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)
```

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

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 $x_1, \ldots, x_n$. 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 $\{x_1, \ldots, x_n\}$ 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 $x^{(1)}, \ldots, x^{(m)}$ are generated from the distribution of X, then a Monte Carlo estimate of the MSE of $\hat{\theta} = \hat{\theta}(x_1, \ldots, x_n)$ is

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

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 $X_1, \ldots, X_n$ is a random sample and $X_{(1)}, \ldots, X_{(n)}$ 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 $T^{(j)}$, $j = 1 \ldots, m$ by repeating:
   (a) Generate $x_1^{(j)}, \ldots, x_n^{(j)}$, iid from the distribution of X.
   (b) Sort $x_1^{(j)}, \ldots, x_n^{(j)}$ 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 $T^{(1)}, \ldots, T^{(m)}$ 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
```


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 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
```

```
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. ◇

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 $X_1, \ldots, X_n$ 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)
```

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. ◇

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 $j = 1, \ldots, m$:
   (a) Generate the j^{th} random sample, $X_1^{(j)}, \ldots, X_n^{(j)}$.
   (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

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

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 $j = 1, \ldots, m$:
   (a) Generate the j^{th} random sample $x_1^{(j)}, \ldots, x_n^{(j)}$ 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

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

Example 6.7 (Empirical Type I error rate)

Suppose that $X_1, \ldots, X_{20}$ 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 nu-

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]. ◇

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 $\hat{\pi}(\theta) \pm \widehat{se}(\hat{\pi}(\theta))$


```
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 $\theta \to \infty$. ◇

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 $X_1, X_2, \ldots, X_n$ 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`*.* ◇

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

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

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.

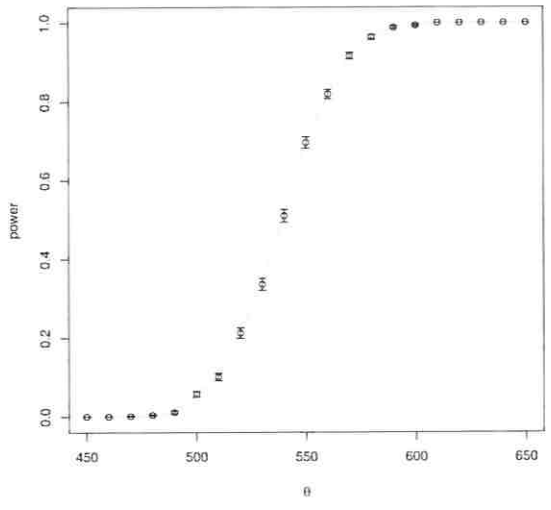


FIGURE 6.1: Empirical power $\hat{\pi}(\theta) \pm \widehat{se}(\hat{\pi}(\theta))$ for the t-test of $H_0 : \theta = 500$ vs $H_1 : \theta > 500$ in Example 6.9.

```
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)
```

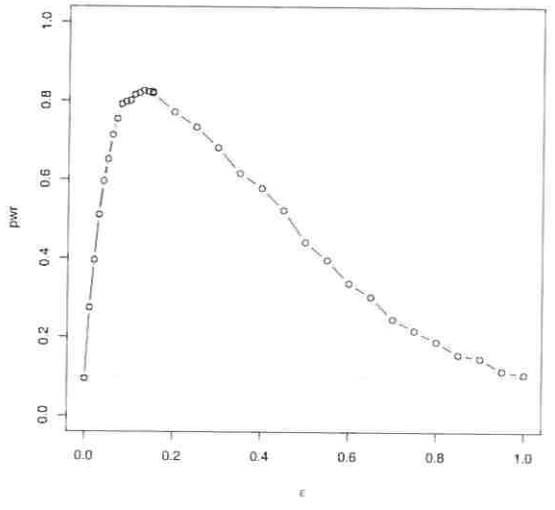



FIGURE 6.2: Empirical power $\hat{\pi}(\varepsilon) \pm \widehat{se}(\hat{\pi}(\varepsilon))$ for the skewness test of normality against ε-contaminated normal scale mixture alternative in Example 6.10.

