c^* will be a function of the variances and the covariance of the estimators.)

5.12 Let $\hat{\theta}^{IS}_f$ be an importance sampling estimator of $\theta = \int g(x)dx$, where the importance function f is a density. Prove that if $g(x)/f(x)$ is bounded, then the variance of the importance sampling estimator $\hat{\theta}^{IS}_f$ is finite.

5.13 Find two importance functions f_1 and f_2 that are supported on $(1,\infty)$ and are 'close' to

$$g(x) = \frac{x^2}{\sqrt{2\pi}}\, e^{-x^2/2}, \qquad x > 1.$$

Which of your two importance functions should produce the smaller variance in estimating

$$\int_1^\infty \frac{x^2}{\sqrt{2\pi}}\, e^{-x^2/2}\, dx$$

by importance sampling? Explain.

5.14 Obtain a Monte Carlo estimate of

$$\int_1^\infty \frac{x^2}{\sqrt{2\pi}}\, e^{-x^2/2}\, dx$$

by importance sampling.

5.15 Obtain the stratified importance sampling estimate in Example 5.13 and compare it with the result of Example 5.10.

R Code

Code to display the plot of importance functions in Figures 5.1(a) and 5.1(b) on page 142.

```
x <- seq(0, 1, .01)
w <- 2
f1 <- exp(-x)
f2 <- (1 / pi) / (1 + x^2)
f3 <- exp(-x) / (1 - exp(-1))
f4 <- 4 / ((1 + x^2) * pi)
g <- exp(-x) / (1 + x^2)

#figure (a)
plot(x, g, type = "l", main = "", ylab = "",
     ylim = c(0,2), lwd = w)
lines(x, g/g, lty = 2, lwd = w)
lines(x, f1, lty = 3, lwd = w)
lines(x, f2, lty = 4, lwd = w)
lines(x, f3, lty = 5, lwd = w)
lines(x, f4, lty = 6, lwd = w)
legend("topright", legend = c("g", 0:4),
       lty = 1:6, lwd = w, inset = 0.02)

#figure (b)
plot(x, g, type = "l", main = "", ylab = "",
    ylim = c(0,3.2), lwd = w, lty = 2)
lines(x, g/f1, lty = 3, lwd = w)
lines(x, g/f2, lty = 4, lwd = w)
lines(x, g/f3, lty = 5, lwd = w)
lines(x, g/f4, lty = 6, lwd = w)
legend("topright", legend = c(0:4),
       lty = 2:6, lwd = w, inset = 0.02)
```

Chapter 6

Monte Carlo Methods in Inference

6.1 Introduction

Monte Carlo methods encompass a vast set of computational tools in modern applied statistics. Monte Carlo integration was introduced in Chapter 5. Monte Carlo methods may refer to any method in statistical inference or numerical analysis where simulation is used. However, in this chapter only a subset of these methods are discussed. This chapter introduces some of the Monte Carlo methods for statistical inference. Monte Carlo methods can be applied to estimate parameters of the sampling distribution of a statistic, mean squared error (MSE), percentiles, or other quantities of interest. Monte Carlo studies can be designed to assess the coverage probability for confidence intervals, to find an empirical Type I error rate of a test procedure, to estimate the power of a test, and to compare the performance of different procedures for a given problem.

In statistical inference there is uncertainty in an estimate. The methods covered in this chapter use repeated sampling from a given probability model, sometimes called *parametric* bootstrap, to investigate this uncertainty. If we can simulate the stochastic process that generated our data, repeatedly drawing samples under identical conditions, then ultimately we hope to have a close replica of the process itself reflected in the samples. Other Monte Carlo methods, such as (nonparametric) bootstrap, are based on resampling from an observed sample. Resampling methods are covered in Chapters 7 and 8. Monte Carlo integration and Markov Chain Monte Carlo methods are covered in Chapters 5 and 9. Methods for generating random variates from specified probability distributions are covered in Chapter 3. See the references in Section 5.1 on some of the early history of Monte Carlo methods, and for general reference see e.g. [63, 84, 228].

6.2 Monte Carlo Methods for Estimation

Suppose $X_1, \ldots, X_n$ is a random sample from the distribution of X. An estimator $\hat{\theta}$ for a parameter θ is an n variate function

$$\hat{\theta} = \hat{\theta}(X_1, \ldots, X_n)$$

of the sample. Functions of the estimator $\hat{\theta}$ are therefore n-variate functions of the data, also. For simplicity, let $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$, and let $x^{(1)}, x^{(2)}, \ldots$ denote a sequence of independent random samples generated from the distribution of X. Random variates from the sampling distribution of $\hat{\theta}$ can be generated by repeatedly drawing independent random samples $x^{(j)}$ and computing $\hat{\theta}^{(j)} = \hat{\theta}(x_1^{(j)}, \ldots, x_n^{(j)})$ for each sample.

6.2.1 Monte Carlo estimation and standard error

Example 6.1 (Basic Monte Carlo estimation)

Suppose that X_1, X_2 are iid from a standard normal distribution. Estimate the mean difference $E|X_1 - X_2|$.

To obtain a Monte Carlo estimate of $\theta = E[g(X_1, X_2)] = E|X_1 - X_2|$ based on m replicates, generate random samples $x^{(j)} = (x_1^{(j)}, x_2^{(j)})$ of size 2 from the standard normal distribution, $j = 1, \ldots, m$. Then compute the replicates $\hat{\theta}^{(j)} = g_j(x_1, x_2) = |x_1^{(j)} - x_2^{(j)}|$, $j = 1, \ldots, m$, and the mean of the replicates

$$\hat{\theta} = \frac{1}{m}\sum_{i=1}^{m} \hat{\theta}^{(j)} = \overline{g(X_1, X_2)} = \frac{1}{m}\sum_{i=1}^{m} |x_1^{(j)} - x_2^{(j)}|.$$

This is easy to implement, as shown below.

```
m <- 1000
g <- numeric(m)
for (i in 1:m) {
    x <- rnorm(2)
    g[i] <- abs(x[1] - x[2])
}
est <- mean(g)
```

One run produces the following estimate.

```
> est
[1] 1.128402
```

One can derive by integration that $E|X_1 - X_2| = 2/\sqrt{\pi} \doteq 1.128379$ and $Var(|X_1 - X_2|) = 2 - 4/\pi$. In this example the standard error of the estimate is $\sqrt{(2 - 4/\pi)/m} \doteq 0.02695850$. ◇

Estimating the standard error of the mean

The standard error of a mean $\overline{X}$ of a sample size n is $\sqrt{Var(X)/n}$. When the distribution of X is unknown we can substitute for F the empirical distribution F_n of the sample $x_1, \dots, x_n$. The "plug-in" estimate of the variance of X is

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

Note that $\widehat{Var}(x)$ is the population variance of the finite pseudo population $\{x_1, \dots, x_n\}$ with cdf F_n. The corresponding estimate of the standard error of $\bar{x}$ is

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

Using the unbiased estimator of $Var(X)$ we have

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

In a Monte Carlo experiment, the sample size is large and the two estimates of standard error are approximately equal.

In Example 6.1 the sample size is m (the number of replicates of $\hat{\theta}$), and the estimate of standard error of $\hat{\theta}$ is

```
> sqrt(sum((g - mean(g))^2)) / m
[1] 0.02708121
```

In Example 6.1 we have the exact value $se(\hat{\theta}) = \sqrt{(2 - 4/\pi)/m} \doteq 0.02695850$ for comparison.

6.2.2 Estimation of MSE

Monte Carlo methods can be applied to estimate the MSE of an estimator. Recall that the MSE of an estimator $\hat{\theta}$ for a parameter θ is defined by $\text{MSE}(\hat{\theta}) = E[(\hat{\theta} - \theta)^2]$. If m (pseudo) random samples $x^{(1)}, \dots, x^{(m)}$ are generated from the distribution of X, then a Monte Carlo estimate of the MSE of $\hat{\theta} = \hat{\theta}(x_1, \dots, x_n)$ is

$$\widehat{MSE} = \frac{1}{m}\sum_{j=1}^{m}(\hat{\theta}^{(j)} - \theta)^2,$$

where $\hat{\theta}^{(j)} = \hat{\theta}(x^{(j)}) = \hat{\theta}(x_1^{(j)}, \dots, x_n^{(j)})$.

Example 6.2 (Estimating the MSE of a trimmed mean)

A trimmed mean is sometimes applied to estimate the center of a continuous symmetric distribution that is not necessarily normal. In this example, we compute an estimate of the MSE of a trimmed mean. Suppose that $X_1, \dots, X_n$ is a random sample and $X_{(1)}, \dots, X_{(n)}$ is the corresponding ordered sample. The trimmed sample mean is computed by averaging all but the largest and smallest sample observations. More generally, the k^{th} level trimmed sample mean is defined by

$$\overline{X}_{[-k]} = \frac{1}{n-2k} \sum_{i=k+1}^{n-k} X_{(i)}.$$

Obtain a Monte Carlo estimate of the MSE($\overline{X}_{[-1]}$) of the first level trimmed mean assuming that the sampled distribution is standard normal.

In this example, the center of the distribution is 0 and the target parameter is $\theta = E[\overline{X}] = E[\overline{X}_{[-1]}] = 0$. We will denote the first level trimmed sample mean by T. A Monte Carlo estimate of MSE(T) based on m replicates can be obtained as follows.

1. Generate the replicates $T^{(j)}$, $j = 1 \dots, m$ by repeating:
   (a) Generate $x_1^{(j)}, \dots, x_n^{(j)}$, iid from the distribution of X.
   (b) Sort $x_1^{(j)}, \dots, x_n^{(j)}$ in increasing order, to obtain $x_{(1)}^{(j)} \leq \cdots \leq x_{(n)}^{(j)}$.
   (c) Compute $T^{(j)} = \frac{1}{n-2} \sum_{i=2}^{n-1} x_{(i)}^{(j)}$.
2. Compute $\widehat{MSE}(T) = \frac{1}{m} \sum_{j=1}^{m} (T^{(j)} - \theta)^2 = \frac{1}{m} \sum_{j=1}^{m} (T^{(j)})^2$.

Then $T^{(1)}, \dots, T^{(m)}$ are independent and identically distributed according to the sampling distribution of the level-1 trimmed mean for a standard normal distribution, and we are computing the sample mean estimate $\widehat{MSE}(T)$ of MSE(T). This procedure can be implemented by writing a `for` loop as shown below (`replicate` can replace the loop; see R note 6.1 on page 161).

```
n <- 20
m <- 1000
tmean <- numeric(m)
for (i in 1:m) {
    x <- sort(rnorm(n))
    tmean[i] <- sum(x[2:(n-1)]) / (n-2)
    }
mse <- mean(tmean^2)

> mse
[1] 0.05176437
> sqrt(sum((tmean - mean(tmean))^2)) / m    #se
[1] 0.007193428
```

The estimate of MSE for the trimmed mean in this run is approximately 0.052 ($\widehat{se} \doteq 0.007$). For comparison, the MSE of the sample mean $\overline{X}$ is $Var(X)/n$, which is $1/20 = 0.05$ in this example. Note that the median is actually a trimmed mean; it trims all but one or two of the observations. The simulation is repeated for the median below.

```
n <- 20
m <- 1000
tmean <- numeric(m)
for (i in 1:m) {
    x <- sort(rnorm(n))
    tmean[i] <- median(x)
    }
mse <- mean(tmean^2)

> mse
[1] 0.07483438
> sqrt(sum((tmean - mean(tmean))^2)) / m    #se
[1] 0.008649554
```

The estimate of MSE for the sample median is approximately 0.075 and $\widehat{se}(\widehat{MSE}) \doteq 0.0086$. ◇

Example 6.3 (MSE of a trimmed mean, cont.)

Compare the MSE of level-k trimmed means for the standard normal and a "contaminated" normal distribution. The contaminated normal distribution in this example is a mixture

$$pN(0, \sigma^2 = 1) + (1-p)N(0, \sigma^2 = 100).$$

The target parameter is the mean, $\theta = 0$. (This example is from [64, 9.7].)

Write a function to estimate $\text{MSE}(\overline{X}_{[-k]})$ for different k and p. To generate the contaminated normal samples, first randomly select σ according to the probability distribution $P(\sigma = 1) = p$; $P(\sigma = 10) = 1 - p$. Note that the normal generator `rnorm` can accept a vector of parameters for standard deviation. After generating the n values for σ, pass this vector as the `sd` argument to `rnorm` (see e.g. Example 3.12 and Example 3.13).

```
n <- 20
K <- n/2 - 1
m <- 1000
mse <- matrix(0, n/2, 6)
```

```
    trimmed.mse <- function(n, m, k, p) {
        #MC est of mse for k-level trimmed mean of
        #contaminated normal pN(0,1) + (1-p)N(0,100)
        tmean <- numeric(m)
        for (i in 1:m) {
            sigma <- sample(c(1, 10), size = n,
                replace = TRUE, prob = c(p, 1-p))
            x <- sort(rnorm(n, 0, sigma))
            tmean[i] <- sum(x[(k+1):(n-k)]) / (n-2*k)
            }
        mse.est <- mean(tmean^2)
        se.mse <- sqrt(mean((tmean-mean(tmean))^2)) / sqrt(m)
        return(c(mse.est, se.mse))
    }


   for (k in 0:K) {
       mse[k+1, 1:2] <- trimmed.mse(n=n, m=m, k=k, p=1.0)
       mse[k+1, 3:4] <- trimmed.mse(n=n, m=m, k=k, p=.95)
       mse[k+1, 5:6] <- trimmed.mse(n=n, m=m, k=k, p=.9)
   }
```

The results of the simulation are shown in Table 6.1. The results in the table are n times the estimates. This comparison suggests that a robust estimator of the mean can lead to reduced MSE for contaminated normal samples. $\diamond$

TABLE 6.1: Estimates of Mean Squared Error for the k^{th} Level Trimmed Mean in Example 6.3 ($n = 20$)

	Normal		$p = 0.95$		$p = 0.90$	
k	$n\widehat{MSE}$	$n\,\widehat{se}$	$n\widehat{MSE}$	$n\,\widehat{se}$	$n\widehat{MSE}$	$n\,\widehat{se}$
0	0.976	0.140	6.229	0.140	11.485	0.140
1	1.019	0.143	1.954	0.143	4.126	0.143
2	1.009	0.142	1.304	0.142	1.956	0.142
3	1.081	0.147	1.168	0.147	1.578	0.147
4	1.048	0.145	1.280	0.145	1.453	0.145
5	1.103	0.149	1.395	0.149	1.423	0.149
6	1.316	0.162	1.349	0.162	1.574	0.162
7	1.377	0.166	1.503	0.166	1.734	0.166
8	1.382	0.166	1.525	0.166	1.694	0.166
9	1.491	0.172	1.646	0.172	1.843	0.172

6.2.3 Estimating a confidence level

One type of problem that arises frequently in statistical applications is the need to evaluate the cdf of the sampling distribution of a statistic, when the density function of the statistic is unknown or intractable. For example, many commonly used estimation procedures are derived under the assumption that the sampled population is normally distributed. In practice, it is often the case that the population is non-normal and in such cases, the true distribution of the estimator may be unknown or intractable. The following examples illustrate a Monte Carlo method to assess the confidence level in an estimation procedure.

If (U, V) is a confidence interval estimate for an unknown parameter θ, then U and V are statistics with distributions that depend on the distribution F_X of the sampled population X. The confidence level is the probability that the interval (U, V) covers the true value of the parameter θ. Evaluating the confidence level is therefore an integration problem.

Note that the sample-mean Monte Carlo approaches to evaluating an integral $\int g(x)dx$ do not require that the function $g(x)$ is specified. It is only necessary that the sample from the distribution $g(X)$ can be generated. It is often the case in statistical applications, that $g(x)$ is in fact not specified, but the variable $g(X)$ is easily generated.

Consider the confidence interval estimation procedure for variance. It is well known that this procedure is sensitive to mild departures from normality. We use Monte Carlo methods to estimate the true confidence level when the normal theory confidence interval for variance is applied to non-normal data. The classical procedure based on the assumption of normality is outlined first.

Example 6.4 (Confidence interval for variance)

If $X_1, \dots, X_n$ is a random sample from a Normal(μ, σ^2) distribution, $n \geq 2$, and S^2 is the sample variance, then

$$V = \frac{(n-1)S^2}{\sigma^2} \sim \chi^2(n-1). \tag{6.1}$$

A one side $100(1-\alpha)\%$ confidence interval is given by $(0, (n-1)S^2/\chi^2_\alpha)$, where χ^2_α is the α-quantile of the $\chi^2(n-1)$ distribution. If the sampled population is normal with variance σ^2, then the probability that the confidence interval contains σ^2 is $1-\alpha$.

The calculation of the 95% upper confidence limit (UCL) for a random sample size $n = 20$ from a Normal$(0, \sigma^2 = 4)$ distribution is shown below.

```
n <- 20
alpha <- .05
x <- rnorm(n, mean=0, sd=2)
UCL <- (n-1) * var(x) / qchisq(alpha, df=n-1)
```

Several runs produce the upper confidence limits UCL = 6.628, UCL = 7.348, UCL = 9.621, etc. All of these intervals contain $\sigma^2 = 4$. In this example, the sampled population is normal with $\sigma^2 = 4$, so the confidence level is exactly

$$P\left(\frac{19S^2}{\chi^2_{.05}(19)} > 4\right) = P\left(\frac{(n-1)S^2}{\sigma^2} > \chi^2_{.05}(n-1)\right) = 0.95.$$

If the sampling and estimation is repeated a large number of times, approximately 95% of the intervals based on (6.1) should contain σ^2, assuming that the sampled population is normal with variance σ^2. $\diamond$

Empirical confidence level is an estimate of the confidence level obtained by simulation. For the simulation experiment, repeat the steps above a large number of times, and compute the proportion of intervals that contain the target parameter.

Monte Carlo experiment to estimate a confidence level

Suppose that $X \sim F_X$ is the random variable of interest and that θ is the target parameter to be estimated.

1. For each replicate, indexed $j = 1, \dots, m$:
   (a) Generate the j^{th} random sample, $X_1^{(j)}, \dots, X_n^{(j)}$.
   (b) Compute the confidence interval C_j for the j^{th} sample.
   (c) Compute $y_j = I(\theta \in C_j)$ for the j^{th} sample.
2. Compute the empirical confidence level $\bar{y} = \frac{1}{m}\sum_{j=1}^{m} y_j$.

The estimator $\bar{y}$ is a sample proportion estimating the true confidence level $1 - \alpha^*$, so $Var(\bar{y}) = (1-\alpha^*)\alpha^*/m$ and an estimate of standard error is $\widehat{se}(\bar{y}) = \sqrt{(1-\bar{y})\bar{y}/m}$.

Example 6.5 (MC estimate of confidence level)

Refer to Example 6.4. In this example we have $\mu = 0$, $\sigma = 2$, $n = 20$, $m = 1000$ replicates, and $\alpha = 0.05$. The sample proportion of intervals that contain $\sigma^2 = 4$ is a Monte Carlo estimate of the true confidence level. This type of simulation can be conveniently implemented by using the `replicate` function.

```
n <- 20
alpha <- .05
UCL <- replicate(1000, expr = {
    x <- rnorm(n, mean = 0, sd = 2)
    (n-1) * var(x) / qchisq(alpha, df = n-1)
    } )
```

```
#count the number of intervals that contain sigma^2=4
sum(UCL > 4)
#or compute the mean to get the confidence level
> mean(UCL > 4)
[1] 0.956
```

The result is that 956 intervals satisfied `(UCL > 4)`, so the empirical confidence level is 95.6% in this experiment. The result will vary but should be close to the theoretical value, 95%. The standard error of the estimate is $(0.95(1-0.95)/1000)^{1/2} \doteq 0.00689$. ◇

R note 6.1 *Notice that in the* `replicate` *function, the lines to be repeatedly executed are enclosed in braces { }. Alternately, the expression argument (*`expr`*) can be a function call:*

```
calcCI <- function(n, alpha) {
    y <- rnorm(n, mean = 0, sd = 2)
    return((n-1) * var(y) / qchisq(alpha, df = n-1))
}

UCL <- replicate(1000, expr = calcCI(n = 20, alpha = .05))
```

The interval estimation procedure based on (6.1) for estimating variance is sensitive to departures from normality, so the true confidence level may be different than the stated confidence level when data are non-normal. The true confidence level depends on the cdf of the statistic S^2. The confidence level is the probability that the interval $(0, (n-1)S^2/\chi^2_\alpha)$ contains the true value of the parameter σ^2, which is

$$P\left(\frac{(n-1)S^2}{\chi^2{}_\alpha} > \sigma^2\right) = P\left(S^2 > \frac{\sigma^2\chi^2_\alpha}{n-1}\right) = 1 - G\left(\frac{\sigma^2\chi^2_\alpha}{n-1}\right),$$

where $G(\cdot)$ is the cdf of S^2. If the sampled population is non-normal, we have the problem of estimating the cdf

$$G(t) = P(S^2 \le c_\alpha) = \int_0^{c_\alpha} g(x)dx,$$

where $g(x)$ is the (unknown) density of S^2 and $c_\alpha = \sigma^2\chi^2_\alpha/(n-1)$. An approximate solution can be computed empirically using Monte Carlo integration to estimate $G(c_\alpha)$. The estimate of $G(t) = P(S^2 \le t) = \int_0^t g(x)dx$, is computed by Monte Carlo integration. It is not necessary to have an explicit formula for $g(x)$, provided that we can sample from the distribution of $g(X)$.

Example 6.6 (Empirical confidence level)

In Example 6.4, what happens if the sampled population is non-normal? For example, suppose that the sampled population is $\chi^2(2)$, which has variance 4, but is distinctly non-normal. We repeat the simulation, replacing the N(0,4) samples with $\chi^2(2)$ samples.

```
n <- 20
alpha <- .05
UCL <- replicate(1000, expr = {
    x <- rchisq(n, df = 2)
    (n-1) * var(x) / qchisq(alpha, df = n-1)
    } )
> sum(UCL > 4)
[1] 773
> mean(UCL > 4)
[1] 0.773
```

In this experiment, only 773 or 77.3% of the intervals contained the population variance, which is far from the 95% coverage under normality. ◇

Remark 6.1 *The problems in Examples 6.1– 6.6 are parametric in the sense that the distribution of the sampled population is specified. The Monte Carlo approach here is sometimes called* parametric bootstrap. *The* ordinary bootstrap *discussed in Chapter 7 is a different procedure. In "parametric" bootstrap, the pseudo random samples are generated from a given probability distribution. In the "ordinary" bootstrap, the samples are generated by resampling from an observed sample. Bootstrap methods in this book refer to* resampling *methods.*

Monte Carlo methods for estimation, including several types of bootstrap confidence interval estimates, are covered in Chapter 7. Bootstrap and jackknife methods for estimating the bias and standard error of an estimate are also covered in Chapter 7. The remainder of this chapter focuses on hypothesis tests, which are also covered in Chapter 8.

6.3 Monte Carlo Methods for Hypothesis Tests

Suppose that we wish to test a hypothesis concerning a parameter θ that lies in a parameter space Θ. The hypotheses of interest are

$$H_0 : \theta \in \Theta_0 \quad \text{vs} \quad H_1 : \theta \in \Theta_1$$

where Θ_0 and Θ_1 partition the parameter space Θ.

Two types of error can occur in statistical hypothesis testing. A Type I error occurs if the null hypothesis is rejected when in fact the null hypothesis is true. A Type II error occurs if the null hypothesis is not rejected when in fact the null hypothesis is false.

The *significance level* of a test is denoted by α, and α is an upper bound on the probability of Type I error. The probability of rejecting the null hypothesis depends on the true value of θ. For a given test procedure, let $\pi(\theta)$ denote the probability of rejecting H_0. Then

$$\alpha = \sup_{\theta \in \Theta_0} \pi(\theta).$$

The probability of Type I error is the conditional probability that the null hypothesis is rejected given that H_0 is true. Thus, if the test procedure is replicated a large number of times under the conditions of the null hypothesis, the observed Type I error rate should be at most (approximately) α.

If T is the test statistic and T^* is the observed value of the test statistic, then T^* is *significant* if the test decision based on T^* is to reject H_0. The *significance probability* or p-value is the smallest possible value of α such that the observed test statistic would be significant.

6.3.1 Empirical Type I error rate

An empirical Type I error rate can be computed by a Monte Carlo experiment. The test procedure is replicated a large number of times under the conditions of the null hypothesis. The empirical Type I error rate for the Monte Carlo experiment is the sample proportion of significant test statistics among the replicates.

Monte Carlo experiment to assess Type I error rate:

1. For each replicate, indexed by $j = 1, \dots, m$:
   (a) Generate the j^{th} random sample $x_1^{(j)}, \dots, x_n^{(j)}$ from the null distribution.
   (b) Compute the test statistic T_j from the j^{th} sample.
   (c) Record the test decision $I_j = 1$ if H_0 is rejected at significance level α and otherwise $I_j = 0$.
2. Compute the proportion of significant tests $\frac{1}{m}\sum_{j=1}^{m} I_j$. This proportion is the observed Type I error rate.

For the Monte Carlo experiment above, the parameter estimated is a probability and the estimate, the observed Type I error rate, is a sample proportion. If we denote the observed Type I error rate by $\hat{p}$, then an estimate of $se(\hat{p})$ is

$$\widehat{se}(\hat{p}) = \sqrt{\frac{\hat{p}(1-\hat{p})}{m}} \leq \frac{0.5}{\sqrt{m}}.$$

The procedure is illustrated below with a simple example.

Example 6.7 (Empirical Type I error rate)

Suppose that $X_1, \dots, X_{20}$ is a random sample from a $N(\mu, \sigma^2)$ distribution. Test $H_0 : \mu = 500$ $H_1 : \mu > 500$ at $\alpha = 0.05$. Under the null hypothesis,

$$T^* = \frac{\overline{X} - 500}{S/\sqrt{20}} \sim t(19),$$

where $t(19)$ denotes the Student t distribution with 19 degrees of freedom. Large values of T^* support the alternative hypothesis. Use a Monte Carlo method to compute an empirical probability of Type I error when $\sigma = 100$, and check that it is approximately equal to $\alpha = 0.05$.

The simulation below illustrates the procedure for the case $\sigma = 100$. The t-test is implemented by `t.test` in R, and we are basing the test decisions on the reported p-values returned by `t.test`.

```
n <- 20
alpha <- .05
mu0 <- 500
sigma <- 100

m <- 10000          #number of replicates
p <- numeric(m)     #storage for p-values
for (j in 1:m) {
    x <- rnorm(n, mu0, sigma)
    ttest <- t.test(x, alternative = "greater", mu = mu0)
    p[j] <- ttest$p.value
    }

p.hat <- mean(p < alpha)
se.hat <- sqrt(p.hat * (1 - p.hat) / m)
print(c(p.hat, se.hat))

[1] 0.050600000 0.002191795
```

The observed Type I error rate in this simulation is 0.0506, and the standard error of the estimate is approximately $\sqrt{0.05 \times 0.95/m} \doteq 0.0022$. Estimates of Type I error probability will vary, but should be close to the nominal rate $\alpha = 0.05$ because all samples were generated under the null hypothesis from the assumed model for a t-test (normal distribution). In this experiment the empirical Type I error rate differs from $\alpha = 0.05$ by less than one standard error.

Theoretically, the probability of rejecting the null hypothesis when $\mu = 500$ is exactly $\alpha = 0.05$ in this example. The simulation really only investigates empirically whether the method of computing the p-value in `t.test` (a numerical algorithm) is consistent with the theoretical value $\alpha = 0.05$. ◇

One of the simplest approaches to testing for univariate normality is the skewness test. In the following example we investigate whether a test based on the asymptotic distribution of the skewness statistic achieves the nominal significance level α under the null hypothesis of normality.

Example 6.8 (Skewness test of normality)

The skewness $\sqrt{\beta_1}$ of a random variable X is defined by

$$\sqrt{\beta_1} = \frac{E[(X - \mu_X)]^3}{\sigma_X^3},$$

where $\mu_X = E[X]$ and $\sigma_X^2 = Var(X)$. (The notation $\sqrt{\beta_1}$ is the classical notation for the signed skewness coefficient.) A distribution is symmetric if $\sqrt{\beta_1} = 0$, positively skewed if $\sqrt{\beta_1} > 0$, and negatively skewed if $\sqrt{\beta_1} < 0$. The sample coefficient of skewness is denoted by $\sqrt{b_1}$, and defined as

$$\sqrt{b_1} = \frac{\frac{1}{n}\sum_{i=1}^n (X_i - \overline{X})^3}{(\frac{1}{n}\sum_{i=1}^n (X_i - \overline{X})^2)^{3/2}}. \tag{6.2}$$

(Note that $\sqrt{b_1}$ is classical notation for the signed skewness statistic.) If the distribution of X is normal, then $\sqrt{b_1}$ is asymptotically normal with mean 0 and variance $6/n$ [59]. Normal distributions are symmetric, and a test for normality based on skewness rejects the hypothesis of normality for large values of $|\sqrt{b_1}|$. The hypotheses are

$$H_0 : \sqrt{\beta_1} = 0; \qquad H_1 : \sqrt{\beta_1} \neq 0,$$

where the sampling distribution of the skewness statistic is derived under the assumption of normality.

However, the convergence of $\sqrt{b_1}$ to its limit distribution is rather slow and the asymptotic distribution is not a good approximation for small to moderate sample sizes.

Assess the Type I error rate for a skewness test of normality at $\alpha = 0.05$ based on the asymptotic distribution of $\sqrt{b_1}$ for sample sizes $n = 10$, 20, 30, 50, 100, and 500.

The vector of critical values `cv` for each of the sample sizes $n = 10$, 20, 30, 50, 100, and 500 are computed under the normal limit distribution and stored in `cv`.

```
n <- c(10, 20, 30, 50, 100, 500) #sample sizes
cv <- qnorm(.975, 0, sqrt(6/n))  #crit. values for each n

asymptotic critical values:
n       10     20     30     50    100    500
cv  1.5182 1.0735 0.8765 0.6790 0.4801 0.2147
```

The asymptotic distribution of $\sqrt{b_1}$ does not depend on the mean and variance of the sampled normal distribution, so the samples can be generated from the standard normal distribution. If the sample size is `n[i]` then H_0 is rejected if $|\sqrt{b_1}| >$ `cv[i]`.

First write a function to compute the sample skewness statistic.

```
sk <- function(x) {
    #computes the sample skewness coeff.
    xbar <- mean(x)
    m3 <- mean((x - xbar)^3)
    m2 <- mean((x - xbar)^2)
    return( m3 / m2^1.5 )
}
```

In the code below, the outer loop varies the sample size n and the inner loop is the simulation for the current n. In the simulation, the test decisions are saved as 1 (reject H_0) or 0 (do not reject H_0) in the vector `sktests`. When the simulation for $n = 10$ ends, the mean of `sktests` gives the sample proportion of significant tests for $n = 10$. This result is saved in `p.reject[1]`. Then the simulation is repeated for $n = 20, 30, 50, 100, 500$, and saved in `p.reject[2:6]`.

```
#n is a vector of sample sizes
#we are doing length(n) different simulations

p.reject <- numeric(length(n)) #to store sim. results
m <- 10000                     #num. repl. each sim.

for (i in 1:length(n)) {
    sktests <- numeric(m)       #test decisions
    for (j in 1:m) {
        x <- rnorm(n[i])
        #test decision is 1 (reject) or 0
        sktests[j] <- as.integer(abs(sk(x)) >= cv[i] )
        }
    p.reject[i] <- mean(sktests) #proportion rejected
}

> p.reject
[1] 0.0129 0.0272 0.0339 0.0415 0.0464 0.0539
```

The results of the simulation are the empirical estimates of Type I error rate summarized below.

```
n              10     20     30     50    100    500
estimate  0.0129 0.0272 0.0339 0.0415 0.0464 0.0539
```

With $m = 10000$ replicates the standard error of the estimate is approximately $\sqrt{0.05 \times 0.95/m} \doteq 0.0022$.

