6

Resampling

6.1 Introduction

Finding a single set of estimates for the parameters in a statistical model is not enough. An assessment of the uncertainty in these estimates is also needed. Standard errors and confidence intervals are common methods for expressing uncertainty.[1] In the past, it was sometimes difficult, if not impossible, to assess uncertainty, especially for complex models. Fortunately, the speed of modern computers, and the innovations in statistical methodology inspired by this speed, have largely overcome this problem. In this chapter we apply a computer simulation technique called the "bootstrap" or "resampling" to find standard errors and confidence intervals. The bootstrap method is very widely applicable and will be used extensively in the remainder of this book. The bootstrap is one way that modern computing has revolutionized statistics. Markov chain Monte Carlo (MCMC) is another; see Chap. 20.

The term "bootstrap" was coined by Bradley Efron (1979) and comes from the phrase "pulling oneself up by one's bootstraps."

When statistics are computed from a randomly chosen sample, then these statistics are random variables. Students often do not appreciate this fact. After all, what could be random about $\overline{Y}$? We just averaged the data, so what is random? The point is that the sample is only one of many possible samples. Each possible sample gives a different value of $\overline{Y}$. Thus, although we only see one value of $\overline{Y}$, it was selected at random from the many possible values and therefore $\overline{Y}$ is a random variable.

Methods of statistical inference such as confidence intervals and hypothesis tests are predicated on the randomness of statistics. For example, the

[1] See Appendices A.16.2 and A.17 for introductions to standard errors and confidence intervals.

D. Ruppert, D.S. Matteson, *Statistics and Data Analysis for Financial Engineering*, Springer Texts in Statistics,
DOI 10.1007/978-1-4939-2614-5_6

confidence coefficient of a confidence interval tells us the probability, before a random sample is taken, that an interval constructed from the sample will contain the parameter. Therefore, by the law of large numbers, the confidence coefficient is also the long-run frequency of intervals that cover their parameter. Confidence intervals are usually derived using probability theory. Often, however, the necessary probability calculations are intractable, and in such cases we can replace theoretical calculations by Monte Carlo simulation.

But how do we simulate sampling from an *unknown* population? The answer, of course, is that we cannot do this exactly. However, a sample is a good representative of the population, and we can simulate sampling from the population by sampling from the sample, which is called *resampling*.

Each resample has the same sample size n as the original sample. The reason for this is that we are trying to simulate the original sampling, so we want the resampling to be as similar as possible to the original sampling. By *bootstrap approximation*, we mean the approximation of the sampling process by resampling.

There are two basic resampling methods, model-free and model-based, which are also known, respectively, as nonparametric and parametric. In this chapter, we assume that we have an i.i.d. sample from some population. For dependent data, resampling requires different techniques, which will be discussed in Sect. 13.6.

In *model-free resampling*, the resamples are drawn *with replacement* from the original sample. Why with replacement? The reason is that only sampling with replacement gives independent observations, and we want the resamples to be i.i.d. just as the original sample. In fact, if the resamples were drawn without replacement, then every resample would be exactly the same as the original sample, so the resamples would show no random variation. This would not be very satisfactory, of course.

Model-based resampling does not take a sample from the original sample. Instead, one assumes that the original sample was drawn i.i.d. from a density in the parametric family, $\{f(\boldsymbol{y}|\boldsymbol{\theta}) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$, so, for an unknown value of $\boldsymbol{\theta}$, $f(\boldsymbol{y}|\boldsymbol{\theta})$ is the population density. The resamples are drawn i.i.d. from the density $f(\boldsymbol{y}|\widehat{\boldsymbol{\theta}})$, where $\widehat{\boldsymbol{\theta}}$ is some estimate of the parameter vector $\boldsymbol{\theta}$.

The number of resamples taken should, in general, be large. Just how large depends on the context and is discussed more fully later. Sometimes thousands or even tens of thousands of resamples are used. We let B denote the number of resamples.

When reading the following section, keep in mind that with resampling, the original sample plays the role of the population, because the resamples are taken from the original sample. Therefore, estimates from the sample play the role of true population parameters.

6.2 Bootstrap Estimates of Bias, Standard Deviation, and MSE

Let θ be a one-dimensional parameter, let $\widehat{\theta}$ be its estimate from the sample, and let $\widehat{\theta}^*_1, \ldots, \widehat{\theta}^*_B$ be estimates from B resamples. Also, define $\overline{\widehat{\theta}^*}$ to be the mean of $\widehat{\theta}^*_1, \ldots, \widehat{\theta}^*_B$. An asterisk indicates a statistic calculated from a resample.

The bias of $\widehat{\theta}$ is defined as $\text{BIAS}(\widehat{\theta}) = E(\widehat{\theta}) - \theta$. Since expectations, which are population averages, are estimated by averaging over resamples, the bootstrap estimate of bias is

$$\text{BIAS}_{\text{boot}}(\widehat{\theta}) = \overline{\widehat{\theta}^*} - \widehat{\theta}. \tag{6.1}$$

Notice that, as discussed in the last paragraph of the previous section, in the bootstrap estimate of bias, the unknown population parameter θ is replaced by the estimate $\widehat{\theta}$ from the sample. The bootstrap standard error for $\widehat{\theta}$ is the sample standard deviation of $\widehat{\theta}^*_1, \ldots, \widehat{\theta}^*_B$, that is,

$$s_{\text{boot}}(\widehat{\theta}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\widehat{\theta}^*_b - \overline{\widehat{\theta}^*})^2}. \tag{6.2}$$

$s_{\text{boot}}(\widehat{\theta})$ estimates the standard deviation of $\widehat{\theta}$.