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. ◇

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

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

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

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

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

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


```
# 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

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
```

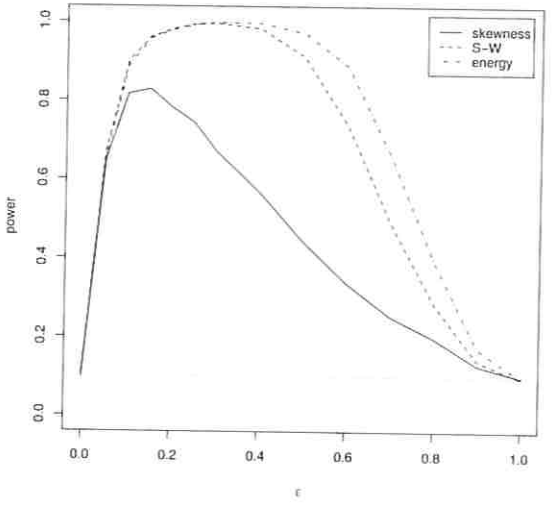



FIGURE 6.3: Empirical power of three tests of normality against a contaminated normal alternative in Example 6.11 ($n = 30$, $\alpha = 0.1$, $se \leq 0.01$)

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

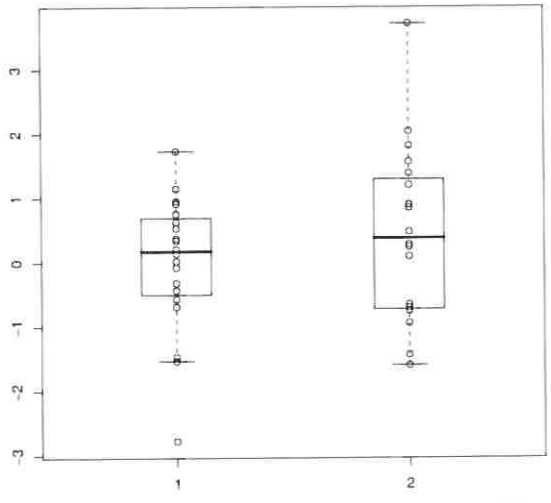


FIGURE 6.4: Boxplots showing extreme points for the Count Five statistic in Example 6.12.

```
> 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

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. ◇

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

[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

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

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 $x_1, \ldots, x_n$ 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

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

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

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

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

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 $\{x_1, \ldots, x_n\}$. 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

$$F \to X \to F_n$$
$$F_n \to X^* \to F_n^*.$$

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

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 $b = 1, \ldots, B$:
   (a) Generate sample $x^{*(b)} = x_1^*, \ldots, x_n^*$ by sampling with replacement from the observed sample $x_1, \ldots, x_n$.
   (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 $\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B)}$.

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

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 $F_n(x)$:

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

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. ◇

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
```

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 $b = 1, \ldots, B$:
   (a) Generate sample $x^{*(b)} = x_1^*, \ldots, x_n^*$ by sampling with replacement from the observed sample $x_1, \ldots, x_n$.
   (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 $\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B)} = R^{(1)}, \ldots, R^{(B)}$.

```
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

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. ◇

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 $\{i_1, \ldots, i_n\}$ 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
```

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].*

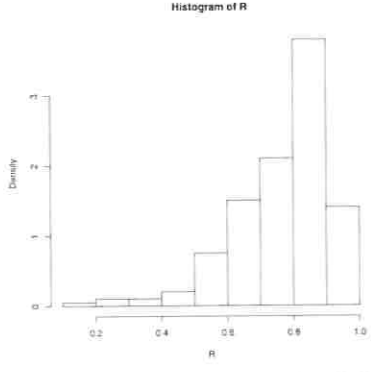


FIGURE 7.1: Bootstrap replicates for law school data in Example 7.2.

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 $x = (x_1, \ldots, x_n)$, 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

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

```
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] 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 $x = (x_1, \ldots, x_n)$ 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,

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

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

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 $x_1, \ldots, x_n$ 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 $i = 1, \ldots, n$ we have

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

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
```

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

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

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

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)
```

```
    #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)
```


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. ◇

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

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

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:

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

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

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*

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

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)
```