The results of the simulation suggest that the asymptotic normal approximation for the distribution of $\sqrt{b_1}$ is not adequate for sample sizes $n \leq 50$, and questionable for sample sizes as large as $n = 500$. For finite samples one should use

$$Var(\sqrt{b_1}) = \frac{6(n-2)}{(n+1)(n+3)},$$

the exact value of the variance [93] (also see [60] or [270]). Repeating the simulation with

```
    cv <- qnorm(.975, 0, sqrt(6*(n-2) / ((n+1)*(n+3))))
    > round(cv, 4)
    [1] 1.1355 0.9268 0.7943 0.6398 0.4660 0.2134
```

produces the simulation results summarized below.

```
    n               10     20     30     50    100    500
    estimate    0.0548 0.0515 0.0543 0.0514 0.0511 0.0479
```

These estimates are closer to the nominal level $\alpha = 0.05$. On skewness tests and other classical tests of normality see [58] or [270]. $\diamond$

6.3.2 Power of a Test

In a test of hypotheses H_0 vs H_1, a Type II error occurs when H_1 is true, but H_0 is not rejected. The *power* of a test is given by the *power function* $\pi : \Theta \to [0, 1]$, which is the probability $\pi(\theta)$ of rejecting H_0 given that the true value of the parameter is θ. Thus, for a given $\theta_1 \in \Theta_1$, the probability of Type II error is $1 - \pi(\theta_1)$. Ideally, we would prefer a test with low probability of error. Type I error is controlled by the choice of the significance level α. Low Type II error corresponds to high power under the alternative hypothesis. Thus, when comparing test procedures for the same hypotheses at the same significance level, we are interested in comparing the power of the tests. In general the comparison is not one problem but many; the power $\pi(\theta_1)$ of a test under the alternative hypothesis depends on the particular value of the alternative θ_1. For the t-test in Example 6.7, $\Theta_1 = (500, \infty)$. In general, however, the set Θ_1 can be more complicated.

If the power function of a test cannot be derived analytically, the power of a test against a fixed alternative $\theta_1 \in \Theta_1$ can be estimated by Monte Carlo methods. Note that the power function is defined for all $\theta \in \Theta$, but the significance level α controls $\pi(\theta) \leq \alpha$ for all $\theta \in \Theta_0$.

Monte Carlo experiment to estimate power of a test against a fixed alternative

1. Select a particular value of the parameter $\theta_1 \in \Theta$.
2. For each replicate, indexed by $j = 1, \ldots, m$:
   (a) Generate the j^{th} random sample $x_1^{(j)}, \ldots, x_n^{(j)}$ under the conditions of the alternative $\theta = \theta_1$.
   (b) Compute the test statistic T_j from the j^{th} sample.
   (c) Record the test decision: set $I_j = 1$ if H_0 is rejected at significance level α, and otherwise set $I_j = 0$.
3. Compute the proportion of significant tests $\hat{\pi}(\theta_1) = \frac{1}{m}\sum_{j=1}^{m} I_j$.

Example 6.9 (Empirical power)

Use simulation to estimate power and plot an empirical power curve for the t-test in Example 6.7. (For a numerical approach that does not involve simulation, see the remark below.)

To plot the curve, we need the empirical power for a sequence of alternatives θ along the horizontal axis. Each point corresponds to a Monte Carlo experiment. The outer `for` loop varies the points θ (`mu`) and the inner `replicate` loop (see R Note 6.1) estimates the power at the current θ.

```
n <- 20
m <- 1000
mu0 <- 500
sigma <- 100
mu <- c(seq(450, 650, 10))  #alternatives
M <- length(mu)
power <- numeric(M)
for (i in 1:M) {
    mu1 <- mu[i]
    pvalues <- replicate(m, expr = {
        #simulate under alternative mu1
        x <- rnorm(n, mean = mu1, sd = sigma)
        ttest <- t.test(x,
                 alternative = "greater", mu = mu0)
        ttest$p.value  } )
    power[i] <- mean(pvalues <= .05)
}
```

The estimated power $\hat{\pi}(\theta)$ values are now stored in the vector `power`. Next, plot the empirical power curve, adding vertical error bars at $\hat{\pi}(\theta) \pm \widehat{se}(\hat{\pi}(\theta))$ using the `errbar` function in the `Hmisc` package [132].

```
library(Hmisc)  #for errbar
plot(mu, power)
abline(v = mu0, lty = 1)
abline(h = .05, lty = 1)

#add standard errors
se <- sqrt(power * (1-power) / m)
errbar(mu, power, yplus = power+se, yminus = power-se,
    xlab = bquote(theta))
lines(mu, power, lty=3)
detach(package:Hmisc)
```

The power curve is shown in Figure 6.1. Note that the empirical power $\hat{\pi}(\theta)$ is small when θ is close to $\theta_0 = 500$, and increasing as θ moves farther away from θ_0, approaching 1 as $\theta \to \infty$. $\diamond$

Remark 6.2 *The non-central t distribution arises in power calculations for t-tests. The general non-central t with parameters (ν, δ) is defined as the distribution of $T(\nu, \delta) = (Z + \delta)/\sqrt{V/\nu}$ where $Z \sim N(0,1)$ and $V \sim \chi^2(\nu)$ are independent.*

Suppose $X_1, X_2, \dots, X_n$ is a random sample from a $N(\mu, \sigma^2)$ distribution, and the t-statistic $T = (\overline{X} - \mu_0)/(S/\sqrt{n})$ is applied to test $H_0 : \mu = \mu_0$. Under the null hypothesis, T has the central $t(n-1)$ distribution, but if $\mu \neq \mu_0$, T has the non-central t distribution with $n-1$ degrees of freedom and non-centrality parameter $\delta = (\mu - \mu_0)\sqrt{n}/\sigma$. A numerical approach to evaluating the cdf of the non-central t distribution, based on an algorithm of Lenth [175], is implemented in the R function `pt`. Also see `power.t.test`. $\diamond$

Example 6.10 (Power of the skewness test of normality)

The skewness test of normality was described in Example 6.8. In this example, we estimate by simulation the power of the skewness test of normality against a contaminated normal (normal scale mixture) alternative described in Example 6.3. The contaminated normal distribution is denoted by

$$(1-\varepsilon)N(\mu = 0, \sigma^2 = 1) + \varepsilon N(\mu = 0, \sigma^2 = 100), \qquad 0 \le \varepsilon \le 1.$$

When $\varepsilon = 0$ or $\varepsilon = 1$ the distribution is normal, but the mixture is non-normal for $0 < \varepsilon < 1$. We can estimate the power of the skewness test for a sequence of alternatives indexed by ε and plot a power curve for the power of the skewness test against *this type of alternative.* For this experiment, the significance level is $\alpha = 0.1$ and the sample size is $n = 30$. The skewness statistic `sk` is implemented in Example 6.8.

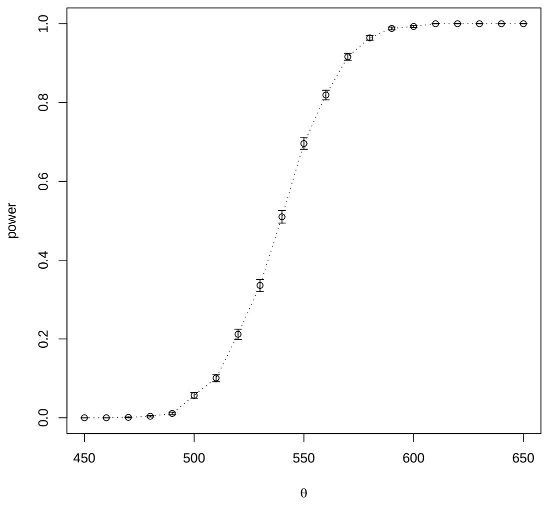


FIGURE 6.1: Empirical power $\hat{\pi}(\theta) \pm \widehat{se}(\hat{\pi}(\theta))$ for the t-test of $H_0 : \theta = 500$ vs $H_1 : \theta > 500$ in Example 6.9.

```
alpha <- .1
n <- 30
m <- 2500
epsilon <- c(seq(0, .15, .01), seq(.15, 1, .05))
N <- length(epsilon)
pwr <- numeric(N)
#critical value for the skewness test
cv <- qnorm(1-alpha/2, 0, sqrt(6*(n-2) / ((n+1)*(n+3))))

for (j in 1:N) {           #for each epsilon
    e <- epsilon[j]
    sktests <- numeric(m)
    for (i in 1:m) {       #for each replicate
        sigma <- sample(c(1, 10), replace = TRUE,
            size = n, prob = c(1-e, e))
        x <- rnorm(n, 0, sigma)
        sktests[i] <- as.integer(abs(sk(x)) >= cv)
        }
    pwr[j] <- mean(sktests)
    }
#plot power vs epsilon
plot(epsilon, pwr, type = "b",
     xlab = bquote(epsilon), ylim = c(0,1))
abline(h = .1, lty = 3)
se <- sqrt(pwr * (1-pwr) / m)  #add standard errors
lines(epsilon, pwr+se, lty = 3)
lines(epsilon, pwr-se, lty = 3)
```

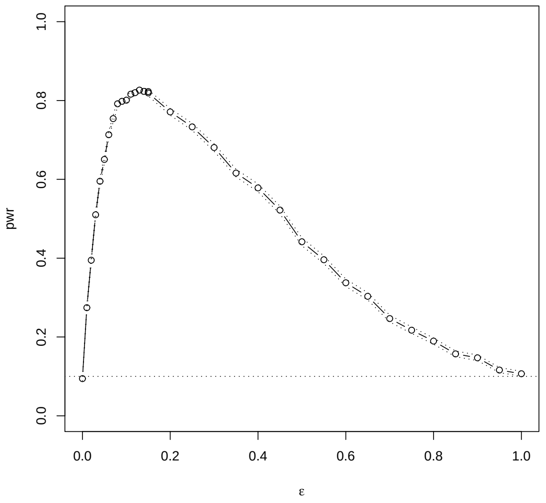


FIGURE 6.2: Empirical power $\hat{\pi}(\varepsilon) \pm \widehat{se}(\hat{\pi}(\varepsilon))$ for the skewness test of normality against ε-contaminated normal scale mixture alternative in Example 6.10.

The empirical power curve is shown in Figure 6.2. Note that the power curve crosses the horizontal line corresponding to $\alpha = 0.10$ at both endpoints, $\varepsilon = 0$ and $\varepsilon = 1$ where the alternative is normally distributed. For $0 < \varepsilon < 1$ the empirical power of the test is greater than 0.10 and highest when ε is about 0.15. $\diamond$

6.3.3 Power comparisons

Monte Carlo methods are often applied to compare the performance of different test procedures. A skewness test of normality was introduced in Example 6.8. There are many tests of normality in the literature (see [58] and [270]). In the following example three tests of univariate normality are compared.

Example 6.11 (Power comparison of tests of normality)

Compare the empirical power of the skewness test of univariate normality with the Shapiro-Wilk [248] test. Also compare the power of the *energy* test [263], which is based on distances between sample elements.

Let $\mathcal{N}$ denote the family of univariate normal distributions. Then the test hypotheses are

$$H_0 : F_X \in \mathcal{N} \qquad H_1 : F_X \notin \mathcal{N}.$$

The Shapiro-Wilk test is based on the regression of the sample order statistics on their expected values under normality, so it falls in the general category of tests based on regression and correlation. The approximate critical values of the statistic are determined by a transformation of the statistic W to normality [235, 236, 237] for sample sizes $7 \leq n \leq 2000$. The Shapiro-Wilk test is implemented by the R function `shapiro.test`.

The *energy* test is based on an energy distance between the sampled distribution and normal distribution, so large values of the statistic are significant. The *energy* test is a test of multivariate normality [263], so the test considered here is the special case $d = 1$. As a test of univariate normality, *energy* performs very much like the Anderson-Darling test [9]. The energy statistic for testing normality is

$$Q_n = n\left[\frac{2}{n}\sum_{i=1}^{n} E\|x_i - X\| - E\|X - X'\| - \frac{1}{n^2}\sum_{i,j=1}^{n}\|x_i - x_j\|\right], \tag{6.3}$$

where X, X' are iid. Large values of Q_n are significant. In the univariate case, the following computing formula is equivalent:

$$Q_n = n\left[\frac{2}{n}\sum_{i=1}^{n}(2Y_i\ \Phi(Y_i) + 2\phi(Y_i)) - \frac{2}{\sqrt{\pi}} - \frac{2}{n^2}\sum_{k=1}^{n}(2k-1-n)Y_{(k)}\right], \tag{6.4}$$

where $Y_i = \frac{X_i - \mu_X}{\sigma_X}$, $Y_{(k)}$ is the k^{th} order statistic of the standardized sample, Φ is the standard normal cdf and ϕ is the standard normal density. If the parameters are unknown, substitute the sample mean and sample standard deviation to to compute $Y_1, \ldots, Y_n$. A computing formula for the multivariate case is given in [263]. The energy test for univariate and multivariate normality is implemented in `mvnorm.etest` in the `energy` package [226].

The skewness test of normality was introduced in Examples 6.8 and 6.10. The sample skewness function `sk` is given in Example 6.8 on page 166.

For this comparison we set significance level $\alpha = 0.1$. The example below compares the power of the tests against the contaminated normal alternatives described in Example 6.3. The alternative is the normal mixture denoted by

$$(1-\varepsilon)N(\mu = 0, \sigma^2 = 1) + \varepsilon N(\mu = 0, \sigma^2 = 100), \qquad 0 \leq \varepsilon \leq 1.$$

When $\varepsilon = 0$ or $\varepsilon = 1$ the distribution is normal, and in this case the empirical Type I error rate should be controlled at approximately the nominal rate $\alpha = 0.1$. If $0 < \varepsilon < 1$ the distributions are non-normal, and we are interested in comparing the empirical power of the tests against these alternatives.

```
# initialize input and output
library(energy)
alpha <- .1
n <- 30
m <- 2500        #try smaller m for a trial run
epsilon <- .1
test1 <- test2 <- test3 <- numeric(m)

#critical value for the skewness test
cv <- qnorm(1-alpha/2, 0, sqrt(6*(n-2) / ((n+1)*(n+3))))

# estimate power
for (j in 1:m) {
    e <- epsilon
    sigma <- sample(c(1, 10), replace = TRUE,
        size = n, prob = c(1-e, e))
    x <- rnorm(n, 0, sigma)
    test1[j] <- as.integer(abs(sk(x)) >= cv)
    test2[j] <- as.integer(
                shapiro.test(x)$p.value <= alpha)
    test3[j] <- as.integer(
                mvnorm.etest(x, R=200)$p.value <= alpha)
    }
print(c(epsilon, mean(test1), mean(test2), mean(test3)))
detach(package:energy)
```

The simulation was repeated for several choices of ε and results saved in a matrix `sim`. Simulation results for $n = 30$ are summarized in Table 6.2 and in Figure 6.3. The plot is obtained as follows.

```
# plot the empirical estimates of power
plot(sim[,1], sim[,2], ylim = c(0, 1), type = "l",
    xlab = bquote(epsilon), ylab = "power")
lines(sim[,1], sim[,3], lty = 2)
lines(sim[,1], sim[,4], lty = 4)
abline(h = alpha, lty = 3)
legend("topright", 1, c("skewness", "S-W", "energy"),
    lty = c(1,2,4), inset = .02)
```

Standard error of the estimates is at most $0.5/\sqrt{m} = 0.01$. Estimates for empirical Type I error rate correspond to $\varepsilon = 0$ and $\varepsilon = 1$. All tests achieve approximately the nominal significance level $\alpha = 0.10$ within one standard error. The tests are at approximately the same significance level, so it is meaningful to compare the results for power.

The simulation results suggest that the Shapiro-Wilk and energy tests are about equally powerful against this type of alternative when $n = 30$ and $\varepsilon < 0.5$. Both have higher power than the skewness test overall and energy appears to have highest power for $0.5 \leq \varepsilon \leq 0.8$.

◇

6.4 Application: "Count Five" Test for Equal Variance

The examples in this section illustrate the Monte Carlo method for a simple two sample test of equal variance.

The two sample "Count Five" test for equality of variance introduced by McGrath and Yeh [193] counts the number of extreme points of each sample relative to the range of the other sample. Suppose the means of the two samples are equal and the sample sizes are equal. An observation in one sample is considered extreme if it is not within the range of the other sample. If either sample has five or more extreme points, the hypothesis of equal variance is rejected.

Example 6.12 (Count Five test statistic)

The computation of the test statistic is illustrated with a numerical example. Compare the side-by-side boxplots in Figure 6.4 and observe that there are some extreme points in each sample with respect to the other sample.

```
    x1 <- rnorm(20, 0, sd = 1)
    x2 <- rnorm(20, 0, sd = 1.5)
    y <- c(x1, x2)

    group <- rep(1:2, each = length(x1))
    boxplot(y ~ group, boxwex = .3, xlim = c(.5, 2.5), main = "")
    points(group, y)

    # now identify the extreme points
    > range(x1)
    [1] -2.782576  1.728505
    > range(x2)
    [1] -1.598917  3.710319
```

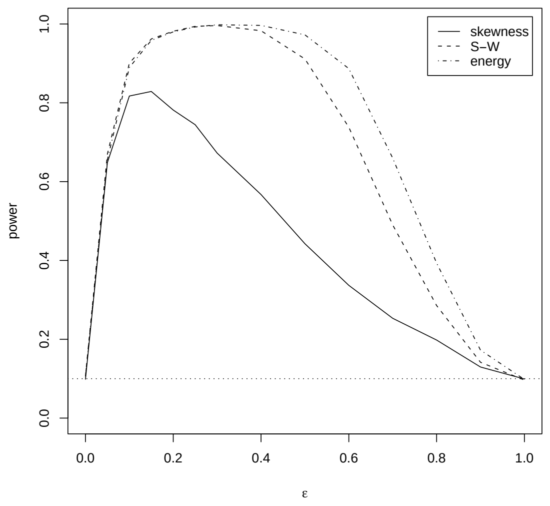


FIGURE 6.3: Empirical power of three tests of normality against a contaminated normal alternative in Example 6.11 ($n = 30$, $\alpha = 0.1$, $se \leq 0.01$)

TABLE 6.2: Empirical Power of Three Tests of Normality against a Contaminated Normal Alternative in Example 6.11 ($n = 30$, $\alpha = 0.1$, $se \leq 0.01$)

ε	skewness test	Shapiro-Wilk	energy test
0.00	0.0984	0.1076	0.1064
0.05	0.6484	0.6704	0.6560
0.10	0.8172	0.9008	0.8896
0.15	0.8236	0.9644	0.9624
0.20	0.7816	0.9816	0.9800
0.25	0.7444	0.9940	0.9924
0.30	0.6724	0.9960	0.9980
0.40	0.5672	0.9828	0.9964
0.50	0.4424	0.9112	0.9724
0.60	0.3368	0.7380	0.8868
0.70	0.2532	0.4900	0.6596
0.80	0.1980	0.2856	0.3932
0.90	0.1296	0.1416	0.1724
1.00	0.0992	0.0964	0.0980

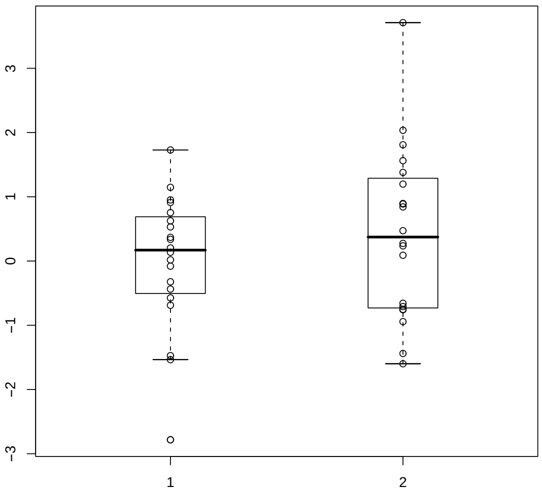


FIGURE 6.4: Boxplots showing extreme points for the Count Five statistic in Example 6.12.

```
> i <- which(x1 < min(x2))
> j <- which(x2 > max(x1))

> x1[i]
[1] -2.782576

> x2[j]
[1] 2.035521 1.809902 3.710319
```

The Count Five statistic is the maximum number of extreme points, $\max(1,3)$, so the Count Five test will not reject the hypothesis of equal variance. Note that we only need the number of extreme points, and the extreme count can be determined without reference to a boxplot as follows.

```
out1 <- sum(x1 > max(x2)) + sum(x1 < min(x2))
out2 <- sum(x2 > max(x1)) + sum(x2 < min(x1))
> max(c(out1, out2))
[1] 3
```

◇

Example 6.13 (Count Five test statistic, cont.)

Consider the case of two independent random samples from the same normal distribution. Estimate the sampling distribution of the maximum number of extreme points, and find the 0.80, 0.90, and 0.95 quantiles of the sampling distribution.

The function `maxout` below counts the maximum number of extreme points of each sample with respect to the range of the other sample. The sampling distribution of the extreme count statistic can be estimated by a Monte Carlo experiment.

```
maxout <- function(x, y) {
    X <- x - mean(x)
    Y <- y - mean(y)
    outx <- sum(X > max(Y)) + sum(X < min(Y))
    outy <- sum(Y > max(X)) + sum(Y < min(X))
    return(max(c(outx, outy)))
}

n1 <- n2 <- 20
mu1 <- mu2 <- 0
sigma1 <- sigma2 <- 1
m <- 1000

# generate samples under H0
stat <- replicate(m, expr={
    x <- rnorm(n1, mu1, sigma1)
    y <- rnorm(n2, mu2, sigma2)
    maxout(x, y)
    })
print(cumsum(table(stat)) / m)
print(quantile(stat, c(.8, .9, .95)))
```

The "Count Five" test criterion looks reasonable for normal distributions. The empirical cdf and quantiles are

```
    1     2     3     4     5     6     7     8     9    10    11
0.149 0.512 0.748 0.871 0.945 0.974 0.986 0.990 0.996 0.999 1.000

80% 90% 95%
  4   5   6
```

Notice that the `quantile` function gives 6 as the 0.95 quantile. However, if $\alpha = 0.05$ is the desired significance level, the critical value 5 appears to be the best choice. The `quantile` function is not always the best way to estimate a critical value. If `quantile` is used, compare the result to the empirical cdf. $\diamond$

The "Count Five" test criterion can be applied for independent random samples when the random variables are similarly distributed and sample sizes are equal. (Random variables X and Y are called *similarly distributed* if Y has the same distribution as $(X - a)/b$ where a and $b > 0$ are constants.) When the data are centered by their respective population means, McGrath and Yeh

[193] show that the Count Five test on the centered data has significance level at most 0.0625.

In practice, the populations means are generally unknown and each sample would be centered by subtracting its sample mean. Also, the sample sizes may be unequal.

Example 6.14 (Count Five test)

Use Monte Carlo methods to estimate the significance level of the test when each sample is centered by subtracting its sample mean. Here again we consider normal distributions. The function `count5test` returns the value 1 (reject H_0) or 0 (do not reject H_0).

```
    count5test <- function(x, y) {
        X <- x - mean(x)
        Y <- y - mean(y)
        outx <- sum(X > max(Y)) + sum(X < min(Y))
        outy <- sum(Y > max(X)) + sum(Y < min(X))
        # return 1 (reject) or 0 (do not reject H0)
        return(as.integer(max(c(outx, outy)) > 5))
    }


    n1 <- n2 <- 20
    mu1 <- mu2 <- 0
    sigma1 <-  sigma2 <- 1
    m <- 10000
    tests <- replicate(m, expr = {
        x <- rnorm(n1, mu1, sigma1)
        y <- rnorm(n2, mu2, sigma2)
        x <- x - mean(x)  #centered by sample mean
        y <- y - mean(y)
        count5test(x, y)
        } )

    alphahat <- mean(tests)
    > print(alphahat)
    [1] 0.0565
```

If the samples are centered by the population mean, we should expect an empirical Type I error rate of about 0.055, from our previous simulation to estimate the quantiles of the `maxout` statistic. In the simulation, each sample was centered by subtracting the sample mean, and the empirical Type I error rate was 0.0565 (se $\doteq$ 0.0022). ◇

Example 6.15 (Count Five test, cont.)

Repeating the previous example, we are estimating the empirical Type I error rate when sample sizes differ and the "Count Five" test criterion is applied. Each sample is centered by subtracting the sample mean.

```
n1 <- 20
n2 <- 30
mu1 <- mu2 <- 0
sigma1 <- sigma2 <- 1
m <- 10000

alphahat <- mean(replicate(m, expr={
    x <- rnorm(n1, mu1, sigma1)
    y <- rnorm(n2, mu2, sigma2)
    x <- x - mean(x)  #centered by sample mean
    y <- y - mean(y)
    count5test(x, y)
    }))

print(alphahat)
[1] 0.1064
```

The simulation result suggests that the "Count Five" criterion does not necessarily control Type I error at $\alpha \leq 0.0625$ when the sample sizes are unequal. Repeating the simulation above with $n_1 = 20$ and $n_2 = 50$, the empirical Type I error rate was 0.2934. See [193] for a method to adjust the test criterion for unequal sample sizes. ◇

Example 6.16 (Count Five, cont.)

Use Monte Carlo methods to estimate the power of the Count Five test, where the sampled distributions are $N(\mu_1 = 0, \sigma_1^2 = 1)$, $N(\mu_2 = 0, \sigma_2^2 = 1.5^2)$, and the sample sizes are $n_1 = n_2 = 20$.

```
# generate samples under H1 to estimate power
sigma1 <- 1
sigma2 <- 1.5

power <- mean(replicate(m, expr={
    x <- rnorm(20, 0, sigma1)
    y <- rnorm(20, 0, sigma2)
    count5test(x, y)
    }))
```

```
> print(power)
[1] 0.3129
```

The empirical power of the test is 0.3129 ($se \leq 0.005$) against the alternative ($\sigma_1 = 1$, $\sigma_2 = 1.5$) with $n_1 = n_2 = 20$. See [193] for power comparisons with other tests for equal variance and applications. ◇

Exercises

6.1 Estimate the MSE of the level k trimmed means for random samples of size 20 generated from a standard Cauchy distribution. (The target parameter θ is the center or median; the expected value does not exist.) Summarize the estimates of MSE in a table for $k = 1, 2, \ldots, 9$.

6.2 Plot the empirical power curve for the t-test in Example 6.9, changing the alternative hypothesis to $H_1 : \mu \neq 500$, and keeping the significance level $\alpha = 0.05$.

6.3 Plot the power curves for the t-test in Example 6.9 for sample sizes 10, 20, 30, 40, and 50, but omit the standard error bars. Plot the curves on the same graph, each in a different color or different line type, and include a legend. Comment on the relation between power and sample size.

6.4 Suppose that $X_1, \ldots, X_n$ are a random sample from a from a lognormal distribution with unknown parameters. Construct a 95% confidence interval for the parameter μ. Use a Monte Carlo method to obtain an empirical estimate of the confidence level.

6.5 Suppose a 95% symmetric t-interval is applied to estimate a mean, but the sample data are non-normal. Then the probability that the confidence interval covers the mean is not necessarily equal to 0.95. Use a Monte Carlo experiment to estimate the coverage probability of the t-interval for random samples of $\chi^2(2)$ data with sample size $n = 20$. Compare your t-interval results with the simulation results in Example 6.4. (The t-interval should be more robust to departures from normality than the interval for variance.)

6.6 Estimate the 0.025, 0.05, 0.95, and 0.975 quantiles of the skewness $\sqrt{b_1}$ under normality by a Monte Carlo experiment. Compute the standard error of the estimates from (2.14) using the normal approximation for the density (with exact variance formula). Compare the estimated quantiles with the quantiles of the large sample approximation $\sqrt{b_1} \approx N(0, 6/n)$.

6.7 Estimate the power of the skewness test of normality against symmetric Beta(α, α) distributions and comment on the results. Are the results different for heavy-tailed symmetric alternatives such as $t(\nu)$?

6.8 Refer to Example 6.16. Repeat the simulation, but also compute the F test of equal variance, at significance level $\hat{\alpha} \doteq 0.055$. Compare the power of the Count Five test and F test for small, medium, and large sample sizes. (Recall that the F test is not applicable for non-normal distributions.)

6.9 Let X be a non-negative random variable with $\mu = E[X] < \infty$. For a random sample $x_1, \dots, x_n$ from the distribution of X, the Gini ratio is defined by

$$G = \frac{1}{2n^2\mu} \sum_{j=1}^{n} \sum_{i=1}^{n} |x_i - x_j|.$$

The Gini ratio is applied in economics to measure inequality in income distribution (see e.g. [163]). Note that G can be written in terms of the order statistics $x_{(i)}$ as

$$G = \frac{1}{n^2\mu} \sum_{i=1}^{n} (2i - n - 1) x_{(i)}.$$

If the mean is unknown, let $\hat{G}$ be the statistic G with μ replaced by $\bar{x}$. Estimate by simulation the mean, median and deciles of $\hat{G}$ if X is standard lognormal. Repeat the procedure for the uniform distribution and Bernoulli(0.1). Also construct density histograms of the replicates in each case.

6.10 Construct an approximate 95% confidence interval for the Gini ratio $\gamma = E[G]$ if X is lognormal with unknown parameters. Assess the coverage rate of the estimation procedure with a Monte Carlo experiment.

Projects

6.A Use Monte Carlo simulation to investigate whether the empirical Type I error rate of the t-test is approximately equal to the nominal significance level α, when the sampled population is non-normal. The t-test is robust to mild departures from normality. Discuss the simulation results for the cases where the sampled population is (i) $\chi^2(1)$, (ii) Uniform(0,2), and (iii) Exponential(rate=1). In each case, test $H_0 : \mu = \mu_0$ vs $H_0 : \mu \neq \mu_0$, where μ_0 is the mean of $\chi^2(1)$, Uniform(0,2), and Exponential(1), respectively.

6.B Tests for association based on Pearson product moment correlation ρ, Spearman's rank correlation coefficient ρ_s, or Kendall's coefficient τ, are implemented in `cor.test`. Show (empirically) that the nonparametric tests based on ρ_s or τ are less powerful than the correlation test when the sampled distribution is bivariate normal. Find an example of an alternative (a bivariate distribution (X, Y) such that X and Y are dependent) such that at least one of the nonparametric tests have better empirical power than the correlation test against this alternative.

6.C Repeat Examples 6.8 and 6.10 for Mardia's multivariate skewness test. Mardia [187] proposed tests of multivariate normality based on multivariate generalizations of skewness and kurtosis. If X and Y are iid, the multivariate population skewness $\beta_{1,d}$ is defined by Mardia as

$$\beta_{1,d} = E\left[(X-\mu)^T\Sigma^{-1}(Y-\mu)\right]^3.$$

Under normality, $\beta_{1,d}=0$. The multivariate skewness statistic is

$$b_{1,d} = \frac{1}{n^2}\sum_{i,j=1}^{n}((X_i-\bar{X})^T\widehat{\Sigma}^{-1}(X_j-\bar{X}))^3, \tag{6.5}$$

where $\hat{\Sigma}$ is the maximum likelihood estimator of covariance. Large values of $b_{1,d}$ are significant. The asymptotic distribution of $nb_{1,d}/6$ is chisquared with $d(d+1)(d+2)/6$ degrees of freedom.

6.D Repeat Example 6.11 for multivariate tests of normality. Mardia [187] defines multivariate kurtosis as

$$\beta_{2,d} = E\left[(X-\mu)^T\Sigma^{-1}(X-\mu)\right]^2.$$

For d-dimensional multivariate normal distributions the kurtosis coefficient is $\beta_{2,d}=d(d+2)$. The multivariate kurtosis statistic is

$$b_{2,d} = \frac{1}{n}\sum_{i=1}^{n}((X_i-\bar{X})^T\widehat{\Sigma}^{-1}(X_i-\bar{X}))^2. \tag{6.6}$$

The large sample test of multivariate normality based on $b_{2,d}$ rejects the null hypothesis at significance level α if

$$\left|\frac{b_{2,d}-d(d+2)}{\sqrt{8d(d+2)/n}}\right| \geq \Phi^{-1}(1-\alpha/2).$$

However, $b_{2,d}$ converges very slowly to the normal limiting distribution. Compare the empirical power of Mardia's skewness and kurtosis tests of multivariate normality with the energy test of multivariate normality `mvnorm.etest` `(energy)` (6.3) [226, 263]. Consider multivariate normal location mixture alternatives where the two samples are generated from `mlbench.twonorm` in the `mlbench` package [174].

Chapter 7

Bootstrap and Jackknife

7.1 The Bootstrap

The bootstrap was introduced in 1979 by Efron [80], with further developments in 1981 [82, 81], 1982 [83], and numerous other publications including the monograph of Efron and Tibshirani [84]. Chernick [45] has an extensive bibliography. Davison and Hinkley [63] is a comprehensive reference with many applications. Also see Barbe and Bertail [19], Shao and Tu [247], and Mammen [186].

Bootstrap methods are a class of nonparametric Monte Carlo methods that estimate the distribution of a population by resampling. Resampling methods treat an observed sample as a finite population, and random samples are generated (resampled) from it to estimate population characteristics and make inferences about the sampled population. Bootstrap methods are often used when the distribution of the target population is not specified; the sample is the only information available.

The term "bootstrap" can refer to nonparametric bootstrap or parametric bootstrap. Monte Carlo methods that involve sampling from a fully specified probability distribution, such as methods of Chapter 6 are sometimes called parametric bootstrap. Nonparametric bootstrap is the subject of this chapter. In nonparametric bootstrap, the distribution is not specified.

The distribution of the finite population represented by the sample can be regarded as a pseudo-population with similar characteristics as the true population. By repeatedly generating random samples from this pseudo-population (resampling), the sampling distribution of a statistic can be estimated. Properties of an estimator such as bias or standard error can be estimated by resampling.

Bootstrap estimates of a sampling distribution are analogous to the idea of density estimation. We construct a histogram of a sample to obtain an estimate of the shape of the density function. The histogram is not the density, but in a nonparametric problem, can be viewed as a reasonable estimate of the density. We have methods to generate random samples from completely specified densities; bootstrap generates random samples from the empirical distribution of the sample.

Suppose that $x = (x_1, \dots, x_n)$ is an observed random sample from a distribution with cdf $F(x)$. If X^* is selected at random from x, then

$$P(X^* = x_i) = \frac{1}{n}, \qquad i = 1, \dots, n.$$

Resampling generates a random sample $X_1^*, \dots, X_n^*$ by sampling with replacement from x. The random variables X_i^* are iid, uniformly distributed on the set $\{x_1, \dots, x_n\}$.

The empirical distribution function (ecdf) $F_n(x)$ is an estimator of $F(x)$. It can be shown that $F_n(x)$ is a sufficient statistic for $F(x)$; that is, all the information about $F(x)$ that is contained in the sample is also contained in $F_n(x)$. Moreover, $F_n(x)$ is itself the distribution function of a random variable; namely the random variable that is uniformly distributed on the set $\{x_1, \dots, x_n\}$. Hence the empirical cdf F_n is the cdf of X^*. Thus in bootstrap, there are two approximations. The ecdf F_n is an approximation to the cdf F_X. The ecdf F_m^* of the bootstrap replicates is an approximation to the ecdf F_n. Resampling from the sample x is equivalent to generating random samples from the distribution $F_n(x)$. The two approximations can be represented by the diagram

$$\begin{aligned} F \to X \to F_n \\ F_n \to X^* \to F_n^*. \end{aligned}$$

To generate a bootstrap random sample by resampling x, generate n random integers $\{i_1, \dots, i_n\}$ uniformly distributed on $\{1, \dots, n\}$ and select the bootstrap sample $x^* = (x_{i_1}, \dots, x_{i_n})$.

Suppose θ is the parameter of interest (θ could be a vector), and $\hat{\theta}$ is an estimator of θ. Then the bootstrap estimate of the distribution of $\hat{\theta}$ is obtained as follows.

1. For each bootstrap replicate, indexed $b = 1, \dots, B$:
   (a) Generate sample $x^{*(b)} = x_1^*, \dots, x_n^*$ by sampling with replacement from the observed sample $x_1, \dots, x_n$.
   (b) Compute the b^{th} replicate $\hat{\theta}^{(b)}$ from the b^{th} bootstrap sample.
2. The bootstrap estimate of $F_{\hat{\theta}}(\cdot)$ is the empirical distribution of the replicates $\hat{\theta}^{(1)}, \dots, \hat{\theta}^{(B)}$.

The bootstrap is applied to estimate the standard error and the bias of an estimator in the following sections. First let us see an example to illustrate the relation between the ecdf F_n and the distribution of the bootstrap replicates.

Example 7.1 (F_n and bootstrap samples)

Suppose that we have observed the sample

$$x = \{2, 2, 1, 1, 5, 4, 4, 3, 1, 2\}.$$

Resampling from x we select 1, 2, 3, 4, or 5 with probabilities 0.3, 0.3, 0.1, 0.2, and 0.1 respectively, so the cdf F_{X^*} of a randomly selected replicate is exactly the ecdf $F_n(x)$:

$$F_{X^*}(x) = F_n(x) = \begin{cases} 0, & x < 1; \\ 0.3, & 1 \leq x < 2; \\ 0.6, & 2 \leq x < 3; \\ 0.7, & 3 \leq x < 4; \\ 0.9, & 4 \leq x < 5; \\ 1, & x \geq 5. \end{cases}$$

Note that if F_n is not close to F_X then the distribution of the replicates will not be close to F_X. The sample x above is actually a sample from a Poisson(2) distribution. Resampling from x a large number of replicates produces a good estimate of F_n but not a good estimate of F_X, because regardless of how many replicates are drawn, the bootstrap samples will never include 0. $\diamond$

7.1.1 Bootstrap Estimation of Standard Error

The bootstrap estimate of standard error of an estimator $\hat{\theta}$ is the sample standard deviation of the bootstrap replicates $\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B)}$.

$$\widehat{se}(\hat{\theta}^*) = \sqrt{\frac{1}{B-1} \sum_{b=1}^{B} (\hat{\theta}^{(b)} - \overline{\hat{\theta}^*})^2}, \tag{7.1}$$

where $\overline{\hat{\theta}^*} = \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}^{(b)}$ [84, (6.6)].

According to Efron and Tibshirani [84, p. 52], the number of replicates needed for good estimates of standard error is not large; $B = 50$ is usually large enough, and rarely is $B > 200$ necessary. (Much larger B will be needed for confidence interval estimation.)

Example 7.2 (Bootstrap estimate of standard error)

The law school data set `law` in the `bootstrap` [271] package is from Efron and Tibshirani [84]. The data frame contains LSAT (average score on law school admission test score) and GPA (average undergraduate grade-point average) for 15 law schools.

```
LSAT 576 635 558 578 666 580 555 661 651 605 653 575 545 572 594
GPA  339 330 281 303 344 307 300 343 336 313 312 274 276 288 296
```

This data set is a random sample from the universe of 82 law schools in `law82` `(bootstrap)`. Estimate the correlation between LSAT and GPA scores, and compute the bootstrap estimate of the standard error of the sample correlation.

1. For each bootstrap replicate, indexed $b = 1, \dots, B$:
   (a) Generate sample $x^{*(b)} = x_1^*, \dots, x_n^*$ by sampling with replacement from the observed sample $x_1, \dots, x_n$.
   (b) Compute the b^{th} replicate $\hat{\theta}^{(b)}$ from the b^{th} bootstrap sample, where $\hat{\theta}$ is the sample correlation R between (`LSAT`, `GPA`).
2. The bootstrap estimate of se(R) is the sample standard deviation of the replicates $\hat{\theta}^{(1)}, \dots, \hat{\theta}^{(B)} = R^{(1)}, \dots, R^{(B)}$.

```
library(bootstrap)    #for the law data
print(cor(law$LSAT, law$GPA))
[1] 0.7763745
print(cor(law82$LSAT, law82$GPA))
[1] 0.7599979
```

The sample correlation is $R = 0.7763745$. The correlation for the universe of 82 law schools is $R = 0.7599979$. Use bootstrap to estimate the standard error of the correlation statistic computed from the sample of scores in `law`.

```
#set up the bootstrap
B <- 200             #number of replicates
n <- nrow(law)       #sample size
R <- numeric(B)      #storage for replicates

#bootstrap estimate of standard error of R
for (b in 1:B) {
    #randomly select the indices
    i <- sample(1:n, size = n, replace = TRUE)
    LSAT <- law$LSAT[i]       #i is a vector of indices
    GPA <- law$GPA[i]
    R[b] <- cor(LSAT, GPA)
}
#output
> print(se.R <- sd(R))
[1] 0.1358393
> hist(R, prob = TRUE)
```

The bootstrap estimate of se(R) is 0.1358393. The normal theory estimate for standard error of R is 0.115. The jackknife-after-bootstrap method of estimating $\widehat{se}(\widehat{se}(\hat{\theta}))$ is covered in Section 7.3. The histogram of the replicates of R is shown in Figure 7.1. $\diamond$

In the next example, the `boot` function in recommended package `boot` [34] is applied to run the bootstrap. See Appendix B.1 for a note about how to write the function for the `statistic` argument in `boot`.

Example 7.3 (Bootstrap estimate of standard error: `boot` function)

Example 7.2 is repeated, using the `boot` function in `boot`. First, write a function that returns $\hat{\theta}^{(b)}$, where the first argument to the function is the sample data, and the second argument is the vector $\{i_1, \dots, i_n\}$ of indices. If the data is `x` and the vector of indices is `i`, we need `x[i,1]` to extract the first resampled variable, and `x[i,2]` to extract the second resampled variable. The code and output is shown below.

```
r <- function(x, i) {
    #want correlation of columns 1 and 2
    cor(x[i,1], x[i,2])
}
```

The printed summary of output from the `boot` function is obtained by the command `boot` or the result can be saved in an object for further analysis. Here we save the result in `obj` and print the summary.

```
library(boot)       #for boot function
> obj <- boot(data = law, statistic = r, R = 2000)
> obj

ORDINARY NONPARAMETRIC BOOTSTRAP

Call: boot(data = law, statistic = r, R = 2000)
Bootstrap Statistics :
     original       bias    std. error
t1* 0.7763745 -0.004795305   0.1303343
```

The observed value $\hat{\theta}$ of the correlation statistic is labeled `t1*`. The bootstrap estimate of standard error of the estimate is $\widehat{se}(\hat{\theta}) \doteq 0.13$, based on 2000 replicates. To compare with formula (7.1), extract the replicates in `$t`.

```
> y <- obj$t
> sd(y)
[1] 0.1303343
```

$\diamond$

R note 7.1 *The syntax and options for the* `boot (boot)` *function and the* `bootstrap (bootstrap)` *function are different. Note that the* `bootstrap` *package [271] is a collection of functions and data for the book by Efron and Tibshirani [84], and the* `boot` *package [34] is a collection of functions and data for the book by Davison and Hinkley [63].*

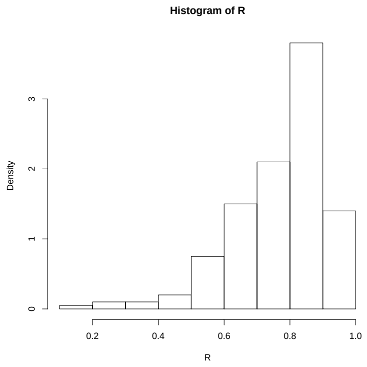


FIGURE 7.1: Bootstrap replicates for law school data in Example 7.2.

7.1.2 Bootstrap Estimation of Bias

If $\hat{\theta}$ is an unbiased estimator of θ, $E[\hat{\theta}] = \theta$. The bias of an estimator $\hat{\theta}$ for θ is

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

Thus, every statistic is an unbiased estimator of its expected value, and in particular, the sample mean of a random sample is an unbiased estimator of the mean of the distribution. An example of a biased estimator is the maximum likelihood estimator of variance, $\hat{\sigma}^2 = \frac{1}{n}\Sigma_{i=1}^n (X_i - \overline{X})^2$, which has expected value $(1 - 1/n)\sigma^2$. Thus, $\hat{\sigma}^2$ underestimates σ^2, and the bias is $-\sigma^2/n$.

The bootstrap estimation of bias uses the bootstrap replicates of $\hat{\theta}$ to estimate the sampling distribution of $\hat{\theta}$. For the finite population $x = (x_1, \dots, x_n)$, the parameter is $\hat{\theta}(x)$ and there are B independent and identically distributed estimators $\hat{\theta}^{(b)}$. The sample mean of the replicates $\{\hat{\theta}^{(b)}\}$ is unbiased for its expected value $E[\hat{\theta}^*]$, so the bootstrap estimate of bias is

$$\widehat{bias}(\hat{\theta}) = \overline{\hat{\theta}^*} - \hat{\theta}, \tag{7.2}$$

where $\overline{\hat{\theta}^*} = \frac{1}{B}\sum_{b=1}^B \hat{\theta}^{(b)}$, and $\hat{\theta} = \hat{\theta}(x)$ is the estimate computed from the original observed sample. (In bootstrap F_n is sampled in place of F_X, so we replace θ with $\hat{\theta}$ to estimate the bias.) Positive bias indicates that $\hat{\theta}$ on average tends to overestimate θ.

Example 7.4 (Boostrap estimate of bias)

In the `law` data of Example 7.2, compute the bootstrap estimate of bias in the sample correlation.

```
#sample estimate for n=15
theta.hat <- cor(law$LSAT, law$GPA)

#bootstrap estimate of bias
B <- 2000   #larger for estimating bias
n <- nrow(law)
theta.b <- numeric(B)

for (b in 1:B) {
    i <- sample(1:n, size = n, replace = TRUE)
    LSAT <- law$LSAT[i]
    GPA <- law$GPA[i]
    theta.b[b] <- cor(LSAT, GPA)
}
bias <- mean(theta.b - theta.hat)
> bias
[1] -0.005797944
```

The estimate of bias is -0.005797944. Note that this is close to the estimate of bias returned by the `boot` function in Example 7.3. See Section 7.3 for the jackknife-after-bootstrap method to estimate the standard error of the bootstrap estimate of bias. ◇

Example 7.5 (Bootstrap estimate of bias of a ratio estimate)

The `patch (bootstrap)` data from Efron and Tibshirani [84, 10.3] contains measurements of a certain hormone in the bloodstream of eight subjects after wearing a medical patch. The parameter of interest is

$$\theta = \frac{E(new) - E(old)}{E(old) - E(placebo)}.$$

If $|\theta| \leq 0.20$, this indicates bioequivalence of the old and new patches. The statistic is $\overline{Y}/\overline{Z}$. Compute a bootstrap estimate of bias in the bioequivalence ratio statistic.

```
data(patch, package = "bootstrap")
> patch
  subject placebo oldpatch newpatch     z     y
1       1    9243    17649    16449  8406 -1200
2       2    9671    12013    14614  2342  2601
3       3   11792    19979    17274  8187 -2705
4       4   13357    21816    23798  8459  1982
5       5    9055    13850    12560  4795 -1290
6       6    6290     9806    10157  3516   351
7       7   12412    17208    16570  4796  -638
8       8   18806    29044    26325 10238 -2719

n <- nrow(patch)  #in bootstrap package
B <- 2000
theta.b <- numeric(B)
theta.hat <- mean(patch$y) / mean(patch$z)

#bootstrap
for (b in 1:B) {
    i <- sample(1:n, size = n, replace = TRUE)
    y <- patch$y[i]
    z <- patch$z[i]
    theta.b[b] <- mean(y) / mean(z)
    }
bias <- mean(theta.b) - theta.hat
se <- sd(theta.b)
print(list(est=theta.hat, bias = bias,
           se = se, cv = bias/se))

$est [1] -0.0713061
$bias [1] 0.007901101
$se [1] 0.1046453
$cv [1] 0.07550363
```

If $|bias|/se \leq 0.25$, it is not usually necessary to adjust for bias [84, 10.3]. The bias is small relative to standard error ($cv < 0.08$), so in this example it is not necessary to adjust for bias. ◇

7.2 The Jackknife

The *jackknife* is another resampling method, proposed by Quenouille [215, 216] for estimating bias, and by Tukey [274] for estimating standard error, a few decades earlier than the bootstrap. Efron [83] is a good introduction to the jackknife.

The jackknife is like a "leave-one-out" type of cross-validation. Let $x = (x_1,\dots,x_n)$ be an observed random sample, and define the i^{th} jackknife sample $x_{(i)}$ to be the subset of x that leaves out the i^{th} observation x_i. That is,

$$x_{(i)} = (x_1,\dots,x_{i-1},x_{i+1},\dots,x_n).$$

If $\hat\theta = T_n(x)$, define the i^{th} jackknife replicate $\hat\theta_{(i)} = T_{n-1}(x_{(i)})$, $i = 1,\dots,n$.

Suppose the parameter $\theta = t(F)$ is a function of the distribution F. Let F_n be the ecdf of a random sample from the distribution F. The "plug-in" estimate of θ is $\hat\theta = t(F_n)$. A "plug-in" $\hat\theta$ is smooth in the sense that small changes in the data correspond to small changes in $\hat\theta$. For example, the sample mean is a plug-in estimate for the population mean, but the sample median is not a plug-in estimate for the population median.

The Jackknife Estimate of Bias

If $\hat\theta$ is a smooth (plug-in) statistic, then $\hat\theta_{(i)} = t(F_{n-1}(x_{(i)}))$, and the jackknife estimate of bias is

$$\widehat{bias}_{jack} = (n-1)(\ \overline{\hat\theta_{(\cdot)}} - \hat\theta\ ), \tag{7.3}$$

where $\overline{\hat\theta_{(\cdot)}} = \frac{1}{n}\sum_{i=1}^n \hat\theta_{(i)}$ is the mean of the estimates from the leave-one-out samples, and $\hat\theta = \hat\theta(x)$ is the estimate computed from the original observed sample.

To see why the jackknife estimator (7.3) has the factor $n-1$, consider the case where θ is the population variance. If $x_1,\dots,x_n$ is a random sample from the distribution of X, the plug-in estimate of the variance of X is

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

The estimator $\hat\theta$ is biased for σ_X^2 with

$$bias(\hat\theta) = E[\hat\theta - \sigma_X^2] = \frac{n-1}{n}\sigma_X^2 - \sigma_X^2 = -\frac{\sigma_X^2}{n}.$$

Each jackknife replicate computes the estimate $\hat\theta_{(i)}$ on a sample size $n-1$, so that the bias in the jackknife replicate is $-\sigma_X^2/(n-1)$. Thus, for $i = 1,\dots,n$ we have

$$\begin{aligned} E[\hat\theta_{(i)} - \hat\theta] &= E[\hat\theta_{(i)} - \theta] - E[\hat\theta - \theta] \\ &= bias(\hat\theta_{(i)}) - bias(\hat\theta) \\ &= -\frac{\sigma_X^2}{n-1} - \left(-\frac{\sigma_X^2}{n}\right) = -\frac{\sigma_X^2}{n(n-1)} = \frac{bias(\hat\theta)}{n-1}. \end{aligned}$$

Thus, the jackknife estimate (7.3) with factor $(n-1)$ gives the correct estimate of bias in the plug-in estimator of variance, which is also the maximum likelihood estimator of variance.

R note 7.2 *(leave-one-out) The [ ] operator provides a very simple way to leave out the i^{th} element of a vector.*

```
x <- 1:5
for (i in 1:5)
    print(x[-i])

[1] 2 3 4 5
[1] 1 3 4 5
[1] 1 2 4 5
[1] 1 2 3 5
[1] 1 2 3 4
```

Note that the jackknife requires only n replications to estimate the bias; the bootstrap estimate of bias typically requires several hundred replicates.

Example 7.6 (Jackknife estimate of bias)

Compute the jackknife estimate of bias for the `patch` data in Example 7.5.

```
data(patch, package = "bootstrap")
n <- nrow(patch)
y <- patch$y
z <- patch$z
theta.hat <- mean(y) / mean(z)
print (theta.hat)

#compute the jackknife replicates, leave-one-out estimates
theta.jack <- numeric(n)
for (i in 1:n)
    theta.jack[i] <- mean(y[-i]) / mean(z[-i])
bias <- (n - 1) * (mean(theta.jack) - theta.hat)

> print(bias)  #jackknife estimate of bias
[1] 0.008002488
```

◇

The jackknife estimate of standard error

A jackknife estimate of standard error [274], [84, (11.5)] is

$$\widehat{se}_{jack} = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}\left(\hat{\theta}_{(i)} - \overline{\hat{\theta}_{(\cdot)}}\right)^2}, \tag{7.4}$$

for smooth statistics $\hat{\theta}$.

To see why the jackknife estimator of standard error (7.4) has the factor $(n-1)/n$, consider the case where θ is the population mean and $\hat{\theta} = \overline{X}$. The standard error of the mean of X is $\sqrt{Var(X)/n}$. A factor of $(n-1)/n$ under the radial makes $\widehat{se}_{jack}$ an unbiased estimator of the standard error of the mean.

We can also consider the plug-in estimate of the standard error of the mean. In the case of a continuous random variable X, the plug-in estimate of the variance of a random sample is the variance of Y, where Y is uniformly distributed on the sample $x_1, \ldots, x_n$. That is,

$$\begin{aligned}\widehat{Var}(Y) &= \frac{1}{n}E[Y - E[Y]]^2 = \frac{1}{n}E[Y - \overline{X}]^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2 \cdot \frac{1}{n} \\ &= \frac{n-1}{n^2}S_X^2 = \frac{n-1}{n}[\widehat{se}(\overline{X})]^2.\end{aligned}$$

Thus, for the jackknife estimator of standard error, a factor of $((n-1)/n)^2$ gives the plug-in estimate of variance. The factors $((n-1)/n)^2$ and $((n-1)/n)$ are approximately equal if n is not small. Efron and Tibshirani [84] remark that the choice of the factor $(n-1)/n$ instead of $((n-1)/n)^2$ is somewhat arbitrary.

Example 7.7 (Jackknife estimate of standard error)

To compute the jackknife estimate of standard error for the `patch` data in Example 7.5, use the jackknife replicates from Example 7.6.

```
se <- sqrt((n-1) *
    mean((theta.jack - mean(theta.jack))^2))
> print(se)
[1] 0.1055278
```

The jackknife estimate of standard error is 0.1055278. From the previous result for the bias, we have the estimated coefficient of variation

```
> .008002488/.1055278
[1] 0.07583298
```

◇

When the Jackknife Fails

The jackknife can fail when the statistic $\hat{\theta}$ is not "smooth." The statistic is a function of the data. Smoothness means that small changes in the data correspond to small changes in the statistic. The median is an example of a statistic that is not smooth.

Example 7.8 (Failure of jackknife)

In this example the jackknife estimate of standard error of the median is computed for a random sample of 10 integers from 1, 2 . . . , 100.

```
n <- 10
x <- sample(1:100, size = n)

#jackknife estimate of se
M <- numeric(n)
for (i in 1:n) {        #leave one out
    y <- x[-i]
    M[i] <- median(y)
}
Mbar <- mean(M)
print(sqrt((n-1)/n * sum((M - Mbar)^2)))

#bootstrap estimate of se
Mb <- replicate(1000, expr = {
        y <- sample(x, size = n, replace = TRUE)
        median(y) })
print(sd(Mb))

# details and results:
# the sample, x:        29 79 41 86 91  5 50 83 51 42
# jackknife medians:    51 50 51 50 50 51 51 50 50 51
# jackknife est. of se: 1.5
# bootstrap medians:    46 50 46 79 79 51 81 65 ...
# bootstrap est. of se: 13.69387
```

Clearly something is wrong here, because the bootstrap estimate and the jackknife estimate are far apart. The jackknife fails, because the median is not smooth. ◇

In this case, when the statistic is not smooth, the delete-d jackknife (leave d observations out on each replicate) can be applied (see Efron and Tibshirani [84, 11.7]). If $\sqrt{n}/d \to 0$ and $n - d \to \infty$ then the delete-d jackknife is consistent for the median. The computing time increases because there are a large number of jackknife replicates when n and d are large.

7.3 Jackknife-after-Bootstrap

In this chapter, bootstrap estimates of standard error and bias have been introduced. These estimates are random variables. If we are interested in the variance of these estimates, one idea is to try the jackknife.

Recall that $\widehat{se}(\hat{\theta})$ is the sample standard deviation of B bootstrap replicates of $\hat{\theta}$. Now, if we leave out the i^{th} observation, the algorithm for estimation of standard error is to resample B replicates from the $n-1$ remaining observations – for each i. In other words, we would replicate the bootstrap itself. Fortunately, there is a way to avoid replicating the bootstrap.

The *jackknife-after-bootstrap* computes an estimate for each "leave-one-out" sample. Let $J(i)$ denote the indices of bootstrap samples that do not contain x_i, and let $B(i)$ denote number of bootstrap samples that do not contain x_i. Then we can compute the jackknife replication leaving out the $B - B(i)$ samples that contain x_i [84, p. 277]. The jackknife estimate of standard error is computed by the formula (7.4). Compute

$$\widehat{se}(\hat{\theta}) = \widehat{se}_{jack}(\widehat{se}_{B(1)}, \ldots, \widehat{se}_{B(n)}),$$

where

$$\widehat{se}_{B(i)} = \sqrt{\frac{1}{B(i)} \sum_{j \in J(i)} \left[\hat{\theta}_{(j)} - \overline{\hat{\theta}_{(J(i))}}\right]^2}, \qquad (7.5)$$

and

$$\overline{\hat{\theta}_{(J(i))}} = \frac{1}{B(i)} \sum_{j \in J(i)} \hat{\theta}_{(j)}$$

is the sample mean of the estimates from the leave-x_i-out jackknife samples.

Example 7.9 (Jackknife-after-bootstrap)

Use the jackknife-after-bootstrap procedure to estimate the standard error of $\widehat{se}(\hat{\theta})$ for the `patch` data in Example 7.7.

```
# initialize
data(patch, package = "bootstrap")
n <- nrow(patch)
y <- patch$y
z <- patch$z
B <- 2000
theta.b <- numeric(B)
# set up storage for the sampled indices
indices <- matrix(0, nrow = B, ncol = n)
```

```
# jackknife-after-bootstrap step 1: run the bootstrap
for (b in 1:B) {
    i <- sample(1:n, size = n, replace = TRUE)
    y <- patch$y[i]
    z <- patch$z[i]
    theta.b[b] <- mean(y) / mean(z)
    #save the indices for the jackknife
    indices[b, ] <- i
    }

#jackknife-after-bootstrap to est. se(se)
se.jack <- numeric(n)
for (i in 1:n) {
    #in i-th replicate omit all samples with x[i]
    keep <- (1:B)[apply(indices, MARGIN = 1,
                 FUN = function(k) {!any(k == i)})]
    se.jack[i] <- sd(theta.b[keep])
}

> print(sd(theta.b))
[1] 0.1027102
> print(sqrt((n-1) * mean((se.jack - mean(se.jack))^2)))
[1] 0.03050501
```

The bootstrap estimate of standard error is 0.1027102 and jackknife-after-bootstrap estimate of its standard error is 0.03050501. ◇

Jackknife-after-bootstrap: Empirical influence values

The empirical influence values in jackknife-after-bootstrap are empirical quantities that measure the difference between each jackknife replicate and the observed statistic. There are several methods for estimating the influence values. One approach uses the usual jackknife differences $\hat{\theta}_{(i)} - \hat{\theta}$, $i = 1, \ldots, n$. The `empinf` function in the `boot` package computes empirical influence values by four methods. The `jack.after.boot` function in the `boot` package [34] produces a plot of empirical influence values. The plots can be used as a diagnostic tool to see the effect or influence of individual observations. See [63, Ch. 3] for examples and a discussion of how to interpret the plots.

7.4 Bootstrap Confidence Intervals

In this section several approaches to obtaining approximate confidence intervals for the target parameter in a bootstrap are discussed. The methods include the *standard normal bootstrap* confidence interval, the *basic bootstrap* confidence interval, the *bootstrap percentile* confidence interval, and the *bootstrap t confidence interval.* Readers are referred to [63] and [84] for theoretical properties and discussion of empirical performance of methods for bootstrap confidence interval estimates.

7.4.1 The Standard Normal Bootstrap Confidence Interval

The standard normal bootstrap confidence interval is the simplest approach, but not necessarily the best. Suppose that $\hat{\theta}$ is an estimator of parameter θ, and assume the standard error of the estimator is $se(\hat{\theta})$. If $\hat{\theta}$ is a sample mean and the sample size is large, then the Central Limit Theorem implies that

$$Z = \frac{\hat{\theta} - E[\hat{\theta}]}{se(\hat{\theta})} \tag{7.6}$$

is approximately standard normal. Hence, if $\hat{\theta}$ is unbiased for θ, then an approximate $100(1-\alpha)\%$ confidence interval for θ is the Z-interval

$$\hat{\theta} \pm z_{\alpha/2} se(\hat{\theta}),$$

where $z_{\alpha/2} = \Phi^{-1}(1-\alpha/2)$. This interval is easy to compute, but we have made several assumptions. To apply the normal distribution, we assume that the distribution of $\hat{\theta}$ is normal or $\hat{\theta}$ is a sample mean and the sample size is large. We have also implicitly assumed that $\hat{\theta}$ is unbiased for θ.

Bias can be estimated and used to center the Z statistic, but the estimator is a random variable, so the transformed variable is not normal. Here we have treated $se(\hat{\theta})$ as a known parameter, but in the bootstrap $se(\hat{\theta})$ is estimated (the sample standard deviation of the replicates).

7.4.2 The Basic Bootstrap Confidence Interval

The basic bootstrap confidence interval transforms the distribution of the replicates by subtracting the observed statistic. The quantiles of the transformed sample are used to determine the confidence limits.

The $100(1-\alpha)\%$ confidence limits for the basic bootstrap confidence interval are

$$(2\hat{\theta} - \hat{\theta}_{1-\alpha/2}, \qquad 2\hat{\theta} - \hat{\theta}_{\alpha/2}). \tag{7.7}$$

To see how the confidence limits in (7.7) are determined, consider first the parametric case. Suppose that T is an estimator of θ and a_α is the α quantile of $T-\theta$. Then

$$P(T-\theta > a_\alpha) = 1-\alpha \Rightarrow P(T-a_\alpha > \theta) = 1-\alpha.$$

Thus, a $100(1-2\alpha)\%$ confidence interval with equal lower and upper tail errors α is given by $(t-a_{1-\alpha},\, t-a_\alpha)$.

In bootstrap the distribution of T is generally unknown, but quantiles can be estimated and an approximate method applied.

Compute the sample α quantiles $\hat\theta_\alpha$ from the ecdf of the replicates $\hat\theta^*$. Denote the α quantile of $\hat\theta^*-\hat\theta$ by b_α. Then $\hat b_\alpha = \hat\theta_\alpha - \hat\theta$ is an estimator of b_α. An approximate upper confidence limit for a $100(1-\alpha)\%$ confidence interval for θ is given by

$$\hat\theta - \hat b_{\alpha/2} = \hat\theta - (\hat\theta_{\alpha/2} - \hat\theta) = 2\hat\theta - \hat\theta_{\alpha/2}.$$

Similarly an approximate lower confidence limit is given by $2\hat\theta - \hat\theta_{1-\alpha/2}$. Thus, a $100(1-\alpha)$ basic bootstrap confidence interval for θ is given by (7.7). See Davison and Hinkley [63, 5.2] for more details.

7.4.3 The Percentile Bootstrap Confidence Interval

A bootstrap percentile interval uses the empirical distribution of the bootstrap replicates as the reference distribution. The quantiles of the empirical distribution are estimators of the quantiles of the sampling distribution of $\hat\theta$, so that these (random) quantiles may match the true distribution better when the distribution of $\hat\theta$ is not normal. Suppose that $\hat\theta^{(1)},\dots,\hat\theta^{(B)}$ are the bootstrap replicates of the statistic $\hat\theta$. From the ecdf of the replicates, compute the $\alpha/2$ quantile $\hat\theta_{\alpha/2}$, and the $1-\alpha/2$ quantile $\hat\theta_{1-\alpha/2}$.

Efron and Tibshirani [84, 13.3] show that the percentile interval has some theoretical advantages over the standard normal interval and somewhat better coverage performance.

Adjustments to percentile methods have been proposed. For example, the *bias-corrected and accelerated* (BCa) percentile intervals (see Section 7.5) are a modified version of percentile intervals that have better theoretical properties and better performance in practice.

The `boot.ci (boot)` function [34] computes five types of bootstrap confidence intervals: basic, normal, percentile, studentized, and BCa. To use this function, first call `boot` for the bootstrap, and pass the returned `boot` object to `boot.ci` (along with other required arguments). For more details see Davison and Hinkley [63, Ch. 5] and the `boot.ci` help topic.

Example 7.10 (Bootstrap confidence intervals for `patch` ratio statistic.)