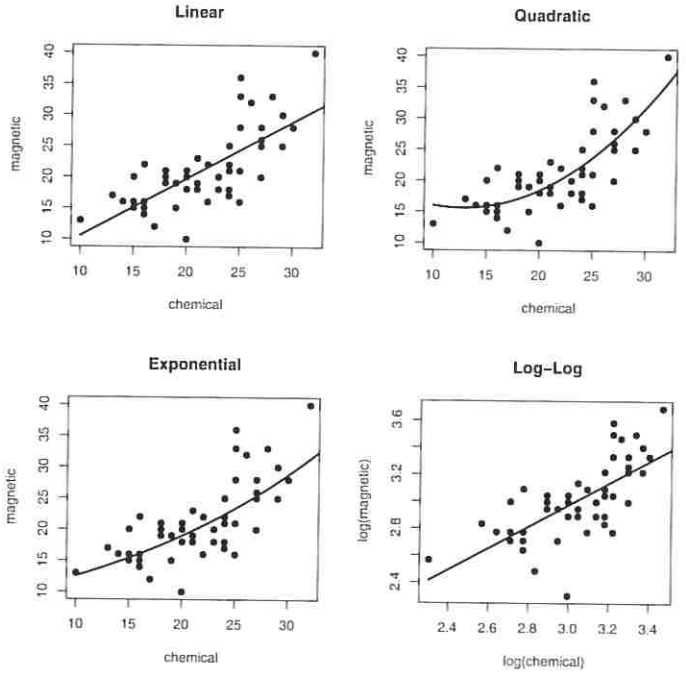



FIGURE 7.2: Four proposed models for `ironslag` data in Example 7.17.

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 $\hat{\sigma}_\varepsilon^2 = \frac{1}{n}\sum_{k=1}^{n} e_k^2$.

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

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. ◇

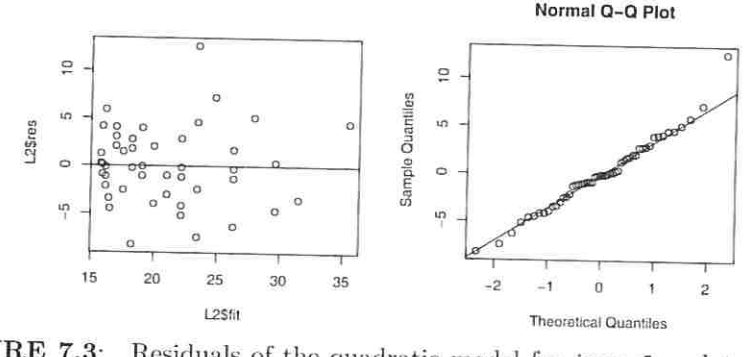


FIGURE 7.3: Residuals of the quadratic model for `ironslag` data, from Example 7.17.

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 Distribution

Suppose that two independent random samples $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$ are observed from the distributions F_X and F_Y, respectively. Let Z be the ordered set $\{X_1, \ldots, X_n, Y_1, \ldots, Y_m\}$, indexed by

$$\nu = \{1, \ldots, n, n+1, \ldots, n+m\} = \{1, \ldots, N\}.$$

Then $Z_i = X_i$ if $1 \le i \le n$ and $Z_i = Y_{i-n}$ if $n+1 \le i \le n+m$. 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

$$\begin{aligned}\{\hat{\theta}^*\} &= \left\{\hat{\theta}(Z,\, \pi_j(\nu)),\, j = 1, \ldots, \binom{N}{n}\right\} \\ &= \{\hat{\theta}^{(j)} \mid \pi_j(\nu) \text{ is a permutation of } \nu\}.\end{aligned}$$

The cdf of $\hat{\theta}^*$ is given by

$$F_{\hat{\theta}^*}(t) = P(\hat{\theta}^* \le t) = \binom{N}{n}^{-1} \sum_{j=1}^{N} I(\hat{\theta}^{(j)} \le t). \tag{8.4}$$

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

$$P(\hat{\theta}^* \ge \hat{\theta}) = \binom{N}{n}^{-1} \sum_{j=1}^{N} I(\hat{\theta}^{(j)} \ge \hat{\theta}),$$

where $\hat{\theta} = \hat{\theta}(Z, \nu)$ is the statistic computed on the observed sample. The ASL

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 $b = 1, \ldots, B$:
   (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

$$\hat{p} = \frac{1 + \#\{\hat{\theta}^{(b)} \ge \hat{\theta}\}}{B+1} = \frac{\left\{1 + \sum_{b=1}^{B} I(\hat{\theta}^{(b)} \ge \hat{\theta})\right\}}{B+1}.$$

   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 $\hat{p} \le \alpha$.