This example illustrates how to obtain the normal, basic, and percentile bootstrap confidence intervals using the `boot` and `boot.ci` functions in the `boot` package. The code generates 95% confidence intervals for the ratio statistic in Example 7.5.

```
library(boot)       #for boot and boot.ci
data(patch, package = "bootstrap")

theta.boot <- function(dat, ind) {
    #function to compute the statistic
    y <- dat[ind, 1]
    z <- dat[ind, 2]
    mean(y) / mean(z)
}
```

Run the bootstrap and compute confidence interval estimates for the bioequivalence ratio.

```
y <- patch$y
z <- patch$z
dat <- cbind(y, z)
boot.obj <- boot(dat, statistic = theta.boot, R = 2000)
```

The output for the bootstrap and bootstrap confidence intervals is below.

```
print(boot.obj)
ORDINARY NONPARAMETRIC BOOTSTRAP
Call: boot(data = dat, statistic = theta.boot, R = 2000)
Bootstrap Statistics :
    original     bias    std. error
t1* -0.0713061 0.01047726   0.1010179

print(boot.ci(boot.obj,
              type = c("basic", "norm", "perc")))

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates
CALL : boot.ci(boot.out = boot.obj, type = c("basic",
    "norm", "perc"))
Intervals :
Level     Normal             Basic             Percentile
95%  (-0.2798, 0.1162 ) (-0.3045, 0.0857 ) (-0.2283, 0.1619 )
Calculations and Intervals on Original Scale
```

Recall that the old and new patches are bioequivalent if $|\theta| \leq 0.20$. Hence, the interval estimates do not support bioequivalence of the old and new patches. Next we compute the bootstrap confidence intervals according to their definitions. Compare the following results with the `boot.ci` output.

```
    #calculations for bootstrap confidence intervals
    alpha <- c(.025, .975)

    #normal
    print(boot.obj$t0 + qnorm(alpha * sd(boot.obj$t)))
 -0.2692975  0.1266853

    #basic
    print(2*boot.obj$t0 -
          quantile(boot.obj$t, rev(alpha), type=1))
    97.5%       2.5%
-0.3018698  0.0857679

    #percentile
    print(quantile(boot.obj$t, alpha, type=6))
     2.5%      97.5%
-0.2283370  0.1618647
```

◇

R note 7.3 *The normal interval computed by* `boot.ci` *corrects for bias. Notice that the* `boot.ci` *normal interval differs from our result by the bias estimate shown in the output from* `boot`*. This is confirmed by reading the source code for the function. To view the source code for this calculation, when the* `boot` *package is loaded, enter the command* `getAnywhere(norm.ci)` *at the console. Also see* `norm.inter` *and [63] for details of calculations of quantiles.*

Example 7.11 (Bootstrap confidence intervals for the correlation statistic)

Compute 95% bootstrap confidence interval estimates for the correlation statistic in the `law` data of Example 7.2.

```
library(boot)
data(law, package = "bootstrap")
boot.obj <- boot(law, R = 2000,
          statistic = function(x, i){cor(x[i,1], x[i,2])})
print(boot.ci(boot.obj, type=c("basic","norm","perc")))
    ...

Intervals :
Level   Normal             Basic              Percentile
95% (0.5182, 1.0448) (0.5916, 1.0994) (0.4534, 0.9611)
Calculations and Intervals on Original Scale
```

All three intervals cover the correlation $\rho = .76$ of the universe of all law schools in `law82`. One reason for the difference in the percentile and normal confidence intervals could be that the sampling distribution of correlation statistic is not close to normal (see the histogram in Figure 7.1). When the sampling distribution of the statistic is approximately normal, the percentile interval will agree with the normal interval. $\diamond$

7.4.4 The Bootstrap t interval

Even if the distribution of $\hat{\theta}$ is normal and $\hat{\theta}$ is unbiased for θ, the normal distribution is not exactly correct for the Z statistic (7.6), because we estimate $se(\hat{\theta})$. Nor can we claim that it is a Student t statistic, because the distribution of the bootstrap estimator $\widehat{se}(\hat{\theta})$ is unknown. The bootstrap t interval does not use a Student t distribution as the reference distribution. Instead, the sampling distribution of a "t type" statistic (a studentized statistic) is generated by resampling. Suppose $x = (x_1, \ldots, x_n)$ is an observed sample. The $100(1-\alpha)\%$ bootstrap t confidence interval is

$$(\hat{\theta} - t^*_{1-\alpha/2}\widehat{se}(\hat{\theta}), \qquad \hat{\theta} - t^*_{\alpha/2}\widehat{se}(\hat{\theta})),$$

where $\widehat{se}(\hat{\theta})$, $t^*_{\alpha/2}$ and $t^*_{1-\alpha/2}$ are computed as outlined below.

Bootstrap t interval (studentized bootstrap interval)

1. Compute the observed statistic $\hat{\theta}$.
2. For each replicate, indexed $b = 1, \ldots, B$:
   (a) Sample with replacement from x to get the b^{th} sample $x^{(b)} = (x_1^{(b)}, \ldots, x_n^{(b)})$.
   (b) Compute $\hat{\theta}^{(b)}$ from the b^{th} sample $x^{(b)}$.
   (c) Compute or estimate the standard error $\widehat{se}(\hat{\theta}^{(b)})$ (a separate estimate for each bootstrap sample; a bootstrap estimate will resample from the current bootstrap sample $x^{(b)}$, not x).
   (d) Compute the b^{th} replicate of the "t" statistic, $t^{(b)} = \frac{\hat{\theta}^{(b)} - \hat{\theta}}{\widehat{se}(\hat{\theta}^{(b)})}$.
3. The sample of replicates $t^{(1)}, \ldots, t^{(B)}$ is the reference distribution for bootstrap t. Find the sample quantiles $t^*_{\alpha/2}$ and $t^*_{1-\alpha/2}$ from the ordered sample of replicates $t^{(b)}$.
4. Compute $\widehat{se}(\hat{\theta})$, the sample standard deviation of the replicates $\hat{\theta}^{(b)}$.
5. Compute confidence limits

$$(\hat{\theta} - t^*_{1-\alpha/2}\widehat{se}(\hat{\theta}), \qquad \hat{\theta} - t^*_{\alpha/2}\widehat{se}(\hat{\theta})).$$

One disadvantage to the bootstrap t interval is that typically the estimates of standard error $\widehat{se}(\hat{\theta}^{(b)})$ must be obtained by bootstrap. This is a bootstrap nested inside a bootstrap. If $B = 1000$, for example, the bootstrap t confidence interval method takes approximately 1000 times longer than any of the other methods.

Example 7.12 (Bootstrap t confidence interval)

This example provides a function to compute a bootstrap t confidence interval for a univariate or a multivariate sample. The required arguments to the function are the sample data `x`, and the function `statistic` that computes the statistic. The default confidence level is 95%, the number of bootstrap replicates defaults to 500, and the number of replicates for estimating standard error defaults to 100.

```
boot.t.ci <-
function(x, B = 500, R = 100, level = .95, statistic){
    #compute the bootstrap t CI
    x <- as.matrix(x);  n <- nrow(x)
    stat <- numeric(B); se <- numeric(B)

    boot.se <- function(x, R, f) {
        #local function to compute the bootstrap
        #estimate of standard error for statistic f(x)
        x <- as.matrix(x); m <- nrow(x)
        th <- replicate(R, expr = {
            i <- sample(1:m, size = m, replace = TRUE)
            f(x[i, ])
            })
        return(sd(th))
    }

    for (b in 1:B) {
        j <- sample(1:n, size = n, replace = TRUE)
        y <- x[j, ]
        stat[b] <- statistic(y)
        se[b] <- boot.se(y, R = R, f = statistic)
    }
    stat0 <- statistic(x)
    t.stats <- (stat - stat0) / se
    se0 <- sd(stat)
    alpha <- 1 - level
    Qt <- quantile(t.stats, c(alpha/2, 1-alpha/2), type = 1)
    names(Qt) <- rev(names(Qt))
    CI <- rev(stat0 - Qt * se0)
}
```

Note that the `boot.se` function is a local function, visible only inside the `boot.t.ci` function. The next example applies the `boot.t.ci` function. ◇

Example 7.13 (Bootstrap t confidence interval for `patch` ratio statistic.)

Compute a 95% bootstrap t confidence interval for the ratio statistic in Examples 7.5 and 7.10.

```
    dat <- cbind(patch$y, patch$z)
    stat <- function(dat) {
        mean(dat[, 1]) / mean(dat[, 2]) }
    ci <- boot.t.ci(dat, statistic = stat, B=2000, R=200)
    print(ci)
```

```
     2.5%       97.5%
-0.2547932  0.4055129
```

The upper confidence limit of the bootstrap t confidence interval is much larger than the three intervals in Example 7.10 and the bootstrap t is the widest interval in this example. ◇

7.5 Better Bootstrap Confidence Intervals

Better bootstrap confidence intervals (see [84, Sec. 14.3]) are a modified version of percentile intervals that have better theoretical properties and better performance in practice. For a $100(1-\alpha)\%$ confidence interval, the usual $\alpha/2$ and $1-\alpha/2$ quantiles are adjusted by two factors: a correction for bias and a correction for skewness. The bias correction is denoted z_0 and the skewness or "acceleration" adjustment is a. The better bootstrap confidence interval is called BCa for "bias corrected" and "adjusted for acceleration."

For a $100(1-\alpha)\%$ BCa bootstrap confidence interval compute

$$\alpha_1 = \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1 - \hat{a}(\hat{z}_0 + z_{\alpha/2})}\right), \tag{7.8}$$

$$\alpha_2 = \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1 - \hat{a}(\hat{z}_0 + z_{1-\alpha/2})}\right), \tag{7.9}$$

where $z_\alpha = \Phi^{-1}(\alpha)$, and $\hat{z}_0$, $\hat{a}$ are given by equations (7.10) and (7.11) below. The BCa interval is

$$(\hat{\theta}^*_{\alpha_1}, \hat{\theta}^*_{\alpha_2}).$$

The upper and lower confidence limits of the BCa confidence interval are the empirical α_1 and α_2 quantiles of the bootstrap replicates.

The bias correction factor is in effect measuring the median bias of the replicates $\hat{\theta}^*$ for $\hat{\theta}$. The estimate of this bias is

$$\hat{z}_0 = \Phi^{-1}\left(\frac{1}{B}\sum_{b=1}^{B} I(\hat{\theta}^{(b)} < \hat{\theta})\right), \tag{7.10}$$

where $I(\cdot)$ is the indicator function. Note that $\hat{z}_0 = 0$ if $\hat{\theta}$ is the median of the bootstrap replicates.

The acceleration factor is estimated from jackknife replicates:

$$\hat{a} = \frac{\sum_{i=1}^{n}(\overline{\theta_{(.)}} - \theta_{(i)})^3}{6\sum_{i=1}^{n}((\overline{\theta_{(.)}} - \theta_{(i)})^2)^{3/2}}, \tag{7.11}$$

which measures skewness.

Other methods for estimating the acceleration have been proposed (see e.g. Shao and Tu [247]). Formula (7.11) is given by Efron and Tibshirani [84, p. 186]. The acceleration factor $\hat{a}$ is so named because it estimates the rate of change of the standard error of $\hat{\theta}$ with respect to the target parameter θ (on a normalized scale). When we use a standard normal bootstrap confidence interval, we suppose that $\hat{\theta}$ is approximately normal with mean θ and constant variance $\sigma^2(\hat{\theta})$ that does not depend on the parameter θ. However, it is not always true that the variance of an estimator has constant variance with respect to the target parameter. Consider, for example, the sample proportion $\hat{p} = X/n$ as an estimator of the probability of success p in a binomial experiment, which has variance $p(1-p)/n$. The acceleration factor aims to adjust the confidence limits to account for the possibility that the variance of the estimator may depend on the true value of the target parameter.

Properties of BCa intervals

There are two important theoretical advantages to BCa bootstrap confidence intervals. The BCa confidence intervals are transformation respecting and BCa intervals have second order accuracy.

Transformation respecting means that if $(\hat{\theta}^*_{\alpha_1}, \hat{\theta}^*_{\alpha_2})$ is a confidence interval for θ, and $t(\theta)$ is a transformation of the parameter θ, then the corresponding interval for $t(\theta)$ is $(t(\hat{\theta}^*_{\alpha_1}), t(\hat{\theta}^*_{\alpha_2}))$. A confidence interval is first order accurate if the error tends to zero at rate $1/\sqrt{n}$ for sample size n, and second order accurate if the error tends to zero at rate $1/n$.

The bootstrap t confidence interval is second order accurate but not transformation respecting. The bootstrap percentile interval is transformation respecting but only first order accurate. The standard normal confidence interval is neither transformation respecting nor second order accurate. See [63] for discussion and comparison of theoretical properties of bootstrap confidence intervals.

Example 7.14 (BCa bootstrap confidence interval)

This example implements a function to compute a BCa confidence interval. The BCa interval is $(\hat{\theta}^*_{\alpha_1}, \hat{\theta}^*_{\alpha_2})$, where $\hat{\theta}^*_{\alpha_1}$ and $\hat{\theta}^*_{\alpha_2}$ are given by equations (7.8)–(7.11). ◇

```
boot.BCa <-
function(x, th0, th, stat, conf = .95) {
    # bootstrap with BCa bootstrap confidence interval
    # th0 is the observed statistic
    # th is the vector of bootstrap replicates
    # stat is the function to compute the statistic

    x <- as.matrix(x)
    n <- nrow(x) #observations in rows
    N <- 1:n
    alpha <- (1 + c(-conf, conf))/2
    zalpha <- qnorm(alpha)

    # the bias correction factor
    z0 <- qnorm(sum(th < th0) / length(th))

    # the acceleration factor (jackknife est.)
    th.jack <- numeric(n)
    for (i in 1:n) {
        J <- N[1:(n-1)]
        th.jack[i] <- stat(x[-i, ], J)
    }
    L <- mean(th.jack) - th.jack
    a <- sum(L^3)/(6 * sum(L^2)^1.5)

    # BCa conf. limits
    adj.alpha <- pnorm(z0 + (z0+zalpha)/(1-a*(z0+zalpha)))
    limits <- quantile(th, adj.alpha, type=6)
    return(list("est"=th0, "BCa"=limits))
}
```

Example 7.15 (BCa bootstrap confidence interval)

Compute a BCa confidence interval for the bioequivalence ratio statistic of Example 7.10 using the function `boot.BCa` provided in Example 7.14.

```
data(patch, package = "bootstrap")
n <- nrow(patch)
B <- 2000
y <- patch$y
z <- patch$z
x <- cbind(y, z)
theta.b <- numeric(B)
theta.hat <- mean(y) / mean(z)

#bootstrap
for (b in 1:B) {
    i <- sample(1:n, size = n, replace = TRUE)
    y <- patch$y[i]
    z <- patch$z[i]
    theta.b[b] <- mean(y) / mean(z)
    }
#compute the BCa interval
stat <- function(dat, index) {
    mean(dat[index, 1]) / mean(dat[index, 2])  }

boot.BCa(x, th0 = theta.hat, th = theta.b, stat = stat)
```

In the result shown below, notice that the probabilities $\alpha/2 = 0.025$ and $1 - \alpha/2 = 0.975$ have been adjusted to 0.0339, and 0.9824.

```
$est
[1] -0.0713061

$BCa
 3.391094%  98.24405%
-0.2252715  0.1916788
```

Thus bioequivalence ($|\theta| \leq 0.20$) is not supported by the BCa confidence interval estimate of θ. ◇

R note 7.4 (Empirical influence values) *By default, the `type="bca"` option of the `boot.ci` function computes empirical influence values by a regression method. The method in example 7.14 corresponds to the "usual jackknife" method of computing empirical jackknife values. See [63, Ch. 5] and the code for `empinf, usual.jack`.*

Example 7.16 (BCa bootstrap confidence interval using `boot.ci`)

Compute a BCa confidence interval for the bioequivalence ratio statistic of Examples 7.5 and 7.10, using the function `boot.ci` provided in the `boot` package [34].

```
boot.obj <- boot(x, statistic = stat, R=2000)
boot.ci(boot.obj, type=c("perc", "bca"))
```

The percentile confidence interval is also given for comparison.

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

CALL : boot.ci(boot.out = boot.obj, type = c("perc", "bca"))

Intervals :
Level      Percentile            BCa
95%   (-0.2368,  0.1824 )   (-0.2221,  0.2175 )
Calculations and Intervals on Original Scale
```

◇

7.6 Application: Cross Validation

Cross validation is a data partitioning method that can be used to assess the stability of parameter estimates, the accuracy of a classification algorithm, the adequacy of a fitted model, and in many other applications. The jackknife could be considered a special case of cross validation, because it is primarily used to estimate bias and standard error of an estimator.

In building a classifier, a researcher can partition the data into training and test sets. The model is estimated using the data in the training set only, and the misclassification rate is estimated by running the classifier on the test set. Similarly, the fit of any model can be assessed by holding back a test set from the model estimation, and then using the test set to see how well the model fits the new test data.

Another version of cross validation is the "n-fold" cross validation, which partitions the data into n test sets (now test points). This "leave-one-out" procedure is like the jackknife. The data could be divided into any number K partitions, so that there are K test sets. Then the model fitting leaves out one test set in turn, so that the models are fitted K times.

Example 7.17 (Model selection)

The `ironslag` (`DAAG`) data [185] has 53 measurements of iron content by two methods, `chemical` and `magnetic` (see "iron.dat" in [126]). A scatterplot of the data in Figure 7.2 suggests that the chemical and magnetic variables are positively correlated, but the relation may not be linear. From the plot, it appears that a quadratic polynomial, or possibly an exponential or logarithmic model might fit the data better than a line.

There are several steps to model selection, but we will focus on the prediction error. The prediction error can be estimated by cross validation, without making strong distributional assumptions about the error variable.

The proposed models for predicting magnetic measurement (Y) from chemical measurement (X) are:

1. Linear: $Y = \beta_0 + \beta_1 X + \varepsilon$.
2. Quadratic: $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \varepsilon$.
3. Exponential: $\log(Y) = \log(\beta_0) + \beta_1 X + \varepsilon$.
4. Log-Log: $\log(Y) = \beta_0 + \beta_1 \log(X) + \varepsilon$.

The code to estimate the parameters of the four models follows. Plots of the predicted response with the data are also constructed for each model and shown in Figure 7.2. To display four plots use `par(mfrow=c(2,2))`.

```
library(DAAG); attach(ironslag)
a <- seq(10, 40, .1)     #sequence for plotting fits

L1 <- lm(magnetic ~ chemical)
plot(chemical, magnetic, main="Linear", pch=16)
yhat1 <- L1$coef[1] + L1$coef[2] * a
lines(a, yhat1, lwd=2)

L2 <- lm(magnetic ~ chemical + I(chemical^2))
plot(chemical, magnetic, main="Quadratic", pch=16)
yhat2 <- L2$coef[1] + L2$coef[2] * a + L2$coef[3] * a^2
lines(a, yhat2, lwd=2)

L3 <- lm(log(magnetic) ~ chemical)
plot(chemical, magnetic, main="Exponential", pch=16)
logyhat3 <- L3$coef[1] + L3$coef[2] * a
yhat3 <- exp(logyhat3)
lines(a, yhat3, lwd=2)

L4 <- lm(log(magnetic) ~ log(chemical))
plot(log(chemical), log(magnetic), main="Log-Log", pch=16)
logyhat4 <- L4$coef[1] + L4$coef[2] * log(a)
lines(log(a), logyhat4, lwd=2)
```

◇

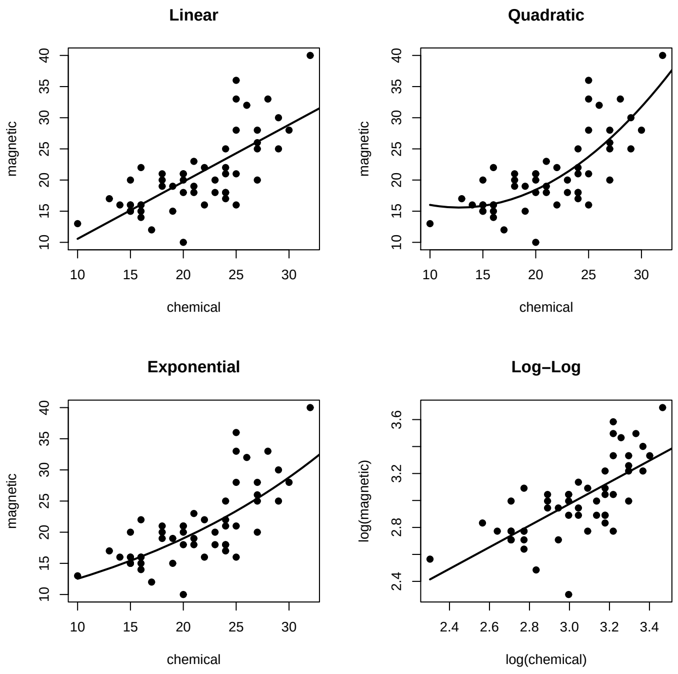


FIGURE 7.2: Four proposed models for `ironslag` data in Example 7.17.

Once the model is estimated, we want to assess the fit. Cross validation can be used to estimate the prediction errors.

Procedure to estimate prediction error by n-fold (leave-one-out) cross validation

1. For $k = 1, \ldots, n$, let observation (x_k, y_k) be the test point and use the remaining observations to fit the model.
   (a) Fit the model(s) using only the $n - 1$ observations in the training set, (x_i, y_i), $i \neq k$.
   (b) Compute the predicted response $\hat{y}_k = \hat{\beta}_0 + \hat{\beta}_1 x_k$ for the test point.
   (c) Compute the prediction error $e_k = y_k - \hat{y}_k$.
2. Estimate the mean of the squared prediction errors $\hat{\sigma}^2_\varepsilon = \frac{1}{n} \sum_{k=1}^{n} e_k^2$.

Example 7.18 (Model selection: Cross validation)

Cross validation is applied to select a model in Example 7.17.

```
n <- length(magnetic)   #in DAAG ironslag
e1 <- e2 <- e3 <- e4 <- numeric(n)

# for n-fold cross validation
# fit models on leave-one-out samples
for (k in 1:n) {
    y <- magnetic[-k]
    x <- chemical[-k]

    J1 <- lm(y ~ x)
    yhat1 <- J1$coef[1] + J1$coef[2] * chemical[k]
    e1[k] <- magnetic[k] - yhat1

    J2 <- lm(y ~ x + I(x^2))
    yhat2 <- J2$coef[1] + J2$coef[2] * chemical[k] +
            J2$coef[3] * chemical[k]^2
    e2[k] <- magnetic[k] - yhat2

    J3 <- lm(log(y) ~ x)
    logyhat3 <- J3$coef[1] + J3$coef[2] * chemical[k]
    yhat3 <- exp(logyhat3)
    e3[k] <- magnetic[k] - yhat3

    J4 <- lm(log(y) ~ log(x))
    logyhat4 <- J4$coef[1] + J4$coef[2] * log(chemical[k])
    yhat4 <- exp(logyhat4)
    e4[k] <- magnetic[k] - yhat4
}
```

The following estimates for prediction error are obtained from the n-fold cross validation.

```
> c(mean(e1^2), mean(e2^2), mean(e3^2), mean(e4^2))
[1] 19.55644 17.85248 18.44188 20.45424
```

According to the prediction error criterion, Model 2, the quadratic model, would be the best fit for the data.

```
> L2
Call:
lm(formula = magnetic ~ chemical + I(chemical^2))
Coefficients:
(Intercept)       chemical  I(chemical^2)
   24.49262       -1.39334        0.05452
```

The fitted regression equation for Model 2 is

$$\hat{Y} = 24.49262 - 1.39334X + 0.05452X^2.$$

The residual plots for Model 2 are shown in Figure 7.3. An easy way to get several residual plots is by `plot(L2)`. Alternately, similar plots can be displayed as follows.

```
par(mfrow = c(2, 2))    #layout for graphs
plot(L2$fit, L2$res)    #residuals vs fitted values
abline(0, 0)            #reference line
qqnorm(L2$res)          #normal probability plot
qqline(L2$res)          #reference line
par(mfrow = c(1, 1))    #restore display
```

Part of the summary for the fitted quadratic model is below.

```
Residuals:
    Min      1Q  Median      3Q     Max
-8.4335 -2.7006 -0.2754  2.5446 12.2665

Residual standard error: 4.098 on 50 degrees of freedom
Multiple R-Squared: 0.5931, Adjusted R-squared: 0.5768
```

In the quadratic model the predictors X and X^2 are highly correlated. See `poly` for another approach with orthogonal polynomials. ◇

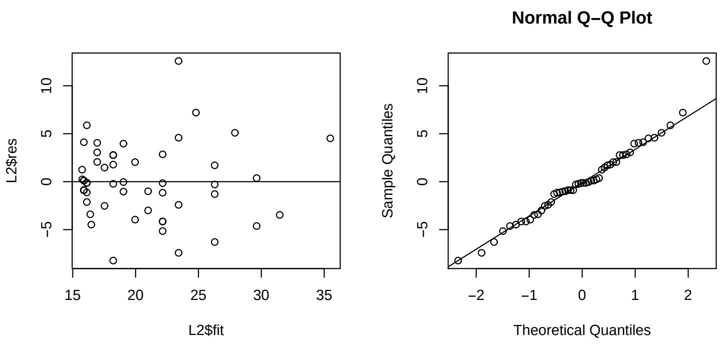


FIGURE 7.3: Residuals of the quadratic model for `ironslag` data, from Example 7.17.

Exercises

7.1 Compute a jackknife estimate of the bias and the standard error of the correlation statistic in Example 7.2.

7.2 Refer to the `law` data `(bootstrap)`. Use the jackknife-after-bootstrap method to estimate the standard error of the bootstrap estimate of $se(R)$.

7.3 Obtain a bootstrap t confidence interval estimate for the correlation statistic in Example 7.2 (`law` data in `bootstrap`).

7.4 Refer to the air-conditioning data set `aircondit` provided in the `boot` package. The 12 observations are the times in hours between failures of air-conditioning equipment [63, Example 1.1]:

$$3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487.$$

Assume that the times between failures follow an exponential model Exp(λ). Obtain the MLE of the hazard rate λ and use bootstrap to estimate the bias and standard error of the estimate.

7.5 Refer to Exercise 7.4. Compute 95% bootstrap confidence intervals for the mean time between failures $1/\lambda$ by the standard normal, basic, percentile, and BCa methods. Compare the intervals and explain why they may differ.

7.6 Efron and Tibshirani discuss the `scor (bootstrap)` test score data on 88 students who took examinations in five subjects [84, Table 7.1], [188, Table 1.2.1]. The first two tests (mechanics, vectors) were closed book and the last three tests (algebra, analysis, statistics) were open book. Each row of the data frame is a set of scores $(x_{i1}, \ldots, x_{i5})$ for the i^{th} student. Use a panel display to display the scatter plots for each pair of test scores. Compare the plot with the sample correlation matrix. Obtain bootstrap estimates of the standard errors for each of the following estimates: $\hat{\rho}_{12} = \hat{\rho}(\text{mec, vec})$, $\hat{\rho}_{34} = \hat{\rho}(\text{alg, ana})$, $\hat{\rho}_{35} = \hat{\rho}(\text{alg, sta})$, $\hat{\rho}_{45} = \hat{\rho}(\text{ana, sta})$.

7.7 Refer to Exercise 7.6. Efron and Tibshirani discuss the following example [84, Ch. 7]. The five-dimensional scores data have a 5×5 covariance matrix Σ, with positive eigenvalues $\lambda_1 > \cdots > \lambda_5$. In principal components analysis,

$$\theta = \frac{\lambda_1}{\sum_{j=1}^{5} \lambda_j}$$

measures the proportion of variance explained by the first principal component. Let $\hat{\lambda}_1 > \cdots > \hat{\lambda}_5$ be the eigenvalues of $\hat{\Sigma}$, where $\hat{\Sigma}$ is the MLE of Σ. Compute the sample estimate

$$\hat{\theta} = \frac{\hat{\lambda}_1}{\sum_{j=1}^{5} \hat{\lambda}_j}$$

of θ. Use bootstrap to estimate the bias and standard error of $\hat{\theta}$.

7.8 Refer to Exercise 7.7. Obtain the jackknife estimates of bias and standard error of $\hat{\theta}$.

7.9 Refer to Exercise 7.7. Compute 95% percentile and BCa confidence intervals for $\hat{\theta}$.

7.10 In Example 7.18, leave-one-out (n-fold) cross validation was used to select the best fitting model. Repeat the analysis replacing the Log-Log model with a cubic polynomial model. Which of the four models is selected by the cross validation procedure? Which model is selected according to maximum adjusted R^2?

7.11 In Example 7.18, leave-one-out (n-fold) cross validation was used to select the best fitting model. Use leave-two-out cross validation to compare the models.

Projects

7.A Conduct a Monte Carlo study to estimate the coverage probabilities of the standard normal bootstrap confidence interval, the basic bootstrap confidence interval, and the percentile confidence interval. Sample from a normal population and check the empirical coverage rates for the sample mean. Find the proportion of times that the confidence intervals miss on the left, and the porportion of times that the confidence intervals miss on the right.

7.B Repeat Project 7.A for the sample skewness statistic. Compare the coverage rates for normal populations (skewness 0) and $\chi^2(5)$ distributions (positive skewness).

Chapter 8

Permutation Tests

8.1 Introduction

Permutation tests are based on resampling, but unlike the ordinary bootstrap, the samples are drawn *without replacement*. Permutation tests are often applied as a nonparametric test of the general hypothesis

$$H_0 : F = G \qquad vs \qquad H_1 : F \neq G, \tag{8.1}$$

where F and G are two unspecified distributions. Under the null hypothesis, two samples from F and G, and the pooled sample, are all random samples from the same distribution F. Replicates of a two sample test statistic that compares the distributions are generated by resampling without replacement from the pooled sample. Nonparametric tests of independence, association, location, common scale, etc. can also be implemented as permutation tests. For example, in a test of multivariate independence

$$H_0 : F_{X,Y} = F_X F_Y \quad \text{vs} \quad H_1 : F_{X,Y} \neq F_X F_Y \tag{8.2}$$

under the null hypothesis the data in a sample need not be matched, and all pairs of samples obtained by permutations of the row labels (observations) of either sample are equally likely. Any statistic that measures dependence can be applied in a permutation test.

Permutation tests also can be applied to multi-sample problems, with similar methodology. For example, to test

$$H_0 : F_1 = \cdots = F_k \quad \text{vs} \quad H_1 : F_i \neq F_j \text{ for some } i, j \tag{8.3}$$

the samples are drawn without replacement from the k pooled samples. Any test statistic for the multi-sample problem can then be applied in a permutation test.

This chapter covers several applications of permutation tests for the general hypotheses (8.1) and (8.2). See Efron and Tibshirani [84, Ch. 15] or Davison and Hinkley for background, examples, and further discussion of permutation tests.

Permutation Distribution

Suppose that two independent random samples $X_1, \dots, X_n$ and $Y_1, \dots, Y_m$ are observed from the distributions F_X and F_Y, respectively. Let Z be the ordered set $\{X_1, \dots, X_n, Y_1, \dots, Y_m\}$, indexed by

$$\nu = \{1, \dots, n, n+1, \dots, n+m\} = \{1, \dots, N\}.$$

Then $Z_i = X_i$ if $1 \leq i \leq n$ and $Z_i = Y_{i-n}$ if $n+1 \leq i \leq n+m$. Let $Z^* = (X^*, Y^*)$ represent a partition of the pooled sample $Z = X \cup Y$, where X^* has n elements and Y^* has $N - n = m$ elements. Then Z^* corresponds to a permutation π of the integers ν, where $Z_i^* = Z_{\pi(i)}$. The number of possible partitions is equal to the number $\binom{N}{n}$ of different ways to select the first n indices of $\pi(\nu)$, hence there are $\binom{N}{n}$ different ways to partition the pooled sample Z into two subsets of size n and m.

The Permutation Lemma [84, p. 207] states that under $H_0 : F_X = F_Y$, a randomly selected Z^* has probability

$$\frac{1}{\binom{N}{n}} = \frac{n!\, m!}{N!}$$

of equaling any of its possible values. That is, if $F_X = F_Y$ then all permutations are equally likely.

If $\hat\theta(X, Y) = \hat\theta(Z, \nu)$ is a statistic, then the *permutation distribution* of $\hat\theta^*$ is the distribution of the replicates

$$\begin{aligned} \{\hat\theta^*\} &= \left\{\hat\theta(Z,\, \pi_j(\nu)),\, j = 1, \dots, \binom{N}{n}\right\} \\ &= \{\hat\theta^{(j)} \mid \pi_j(\nu) \text{ is a permutation of } \nu\}. \end{aligned}$$

The cdf of $\hat\theta^*$ is given by

$$F_{\theta^*}(t) = P(\hat\theta^* \leq t) = \binom{N}{n}^{-1} \sum_{j=1}^{N} I(\hat\theta^{(j)} \leq t). \tag{8.4}$$

Thus, if $\hat\theta$ is applied to test a hypothesis and large values of $\hat\theta$ are significant, then the permutation test rejects the null hypothesis when $\hat\theta$ is large relative to the distribution of the permutation replicates. The achieved significance level (ASL) of the observed statistic $\hat\theta$ is the probability

$$P(\hat\theta^* \geq \hat\theta) = \binom{N}{n}^{-1} \sum_{j=1}^{N} I(\hat\theta^{(j)} \geq \hat\theta),$$

where $\hat\theta = \hat\theta(Z, \nu)$ is the statistic computed on the observed sample. The ASL for a lower-tail or two-tail test based on $\hat\theta$ is computed in a similar way.

In practice, unless the sample size is very small, evaluating the test statistic for all of the $\binom{N}{n}$ permutations is computationally excessive. An approximate permutation test is implemented by randomly drawing a large number of samples without replacement.

Approximate permutation test procedure

1. Compute the observed test statistic $\hat{\theta}(X, Y) = \hat{\theta}(Z, \nu)$.
2. For each replicate, indexed $b = 1, \dots, B$:
   (a) Generate a random permutation $\pi_b = \pi(\nu)$.
   (b) Compute the statistic $\hat{\theta}^{(b)} = \hat{\theta}^*(Z, \pi_b)$.
3. If large values of $\hat{\theta}$ support the alternative, compute the ASL (the empirical p-value) by

$$\hat{p} = \frac{1 + \#\{\hat{\theta}^{(b)} \geq \hat{\theta}\}}{B+1} = \frac{\left\{1 + \sum_{b=1}^{B} I(\hat{\theta}^{(b)} \geq \hat{\theta})\right\}}{B+1}.$$

   For a lower-tail or two-tail test $\hat{p}$ is computed in a similar way.
4. Reject H_0 at significance level α if $\hat{p} \leq \alpha$.

The formula for $\hat{p}$ is given by Davison and Hinkley [63, p. 159], who state that "at least 99 and at most 999 random permutations should suffice."

Methods for implementing an approximate permutation test are illustrated in the examples that follow. Although the `boot` function [34] can be used to generate the replicates, it is not necessary to use `boot`. For a multivariate permutation test using `boot` see the examples in Section 8.3.

Example 8.1 (Permutation distribution of a statistic)

The permutation distribution of a statistic is illustrated for a small sample, from the `chickwts` data in R. Weights in grams are recorded, for six groups of newly hatched chicks fed different supplements. There are six types of feed supplements. A quick graphical summary of the data can be displayed by `boxplot(formula(chickwts))`. The plot (not shown) suggests that soybean and linseed groups may be similar. The distribution of weights for these two groups are compared below.

```
attach(chickwts)
x <- sort(as.vector(weight[feed == "soybean"]))
y <- sort(as.vector(weight[feed == "linseed"]))
detach(chickwts)
```

The ordered chick weights for the two samples are

```
X: 158 171 193 199 230 243 248 248 250 267 271 316 327 329
Y: 141 148 169 181 203 213 229 244 257 260 271 309
```

The groups can be compared in several ways. For example, sample means, sample medians, or other trimmed means can be compared. More generally, one can ask whether the distributions of the two variables differ and compare the groups by any statistic that measures a distance between two samples.

Consider the sample mean. If the two samples are drawn from normal populations with equal variances, we can apply the two-sample t-test. The sample means are $\overline{X} = 246.4286$ and $\overline{Y} = 218.7500$. The two sample t statistic is $T = 1.3246$. In this problem, however, the distributions of the weights are unknown. The achieved significance level of T can be computed from the permutation distribution without requiring distributional assumptions.

The sample sizes are $n = 14$ and $m = 12$, so there are a total of

$$\binom{n+m}{n} = \binom{26}{14} = \frac{26!}{14!\,12!} = 9,657,700$$

different partitions of the pooled sample into two subsets of size 14 and 12. Thus, even for small samples, enumerating all possible partitions of the pooled sample is not practical. An alternate approach is to generate a large number of the permutation samples, to obtain the approximate permutation distribution of the replicates. Draw a random sample of n indices from 1:N without replacement, which determines a randomly selected partition (X^*, Y^*). In this way we can generate a large number of the permutation samples. Then compare the observed statistic T to the replicates T^*.

The approximate permutation test procedure is illustrated below with the two-sample t statistic.

```
R <- 999              #number of replicates
z <- c(x, y)          #pooled sample
K <- 1:26
reps <- numeric(R)    #storage for replicates
t0 <- t.test(x, y)$statistic

for (i in 1:R) {
    #generate indices k for the first sample
    k <- sample(K, size = 14, replace = FALSE)
    x1 <- z[k]
    y1 <- z[-k]      #complement of x1
    reps[i] <- t.test(x1, y1)$statistic
    }
p <- mean(c(t0, reps) >= t0)

> p
[1] 0.101
```

The value of $\hat{p}$ is the proportion of replicates T^* that are at least as large as the observed test statistic (an approximate p-value). For a two-tail test the ASL is $2\hat{p}$ if $\hat{p} \leq 0.5$ (it is $2(1 - \hat{p})$ if $\hat{p} > 0.5$). The ASL is 0.202 so the null hypothesis is not rejected. For comparison, the two-sample t-test reports p-value $= 0.198$. A histogram of the replicates of T is displayed by

```
hist(reps, main = "", freq = FALSE, xlab = "T (p = 0.202)",
    breaks = "scott")
points(t0, 0, cex = 1, pch = 16)      #observed T
```

which is shown in Figure 8.1. ◇

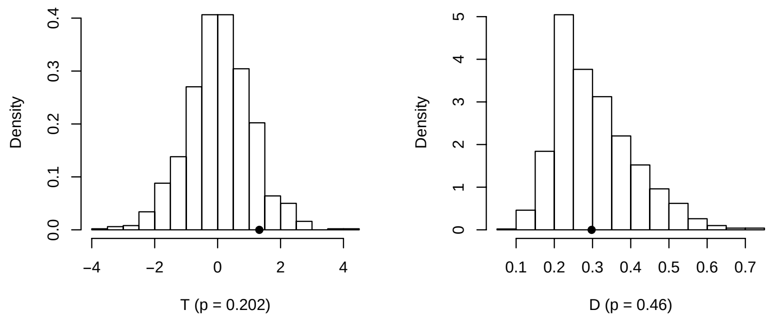


FIGURE 8.1: Permutation distribution of replicates in Example 8.1 (left) and Example 8.2 (right).

8.2 Tests for Equal Distributions

Suppose that $X = (X_1, \ldots, X_n)$ and $Y = (Y_1, \ldots, Y_m)$ are independent random samples from distributions F and G respectively, and we wish to test the hypothesis $H_0 : F = G$ vs the alternative $H_1 : F \neq G$. Under the null hypothesis, samples X, Y, and the pooled sample $Z = X \cup Y$, are all random samples from the same distribution F. Moreover, under H_0, any subset X^* of size n from the pooled sample, and its complement Y^*, also represent independent random samples from F.

Suppose that $\hat{\theta}$ is a two-sample statistic that measures the distance in some sense between F and G. Without loss of generality, we can suppose that large values of $\hat{\theta}$ support the alternative $F \neq G$. By the permutation lemma, under the null hypothesis all values of $\hat{\theta}^* = \hat{\theta}(X^*, Y^*)$ are equally likely. The permutation distribution of $\hat{\theta}^*$ is given by (8.4), and an exact permutation test or the approximate permutation test procedure given on page 217 can be applied.

Two-sample tests for univariate data

To apply a permutation test of equal distributions, choose a test statistic that measures the difference between two distributions. For example, the two-sample Kolmogorov-Smirnov (K-S) statistic or the two-sample Cramér-von Mises statistic can be applied in the univariate case. Many other statistics are in the literature, although the K-S statistic is one of the most widely applied for univariate distributions. It is applied in the following example.

Example 8.2 (Permutation distribution of the K-S statistic)

In Example 8.1 the means of the soybean and linseed groups were compared. Suppose now that we are interested in testing for any type of difference in the two groups. The hypotheses of interest are $H_0 : F = G$ vs $H_1 : F \neq G$, where F is the distribution of weight of chicks fed soybean supplements and G is the distribution of weight of chicks fed linseed supplements. The Kolmogorov-Smirnov statistic D is the maximum absolute difference between the ecdf's of the two samples, defined by

$$D = \sup_{1 \leq i \leq N} |F_n(z_i) - G_m(z_i)|,$$

where F_n is the ecdf of the first sample $x_1, \ldots, x_n$ and G_m is the ecdf of the second sample $y_1, \ldots, y_m$. Note that $0 \leq D \leq 1$ and large values of D support the alternative $F \neq G$. The observed value of $D = D(X, Y) = 0.2976190$ can be computed using `ks.test`. To determine whether this value of D is strong evidence for the alternative, we compare D with the replicates $D^* = D(X^*, Y^*)$.

```
R <- 999               #number of replicates
z <- c(x, y)           #pooled sample
K <- 1:26
D <- numeric(R)        #storage for replicates
options(warn = -1)
D0 <- ks.test(x, y, exact = FALSE)$statistic
for (i in 1:R) {
    #generate indices k for the first sample
    k <- sample(K, size = 14, replace = FALSE)
    x1 <- z[k]
    y1 <- z[-k]        #complement of x1
    D[i] <- ks.test(x1, y1, exact = FALSE)$statistic
    }
```

```
p <- mean(c(D0, D) >= D0)
options(warn = 0)
> p
[1] 0.46
```

The approximate ASL 0.46 does not support the alternative hypothesis that distributions differ. A histogram of the replicates of D is displayed by

```
hist(D, main = "", freq = FALSE, xlab = "D (p = 0.46)",
    breaks = "scott")
points(D0, 0, cex = 1, pch = 16)      #observed D
```

which is shown in Figure 8.1. ◇

R note 8.1 *In Example 8.2 the Kolmogorov-Smirnov test* `ks.test` *generates a warning each time it tries to compute a p-value, because there are ties in the data. We are not using the p-value, so it is safe to ignore these warnings. Display of warnings or messages at the console regarding warnings can be suppressed by* `options(warn = -1)`*. The default value is* `warn = 0`*.*

Example 8.3 (Two-sample K-S test)

Test whether the distributions of chick weights for the sunflower and linseed groups differ. The K-S test can be applied as in Example 8.2.

```
attach(chickwts)
x <- sort(as.vector(weight[feed == "sunflower"]))
y <- sort(as.vector(weight[feed == "linseed"]))
detach(chickwts)
```

The sample sizes are $n = m = 12$, and the observed K-S test statistic is $D = 0.8333$. The summary statistics below suggest that the distributions of weights for these two groups may differ.

```
> summary(cbind(x, y))
       x               y
Min.   :226.0   Min.   :141.0
1st Qu.:312.8   1st Qu.:178.0
Median :328.0   Median :221.0
Mean   :328.9   Mean   :218.8
3rd Qu.:340.2   3rd Qu.:257.8
Max.   :423.0   Max.   :309.0
```

Repeating the simulation in Example 8.2 with the sunflower sample replacing the soybean sample produces the following result.

```
p <- mean(c(D0, D) >= D0)
> p
[1] 0.001
```

Thus, none of the replicates are as large as the observed test statistic. Here the sample evidence supports the alternative hypothesis that the distributions differ. ◇

Another univariate test for the two-sample problem is the Cramér-von Mises test [56, 281]. The Cramér-von Mises statistic, which estimates the integrated squared distance between the distributions, is defined by

$$W_2 = \frac{mn}{(m+n)^2}\left[\sum_{i=1}^{n}(F_n(x_i) - G_m(x_i))^2 + \sum_{j=1}^{m}(F_n(y_j) - G_m(y_j))^2\right],$$

where F_n is the ecdf of the sample $x_1, \dots, x_n$ and G_m is the ecdf of the sample $y_1, \dots, y_m$. Large values of W_2 are significant. The implementation of the Cramér-von Mises test is left as an exercise.

The multivariate tests discussed in the next section can also be applied for testing $H_0 : F = G$ in the univariate case.

8.3 Multivariate Tests for Equal Distributions

Classical approaches to the two-sample problem in the univariate case based on comparing empirical distribution functions, such as the Kolmogorov–Smirnov and Cramér-von Mises tests, do not have a natural distribution free extension to the multivariate case. Multivariate tests based on maximum likelihood depend on distributional assumptions about the underlying populations. Hence although likelihood tests may apply in special cases, they do not apply to the general two-sample or k-sample problem, and may not be robust to departures from these assumptions.

Many of the procedures that are available for the multivariate two-sample problem (8.1) require a computational approach for implementation. Bickel [27] constructed a consistent distribution free multivariate extension of the univariate Smirnov test by conditioning on the pooled sample. Friedman and Rafsky [101] proposed distribution free multivariate generalizations of the Wald-Wolfowitz runs test and Smirnov test for the two-sample problem, based on the minimal spanning tree of the pooled sample. A class of consistent, asymptotically distribution free tests for the multivariate problem is based on nearest neighbors [28, 139, 240]. The nearest neighbor tests apply to testing the k-sample hypothesis when all distributions are continuous. A multivariate nonparametric test for equal distributions was developed independently by Baringhaus and Franz [20] and Székely and Rizzo [261, 262], which is implemented as an approximate permutation test. We will discuss the latter two, the nearest neighbor tests and the energy test [226, 261].

In the following sections multivariate samples will be denoted by boldface type. Suppose that

$$\mathbf{X} = \{X_1, \dots, X_{n_1}\} \in \mathbb{R}^d, \quad \mathbf{Y} = \{Y_1, \dots, Y_{n_2}\} \in \mathbb{R}^d,$$

are independent random samples, $d \geq 1$. The pooled data matrix is $\mathbf{Z}$, an $n \times d$ matrix with observations in rows:

$$\mathbf{Z}_{n\times d} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & & \vdots \\ x_{n_1,1} & x_{n_1,2} & \cdots & x_{n_1,d} \\ y_{1,1} & y_{1,2} & \cdots & y_{1,d} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,d} \\ \vdots & \vdots & & \vdots \\ y_{n_2,1} & y_{n_2,2} & \cdots & y_{n_2,d} \end{bmatrix}, \tag{8.5}$$

where $n = n_1 + n_2$.

Nearest neighbor tests

A multivariate test for equal distributions is based on nearest neighbors. The nearest neighbor (NN) tests are a type of test based on ordered distances between sample elements, which can be applied when the distributions are continuous.

Usually the distance is the Euclidean norm $\|z_i - z_j\|$. The NN tests are based on the first through r^{th} nearest neighbor coincidences in the pooled sample. Consider the simplest case, $r = 1$. For example, if the observed samples are the weights in Example 8.3

```
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
x  423  340  392  339  341  226  320  295  334   322   297   318
y  309  229  181  141  260  203  148  169  213   257   244   271
```

then the first nearest neighbor of $x_1 = 423$ is $x_3 = 392$, which are in the same sample. The first nearest neighbor of $x_6 = 226$ is $y_2 = 229$, in different samples. In general, if the sampled distributions are equal, then the pooled sample has on average less nearest neighbor coincidences than under the alternative hypothesis. In this example, most of the nearest neighbors are found in the same sample.

Let $\mathbf{Z} = \{X_1, \dots, X_{n_1}, Y_1, \dots, Y_{n_2}\}$ as in (8.5). Denote the first nearest neighbor of Z_i by $NN_1(Z_i)$. Count the number of first nearest neighbor coincidences by the indicator function $I_i(1)$, which is defined by

$$I_i(1) = 1 \text{ if } Z_i \text{ and } NN_1(Z_i) \text{ belong to the same sample;}$$
$$I_i(1) = 0 \text{ if } Z_i \text{ and } NN_1(Z_i) \text{ belong to different samples.}$$

The first nearest neighbor statistic is the proportion of first nearest neighbor coincidences

$$T_{n,1} = \frac{1}{n} \sum_{i=1}^{n} I_i(1),$$

where $n = n_1 + n_2$. Large values of $T_{n,1}$ support the alternative hypothesis that the distributions differ.

Similarly, denote the second nearest neighbor of a sample element Z_i by $NN_2(Z_i)$ and define the indicator function $I_i(2)$, which is 1 if $NN_2(Z_i)$ is in the same sample as Z_i and otherwise $I_i(2) = 0$. The second nearest neighbor statistic is based on the first and second nearest neighbor coincidences, defined by

$$T_{n,2} = \frac{1}{2n} \sum_{i=1}^{n} (I_i(1) + I_i(2)).$$

In general, the r^{th} nearest neighbor of Z_i is defined to be the sample element Z_j satisfying $\|Z_i - Z_\ell\| < \|Z_i - Z_j\|$ for exactly $r-1$ indices $1 \le \ell \le n$, $\ell \ne i$. Denote the r^{th} nearest neighbor of a sample element Z_i by $NN_r(Z_i)$. For $i = 1, \dots, n$ the indicator function $I_i(r)$ is defined by $I_i(r) = 1$ if Z_i and $NN_r(Z_i)$ belong to the same sample, and otherwise $I_i(r) = 0$. The J^{th} nearest neighbor statistic measures the proportion of first through J^{th} nearest neighbor coincidences:

$$T_{n,J} = \frac{1}{nJ} \sum_{i=1}^{n} \sum_{r=1}^{J} I_i(r). \tag{8.6}$$

Under the hypothesis of equal distributions, the pooled sample has on average less nearest neighbor coincidences than under the alternative hypothesis, so the test rejects the null hypothesis for large values of $T_{n,J}$. Henze [139] proved that the limiting distribution of a class of nearest neighbor statistics is normal for any distance generated by a norm on $\mathbb{R}^d$. Schilling [240] derived the mean and variance of the distribution of $T_{n,2}$ for selected values of n_1/n and d in the case of Euclidean norm. In general, the parameters of the normal distribution may be difficult to obtain analytically. If we condition on the pooled sample to implement an exact permutation test, the procedure is distribution free. The test can be implemented as an approximate permutation test, following the procedure outlined on page 217.

Remark 8.1 *Nearest neighbor statistics are functions of the ordered distances between sample elements. The sampled distributions are assumed to be continuous, so there are no ties. Thus, resampling without replacement is the correct resampling method and the permutation test rather than the ordinary bootstrap should be applied. In the ordinary bootstrap, many ties would occur in the bootstrap samples.*

Searching for nearest neighbors is not a trivial computational problem, but fast algorithms have been developed [25, 12, 13]. A fast nearest neighbor method `nn` is available in the `knnFinder` package. The algorithm uses a kd-tree. According to the package author Kemp [160], "The advantage of the kd-tree is that it runs in $O(M \log M)$ time ... where M is the number of data points using Bentley's kd-tree."

Example 8.4 (Finding nearest neighbors)

The following numerical example illustrates the usage of the `nn (knnFinder)` [160] function as a method to find the indices of the first through r^{th} nearest neighbors. The pooled data matrix $\mathbf{Z}$ is assumed to be in the layout (8.5).

```
library(knnFinder)  #for nn function

#generate a small multivariate data set
x <- matrix(rnorm(12), 3, 4)
y <- matrix(rnorm(12), 3, 4)

z <- rbind(x, y)
o <- rep(0, nrow(z))

DATA <- data.frame(cbind(z, o))
NN <- nn(DATA, p = nrow(z)-1)
```

In the distance matrix below, for example, the first through fifth nearest neighbors of Z_1 are Z_4, Z_2, Z_3, Z_5, Z_6.

```
> D <- dist(z)
> round(as.matrix(D), 2)

     1    2    3    4    5    6
1 0.00 2.29 2.69 1.88 2.87 3.94
2 2.29 0.00 2.60 3.27 2.45 3.69
3 2.69 2.60 0.00 2.14 0.43 3.52
4 1.88 3.27 2.14 0.00 2.52 3.66
5 2.87 2.45 0.43 2.52 0.00 3.48
6 3.94 3.69 3.52 3.66 3.48 0.00
```

The index matrix returned by the function `nn` identifies the nearest neighbors as follows. The i^{th} row of `$nn.idx` on the next page contains the subscripts (indices) of $NN_1(Z_i)$, $NN_2(Z_i), \ldots$, the nearest neighbors of Z_i. According to the first row of the index matrix `$nn.idx`, the indices of the first through fifth nearest neighbors of Z_1 are 4, 2, 3, 5, and 6, respectively.

```
> NN$nn.idx
  X1 X2 X3 X4 X5
1  4  2  3  5  6
2  1  5  3  4  6
3  5  4  2  1  6
4  1  3  5  2  6
5  3  2  4  1  6
6  5  3  4  2  1

> round(NN$nn.dist, 2)
    X1   X2   X3   X4   X5
1 1.88 2.29 2.69 2.87 3.94
2 2.29 2.45 2.60 3.27 3.69
3 0.43 2.14 2.60 2.69 3.52
4 1.88 2.14 2.52 3.27 3.66
5 0.43 2.45 2.52 2.87 3.48
6 3.48 3.52 3.66 3.69 3.94
```

In this small data set it is easy to compute the nearest neighbor statistics. For example, $T_{n,1} = 2/6 \doteq 0.333$ and

$$T_{n,2} = \frac{1}{2n}\sum_{i=1}^{n}(I_i(1) + I_i(2)) = \frac{1}{12}(2+1) = 0.25.$$

◇

Example 8.5 (Nearest neighbor statistic)

In this example a method of computing nearest neighbor statistics from the result of `nn (knnFinder)` is shown. Compute $T_{n,3}$ for the `chickwts` data from Example 8.3.

```
library(knnFinder)
attach(chickwts)
x <- as.vector(weight[feed == "sunflower"])
y <- as.vector(weight[feed == "linseed"])
detach(chickwts)

z <- c(x, y)
o <- rep(0, length(z))
z <- as.data.frame(cbind(z, o))
NN <- nn(z, p=3)
```

The data and the index matrix `NN$nn.idx` are shown on the facing page.

```
pooled sample    $nn.idx
      [,1]          X1 X2 X3
 [1,]  423        1   3  5  2
 [2,]  340        2   4  5  9
 [3,]  392        3   1  5  2
 [4,]  339        4   2  5  9
 [5,]  341        5   2  4  9
 [6,]  226        6  14 21 23
 [7,]  320        7  12 10 13
 [8,]  295        8  11 13 12
 [9,]  334        9   4  2  5
[10,]  322        10  7 12  9
[11,]  297        11  8 13 12
[12,]  318        12  7 10 13      I=1 if index <= 12
--------------------------------------------------------
[13,]  309        13 12  7 11      I=1 if index > 12
[14,]  229        14  6 23 21
[15,]  181        15 20 18 21
[16,]  141        16 19 20 15
[17,]  260        17 22 24 23
[18,]  203        18 21 15  6
[19,]  148        19 16 20 15
[20,]  169        20 15 19 16
[21,]  213        21 18  6 14
[22,]  257        22 17 23 24
[23,]  244        23 22 14 17
[24,]  271        24 17 22  8
```

The first three nearest neighbors of each sample element Z_i are in the i^{th} row. In the first block, count the number of entries that are between 1 and $n_1 = 12$. In the second block, count the number of entries that are between $n_1 + 1 = 13$ and $n_1 + n_2 = 24$.

```
block1 <- NN$nn.idx[1:12, ]
block2 <- NN$nn.idx[13:24, ]
i1 <- sum(block1 < 12.5)
i2 <- sum(block2 > 12.5)

> c(i1, i2)
[1] 29 29
```

Then

$$T_{n,3} = \frac{1}{3n} \sum_{i=1}^{n} \sum_{j=1}^{3} I_i(j) = \frac{1}{3(24)}(29 + 29) = \frac{58}{72} = 0.8055556.$$

◇

Example 8.6 (Nearest neighbor test)

The permutation test for $T_{n,3}$ in Example 8.5 can be applied using the `boot` function in the `boot` package [34] as follows.

```
library(boot)
Tn3 <- function(z, ix, sizes) {
    n1 <- sizes[1]
    n2 <- sizes[2]
    n <- n1 + n2
    z <- z[ix, ]
    o <- rep(0, NROW(z))
    z <- as.data.frame(cbind(z, o))
    NN <- nn(z, p=3)
    block1 <- NN$nn.idx[1:n1, ]
    block2 <- NN$nn.idx[(n1+1):n, ]
    i1 <- sum(block1 < n1 + .5)
    i2 <- sum(block2 > n1 + .5)
    return((i1 + i2) / (3 * n))
}
N <- c(12, 12)
boot.obj <- boot(data = z, statistic = Tn3,
    sim = "permutation", R = 999, sizes = N)
```

Note: The permutation samples can also be generated by the `sample` function. The result of the simulation is

```
> boot.obj
DATA PERMUTATION
Call: boot(data = z, statistic = Tn3, R = 999,
    sim = "permutation", sizes = N)

Bootstrap Statistics :
     original     bias    std. error
t1* 0.8055556 -0.3260066  0.07275428
```

The output from `boot` does not include a p-value, of course, because `boot` has no way of knowing what hypotheses are being tested. What is printed at the console is a summary of the `boot` object. The `boot` object itself is a list that contains several things including the permutation replicates of the test statistic. The test decision can be obtained from the observed statistic in `$t0` and the replicates in `$t`.

```
> tb <- c(boot.obj$t, boot.obj$t0)
> mean(tb >= boot.obj$t0)
[1] 0.001
```

The ASL is $\hat{p} = 0.001$, so the hypothesis of equal distributions is rejected. The histogram of replicates of $T_{n,3}$ is shown in Figure 8.2.

```
hist(tb, freq=FALSE, main="",
     xlab="replicates of T(n,3) statistic")
points(boot.obj$t0, 0, cex=1, pch=16)
```

◇

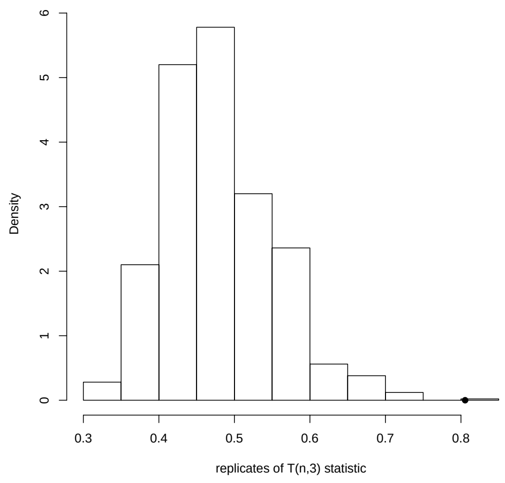


FIGURE 8.2: Permutation distribution of $T_{n,3}$ in Example 8.6.

The multivariate r^{th} nearest neighbor test can be implemented by an approximate permutation test. The steps are to write a general function that computes the statistic $T_{n,r}$ for any given (n_1, n_2, r) and permutation of the row indices of the pooled sample. Then apply `boot` or generate permutations using `sample`, similar to the implementation of the permutation test shown in Example 8.6.

Energy test for equal distributions

The *energy* distance or e-distance statistic $\mathcal{E}_n$ is defined by

$$\mathcal{E}_n = e(\mathbf{X}, \mathbf{Y}) = \frac{n_1 n_2}{n_1 + n_2} \Bigg(\frac{2}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \|X_i - Y_j\| - \frac{1}{n_1^2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_1} \|X_i - X_j\| - \frac{1}{n_2^2} \sum_{i=1}^{n_2} \sum_{j=1}^{n_2} \|Y_i - Y_j\| \Bigg). \quad (8.7)$$

On the name "energy" and concept of energy statistics in general see [258, 259]. The non-negativity of $e(\mathbf{X}, \mathbf{Y})$ is a special case of the following inequality. If X, X', Y, Y' are independent random vectors in $\mathbb{R}^d$ with finite expectations, $X \overset{D}{=} X'$ and $Y \overset{D}{=} Y'$, then

$$2E\|X - Y\| - E\|X - X'\| - E\|Y - Y'\| \geq 0, \tag{8.8}$$

and equality holds if and only if X and Y are identically distributed [262, 263]. The $\mathcal{E}$ distance between the distribution of X and Y is

$$\mathcal{E}(X, Y) = 2E\|X - Y\| - E\|X - X'\| - E\|Y - Y'\|$$

and the empirical distance $\mathcal{E}_n = e(\mathbf{X}, \mathbf{Y})$ is a constant times the plug-in estimator of $\mathcal{E}(X, Y)$.

Clearly large e-distance corresponds to different distributions, and measures the distance between distributions in a similar sense as the univariate empirical distribution function (edf) statistics. In contrast to edf statistics, however, e-distance does not depend on the notion of a sorted list, and e-distance is by definition a multivariate measure of distance between distributions.

If X and Y are not identically distributed, and $n = n_1 + n_2$, then $E[\mathcal{E}_n]$ is asymptotically a positive constant times n. As the sample size n tends to infinity, under the null hypothesis $E[\mathcal{E}_n]$ tends to a positive constant, while under the alternative hypothesis $E[\mathcal{E}_n]$ tends to infinity. Not only the expected value of $\mathcal{E}_n$, but $\mathcal{E}_n$ itself, converges (in distribution) under the null hypothesis, and tends to infinity (stochastically) otherwise. A test for equal distributions based on $\mathcal{E}_n$ is universally consistent against all alternatives with finite first moments [261, 262]. The asymptotic distribution of $\mathcal{E}_n$ is a quadratic form of centered Gaussian random variables, with coefficients that depend on the distributions of X and Y.

To implement the test, suppose that $\mathbf{Z}$ is the $n \times d$ data matrix of the pooled sample as in (8.5). The permutation operation is applied to the row indices of $\mathbf{Z}$. The calculation of the test statistic has $O(n^2)$ time complexity, where $n = n_1 + n_2$ is the size of the pooled sample. (In the univariate case the statistic can be written as a linear combination of the order statistics, with $O(n \log n)$ complexity.)

Example 8.7 (Two-sample energy statistic)

The approximate permutation energy test is implemented in `eqdist.etest` in the `energy` package [226]. However, in order to illustrate the details of the implementation for a multivariate permutation test, we provide an R version below. Note that the `energy` implementation is considerably faster than the example below, because in `eqdist.etest` the calculation of the test statistic is implemented in an external C library.

The $\mathcal{E}_n$ statistic is a function of the pairwise distances between sample elements. The distances remain invariant under any permutation of the indices, so it is *not* necessary to recalculate distances for each permutation sample.

However, it *is* necessary to provide a method for looking up the correct distance in the original distance matrix given the permutation of indices.

```
edist.2 <- function(x, ix, sizes) {
    # computes the e-statistic between 2 samples
    # x:          Euclidean distances of pooled sample
    # sizes:      vector of sample sizes
    # ix:         a permutation of row indices of x

    dst <- x
    n1 <- sizes[1]
    n2 <- sizes[2]
    ii <- ix[1:n1]
    jj <- ix[(n1+1):(n1+n2)]
    w <- n1 * n2 / (n1 + n2)

    # permutation applied to rows & cols of dist. matrix
    m11 <- sum(dst[ii, ii]) / (n1 * n1)
    m22 <- sum(dst[jj, jj]) / (n2 * n2)
    m12 <- sum(dst[ii, jj]) / (n1 * n2)
    e <- w * ((m12 + m12) - (m11 + m22))
    return (e)
}
```

Below, the simulated samples in $\mathbb{R}^d$ are generated from distributions that differ in location. The first distribution is centered at $\mu_1 = (0, \ldots, 0)^T$ and the second distribution is centered at $\mu_2 = (a, \ldots, a)^T$.

```
d <- 3
a <- 2 / sqrt(d)
x <- matrix(rnorm(20 * d), nrow = 20, ncol = d)
y <- matrix(rnorm(10 * d, a, 1), nrow = 10, ncol = d)
z <- rbind(x, y)
dst <- as.matrix(dist(z))

> edist.2(dst, 1:30, sizes = c(20, 10))
[1] 9.61246
```

The observed value of the test statistic is $\mathcal{E}_n = 9.61246$. ◇

The function `edist.2` is designed to be used with the `boot (boot)` function [34] to perform the permutation test. Alternately, generate the permutation vectors `ix` using the `sample` function. The `boot` method is shown in the following example.

Example 8.8 (Two-sample energy test)

This example shows how to apply the `boot` function to perform an approximate permutation test using a multivariate test statistic function. Apply the permutation test to the data matrix `z` in Example 8.7.

```
library(boot)  #for boot function
dst <- as.matrix(dist(z))
N <- c(20, 10)

boot.obj <- boot(data = dst, statistic = edist.2,
    sim = "permutation", R = 999, sizes = N)

> boot.obj

DATA PERMUTATION

Call: boot(data = dst, statistic = edist.2, R = 999,
      sim = "permutation", sizes = N)

Bootstrap Statistics :
    original    bias    std. error
t1*  9.61246 -7.286621    1.025068
```

The permutation vectors generated by `boot` will have the same length as the `data` argument. If `data` is a vector then the permutation vector generated by `boot` will have length equal to the `data` vector. If `data` is a matrix, then the permutation vector will have length equal to the number of rows of the matrix. For this reason, it is necessary to convert the `dist` object to an $n \times n$ distance matrix.

The ASL is computed from the replicates in the bootstrap object.

```
e <- boot.obj$t0
tb <- c(e, boot.obj$t)
mean(tb >= e)
[1] 0.001

hist(tb, main = "", breaks="scott", freq=FALSE,
    xlab="Replicates of e")
points(e, 0, cex=1, pch=16)
```

None of the replicates exceed the observed value 9.61246 of the test statistic. The approximate achieved significance level is 0.001, and we reject the hypothesis of equal distributions. Replicates of $\mathcal{E}_n$ are shown in Figure 8.3(a).

The large estimate of bias reported by the boot function gives an indication that the test statistic is large, because $\mathcal{E}(X, Y) \geq 0$ and $\mathcal{E}(X, Y) = 0$ if and only if the sampled distributions are equal.

Finally, let us check the result of the test when the sampled distributions are identical.