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)
```

```
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
```


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. ◇

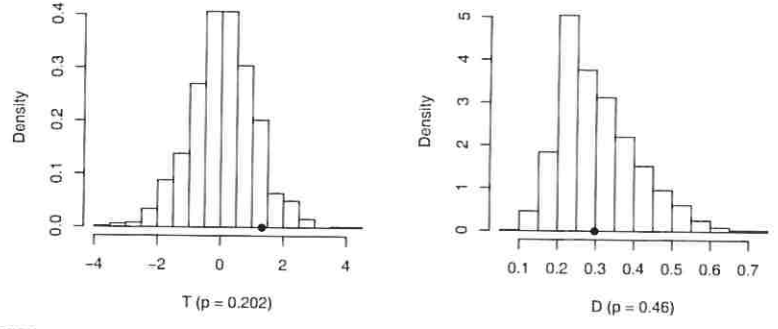


FIGURE 8.1: Permutation distribution of replicates in Example 8.1 (left) and Example 8.2 (right).

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

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

$$D = \sup_{1 \le i \le N} |F_n(z_i) - G_m(z_i)|,$$

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 $0 \le D \le 1$ 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 $x_1, \ldots, x_n$ and G_m is the ecdf of the sample $y_1, \ldots, y_m$. 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

In the following sections multivariate samples will be denoted by boldface type. Suppose that

$$\mathbf{X} = \{X_1, \ldots, X_{n_1}\} \in \mathbb{R}^d, \quad \mathbf{Y} = \{Y_1, \ldots, Y_{n_2}\} \in \mathbb{R}^d,$$

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 $\mathbf{Z} = \{X_1, \ldots, X_{n_1}, Y_1, \ldots, Y_{n_2}\}$ 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

$$I_i(1) = 1 \text{ if } Z_i \text{ and } NN_1(Z_i) \text{ belong to the same sample;}$$

```
> 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)
```



```
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.


```
hist(tb, freq=FALSE, main="",
     xlab="replicates of T(n,3) statistic")
points(boot.obj$t0, 0, cex=1, pch=16)
```

◇

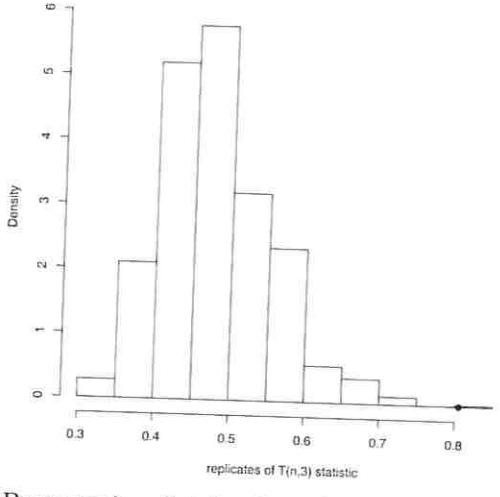


FIGURE 8.2: Permutation distribution of $T_{n,3}$ in Example 8.6.

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

$$\mathcal{E}_n = e(\mathbf{X}, \mathbf{Y}) = \frac{n_1 n_2}{n_1 + n_2}\left(\frac{2}{n_1 n_2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\|X_i - Y_j\| - \frac{1}{n_1^2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_1}\|X_i - X_j\| - \frac{1}{n_2^2}\sum_{i=1}^{n_2}\sum_{j=1}^{n_2}\|Y_i - Y_j\|\right). \quad (8.7)$$

On the name "energy" and

equality. 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,

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)
}
```

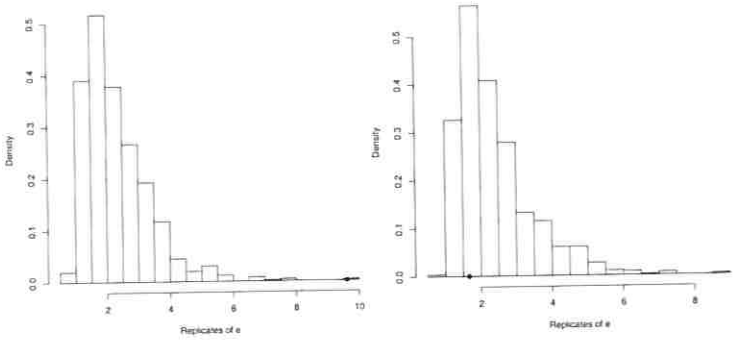
Below, the simulated samples in $\mathbb{R}^d$ are generated from distributions that differ in location. The first distribution is centered at $\mu_1 = (0, \ldots, 0)^T$ and the second distribution is centered at $\mu_2 = (a, \ldots, a)^T$.