```
    d <- 3
    a <- 0
    x <- matrix(rnorm(20 * d), nrow = 20, ncol = d)
    y <- matrix(rnorm(10 * d, a, 1), nrow = 10, ncol = d)
    z <- rbind(x, y)
    dst <- as.matrix(dist(z))

    N <- c(20, 10)
    dst <- as.matrix(dist(z))
    boot.obj <- boot(data = dst, statistic = edist.2,
        sim="permutation", R=999, sizes=N)
    > boot.obj
    ...
    Bootstrap Statistics :
        original    bias    std. error
    t1* 1.664265 0.7325929    1.051064

    e <- boot.obj$t0
    E <- c(boot.obj$t, e)

    mean(E >= e)
[1] 0.742
    hist(E, main = "", breaks="scott",
        xlab="Replicates of e", freq=FALSE)
    points(e, 0, cex=1, pch=16)
```

In the second example the approximate achieved significance level is 0.742 and the hypothesis of equal distributions is not rejected. Notice that the estimate of bias here is small. The histogram of replicates is shown in Figure 8.3(b). ◇

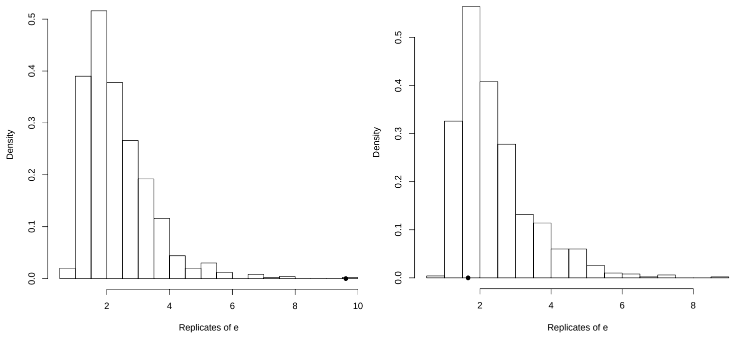
The $\mathcal{E}$ distance and two-sample e-statistic $\mathcal{E}_n$ are easily generalized to the k-sample problem. See e.g. the function `edist (energy)`, which returns a dissimilarity object like the `dist` object.

Example 8.9 (k-sample energy distances)

The function `edist.2` in Example 8.7 is a two-sample version of the function `edist` in the `energy` package [226], which summarizes the empirical $\mathcal{E}$-distances between $k \geq 2$ samples. The syntax is

```
    edist(x, sizes, distance=FALSE, ix=1:sum(sizes), alpha=1)
```

The argument `alpha` is an exponent $0 < \alpha \leq 2$ on the Euclidean distance.



(a) (b)

FIGURE 8.3: Permutation distribution of the two-sample e-statistic replicates in Example 8.7.

It can be shown that for all $0 < \alpha < 2$ the corresponding $e^{(\alpha)}$-distance determines a statistically consistent test of equal distributions for all random vectors with finite first moments [262].

Consider the four-dimensional iris data. Compute the e-distance matrix for the three species of iris.

```
library(energy)  #for edist
z <- iris[ , 1:4]
dst <- dist(z)

> edist(dst, sizes = c(50, 50, 50), distance = TRUE)
          1         2
2 123.55381
3 195.30396  38.85415
```

A test for the k-sample hypothesis of equal distributions is based on k-sample e-distances with a suitable weight function. ◇

Comparison of nearest neighbor and energy tests

Example 8.10 (Power comparison)

In a simulation experiment, we compared the empirical power of the third nearest neighbor test based on $T_{n,3}$ (8.6) and the energy test based on $\mathcal{E}_n$ (8.7). The distributions compared,

$$F_1 = N_2(\mu = (0,0)^2, \Sigma = I_2), \qquad F_2 = N_2(\mu = (0,\delta)^T, \Sigma = I_2),$$

differ in location. The empirical power was estimated for $\delta = 0, 0.5, 0.75, 1$, from a simulation of permutation tests on 10,000 pairs of samples. Each permutation test decision was based on 499 permutation replicates (each entry in the table required $5 \cdot 10^6$ calculations of the test statistic). Empirical results are given below for selected alternatives, sample sizes, and dimension, at significance level $\alpha = 0.1$. Both the $\mathcal{E}_n$ and $T_{n,3}$ statistics achieved approximately correct empirical significance in our simulations (see case $\delta = 0$ in Table 8.1), although the Type I error rate for $T_{n,3}$ may be slightly inflated when n is small.

TABLE 8.1: Significant Tests (nearest whole percent at $\alpha = 0.1$, $se \leq 0.5\%$) of Bivariate Normal Location Alternatives $F_1 = N_2((0,0)^T, I_2)$, $F_2 = N_2((0,\delta)^T, I_2)$

		$\delta = 0$		$\delta = 0.5$		$\delta = 0.75$		$\delta = 1$	
n_1	n_2	$\mathcal{E}_n$	$T_{n,3}$	$\mathcal{E}_n$	$T_{n,3}$	$\mathcal{E}_n$	$T_{n,3}$	$\mathcal{E}_n$	$T_{n,3}$
10	10	10	12	23	19	40	29	58	42
15	15	9	11	30	21	53	34	75	52
20	20	10	12	37	23	64	38	86	58
25	25	10	11	43	25	73	42	93	65
30	30	10	11	48	25	81	47	96	70
40	40	11	10	59	28	90	52	99	78
50	50	10	11	69	29	95	58	100	82
75	75	10	11	85	37	99	69	100	93
100	100	10	10	92	40	100	79	100	100

These alternatives differ in location only, and the empirical evidence summarized in Table 8.1 suggests that $\mathcal{E}_n$ is more powerful than $T_{n,3}$ against this class of alternatives. $\diamond$

8.4 Application: Distance Correlation

A test of independence of random vectors $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}^q$

$$H_0 : F_{XY} = F_X F_Y \qquad vs \qquad H_1 : F_{XY} \neq F_X F_Y$$

can be implemented as a permutation test. The permutation test does not require distributional assumptions, or any type of model specification for the dependence structure. Not many universally consistent nonparametric tests exist for the general hypothesis above. In this section we will discuss a new multivariate nonparametric test of independence based on distance correlation [265] that is consistent against all dependent alternatives with finite first moments. The test will be implemented as a permutation test.

Distance Correlation

Distance correlation is a new measure of dependence between random vectors introduced by Székely, Rizzo, and Bakirov [265]. For all distributions with finite first moments, distance correlation $\mathcal{R}$ generalizes the idea of correlation in two fundamental ways:

1. $\mathcal{R}(X, Y)$ is defined for X and Y in arbitrary dimension.
2. $\mathcal{R}(X, Y) = 0$ characterizes independence of X and Y.

Distance correlation satisfies $0 \leq \mathcal{R} \leq 1$, and $\mathcal{R} = 0$ only if X and Y are independent. Distance covariance $\mathcal{V}$ provides a new approach to the problem of testing the joint independence of random vectors. The formal definitions of the population coefficients $\mathcal{V}$ and $\mathcal{R}$ are given in [265]. The definitions of the empirical coefficients are as follows.

Definition 8.1 *The empirical distance covariance $\mathcal{V}_n(\mathbf{X}, \mathbf{Y})$ is the nonnegative number defined by*

$$\mathcal{V}_n^2(\mathbf{X}, \mathbf{Y}) = \frac{1}{n^2} \sum_{k,\, l=1}^{n} A_{kl} B_{kl}, \tag{8.9}$$

where A_{kl} and B_{kl} are defined in equations (8.11-8.12) below. Similarly, $\mathcal{V}_n(\mathbf{X})$ is the nonnegative number defined by

$$\mathcal{V}_n^2(\mathbf{X}) = \mathcal{V}_n^2(\mathbf{X}, \mathbf{X}) = \frac{1}{n^2} \sum_{k,\, l=1}^{n} A_{kl}^2 \,. \tag{8.10}$$

The formulas for A_{kl} and B_{kl} in (8.9–8.10) are given by

$$A_{kl} = a_{kl} - \bar{a}_{k.} - \bar{a}_{.\,l} + \bar{a}_{..}\,; \tag{8.11}$$

$$B_{kl} = b_{kl} - \bar{b}_{k.} - \bar{b}_{.\,l} + \bar{b}_{..}\,, \tag{8.12}$$

where

$$a_{kl} = \|X_k - X_l\|_p, \qquad b_{kl} = \|Y_k - Y_l\|_q, \qquad k, l = 1, \dots, n,$$

and the subscript "." denotes that the mean is computed for the index that it replaces. Note that these formulas are similar to computing formulas in analysis of variance, so the distance covariance statistic is very easy to compute. Although it may not be obvious that $\mathcal{V}_n^2(\mathbf{X}, \mathbf{Y}) \geq 0$, this fact as well as the motivation for the definition of $\mathcal{V}_n$ is explained in [265].

Definition 8.2 *The empirical distance correlation $\mathcal{R}_n(\mathbf{X}, \mathbf{Y})$ is the square root of*

$$\mathcal{R}_n^2(\mathbf{X}, \mathbf{Y}) = \begin{cases} \frac{\mathcal{V}_n^2(\mathbf{X}, \mathbf{Y})}{\sqrt{\mathcal{V}_n^2(\mathbf{X})\mathcal{V}_n^2(\mathbf{Y})}}, & \mathcal{V}_n^2(\mathbf{X})\mathcal{V}_n^2(\mathbf{Y}) > 0; \\ 0, & \mathcal{V}_n^2(\mathbf{X})\mathcal{V}_n^2(\mathbf{Y}) = 0. \end{cases} \tag{8.13}$$

The asymptotic distribution of $n\mathcal{V}_n^2$ is a quadratic form of centered Gaussian random variables, with coefficients that depend on the distributions of X and Y. For the general problem of testing independence when the distributions of X and Y are unknown, the test based on $n\mathcal{V}_n^2$ can be implemented as a permutation test.

Before proceeding to the details of the permutation test, we implement the calculation of the distance covariance statistic (dCov).

Example 8.11 (Distance covariance statistic)

In the distance covariance function `dCov`, operations on the rows and columns of the distance matrix generate the matrix with entries A_{kl}. Note that each term

$$A_{kl} = a_{kl} - \bar{a}_{k.} - \bar{a}_{.\,l} + \bar{a}_{..}; \qquad a_{kl} = \|X_k - X_l\|$$

is a function of the distance matrix of the X sample. In the function `Akl`, the `sweep` operator is used twice. The first `sweep` subtracts $\bar{a}_{.\,l}$, the row means, from the distances a_{kl}. The second `sweep` subtracts $\bar{a}_{k\,.}$, the column means, from the result of the first `sweep`. (The column means and row means are equal because the distance matrix is symmetric.) If the samples are `x` and `y`, then the matrix $A = (A_{kl})$ is returned by `Akl(x)` and the matrix $B = (B_{kl})$ is returned by `Akl(y)`. The remaining calculations are simple functions of these two matrices.

```
dCov <- function(x, y) {
    x <- as.matrix(x)
    y <- as.matrix(y)
    n <- nrow(x)
    m <- nrow(y)
    if (n != m || n < 2) stop("Sample sizes must agree")
    if (! (all(is.finite(c(x, y)))))
        stop("Data contains missing or infinite values")

    Akl <- function(x) {
        d <- as.matrix(dist(x))
        m <- rowMeans(d)
        M <- mean(d)
        a <- sweep(d, 1, m)
        b <- sweep(a, 2, m)
        return(b + M)
    }
    A <- Akl(x)
    B <- Akl(y)
    dCov <- sqrt(mean(A * B))
    dCov
}
```

A simple example to try out the dCov function is the following. Compute $\mathcal{V}_n$ for the bivariate distributions of iris setosa (petal length, petal width) and (sepal length, sepal width).

```
z <- as.matrix(iris[1:50, 1:4])
x <- z[ , 1:2]
y <- z[ , 3:4]
# compute the observed statistic
> dCov(x, y)
[1] 0.06436159
```

The returned value is $\mathcal{V}_n = 0.06436159$. Here $n = 50$ so the test statistic for a test of independence is $n\mathcal{V}_n^2 \doteq 0.207$. $\diamond$

Example 8.12 (Distance correlation statistic)

The distance covariance must be computed to get the distance correlation statistic. Rather than call the distance covariance function three times, which means repeated calculation of the distances and the A and B matrices, it is more efficient to combine all operations in one function.

```
DCOR <- function(x, y) {
    x <- as.matrix(x)
    y <- as.matrix(y)
    n <- nrow(x)
    m <- nrow(y)
    if (n != m || n < 2) stop("Sample sizes must agree")
    if (! (all(is.finite(c(x, y)))))
        stop("Data contains missing or infinite values")
    Akl <- function(x) {
        d <- as.matrix(dist(x))
        m <- rowMeans(d)
        M <- mean(d)
        a <- sweep(d, 1, m)
        b <- sweep(a, 2, m)
        return(b + M)
    }
    A <- Akl(x)
    B <- Akl(y)
    dCov <- sqrt(mean(A * B))
    dVarX <- sqrt(mean(A * A))
    dVarY <- sqrt(mean(B * B))
    dCor <- sqrt(dCov / sqrt(dVarX * dVarY))
    list(dCov=dCov, dCor=dCor, dVarX=dVarX, dVarY=dVarY)
}
```

Applying the function `DCOR` to the `iris` data we obtain all of the distance dependence statistics in one step.

```
z <- as.matrix(iris[1:50, 1:4])
x <- z[ , 1:2]
y <- z[ , 3:4]

> unlist(DCOR(x, y))
      dCov       dCor      dVarX      dVarY
0.06436159 0.61507138 0.28303069 0.10226284
```

◇

Permutation tests of independence

A permutation test of independence is implemented as follows. Suppose that $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}^q$ and $Z = (X, Y)$. Then Z is a random vector in $\mathbb{R}^{p+q}$. In the following, we suppose that a random sample is in an $n \times (p+q)$ data matrix $\mathbf{Z}$ with observations in rows:

$$\mathbf{Z}_{n\times d} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,p} & y_{1,1} & y_{1,2} & \cdots & y_{1,q} \\ x_{2,1} & x_{2,2} & \dots & x_{2,p} & y_{2,1} & y_{2,2} & \cdots & y_{2,q} \\ \vdots & \vdots & & \vdots & & & & \\ x_{n,1} & x_{n,2} & \dots & x_{n,p} & y_{n,1} & y_{n,2} & \cdots & x_{n,q} \end{bmatrix}.$$

Let ν_1 be the row labels of the X sample and let ν_2 be the row labels of the Y sample. Then (Z, ν_1, ν_2) is the sample from the joint distribution of X and Y. If X and Y are dependent, the samples must be paired and the ordering of labels ν_2 cannot be changed independently of ν_1. Under independence, the samples X and Y need not be matched. Any permutation of the row labels of the X or Y sample generates a permutation replicate. The permutation test procedure for independence permutes the row indices of one of the samples (it is not necessary to permute both ν_1 and ν_2).

Approximate permutation test procedure for independence

Let $\hat{\theta}$ be a two sample statistic for testing multivariate independence.

1. Compute the observed test statistic $\hat{\theta}(X, Y) = \hat{\theta}(Z, \nu_1, \nu_2)$.
2. For each replicate, indexed $b = 1, \dots, B$:
   (a) Generate a random permutation $\pi_b = \pi(\nu_2)$.
   (b) Compute the statistic $\hat{\theta}^{(b)} = \hat{\theta}^*(Z, \pi_b) = \hat{\theta}(X, Y^*, \pi(\nu_2))$.
3. If large values of $\hat{\theta}$ support the alternative, compute the ASL by

$$\hat{p} = \frac{1 + \#\{\hat{\theta}^{(b)} \geq \hat{\theta}\}}{B+1} = \frac{\left\{1 + \sum_{b=1}^{B} I(\hat{\theta}^{(b)} \geq \hat{\theta})\right\}}{B+1}.$$

The ASL for a lower-tail or two-tail test based on $\hat{\theta}$ is computed in a similar way.

4. Reject H_0 at significance level α if $\hat{p} \leq \alpha$.

Example 8.13 (Distance covariance test)

This example tests whether the bivariate distributions (petal length, petal width) and (sepal length, sepal width) of iris setosa are independent. To implement a permutation test, write a function to compute the replicates of the test statistic $n\mathcal{V}_n^2$ that takes as its first argument the data matrix and as its second argument the permutation vector.

```
ndCov2 <- function(z, ix, dims) {
    #dims contains dimensions of x and y
    p <- dims[1]
    q1 <- dims[2] + 1
    d <- p + dims[2]
    x <- z[ , 1:p]      #leave x as is
    y <- z[ix, q1:d]    #permute rows of y
    return(nrow(z) * dCov(x, y)^2)
}

library(boot)
z <- as.matrix(iris[1:50, 1:4])
boot.obj <- boot(data = z, statistic = ndCov2, R = 999,
    sim = "permutation", dims = c(2, 2))

tb <- c(boot.obj$t0, boot.obj$t)
hist(tb, nclass="scott", xlab="", main="",
        freq=FALSE)
points(boot.obj$t0, 0, cex=1, pch=16)

> mean(tb >= boot.obj$t0)
[1] 0.066
> boot.obj
DATA PERMUTATION
Call: boot(data = z, statistic = ndCov2, R = 999,
      sim = "permutation", dims = c(2, 2))
Bootstrap Statistics :
    original      bias    std. error
t1* 0.2071207 -0.05991699   0.0353751
```

The achieved significance level is 0.066 so the null hypothesis of independence is rejected at $\alpha = 0.10$. The histogram of replicates of the dCov statistic is shown in Figure 8.4. $\diamond$

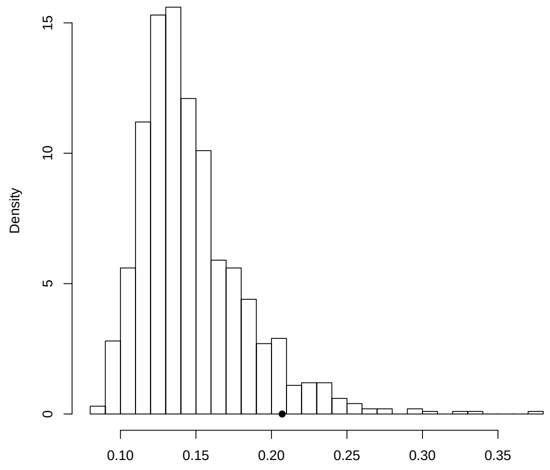


FIGURE 8.4: Permutation replicates of dCov in Example 8.13.

One of the advantages of the dCov test is that it is sensitive to all types of dependence structures in data. Procedures based on the classical definition of covariance, or measures of association based on ranks are generally less effective against non-monotone types of dependence. An alternative with non-monotone dependence is tested in the following example.

Example 8.14 (Power of dCov)

Consider the data generated by the following nonlinear model. Suppose that

$$Y_{ij} = X_{ij}\varepsilon_{ij}, \qquad i = 1, \ldots, n,\ j = 1, \ldots, 5,$$

where $X \sim N_5(0, I_5)$ and $\varepsilon \sim N_5(0, \sigma^2 I_5)$ are independent. Then X and Y are dependent, but if the parameter σ is large, the dependence can be hard to detect. We compared the permutation test implementation of *dCov* with the parametric Wilks Lambda (W) likelihood ratio test [296] using Bartlett's approximation for the critical value (see e.g. [188, Sec. 5.3.2b]). Recall that Wilks Lambda tests whether the covariance $\Sigma_{12} = Cov(X, Y)$ is the zero matrix.

From a power comparison with 10,000 test decisions for each of the sample sizes we have obtained the results shown in Table 8.2 and Figure 8.5. Figure 8.5 shows a plot of power vs sample size. Table 8.2 reports the empirical power for a subset of the cases in the plot.

The dCov test is clearly more powerful in this empirical comparison. This example illustrates that the parametric Wilks Lambda test based on product-moment correlation is not always powerful against non-monotone types of dependence. The dCov test is statistically consistent with power approaching 1 as $n \to \infty$ (theoretically and empirically). $\diamond$

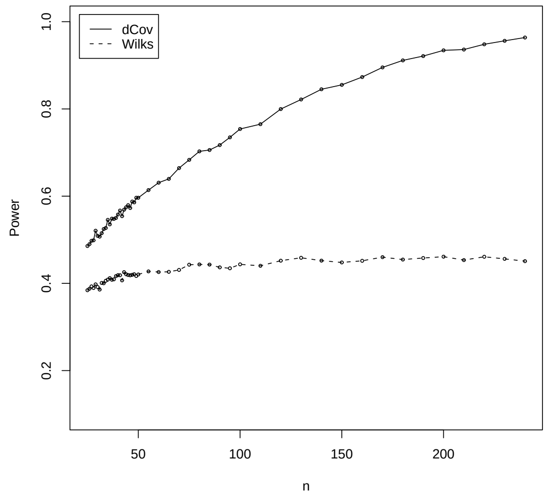


FIGURE 8.5: Empirical power comparison of the distance covariance test dCov and Wilks Lambda W in Example 8.14.

TABLE 8.2: Example 8.14: Percent of Significant Tests of Independence of $Y = X\varepsilon$ at $\alpha = 0.1$ ($se \leq 0.5\%$)

n	*dCov*	*W*	n	*dCov*	*W*	n	*dCov*	*W*
25	48.56	38.43	55	61.39	42.74	100	75.40	44.36
30	50.89	39.16	60	63.09	42.60	120	79.97	45.20
35	54.56	40.86	65	63.96	42.64	140	84.51	45.21
40	55.79	41.88	70	66.43	43.08	160	87.31	45.17
45	57.93	41.91	75	68.32	44.28	180	91.13	45.46
50	59.63	42.05	80	70.27	44.34	200	93.43	46.12

For properties of distance covariance and distance correlation, proofs of convergence and consistency, and more empirical results, see [265]. The distance correlation and covariance statistics and the corresponding permutation tests are provided in the `energy` package [226].

Exercises

8.1 Implement the two-sample Cramér-von Mises test for equal distributions as a permutation test. Apply the test to the data in Examples 8.1 and 8.2.

8.2 Implement the bivariate Spearman rank correlation test for independence [255] as a permutation test. The Spearman rank correlation test statistic can be obtained from function `cor` with `method = "spearman"`. Compare the achieved significance level of the permutation test with the p-value reported by `cor.test` on the same samples.

8.3 The Count 5 test for equal variances in Section 6.4 is based on the maximum number of extreme points. Example 6.15 shows that the Count 5 criterion is not applicable for unequal sample sizes. Implement a permutation test for equal variance based on the maximum number of extreme points that applies when sample sizes are not necessarily equal.

8.4 Complete the steps to implement a r^{th}-nearest neighbors test for equal distributions. Write a function to compute the test statistic. The function should take the data matrix as its first argument, and an index vector as the second argument. The number of nearest neighbors r should follow the index argument.

Projects

8.A Replicate the power comparison in Example 8.10, reducing the number of permutation tests from 10000 to 2000 and number of replicates from 499 to 199. Use the `eqdist.etest (energy)` version of the energy test.

8.B The `aml (boot)` [34] data contains estimates of the times to remission for two groups of patients with acute myelogenous leukaemia (AML). One group received maintenance chemotherapy treament and the other group did not. See the description in the `aml` data help topic. Following Davison and Hinkley [63, Example 4.12], compute the log-rank statistic and apply a permutation test procedure to test whether the survival distributions of the two groups are equal.

Chapter 9

Markov Chain Monte Carlo Methods

9.1 Introduction

Markov Chain Monte Carlo (MCMC) methods encompass a general framework of methods introduced by Metropolis et al. [197] and Hastings [138] for Monte Carlo integration. Recall (see Section 5.2) that Monte Carlo integration estimates the integral

$$\int_A g(t)dt$$

with a sample mean, by restating the integration problem as an expectation with respect to some density function $f(\cdot)$. The integration problem then is reduced to finding a way to generate samples from the target density $f(\cdot)$.

The MCMC approach to sampling from $f(\cdot)$ is to construct a Markov chain with stationary distribution $f(\cdot)$, and run the chain for a sufficiently long time until the chain converges (approximately) to its stationary distribution.

This chapter is a brief introduction to MCMC methods, with the goal of understanding the main ideas and how to implement some of the methods in R. In the following sections, methods of constructing the Markov chains are illustrated, such as the Metropolis and Metropolis-Hastings algorithms, and the Gibbs sampler, with applications. Methods of checking for convergence are briefly discussed. In addition to references listed in Section 5.1, see e.g. Casella and George [40], Chen, Shao, and Ibrahim [44], Chib and Greenberg [47], Gamerman [103], Gelman et al. [108], or Tierney [272]. For a thorough, accessible treatment with applications, see Gilks, Richardson, and Spiegelhalter [120]. For reference on Monte Carlo methods including extensive treatment of MCMC methods see Robert and Casella [228].

9.1.1 Integration problems in Bayesian inference

Many applications of Markov Chain Monte Carlo methods are problems that arise in Bayesian inference. From a Bayesian perspective, in a statistical model both the observables and the parameters are random. Given observed data $x = \{x_1, \ldots, x_n\}$, and parameters θ, x depends on the *prior* distribution

$f_\theta(\theta)$. This dependence is expressed by the likelihood $f(x_1,\dots,x_n|\theta)$. The joint distribution of (x,θ) is therefore

$$f_{x,\theta}(x,\theta) = f_{x|\theta}(x_1,\dots,x_n|\theta)f_\theta(\theta).$$

One can then update the distribution of θ conditional on the information in the sample $x = \{x_1,\dots,x_n\}$, so that by Bayes Theorem the *posterior* distribution of θ is given by

$$f_{\theta|x}(\theta|x) = \frac{f_{x|\theta}(x_1,\dots,x_n|\theta)f_\theta(\theta)}{\int f_{x|\theta}(x_1,\dots,x_n|\theta)f_\theta(\theta)d\theta} = \frac{f_{x|\theta}(x)f_\theta(\theta)}{\int f_{x|\theta}(x)f_\theta(\theta)d\theta}.$$

Then the conditional expectation of a function $g(\theta)$ with respect to the posterior density is

$$E[g(\theta|x)] = \int g(\theta)f_{\theta|x}(\theta)\,d\theta = \frac{\int g(\theta)f_{x|\theta}(x)f_\theta(\theta)d\theta}{\int f_{x|\theta}(x)f_\theta(\theta)d\theta}. \qquad (9.1)$$

To state the problem in more general terms,

$$E[g(Y)] = \frac{\int g(t)\pi(t)\,dt}{\int \pi(t)\,dt}, \qquad (9.2)$$

where $\pi(\cdot)$ is (proportional to) a density or a likelihood. If $\pi(\cdot)$ is a density function, then (9.2) is just the usual definition $E[g(Y)] = \int g(t)f_Y(t)dt$. If $\pi(\cdot)$ is a likelihood, then the normalizing constant in the denominator is needed. In Bayesian analysis, $\pi(\cdot)$ is a posterior density. The expectation (9.2) can be evaluated even if $\pi(\cdot)$ is known only up to a constant. This simplifies the problem because in practice the normalizing constant for a posterior density $f_{\theta|x}(\theta)$ is often difficult to evaluate.

The practical problem, however, is that the integrations in (9.2) are often mathematically intractable, and difficult to compute by numerical methods, especially in higher dimensions. Markov Chain Monte Carlo provides a method for this type of integration problem.

9.1.2 Markov Chain Monte Carlo Integration

The Monte Carlo estimate of $E[g(\theta)] = \int g(\theta)f_{\theta|x}(\theta)d\theta$ is the sample mean

$$\overline{g} = \frac{1}{m}\sum_{i=1}^{m} g(x_i),$$

where $x_1,\dots,x_m$ is a sample from the distribution with density $f_{\theta|x}$. If $x_1,\dots,x_m$ are independent (it is a random sample) then by the laws of large numbers, the sample mean $\overline{g}$ converges in probability to $E[g(\theta)]$ as sample size n tends to infinity. In this case, one can in principle draw as large a Monte

Carlo sample as required to obtain the desired precision in the estimate $\overline{g}$. Here the first "MC" in "MCMC" is not needed; Monte Carlo integration can be used.

However, in a problem such as (9.1) it may be quite difficult to implement a method for generating independent observations from the density $f_{\theta|x}$. Nevertheless, even if the sample observations are dependent, a Monte Carlo integration can be applied if the observations can be generated so that their joint density is roughly the same as the joint density of a random sample. This is where the first "MC" comes to the rescue. Markov Chain Monte Carlo methods estimate the integral in (9.1) or (9.2) by *Monte Carlo* integration, and the *Markov Chain* provides the sampler that generates the random observations from the target distribution.

By a generalization of the strong law of large numbers, if $\{X_0, X_1, X_2, \dots\}$ is a realization of an irreducible, ergodic Markov Chain with stationary distribution π, then

$$\overline{g(X)}_m = \frac{1}{m} \sum_{t=0}^{m} g(X_t)$$

converges with probability one to $E[g(X)]$ as $m \to \infty$, where X has the stationary distribution π and the expectation is taken with respect to π (provided the expectation exists).

For a brief review of discrete-time discrete-state-space Markov Chains see Section 2.8. For an introduction to Markov chains and stochastic processes see Ross [234].

9.2 The Metropolis-Hastings Algorithm

The *Metropolis-Hastings algorithms* are a class of Markov Chain Monte Carlo methods including the special cases of the Metropolis sampler, the Gibbs sampler, the independence sampler, and the random walk. The main idea is to generate a Markov Chain $\{X_t | t = 0, 1, 2, \dots\}$ such that its stationary distribution is the target distribution. The algorithm must specify, for a given state X_t, how to generate the next state X_{t+1}. In all of the Metropolis-Hastings (M-H) sampling algorithms, there is a candidate point Y generated from a proposal distribution $g(\cdot|X_t)$. If this candidate point is accepted, the chain moves to state Y at time $t+1$ and $X_{t+1} = Y$; otherwise the chain stays in state X_t and $X_{t+1} = X_t$. Note that the proposal distribution can depend on the previous state X_t. For example, if the proposal distribution is normal, one choice for $g(\cdot|X_t)$ might be Normal$(\mu_t = X_t, \sigma^2)$ for some fixed σ^2.

The choice of proposal distribution is very flexible, but the chain generated by this choice must satisfy certain regularity conditions. The proposal distribution must be chosen so that the generated chain will converge to a stationary distribution – the target distribution f. Required conditions for the generated chain are irreducibility, positive recurrence, and aperiodicity (see [229]). A proposal distribution with the same support set as the target distribution will usually satisfy these regularity conditions. Refer to [121, Ch. 7-8], [228, Ch. 7] or [229] for further details on the choice of proposal distribution.

9.2.1 Metropolis-Hastings Sampler

The *Metropolis-Hastings sampler* generates the Markov chain $\{X_0, X_1, \dots\}$ as follows.

1. Choose a proposal distribution $g(\cdot|X_t)$ (subject to regularity conditions stated above).
2. Generate X_0 from a distribution g.
3. Repeat (until the chain has converged to a stationary distribution according to some criterion):
   (a) Generate Y from $g(\cdot|X_t)$.
   (b) Generate U from Uniform(0,1).
   (c) If
   $$U \le \frac{f(Y)g(X_t|Y)}{f(X_t)g(Y|X_t)}$$
   accept Y and set $X_{t+1} = Y$; otherwise set $X_{t+1} = X_t$.
   (d) Increment t.

Observe that in step (3c) the candidate point Y is accepted with probability

$$\alpha(X_t, Y) = \min\left(1, \frac{f(Y)g(X_t|Y)}{f(X_t)g(Y|X_t)}\right), \tag{9.3}$$

so that it is only necessary to know the density of the target distribution f up to a constant.

Assuming that the proposal distribution satisfies the regularity conditions, the Metropolis-Hastings chain will converge to a unique stationary distribution π. The algorithm is designed so that the stationary distribution of the Metropolis-Hastings chain is indeed the target distribution, f.

Suppose (r, s) are two elements of the state space of the chain, and without loss of generality suppose that $f(s)g(r|s) \ge f(r)g(s|r)$. Thus, $\alpha(r, s) = 1$ and the joint density of (X_t, X_{t+1}) at (r, s) is $f(r)g(s|r)$. The joint density of (X_t, X_{t+1}) at (s, r) is

$$f(s)g(r|s)\ \alpha(s, r) = f(s)g(r|s)\left(\frac{f(r)g(s|r)}{f(s)g(r|s)}\right) = f(r)g(s|r).$$

The transition kernel is

$$K(r, s) = \alpha(r, s)g(s|r) + I(s = r)\left[1 - \int_\alpha (r, s)g(s|r)ds\right].$$

(The second term in $K(r, s)$ arises when the candidate point is rejected and $X_{t+1} = X_t$.) Hence we have the system of equations

$$\alpha(r,s)f(r)g(s|r) = \alpha(s,r)f(s)g(r|s),$$
$$I(s=r)\left[1 - \int_\alpha (r,s)g(s|r)ds]f(r)\right] = I(r=s)\left[1 - \int_\alpha (s,r)g(r|s)ds]f(s)\right]$$

for the Metropolis-Hastings chain, and f satisfies the detailed balance condition $K(s,r)f(s) = K(r,s)f(r)$. Therefore f is the stationary distribution of the chain. See Theorems 6.46 and 7.2 in [228].

Example 9.1 (Metropolis-Hastings sampler)

Use the Metropolis-Hastings sampler to generate a sample from a Rayleigh distribution. The Rayleigh density [156, (18.76)] is

$$f(x) = \frac{x}{\sigma^2}\, e^{-x^2/(2\sigma^2)}, \qquad x \ge 0,\, \sigma > 0.$$

The Rayleigh distribution is used to model lifetimes subject to rapid aging, because the hazard rate is linearly increasing. The mode of the distribution is at σ, $E[X] = \sigma\sqrt{\pi/2}$ and $Var(X) = \sigma^2(4-\pi)/2$.

For the proposal distribution, try the chisquared distribution with degrees of freedom X_t. Implementation of a Metropolis-Hastings sampler for this example is as follows. Note that the base of the array in R is 1, so we initialize the chain at X_0 in `x[1]`.

1. Set $g(\cdot|X)$ to the density of $\chi^2(X)$.
2. Generate X_0 from distribution $\chi^2(1)$ and store in `x[1]`.
3. Repeat for $i = 2, \ldots, N$:
   (a) Generate Y from $\chi^2(\text{df} = X_t) = \chi^2($df=`x[i-1]`$)$.
   (b) Generate U from Uniform(0, 1).
   (c) With $X_t =$ `x[i-1]`, compute

   $$r(X_t, Y) = \frac{f(Y)g(X_t|Y)}{f(X_t)g(Y|X_t)},$$

   where f is the Rayleigh density with parameter σ, $g(Y|X_t)$ is the $\chi^2(\text{df} = X_t)$ density evaluated at Y, and $g(X_t|Y)$ is the $\chi^2(\text{df} = Y)$ density evaluated at X_t.

   If $U \le r(X_t, Y)$ accept Y and set $X_{t+1} = Y$; otherwise set $X_{t+1} = X_t$. Store X_{t+1} in `x[i]`.
   (d) Increment t.

The constants in the densities cancel, so

$$r(x_t, y) = \frac{f(y)g(x_t|y)}{f(x_t)g(y|x_t)} = \frac{ye^{-y^2/2\sigma^2}}{x_t e^{-x_t^2/2\sigma^2}} \times \frac{\Gamma(\frac{x_t}{2})2^{x_t/2}x_t^{y/2-1}\, e^{-x_t/2}}{\Gamma(\frac{y}{2})2^{y/2}y^{x_t/2-1}\, e^{-y/2}}.$$

This ratio can be simplified further, but in the following simulation for clarity we will evaluate the Rayleigh and chisquare densities separately. The following function evaluates the Rayleigh(σ) density.

```
f <- function(x, sigma) {
    if (any(x < 0)) return (0)
    stopifnot(sigma > 0)
    return((x / sigma^2) * exp(-x^2 / (2*sigma^2)))
}
```

In the simulation below, a Rayleigh($\sigma = 4$) sample is generated using the chisquare proposal distribution. At each transition, the candidate point Y is generated from $\chi^2(\nu = X_{i-1})$

```
xt <- x[i-1]
y <- rchisq(1, df = xt)
```

and for each `y`, the numerator and denominator of $r(X_{i-1}, Y)$ are computed in `num` and `den`. The counter `k` records the number of rejected candidate points.

```
m <- 10000
sigma <- 4
x <- numeric(m)
x[1] <- rchisq(1, df=1)
k <- 0
u <- runif(m)

for (i in 2:m) {
    xt <- x[i-1]
    y <- rchisq(1, df = xt)
    num <- f(y, sigma) * dchisq(xt, df = y)
    den <- f(xt, sigma) * dchisq(y, df = xt)
    if (u[i] <= num/den) x[i] <- y else {
         x[i] <- xt
         k <- k+1     #y is rejected
         }
    }

> print(k)
[1] 4009
```

In this example, approximately 40% of the candidate points are rejected, so the chain is somewhat inefficient.

To see the generated sample as a realization of a stochastic process, we can plot the sample vs the time index. The following code will display a partial plot starting at time index 5000.

```
index <- 5000:5500
y1 <- x[index]
plot(index, y1, type="l", main="", ylab="x")
```

The plot is shown in Figure 9.1. Note that at times the candidate point is rejected and the chain does not move at these time points; this corresponds to the short horizontal paths in the graph. ◇

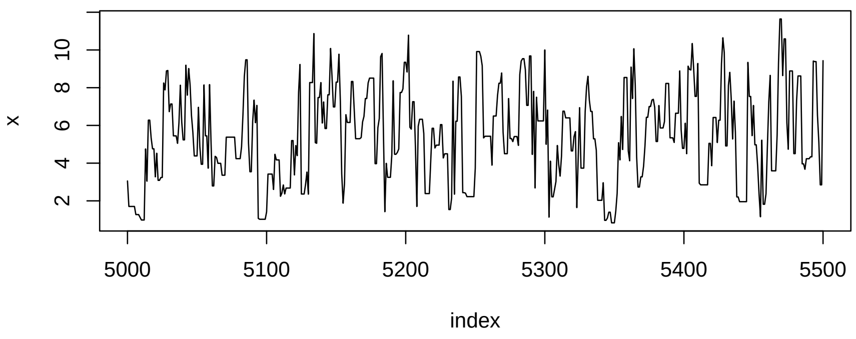


FIGURE 9.1: Part of a chain generated by a Metropolis-Hastings sampler of a Rayleigh distribution in Example 9.1.

Example 9.1 is a simple example intended to illustrate how to implement a Metropolis-Hastings sampler. There are better ways to generate samples from Rayleigh distributions. In fact, an explicit formula for the quantiles of the Rayleigh distribution are given by

$$x_q = F^{-1}(q) = \sigma\{-2\log(1-q)\}^{1/2}, \qquad 0 < q < 1. \tag{9.4}$$

Using F^{-1} one could write a simple generator for Rayleigh using the inverse transform method of Section 3.2.1 with antithetic sampling (Section 5.4).

Example 9.2 (Example 9.1, cont.)

The following code compares the quantiles of the target Rayleigh($\sigma = 4$) distribution with the quantiles of the generated chain in a quantile-quantile plot (QQ plot).

```
b <- 2001       #discard the burnin sample
y <- x[b:m]
a <- ppoints(100)
QR <- sigma * sqrt(-2 * log(1 - a))  #quantiles of Rayleigh
Q <- quantile(x, a)

qqplot(QR, Q, main="",
    xlab="Rayleigh Quantiles", ylab="Sample Quantiles")

hist(y, breaks="scott", main="", xlab="", freq=FALSE)
lines(QR, f(QR, 4))
```

The histogram of the generated sample with the Rayleigh($\sigma = 4$) density superimposed is shown in Figure 9.2(a) and the QQ plot is shown in Figure 9.2(b). The QQ plot is an informal approach to assessing the goodness-of-fit of the generated sample with the target distribution. From the plot, it appears that the sample quantiles are in approximate agreement with the theoretical quantiles. ◇

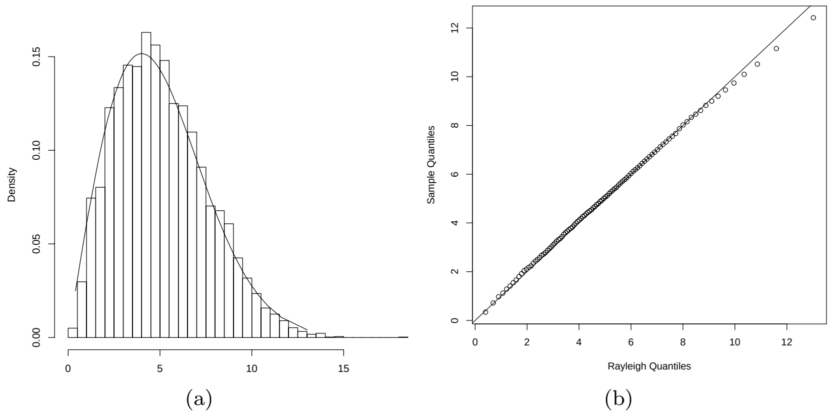


FIGURE 9.2: Histogram with target Rayleigh density and QQ plot for a Metropolis-Hastings chain in Example 9.1.

9.2.2 The Metropolis Sampler

The Metropolis-Hastings sampler [138, 197] is a generalization of the *Metropolis sampler* [197]. In the Metropolis algorithm, the proposal distribution is symmetric. That is, the proposal distribution $g(\cdot|X_t)$ satisfies

$$g(X|Y) = g(Y|X),$$

so that in (9.3) the proposal distribution g cancels from

$$r(X_t, Y) = \frac{f(Y)g(X_t|Y)}{f(X_t)g(Y|X_t)},$$

and the candidate point Y is accepted with probability

$$\alpha(X_t, Y) = \min\left(1,\ \frac{f(Y)}{f(X_t)}\right).$$

9.2.3 Random Walk Metropolis

The *random walk Metropolis* sampler is an example of a Metropolis sampler. Suppose the candidate point Y is generated from a symmetric proposal distribution $g(Y|X_t) = g(|X_t - Y|)$. Then at each iteration, a random increment Z is generated from $g(\cdot)$, and Y is defined by $Y = X_t + Z$. For example, the random increment might be normal with zero mean, so that the candidate point is $Y|X_t \sim Normal(X_t, \sigma^2)$ for some fixed $\sigma^2 > 0$.

Convergence of the random walk Metropolis is often sensitive to the choice of scale parameter. When variance of the increment is too large, most of the candidate points are rejected and the algorithm is very inefficient. If the variance of the increment is too small, the candidate points are almost all accepted, so the random walk Metropolis generates a chain that is almost like a true random walk, which is also inefficient. One approach to selecting the scale parameter is to monitor the acceptance rates, which should be in the range [0.15, 0.5] [230].

Example 9.3 (Random walk Metropolis)

Implement the random walk version of the Metropolis sampler to generate the target distribution Student t with ν degrees of freedom, using the proposal distribution $Normal(X_t, \sigma^2)$. In order to see the effect of different choices of variance of the proposal distribution, try repeating the simulation with different choices of σ.

The $t(\nu)$ density is proportional to $(1 + x^2/\nu)^{-(\nu+1)/2}$, so

$$r(x_t, y) = \frac{f(Y)}{f(X_t)} = \frac{\left(1 + \frac{y^2}{\nu}\right)^{-(\nu+1)/2}}{\left(1 + \frac{x_t^2}{\nu}\right)^{-(\nu+1)/2}}.$$

In this simulation below, the t densities in $r(x_{i-1}, y)$ will be computed by the `dt` function. Then `y` is accepted or rejected and X_i generated by

```
if (u[i] <= dt(y, n) / dt(x[i-1], n))
    x[i] <- y
else
    x[i] <- x[i-1]
```

These steps are combined into a function to generate the chain, given the parameters n and σ, initial value X_0, and the length of the chain, N.

```
rw.Metropolis <- function(n, sigma, x0, N) {
    x <- numeric(N)
    x[1] <- x0
    u <- runif(N)
    k <- 0
    for (i in 2:N) {
        y <- rnorm(1, x[i-1], sigma)
            if (u[i] <= (dt(y, n) / dt(x[i-1], n)))
            x[i] <- y  else {
                x[i] <- x[i-1]
                k <- k + 1
            }
        }
    return(list(x=x, k=k))
    }
```

Four chains are generated for different variances σ^2 of the proposal distribution.

```
n <- 4  #degrees of freedom for target Student t dist.
N <- 2000
sigma <- c(.05, .5, 2,  16)

x0 <- 25
rw1 <- rw.Metropolis(n, sigma[1], x0, N)
rw2 <- rw.Metropolis(n, sigma[2], x0, N)
rw3 <- rw.Metropolis(n, sigma[3], x0, N)
rw4 <- rw.Metropolis(n, sigma[4], x0, N)

#number of candidate points rejected
> print(c(rw1$k, rw2$k, rw3$k, rw4$k))
[1]   14  136  891 1798
```

Only the third chain has a rejection rate in the range [0.15, 0.5]. The plots in Figure 9.3 show that the random walk Metropolis sampler is very sensitive to the variance of the proposal distribution. Recall that the variance of the

$t(\nu)$ distribution is $\nu/(\nu-2)$, $\nu > 2$. Here $\nu = 4$ and the standard deviation of the target distribution is $\sqrt{2}$.

In the first plot of Figure 9.3 with $\sigma = 0.05$, the ratios $r(X_t, Y)$ tend to be large and almost every candidate point is accepted. The increments are small and the chain is almost like a true random walk. Chain 1 has not converged to the target in 2000 iterations. The chain in the second plot generated with $\sigma = 0.5$ is converging very slowly and requires a much longer burn-in period. In the third plot ($\sigma = 2$) the chain is mixing well and converging to the target distribution after a short burn-in period of about 500. Finally, in the fourth plot, where $\sigma = 16$, the ratios $r(X_t, Y)$ are smaller and most of the candidate points are rejected. The fourth chain converges, but it is inefficient.

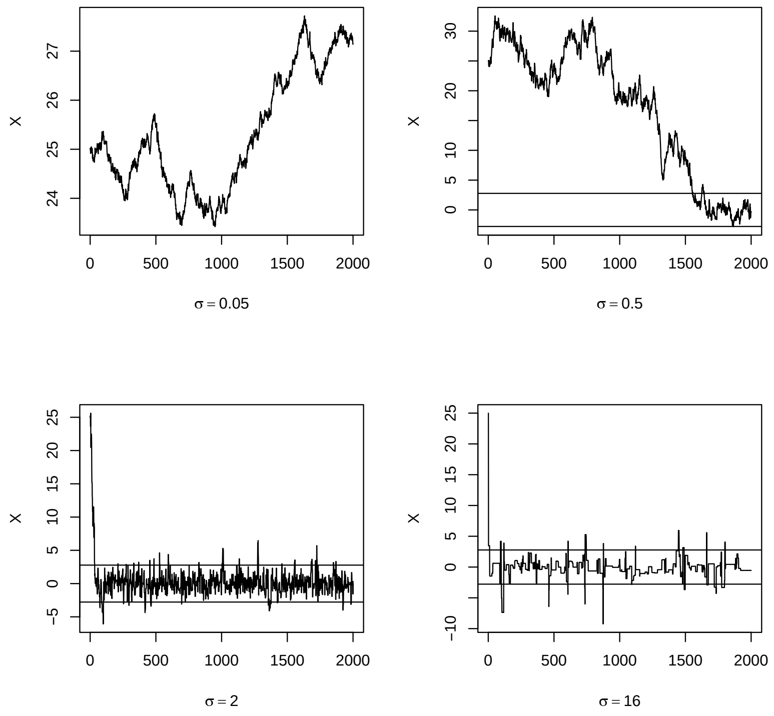


FIGURE 9.3: Random walk Metropolis chains generated by proposal distributions with different variances in Example 9.3.

◇

Example 9.4 (Example 9.3, cont.)

Usually in MCMC problems one does not have the theoretical quantiles of the target distribution available for comparison, but in this case the output of the random walk Metropolis chains in Example 9.3 can be compared with the theoretical quantiles of the target distribution. Discard the burn-in values in the first 500 rows of each chain. The quantiles are computed by the `apply` function (applying `quantile` to the columns of the matrix). The quantiles of the target distribution and the sample quantiles of the four chains `rw1`, `rw2`, `rw3`, and `rw4` are in Table 9.1.

```
a <- c(.05, seq(.1, .9, .1), .95)
Q <- qt(a, n)
rw <- cbind(rw1$x, rw2$x, rw3$x, rw4$x)
mc <- rw[501:N, ]
Qrw <- apply(mc, 2, function(x) quantile(x, a))
print(round(cbind(Q, Qrw), 3))          #not shown
xtable::xtable(round(cbind(Q, Qrw), 3)) #latex format
```

◇

TABLE 9.1: Quantiles of Target Distribution and Chains in Example 9.4

	Q	rw1	rw2	rw3	rw4
5%	−2.13	23.66	−1.16	−1.92	−2.40
10%	−1.53	23.77	−0.39	−1.47	−1.35
20%	−0.94	23.99	0.67	−1.01	−0.90
30%	−0.57	24.29	4.15	−0.63	−0.64
40%	−0.27	24.68	9.81	−0.25	−0.47
50%	0.00	25.29	17.12	0.01	−0.15
60%	0.27	26.14	18.75	0.27	0.06
70%	0.57	26.52	21.79	0.59	0.25
80%	0.94	26.93	25.42	0.92	0.52
90%	1.53	27.27	28.51	1.55	1.18
95%	2.13	27.39	29.78	2.37	1.90

R note 9.1 *Table 9.1 was exported to LaTeXformat by the* `xtable` *function in the* `xtable` *package [61].*

Example 9.5 (Bayesian inference: A simple investment model)

In general, the returns on different investments are not independent. To reduce risk, portfolios are sometimes selected so that returns of securities are negatively correlated. Rather than the correlation of returns, here the daily performance is ranked. Suppose five stocks are tracked for 250 trading days (one year), and each day the "winner" is picked based on maximum return relative to the market. Let X_i be the number of days that security i is a winner. Then the observed vector of frequencies $(x_1, \dots, x_5)$ is an observation from the joint distribution of $(X_1, \dots, X_5)$. Based on historical data, suppose that the prior odds of an individual security being a winner on any given day are $[1 : (1-\beta) : (1-2\beta) : 2\beta : \beta]$, where $\beta \in (0, 0.5)$ is an unknown parameter. Update the estimate of β for the current year of winners.

According to this model, the multinomial joint distribution of $X_1, \dots, X_5$ has the probability vector

$$p = \left(\frac{1}{3}, \frac{(1-\beta)}{3}, \frac{(1-2\beta)}{3}, \frac{2\beta}{3}, \frac{\beta}{3}\right).$$

The posterior distribution of β given $(x_1, \dots, x_5)$ is therefore

$$Pr[\beta|(x_1, \dots, x_5)] = \frac{250!}{x_1!x_2!x_3!x_4!x_5!} p_1^{x_1} p_2^{x_2} p_3^{x_3} p_4^{x_4} p_5^{x_5}.$$

In this example, we cannot directly simulate random variates from the posterior distribution. One approach to estimating β is to generate a chain that converges to the posterior distribution and estimate β from the generated chain. Use the random walk Metropolis sampler with a uniform proposal distribution to generate the posterior distribution of β. The candidate point Y is accepted with probability

$$\alpha(X_t, Y) = \min\left(1, \ \frac{f(Y)}{f(X_t)}\right).$$

The multinomial coefficient cancels from the ratio in $\alpha(X, Y)$, so that

$$\frac{f(Y)}{f(X)} = \frac{(1/3)^{x_1}((1-Y)/3)^{x_2}((1-2Y)/3)^{x_3}\,((2Y)/3)^{x_4}(Y/3)^{x_5}}{(1/3)^{x_1}((1-X)/3)^{x_2}((1-2X)/3)^{x_3}\,((2X)/3)^{x_4}(X/3)^{x_5}}.$$

The ratio can be further simplified, but the numerator and denominator are evaluated separately in the implementation below. In order to check the results, start by generating the observed frequencies from a distribution with specified β.

```
b <- .2             #actual value of beta
w <- .25            #width of the uniform support set
m <- 5000           #length of the chain
burn <- 1000        #burn-in time
days <- 250
x <- numeric(m)     #the chain
```

```
# generate the observed frequencies of winners
i <- sample(1:5, size=days, replace=TRUE,
        prob=c(1, 1-b, 1-2*b, 2*b, b))
win <- tabulate(i)
> print(win)
[1] 82 72 45 34 17
```

The tabulated frequencies in win are the simulated numbers of trading days that each of the stocks were the daily winner. Based on this year's observed distribution of winners, we want to estimate the parameter β.

The following function prob computes the target density (without the constant).

```
prob <- function(y, win) {
    # computes (without the constant) the target density
    if (y < 0 || y >= 0.5)
        return (0)
    return((1/3)^win[1] *
        ((1-y)/3)^win[2] * ((1-2*y)/3)^win[3] *
            ((2*y)/3)^win[4] * (y/3)^win[5])
}
```

Finally the random walk Metropolis chain is generated. Two sets of uniform random variates are required; one for generating the proposal distribution and another for the decision to accept or reject the candidate point.

```
u <- runif(m)         #for accept/reject step
v <- runif(m, -w, w)  #proposal distribution
x[1] <- .25
for (i in 2:m) {
    y <- x[i-1] + v[i]
    if (u[i] <= prob(y, win) / prob(x[i-1], win))
        x[i] <- y  else
            x[i] <- x[i-1]
}
```

The plot of the chains in Figure 9.4(a) shows that the chain has converged, approximately, to the target distribution. Now the generated chain provides an estimate of β, after discarding a burn-in sample. From the histogram of the sample in Figure 9.4(b) the plausible values for β are close to 0.2.

The original sample table of relative frequencies, and the MCMC estimates of the multinomial probabilities are given below.

```
> print(win)
[1] 82 72 45 34 17
> print(round(win/days, 3))
[1] 0.328 0.288 0.180 0.136 0.068
```

```
> print(round(c(1, 1-b, 1-2*b, 2*b, b)/3, 3))
[1] 0.333 0.267 0.200 0.133 0.067
> xb <- x[(burn+1):m]
> print(mean(xb))
[1] 0.2101277
```

The sample mean of the generated chain is 0.2101277 (the simulated year of winners table was generated with $\beta = 0.2$). ◇

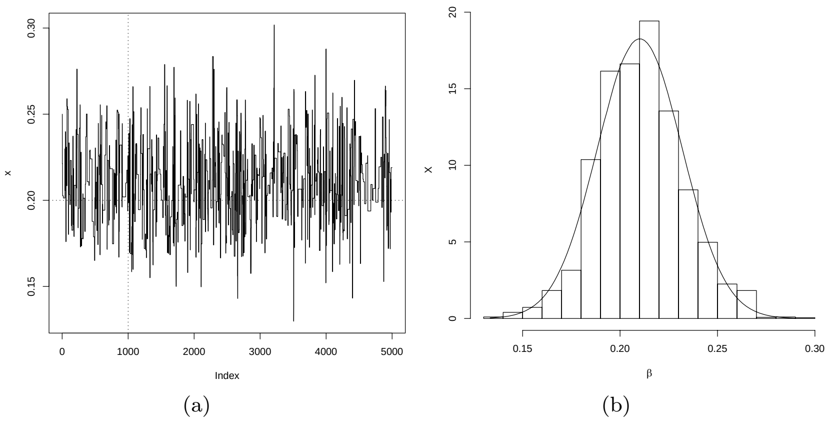


FIGURE 9.4: Random walk Metropolis chain for β in Example 9.5.

9.2.4 The Independence Sampler

Another special case of the Metropolis-Hastings sampler is the independence sampler [272]. The proposal distribution in the independence sampling algorithm does not depend on the previous value of the chain. Thus, $g(Y|X_t) = g(Y)$ and the acceptance probability (9.3) is

$$\alpha(X_t, Y) = \min\left(1, \frac{f(Y)g(X_t)}{f(X_t)g(Y)}\right).$$

The independence sampler is easy to implement and tends to work well when the proposal density is a close match to the target density, but otherwise does not perform well. Roberts [229] discusses convergence of the independence sampler, and comments that "it is rare for the independence sampler to be useful as a stand-alone algorithm." Nevertheless, we illustrate the procedure in the following example, because the independence sampler can be useful in hybrid MCMC methods (see e.g. [119]).

Example 9.6 (Independence sampler)

Assume that a random sample $(z_1, \dots, z_n)$ from a two-component normal mixture is observed. The mixture is denoted by

$$pN(\mu_1, \sigma_1^2) + (1-p)N(\mu_2, \sigma_2^2),$$

and the density of the mixture (see Chapter 3) is

$$f^*(z) = pf_1(z) + (1-p)f_2(z),$$

where f_1 and f_2 are the densities of the two normal distributions, respectively. If the densities f_1 and f_2 are completely specified, the problem is to estimate the mixing parameter p given the observed sample. Generate a chain using an independence sampler that has the posterior distribution of p as the target distribution.

The proposal distribution should be supported on the set of valid probabilities p; that is, the interval $(0,1)$. The most obvious choices are the beta distributions. With no prior information on p, one might consider the Beta(1,1) proposal distribution (Beta(1,1) is Uniform(0,1)). The candidate point Y is accepted with probability

$$\alpha(X_t, Y) = \min\left(1, \frac{f(Y)g(X_t)}{f(X_t)g(Y)}\right),$$

where $g(\cdot)$ is the Beta proposal density. Thus, if the proposal distribution is Beta(a,b), then $g(y) \propto y^{a-1}(1-y)^{b-1}$ and Y is accepted with probability $\min(1, f(y)g(x_t)/f(x_t)g(y))$, where

$$\frac{f(y)g(x_t)}{f(x_t)g(y)} = \frac{x_t^{a-1}(1-x_t)^{b-1}\prod_{j=1}^{n}[yf_1(z_j) + (1-y)f_2(z_j)]}{y^{a-1}(1-y)^{b-1}\prod_{j=1}^{n}[x_tf_1(z_j) + (1-x_t)f_2(z_j)]}.$$

In the following simulation the proposal distribution is Uniform(0,1). The simulated data is generated from the normal mixture

$$0.2N(0,1) + 0.8N(5,1).$$

The first steps are to initialize constants and generate the observed sample. Then an observed sample is generated. To generate the chain, all random numbers can be generated in advance because the candidate Y does not depend on X_t.

```
m <- 5000 #length of chain
xt <- numeric(m)
a <- 1              #parameter of Beta(a,b) proposal dist.
b <- 1              #parameter of Beta(a,b) proposal dist.
p <- .2             #mixing parameter
n <- 30             #sample size
mu <- c(0, 5)       #parameters of the normal densities
sigma <- c(1, 1)

# generate the observed sample
i <- sample(1:2, size=n, replace=TRUE, prob=c(p, 1-p))
x <- rnorm(n, mu[i], sigma[i])

# generate the independence sampler chain
u <- runif(m)
y <- rbeta(m, a, b)      #proposal distribution
xt[1] <- .5

for (i in 2:m) {
    fy <- y[i] * dnorm(x, mu[1], sigma[1]) +
            (1-y[i]) * dnorm(x, mu[2], sigma[2])
    fx <- xt[i-1] * dnorm(x, mu[1], sigma[1]) +
            (1-xt[i-1]) * dnorm(x, mu[2], sigma[2])

    r <- prod(fy / fx) *
           (xt[i-1]^(a-1) * (1-xt[i-1])^(b-1)) /
             (y[i]^(a-1) * (1-y[i])^(b-1))

    if (u[i] <= r) xt[i] <- y[i] else
        xt[i] <- xt[i-1]
    }

plot(xt, type="l", ylab="p")
hist(xt[101:m], main="", xlab="p", prob=TRUE)
print(mean(xt[101:m]))
```

The histogram of the generated sample after discarding the first 100 points is shown in Figure 9.5 on the next page. The mean of the remaining sample is 0.2516. The time plot of the generated chain is shown in Figure 9.6(a), which mixes well and converges quickly to a stationary distribution.

For comparison, we repeated the simulation with a Beta(5,2) proposal distribution. In this simulation the sample mean of the chain after discarding the burn-in sample is 0.2593, but the chain that is generated, shown in Figure 9.6(b) on the following page, is not very efficient. ◇

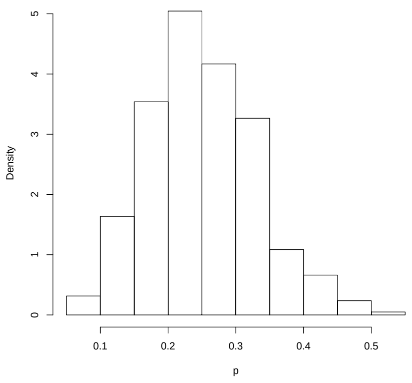


FIGURE 9.5: Distribution of the independence sampler chain for p with proposal distribution Beta(1, 1) in Example 9.6, after discarding a burn-in sample of length 100.

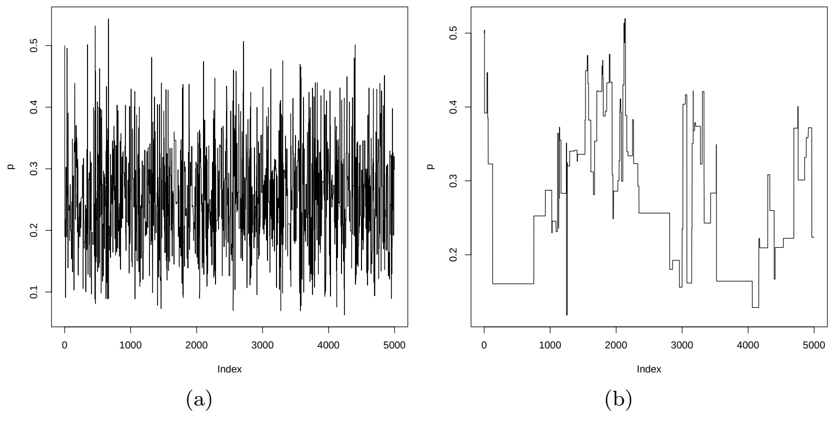


FIGURE 9.6: Chain generated by independence sampler for p with proposal distribution Beta(1, 1) (left) and Beta(5, 2) (right) in Example 9.6.

9.3 The Gibbs Sampler

The Gibbs sampler was named by Geman and Geman [111], because of its application to analysis of Gibbs lattice distributions. However, it is a general method that can be applied to a much wider class of distributions [111, 106, 105]. It is another special case of the Metropolis-Hastings sampler. See the introduction to Gibbs sampling by Casella and George [40].

The Gibbs sampler is often applied when the target is a multivariate distribution. Suppose that all the univariate conditional densities are fully specified and it is reasonably easy to sample from them. The chain is generated by sampling from the marginal distributions of the target distribution, and every candidate point is therefore accepted.

Let $X = (X_1, \dots, X_d)$ be a random vector in $\mathbb{R}^d$. Define the $d-1$ dimensional random vectors

$$X_{(-j)} = (X_1, \dots, X_{j-1}, X_{j+1}, \dots, X_d),$$

and denote the corresponding univariate conditional density of X_j given $X_{(-j)}$ by $f(X_j|X_{(-j)})$. The Gibbs sampler generates the chain by sampling from each of the d conditional densities $f(X_j|X_{(-j)})$.

In the following algorithm for the Gibbs sampler, we denote X_t by $X(t)$.

```
1. Initialize $X(0)$ at time $t = 0$.
2. For each iteration, indexed $t = 1, 2, \dots$ repeat:
    (a) Set $x_1 = X_1(t-1)$.
    (b) For each coordinate $j = 1, \dots, d$
        (a) Generate $X_j^*(t)$ from $f(X_j|x_{(-j)})$.
        (b) Update $x_j = X_j^*(t)$.
    (c) Set $X(t) = (X_1^*(t), \dots, X_d^*(t))$ (every candidate is accepted).
    (d) Increment $t$.
```

Example 9.7 (Gibbs sampler: Bivariate distribution)

Generate a bivariate normal distribution with mean vector (μ_1, μ_2), variances σ_1^2, σ_2^2, and correlation ρ, using Gibbs sampling.

In the bivariate case, $X = (X_1, X_2)$, $X_{(-1)} = X_2$, $X_{(-2)} = X_1$. The conditional densities of a bivariate normal distribution are univariate normal with parameters

$$E[X_2|x_1] = \mu_1 + \rho\frac{\sigma_2}{\sigma_1}(x_1 - \mu_1),$$

$$\mathrm{Var}(X_2|x_1) = (1-\rho^2)\sigma_2^2,$$

and the chain is generated by sampling from

$$f(x_1|x_2) \sim \text{Normal}(\mu_1 + \frac{\rho\sigma_1}{\sigma_2}(x_2 - \mu_2),\ (1-\rho^2)\sigma_1^2),$$
$$f(x_2|x_1) \sim \text{Normal}(\mu_2 + \frac{\rho\sigma_2}{\sigma_1}(x_1 - \mu_1),\ (1-\rho^2)\sigma_2^2).$$

For a bivariate distribution (X_1, X_2), at each iteration the Gibbs sampler

1. Sets $(x_1, x_2) = X(t-1)$;
2. Generates $X_1^*(t)$ from $f(X_1|x_2)$;
3. Updates $x_1 = X_1^*(t)$;
4. Generates $X_2^*(t)$ from $f(X_2|x_1)$;
5. Sets $X(t) = (X_1^*(t), X_2^*(t))$.