```
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



(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 $\delta = 0, 0.5, 0.75, 1$,

mutation 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. ◇

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

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

$$A_{kl} = a_{kl} - \bar{a}_{k\cdot} - \bar{a}_{\cdot l} + \bar{a}_{\cdot\cdot}\,; \tag{8.11}$$
$$B_{kl} = b_{kl} - \bar{b}_{k\cdot} - \bar{b}_{\cdot l} + \bar{b}_{\cdot\cdot}\,, \tag{8.12}$$

where

$$a_{kl} = \|X_k - X_l\|_p, \qquad b_{kl} = \|Y_k - Y_l\|_q, \qquad k, l = 1, \ldots, n,$$

and the subscript "$\cdot$" 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

$$A_{kl} = a_{kl} - \bar{a}_{k\cdot} - \bar{a}_{\cdot l} + \bar{a}_{\cdot\cdot}; \qquad a_{kl} = \|X_k - X_l\|$$

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 $\bar{a}_{\cdot l}$, the row means, from the distances a_{kl}. The second `sweep` subtracts $\bar{a}_{k\cdot}$, 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 $n\mathcal{V}_n^2 \doteq 0.207$. ◇

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:

$$\mathbf{Z}_{n\times d} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,p} & y_{1,1} & y_{1,2} & \dots & y_{1,q} \\ x_{2,1} & x_{2,2} & \dots & x_{2,p} & y_{2,1} & y_{2,2} & \dots & y_{2,q} \\ \vdots & \vdots & & \vdots & & & & \\ x_{n,1} & x_{n,2} & \dots & x_{n,p} & y_{n,1} & y_{n,2} & \dots & x_{n,q} \end{bmatrix}.$$

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 :
```

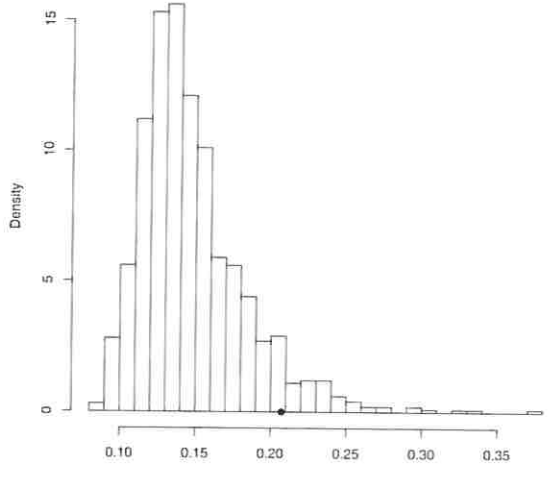



FIGURE 8.4: Permutation replicates of dCov in Example 8.13.

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.

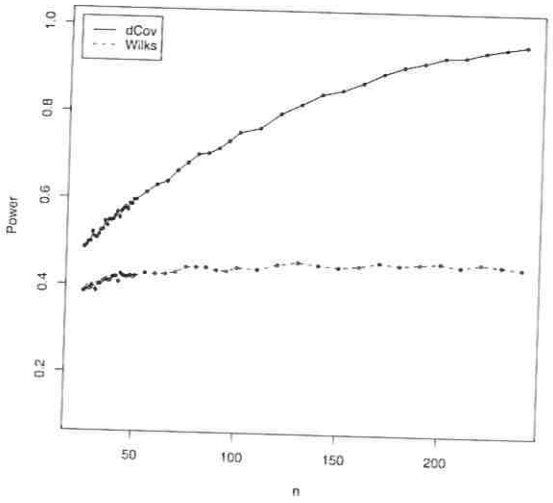


FIGURE 8.5: Empirical power comparison of the distance covariance test dCov and Wilks Lambda W in Example 8.14.

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.