```
#initialize constants and parameters
N <- 5000               #length of chain
burn <- 1000            #burn-in length
X <- matrix(0, N, 2)    #the chain, a bivariate sample

rho <- -.75             #correlation
mu1 <- 0
mu2 <- 2
sigma1 <- 1
sigma2 <- .5
s1 <- sqrt(1-rho^2)*sigma1
s2 <- sqrt(1-rho^2)*sigma2

###### generate the chain #####

X[1, ] <- c(mu1, mu2)            #initialize

for (i in 2:N) {
    x2 <- X[i-1, 2]
    m1 <- mu1 + rho * (x2 - mu2) * sigma1/sigma2
    X[i, 1] <- rnorm(1, m1, s1)
    x1 <- X[i, 1]
    m2 <- mu2 + rho * (x1 - mu1) * sigma2/sigma1
    X[i, 2] <- rnorm(1, m2, s2)
}

b <- burn + 1
x <- X[b:N, ]
```

The first 1000 observations are discarded from the chain in matrix X and the remaining observations are in x. Summary statistics for the column means, the sample covariance, and correlation matrices are shown below.

```
# compare sample statistics to parameters
> colMeans(x)
[1] -0.03030001  2.01176134
> cov(x)
           [,1]       [,2]
[1,]  1.0022207 -0.3757518
[2,] -0.3757518  0.2482327
> cor(x)
              [,1]       [,2]
[1,]  1.0000000 -0.7533379
[2,] -0.7533379  1.0000000

plot(x, main="", cex=.5, xlab=bquote(X[1]),
     ylab=bquote(X[2]), ylim=range(x[,2]))
```

The sample means, variances, and correlation are close to the true parameters, and the plot in Figure 9.7 exhibits the elliptical symmetry of the bivariate normal, with negative correlation. (The version printed is a randomly selected subset of 1000 generated variates after discarding the burn-in sample.) ◇

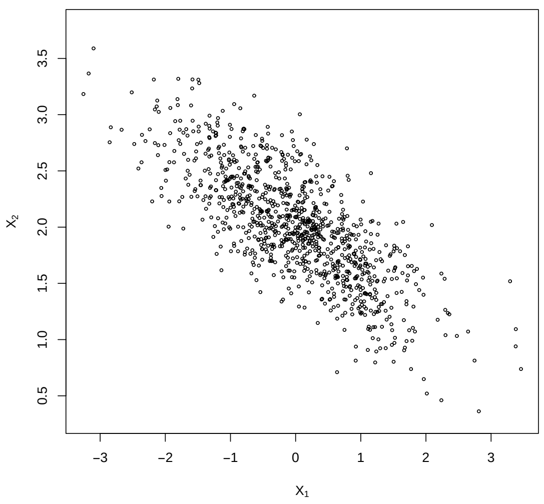


FIGURE 9.7: Bivariate normal chain generated by the Gibbs sampler in Example 9.7.

9.4 Monitoring Convergence

In several examples using various Metropolis-Hastings algorithms, we have seen that some generated chains have not converged to the target distribution. In general, for an arbitrary Metropolis-Hastings sampler the number of iterations that are sufficient for approximate convergence to the target distribution or what length burn-in sample is required are unknown. Moreover, Gelman and Rubin [110] provide examples of slow convergence that cannot be detected by examining a single chain. A single chain may appear to have converged because the generated values have a small variance within a local part of the support set of the target distribution, but in reality the chain has not explored all of the support set. By examining several parallel chains, slow convergence should be more evident, particularly if the initial values of the chain are overdispersed with respect to the target distribution. Methods have been proposed in the literature for monitoring the convergence of MCMC chains (see e.g. [33, 54, 116, 138, 227, 219]). In this section we discuss and illustrate the approach suggested by Gelman and Rubin [107, 109] for monitoring convergence of Metropolis-Hastings chains.

9.4.1 The Gelman-Rubin Method

The Gelman-Rubin [107, 109] method of monitoring convergence of a M-H chain is based on comparing the behavior of several generated chains with respect to the variance of one or more scalar summary statistics. The estimates of the variance of the statistic are analogous to estimates based on between-sample and within-sample mean squared errors in a one-way analysis of variance (ANOVA).

Let ψ be a scalar summary statistic that estimates some parameter of the target distribution. Generate k chains $\{X_{ij} : 1 \leq i \leq k, 1 \leq j \leq n\}$ of length n. (Here the chains are indexed with initial time $t = 1$.) Compute $\{\psi_{in} = \psi(X_{i1}, \dots, X_{in})\}$ for each chain at time n. We expect that if the chains are converging to the target distribution as $n \to \infty$, then the sampling distribution of the statistics $\{\psi_{in}\}$ should be converging to a common distribution.

The Gelman-Rubin method uses the between-sequence variance of ψ and the within-sequence variance of ψ to estimate an upper bound and a lower bound for variance of ψ, converging to variance ψ from above and below, respectively, as the chain converges to the target distribution.

Consider the chains up to time n to represent data from a balanced one-way ANOVA on k groups with n observations. Compute the estimates of between-sample and within-sample variance analogous to the sum of squares for treatments and the sum of squares for error, and the corresponding mean squared errors as in ANOVA.

The between-sequence variance is

$$B = \frac{1}{k-1}\sum_{i=1}^{k}\sum_{j=1}^{n}(\overline{\psi}_{i\cdot} - \overline{\psi}_{\cdot\cdot})^2 = \frac{n}{k-1}\sum_{i=1}^{k}(\overline{\psi}_{i\cdot} - \overline{\psi}_{\cdot\cdot})^2,$$

where

$$\overline{\psi}_{i\cdot} = (1/n)\sum_{j=1}^{n}\psi_{ij}, \qquad \overline{\psi}_{\cdot\cdot} = (1/(nk))\sum_{i=1}^{k}\sum_{j=1}^{n}\psi_{ij}.$$

Within the i^{th} sequence, the sample variance is

$$s_i^2 = \frac{1}{n}\sum_{j=1}^{n}(\psi_{ij} - \overline{\psi}_{i\cdot})^2,$$

and the pooled estimate of within sample variance is

$$W = \frac{1}{nk-k}\sum_{i=1}^{k}(n-1)s_i^2 = \frac{1}{k}\sum_{i=1}^{k}s_i^2.$$

The between-sequence and within-sequence estimates of variance are combined to estimate an upper bound for $Var(\psi)$

$$\widehat{Var}(\psi) = \frac{n-1}{n}W + \frac{1}{n}B. \tag{9.5}$$

If the chains were random samples from the target distribution, (9.5) is an unbiased estimator of $Var(\psi)$. In this application (9.5) is positively biased for the variance of ψ if the initial values of the chain are over-dispersed, but converges to $Var(\psi)$ as $n \to \infty$. On the other hand, if the chains have not converged by time n, the chains have not yet mixed well across the entire support set of the target distribution so the within-sample variance W underestimates the variance of ψ. As $n \to \infty$ we have the expected value of (9.5) converging to $Var(\psi)$ from above and W converging to $Var(\psi)$ from below. If $\widehat{Var}(\psi)$ is large relative to W this suggests that the chain has not converged to the target distribution by time n.

The Gelman-Rubin statistic is the *estimated potential scale reduction*

$$\sqrt{\hat{R}} = \sqrt{\frac{\widehat{Var}(\psi)}{W}}, \tag{9.6}$$

which can be interpreted as measuring the factor by which the standard deviation of ψ could be reduced by extending the chain. The factor $\sqrt{\hat{R}}$ decreases to 1 as the length of the chain tends to infinity, so $\sqrt{\hat{R}}$ should be close to 1 if the chains have approximately converged to the target distribution. Gelman [107] suggests that $\hat{R}$ should be less than 1.1 or 1.2.

Example 9.8 (Gelman-Rubin method of monitoring convergence)

This example illustrates the Gelman-Rubin method of monitoring convergence of a Metropolis chain. The target distribution is Normal(0,1), and the proposal distribution is Normal(X_t, σ^2). The scalar summary statistic ψ_{ij} is the mean of the i^{th} chain up to time j. After generating all chains the diagnostic statistics are computed in the `Gelman.Rubin` function below.

```
Gelman.Rubin <- function(psi) {
    # psi[i,j] is the statistic psi(X[i,1:j])
    # for chain in i-th row of X
    psi <- as.matrix(psi)
    n <- ncol(psi)
    k <- nrow(psi)

    psi.means <- rowMeans(psi)     #row means
    B <- n * var(psi.means)        #between variance est.
    psi.w <- apply(psi, 1, "var")  #within variances
    W <- mean(psi.w)               #within est.
    v.hat <- W*(n-1)/n + (B/n)     #upper variance est.
    r.hat <- v.hat / W             #G-R statistic
    return(r.hat)
    }
```

Since several chains are to be generated, the M-H sampler is written as a function `normal.chain`.

```
normal.chain <- function(sigma, N, X1) {
    #generates a Metropolis chain for Normal(0,1)
    #with Normal(X[t], sigma) proposal distribution
    #and starting value X1
    x <- rep(0, N)
    x[1] <- X1
    u <- runif(N)

    for (i in 2:N) {
        xt <- x[i-1]
        y <- rnorm(1, xt, sigma)     #candidate point
        r1 <- dnorm(y, 0, 1) * dnorm(xt, y, sigma)
        r2 <- dnorm(xt, 0, 1) * dnorm(y, xt, sigma)
        r <- r1 / r2
        if (u[i] <= r) x[i] <- y else
             x[i] <- xt
        }
    return(x)
    }
```

In the following simulation, the proposal distribution has a small variance $\sigma^2 = 0.04$. When the variance is small relative to the target distribution, the chains are usually converging slowly.

```
sigma <- .2      #parameter of proposal distribution
k <- 4           #number of chains to generate
n <- 15000       #length of chains
b <- 1000        #burn-in length

#choose overdispersed initial values
x0 <- c(-10, -5, 5, 10)

#generate the chains
X <- matrix(0, nrow=k, ncol=n)
for (i in 1:k)
    X[i, ] <- normal.chain(sigma, n, x0[i])

#compute diagnostic statistics
psi <- t(apply(X, 1, cumsum))
for (i in 1:nrow(psi))
    psi[i,] <- psi[i,] / (1:ncol(psi))
print(Gelman.Rubin(psi))

#plot psi for the four chains
par(mfrow=c(2,2))
for (i in 1:k)
    plot(psi[i, (b+1):n], type="l",
        xlab=i, ylab=bquote(psi))
par(mfrow=c(1,1)) #restore default

#plot the sequence of R-hat statistics
rhat <- rep(0, n)
for (j in (b+1):n)
    rhat[j] <- Gelman.Rubin(psi[,1:j])
plot(rhat[(b+1):n], type="l", xlab="", ylab="R")
abline(h=1.1, lty=2)
```

The plots of the four sequences of the summary statistic (the mean) ψ are shown in Figure 9.8 from time 1001 to 15000. Rather than interpret the plots, one can refer directly to the value of the factor $\hat{R}$ to monitor convergence. The value $\hat{R} = 1.447811$ at time $n = 5000$ suggests that the chain should be extended. The plot of $\hat{R}$ (Figure 9.9(a)) over time 1001 to 15000 suggests that the chain has approximately converged to the target distribution within approximately 10000 iterations ($\hat{R} = 1.1166$). The dashed line on the plot is at $\hat{R} = 1.1$. Some intermediate values are 1.2252, 1.1836, 1.1561, and 1.1337 at times 6000, 7000, 8000, and 9000, respectively. The value of $\hat{R}$ is less than 1.1 within time 11200.

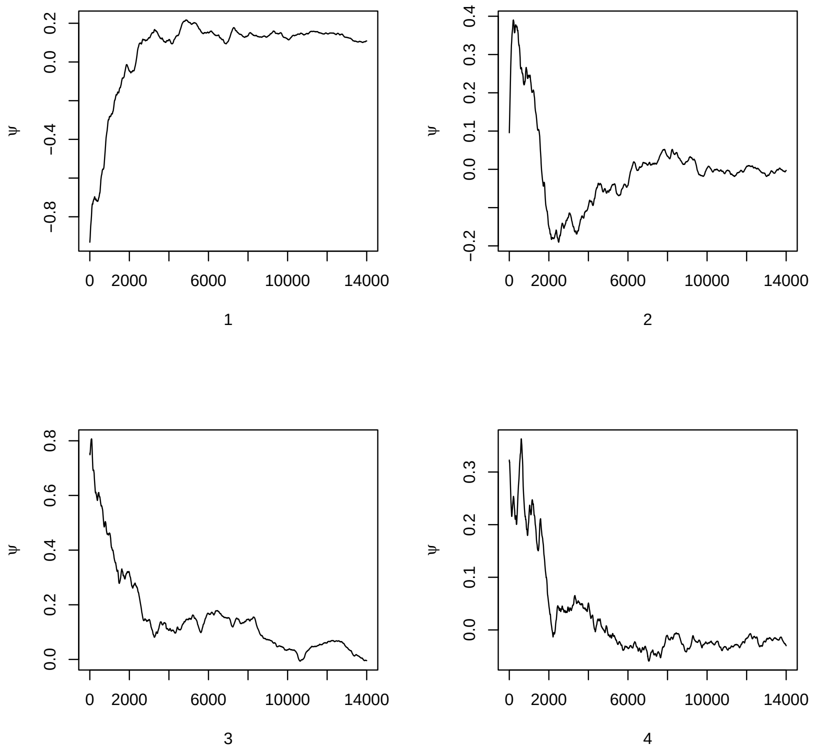


FIGURE 9.8: Sequences of the running means ψ for four Metropolis-Hastings chains in Example 9.8.

For comparison the simulation is repeated, where the variance of the proposal distribution is $\sigma^2 = 4$. The plot of $\hat{R}$ is shown in Figure 9.9(b) for time 1001 to 15000. From this plot it is evident that the chain is converging faster than when the proposal distribution had a very small variance. The value of $\hat{R}$ is below 1.2 within 2000 iterations and below 1.1 within 4000 iterations. $\diamond$

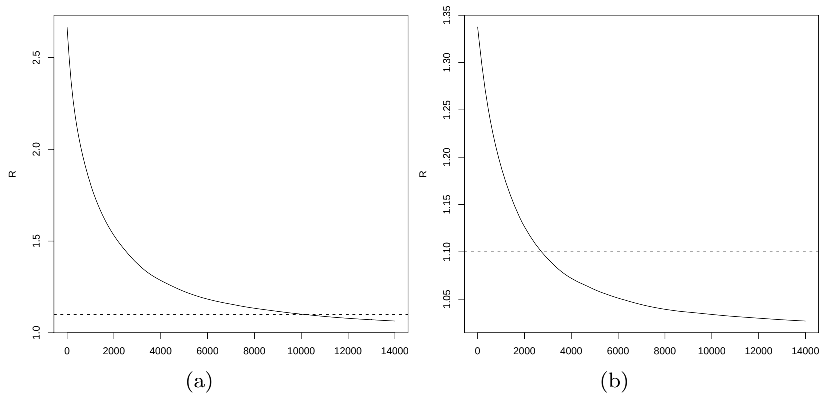


FIGURE 9.9: Sequence of the Gelman-Rubin $\hat{R}$ for four Metropolis-Hastings chains in Example 9.8 (a) $\sigma = 0.2$, (b) $\sigma = 2$.

9.5 Application: Change Point Analysis

A Poisson process is often chosen to model the frequency of rare events. Poisson processes are discussed in Section 3.7. A homogeneous Poisson process $\{X(t), t \geq 0\}$ with constant rate λ is a counting process with independent and stationary increments, such that $X(0) = 0$ and the number of events $X(t)$ in $[0, t]$ has the Poisson(λt) distribution.

Suppose that the parameter λ, which is the expected number of events that occur in a unit of time, has changed at some point in time k. That is, $X_t \sim$ Poisson(μt) for $0 < t \leq k$ and $X_t \sim$ Poisson(λt) for $k < t$. Given a sample of n observations from this process, the problem is to estimate μ, λ and k.

For a specific application, consider the following well known example. The `coal` data in the `boot` package [34] gives the dates of 191 explosions in coal mines which resulted in 10 or more fatalities from March 15, 1851 until March 22, 1962. The data are given in [126], originally from [153]. This problem has been discussed by many authors, including e.g. [36, 37, 63, 121, 171, 192]. A Bayesian model and Gibbs sampling can be applied to estimate the change point in the annual number of coal mining disasters.

Example 9.9 (Coal mining disasters)

In the `coal` data, the date of the disaster is given. The integer part of the date gives the year. For simplicity truncate the fractional part of the year. As a first step, tabulate the number of disasters per year and create a time plot.

```
library(boot)     #for coal data
data(coal)
year <- floor(coal)
y <- table(year)
plot(y)  #a time plot
```

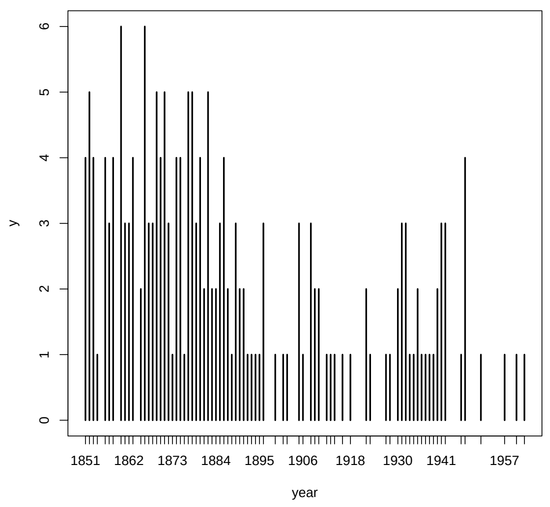


FIGURE 9.10: Number of annual coal mining disasters in Example 9.9.

From the plot in Figure 9.10 it appears that a change in the average number of disasters per year may have occurred somewhere around the turn of the century. Note that vector of frequencies returned by `table` omits the years where there are zero counts, so for the change point analysis `tabulate` is applied.

```
y <- floor(coal[[1]])
y <- tabulate(y)
y <- y[1851:length(y)]

Sequence of annual number of coal mining disasters:

4 5 4 1 0 4 3 4 0 6 3 3 4 0 2 6 3 3 5 4 5 3 1 4 4
1 5 5 3 4 2 5 2 2 3 4 2 1 3 2 2 1 1 1 1 3 0 0 1 0
1 1 0 0 3 1 0 3 2 2 0 1 1 1 0 1 0 1 0 0 0 2 1 0 0
0 1 1 0 2 3 3 1 1 2 1 1 1 1 2 3 3 0 0 0 1 4 0 0 0
1 0 0 0 0 0 1 0 0 1 0 1
```

Let Y_i be the number of disasters in year i, where 1851 is year 1. Assume that the change point occurs at year k, and the number of disasters in year i is a Poisson random variable, where

$$\begin{aligned} Y_i &\sim \text{Poisson}(\mu), \qquad i = 1, \ldots, k, \\ Y_i &\sim \text{Poisson}(\lambda), \qquad i = k+1, \ldots, n. \end{aligned}$$

There are $n = 112$ observations ending with year 1962.

Assume the Bayesian model with independent priors

$$\begin{aligned} k &\sim \text{Uniform}\,\{1, 2, \ldots, n\}, \\ \mu &\sim \text{Gamma}(0.5, b_1), \\ \lambda &\sim \text{Gamma}(0.5, b_2), \end{aligned}$$

introducing additional parameters b_1 and b_2, independently distributed as a positive multiple of a chisquare random variable. That is,

$$\begin{aligned} b_1 | Y, \mu, \lambda, b_2, k &\sim \text{Gamma}(0.5,\ \mu + 1), \\ b_2 | Y, \mu, \lambda, b_1, k &\sim \text{Gamma}(0.5,\ \lambda + 1). \end{aligned}$$

Let $S_k = \sum_{i=1}^k Y_i$, and $S'_k = S_n - S_k$ To apply the Gibbs sampler, the fully specified conditional distributions are needed. The conditional distributions for μ, λ, b_1, and b_2 are given by

$$\begin{aligned} \mu \,|\, y, \lambda, b_1, b_2, k &\sim \text{Gamma}(0.5 + S_k,\ k + b_1); \\ \lambda \,|\, y, \mu, b_1, b_2, k &\sim \text{Gamma}(0.5 + S'_k,\ n - k + b_2); \\ b_1 \,|\, y, \mu, \lambda, b_2, k &\sim \text{Gamma}(0.5, \mu + 1); \\ b_2 \,|\, y, \mu, \lambda, b_1, k &\sim \text{Gamma}(0.5, \lambda + 1), \end{aligned}$$

and the posterior density of the change point k is

$$f(k | Y, \mu, \lambda, b_1, b_2) = \frac{L(Y; k, \mu, \lambda)}{\sum_{j=1}^n L(Y; j, \mu, \lambda)}, \tag{9.7}$$

where

$$L(Y; k, \mu, \lambda) = e^{k(\lambda - \mu)} \left(\frac{\mu}{\lambda}\right)^{S_k}$$

is the likelihood function.

For the change point analysis with the model specified on the previous page, the Gibbs sampler algorithm is as follows ($G(a, b)$ denotes the Gamma(shape= a, rate= b) distribution).

1. Initialize k by a random draw from 1:n, and initialize λ, μ, b_1, b_2 to 1.
2. For each iteration, indexed $t = 1, 2, \ldots$ repeat:
   (a) Generate $\mu(t)$ from $G(0.5 + S_{k(t-1)},\ k(t-1) + b_1(t-1))$.
   (b) Generate $\lambda(t)$ from $G(0.5 + S'_{k(t-1)},\ n - k(t-1) + b_2(t-1))$.
   (c) Generate $b_1(t)$ from $G(0.5,\ \mu(t) + 1)$.
   (d) Generate $b_2(t)$ from $G(0.5,\ \lambda(t) + 1)$.
   (e) Generate $k(t)$ from the multinomial distribution defined by (9.7) using the updated values of λ, μ, b_1, b_2.
   (f) $X(t) = (\mu(t), \lambda(t), b_1(t), b_2(t), k(t))$ (every candidate is accepted).
   (g) Increment t.

The implementation of the Gibbs sampler for this problem is shown on the facing page.

From the output of the Gibbs sampler below, the following sample means are obtained after discarding a burn-in sample of size 200. The estimated change point is $k \doteq 40$. From year $k = 1$ (1851) to $k = 40$ (1890) the estimated Poisson mean is $\hat{\mu} \doteq 3.1$, and from year $k = 41$ (1891) forward the estimated Poisson mean is $\hat{\lambda} \doteq 0.93$.

```
b <- 201
j <- k[b:m]
> print(mean(k[b:m]))
[1] 39.935
> print(mean(lambda[b:m]))
[1] 0.9341033
> print(mean(mu[b:m]))
[1] 3.108575
```

Histograms and plots of the chains are shown in Figures 9.11 and 9.12. Code to generate the plots is given on page 279. ◇

```
# Gibbs sampler for the coal mining change point

# initialization
n <- length(y)    #length of the data
m <- 1000         #length of the chain
mu <- lambda <- k <- numeric(m)
L <- numeric(n)
k[1] <- sample(1:n, 1)
mu[1] <- 1
lambda[1] <- 1
b1 <- 1
b2 <- 1



# run the Gibbs sampler
for (i in 2:m) {
    kt <- k[i-1]

    #generate mu
    r <- .5 + sum(y[1:kt])
    mu[i] <- rgamma(1, shape = r, rate = kt + b1)

    #generate lambda
    if (kt + 1 > n) r <- .5 + sum(y) else
        r <- .5 + sum(y[(kt+1):n])
    lambda[i] <- rgamma(1, shape = r, rate = n - kt + b2)

    #generate b1 and b2
    b1 <- rgamma(1, shape = .5, rate = mu[i]+1)
    b2 <- rgamma(1, shape = .5, rate = lambda[i]+1)

    for (j in 1:n) {
        L[j] <- exp((lambda[i] - mu[i]) * j) *
                     (mu[i] / lambda[i])^sum(y[1:j])
        }
    L <- L / sum(L)

    #generate k from discrete distribution L on 1:n
    k[i] <- sample(1:n, prob=L, size=1)
}
```

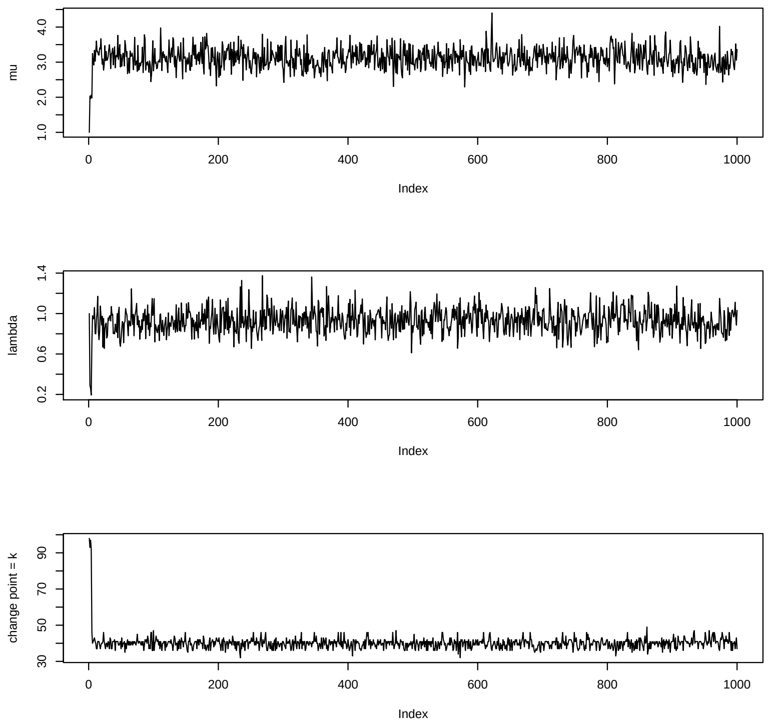


FIGURE 9.11: Output of the Gibbs sampler in Example 9.9.

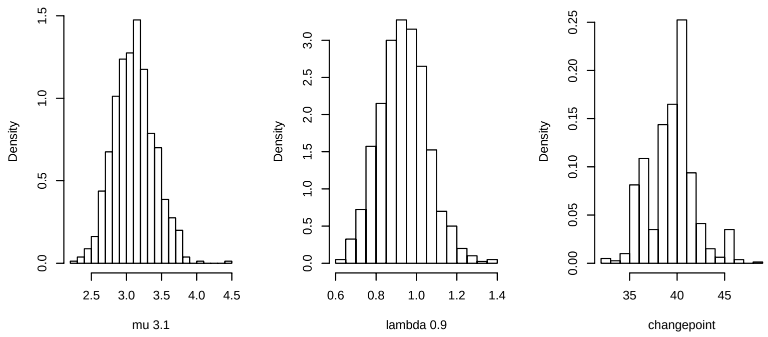


FIGURE 9.12: Distribution of μ, λ, and k from the change point analysis for coal mining disasters in Example 9.9.

Several contributed packages for R offer implementations of the methods in this chapter. See, for example, the packages `mcmc` and `MCMCpack` [117, 191]. The `coda` (Convergence Diagnosis and Output Analysis) package [212] provides utilities that summarize, plot, and diagnose convergence of `mcmc` objects created by functions in `MCMCpack`. Also see `mcgibbsit` [291]. For implementation of Bayesian methods in general, see the task view on CRAN "Bayesian Inference" for a description of several packages.

Exercises

9.1 Repeat Example 9.1 for the target distribution Rayleigh($\sigma = 2$). Compare the performance of the Metropolis-Hastings sampler for Example 9.1 and this problem. In particular, what differences are obvious from the plot corresponding to Figure 9.1?

9.2 Repeat Example 9.1 using the proposal distribution $Y \sim$ Gamma($X_t, 1$) (shape parameter X_t and rate parameter 1).

9.3 Use the Metropolis-Hastings sampler to generate random variables from a standard Cauchy distribution. Discard the first 1000 of the chain, and compare the deciles of the generated observations with the deciles of the standard Cauchy distribution (see `qcauchy` or `qt` with df=1). Recall that a Cauchy(θ, η) distribution has density function

$$f(x) = \frac{1}{\theta\pi(1 + [(x - \eta)/\theta]^2)}, \qquad -\infty < x < \infty,\ \theta > 0.$$

The standard Cauchy has the Cauchy($\theta = 1, \eta = 0$) density. (Note that the standard Cauchy density is equal to the Student t density with one degree of freedom.)

9.4 Implement a random walk Metropolis sampler for generating the standard Laplace distribution (see Exercise 3.2). For the increment, simulate from a normal distribution. Compare the chains generated when different variances are used for the proposal distribution. Also, compute the acceptance rates of each chain.

9.5 What effect, if any, does the width w have on the mixing of the chain in Example 9.5? Repeat the simulation keeping the random number seed fixed, trying different proposal distributions based on the random increments from Uniform($-w, w$), varying w.

9.6 Rao [220, Sec. 5g] presented an example on genetic linkage of 197 animals in four categories (also discussed in [67, 106, 171, 266]). The group sizes are

$(125, 18, 20, 34)$. Assume that the probabilities of the corresponding multinomial distribution are

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

Estimate the posterior distribution of θ given the observed sample, using one of the methods in this chapter.

9.7 Implement a Gibbs sampler to generate a bivariate normal chain (X_t, Y_t) with zero means, unit standard deviations, and correlation 0.9. Plot the generated sample after discarding a suitable burn-in sample. Fit a simple linear regression model $Y = \beta_0 + \beta_1 X$ to the sample and check the residuals of the model for normality and constant variance.

9.8 This example appears in [40]. Consider the bivariate density

$$f(x, y) \propto \binom{n}{x} y^{x+a-1}(1-y)^{n-x+b-1}, \quad x = 0, 1, \ldots, n,\ 0 \leq y \leq 1.$$

It can be shown (see e.g. [23]) that for fixed a, b, n, the conditional distributions are Binomial(n, y) and Beta$(x + a, n - x + b)$. Use the Gibbs sampler to generate a chain with target joint density $f(x, y)$.

9.9 Modify the Gelman-Rubin convergence monitoring given in Example 9.8 so that only the final value of $\hat{R}$ is computed, and repeat the example, omitting the graphs.

9.10 Refer to Example 9.1. Use the Gelman-Rubin method to monitor convergence of the chain, and run the chain until the chain has converged approximately to the target distribution according to $\hat{R} < 1.2$. (See Exercise 9.9.) Also use the `coda` [212] package to check for convergence of the chain by the Gelman-Rubin method. Hints: See the help topics for the `coda` functions `gelman.diag`, `gelman.plot`, `as.mcmc`, and `mcmc.list`.

9.11 Refer to Example 9.5. Use the Gelman-Rubin method to monitor convergence of the chain, and run the chain until the chain has converged approximately to the target distribution according to $\hat{R} < 1.2$. Also use the `coda` [212] package to check for convergence of the chain by the Gelman-Rubin method. (See Exercises 9.9 and 9.10.)

9.12 Refer to Example 9.6. Use the Gelman-Rubin method to monitor convergence of the chain, and run the chain until the chain has converged approximately to the target distribution according to $\hat{R} < 1.2$. Also use the `coda` [212] package to check for convergence of the chain by the Gelman-Rubin method. (See Exercises 9.9 and 9.10.)

R Code

Code for Figure 9.3 on page 255

Reference lines are added at the $t_{0.025}(\nu)$ and $t_{0.975}(\nu)$ quantiles.

```
par(mfrow=c(2,2))  #display 4 graphs together
refline <- qt(c(.025, .975), df=n)
rw <- cbind(rw1$x, rw2$x, rw3$x,  rw4$x)
for (j in 1:4) {
    plot(rw)[,j], type="l",
         xlab=bquote(sigma == .(round(sigma[j],3))),
         ylab="X", ylim=range(rw[,j]))
    abline(h=refline)
}
par(mfrow=c(1,1)) #reset to default
```

Code for Figures 9.4(a) on page 259 and 9.4(b) on page 259

```
plot(x, type="l")
abline(h=b, v=burn, lty=3)
xb <- x[- (1:burn)]
hist(xb, prob=TRUE, xlab=bquote(beta), ylab="X", main="")
z <- seq(min(xb), max(xb), length=100)
lines(z, dnorm(z, mean(xb), sd(xb)))
```

Code for Figure 9.11 on page 276

```
# plots of the chains for Gibbs sampler output

par(mfcol=c(3,1), ask=TRUE)
plot(mu, type="l", ylab="mu")
plot(lambda, type="l", ylab="lambda")
plot(k, type="l", ylab="change point = k")
```

Code for Figure 9.12 on page 276

```
# histograms from the Gibbs sampler output

par(mfrow=c(2,3))
labelk <- "changepoint"
label1 <- paste("mu", round(mean(mu[b:m]), 1))
label2 <- paste("lambda", round(mean(lambda[b:m]), 1))

hist(mu[b:m], main="", xlab=label1,
     breaks = "scott", prob=TRUE) #mu posterior
hist(lambda[b:m], main="", xlab=label2,
     breaks = "scott", prob=TRUE) #lambda posterior
hist(j, breaks=min(j):max(j), prob=TRUE, main="",
    xlab = labelk)
par(mfcol=c(1,1), ask=FALSE)  #restore display
```