The mean-squared error (MSE) of $\widehat{\theta}$ is $E(\widehat{\theta} - \theta)^2$ and is estimated by

$$\text{MSE}_{\text{boot}}(\widehat{\theta}) = \frac{1}{B}\sum_{b=1}^{B}(\widehat{\theta}^*_b - \widehat{\theta})^2.$$

As in the estimation of bias, when estimating MSE, the unknown θ is replaced by $\widehat{\theta}$. The MSE reflects both bias and variability and, in fact,

$$\text{MSE}_{\text{boot}}(\widehat{\theta}) \approx \text{BIAS}^2_{\text{boot}}(\widehat{\theta}) + s^2_{\text{boot}}(\widehat{\theta}). \tag{6.3}$$

We would have equality in (6.3), rather than an approximation, if in the denominator of (6.1) we used B rather than $B - 1$. Since B is usually large, the error of the approximation is typically very small.

6.2.1 Bootstrapping the MLE of the *t*-Distribution

Functions that compute the MLE, such as, `fitdistr()` in `R`, usually compute standard errors for the MLE along with the estimates themselves. The standard errors are justified theoretically by an "asymptotic" or "large-sample" approximation, called the CLT (central limit theorem) for the maximum likelihood estimator.[2] This approximation becomes exact only as the sample size

[2] See Sect. 5.10.

increases to ∞. Since a sample size is always finite, one cannot be sure of the accuracy of the standard errors. Computing standard errors by the bootstrap can serve as a check on the accuracy of the large-sample approximation, as illustrated in the following example.

Example 6.1. Bootstrapping GE Daily Returns

This example uses the GE daily returns from January 3, 1969, to December 31, 1998, in the data set `CRSPday` in `R`'s `Ecdat` package. The sample size is 2,528 and the number of resamples is $B = 1{,}000$. The t-distribution was fit using `fitdistr()` in `R` and the model-free bootstrap was used. The first and third lines in Table 6.1 are the estimates and standard errors returned by `fitdistr()`, which uses observed Fisher information to calculate standard errors. The second and fourth lines have the results from bootstrapping. The differences between "Estimate" and "Bootstrap mean" are the bootstrap estimates of bias. We can see that the biases are small relative to the standard errors in the row labeled "SE." Small, and even negligible, bias is common when the sample size is in the thousands, as in this example.

Table 6.1. *Estimates from fitting a t-distribution to the* 2,528 *GE daily returns. "Estimate" = MLE. "SE" is standard error from observed Fisher information returned by the* `R` *function* `fitdistr()`*. "Bootstrap mean" and "Bootstrap SE" are the sample mean and standard deviation of the maximum likelihood estimates from* 1,000 *bootstrap samples.* ν *is the degrees-of-freedom parameter. The model-free bootstrap was used.*

	μ	σ	ν
Estimate	0.000879	0.0113	6.34
Bootstrap mean	0.000874	0.0113	6.30
SE	0.000253	0.000264	0.73
Bootstrap SE	0.000252	0.000266	0.82

It is reassuring that "SE" and "Bootstrap SE" agree as closely as they do in Table 6.1. This is an indication that both are reliable estimates of the uncertainty in the parameter estimates. Such close agreement is more likely with samples as large as this one. □

```
library(bootstrap)
library(MASS)
set.seed("3857")
data(CRSPday, package = "Ecdat")
ge = CRSPday[ ,4]
nboot = 1000
t_mle = function(x){as.vector(fitdistr(x, "t")$estimate)}
results = bootstrap(ge, nboot, t_mle)
```

```
9  rowMeans(results$thetastar[ , ])
10 apply(results$thetastar[,], 1, sd)
11 fitdistr(ge, "t")
```

The code above computes the results reported in Table 6.1. The bootstrap was performed at line 8 by the function `bootstrap()` in the `bootstrap` package which is loaded at line 1. The function `bootstrap()` has three arguments, the data, the value of B, and the function that computes the statistic to be bootstrapped; in this example that function is `t_mle()` which is defined at line 7. The function `fitdistr()` is in the package `MASS` that is loaded at line 2. Lines 9, 10, and 11 compute, respectively, the bootstrap mean, the bootstrap SEs, and the MLE and its standard errors.

Example 6.2. Bootstrapping GE daily returns, continued

To illustrate the bootstrap for a smaller sample size, we now use only the first 250 daily GE returns, approximately the first year of data. The number of bootstrap samples is 1,000. The results are in Table 6.2. For μ and s, the results in Tables 6.1 and 6.2 are comparable though the standard errors in Table 6.2 are, of course, larger because of the smaller sample size. For the parameter ν, the results in Table 6.2 are different in two respects from those in Table 6.1. First, the estimate and the bootstrap mean differ by more than 1, a sign that there is some bias. Second, the bootstrap standard deviation is 2.99, considerably larger than the SE, which is only 1.97. This suggests that the SE, which is based on large-sample theory, specifically the CLT for the MLE, is not an accurate measure of uncertainty in the parameter ν, at least not for the smaller sample.

Table 6.2. *Estimates from fitting a t-distribution to the first* 250 *GE daily returns. Notation as in Table* 6.1*. The nonparametric bootstrap was used.*

	μ	σ	ν
Estimate	0.00142	0.01055	5.52
Bootstrap mean	0.00145	0.01064	6.77
SE	0.000764	0.000817	1.98
Bootstrap SE	0.000777	0.000849	2.99

To gain some insight about why the results of ν in these two tables disagree, kernel density estimates of the two bootstrap samples were plotted in Fig. 6.1. We see that with the smaller sample size in panel (a), the density is bimodal and has noticeable right skewness. The density with the full sample is unimodal and has much less skewness.

Tail-weight parameters such as ν are difficult to estimate unless the sample size is in the thousands. With smaller sample sizes, such as 250, there will

not be enough extreme observations to obtain a precise estimate of the tail-weight parameters. This problem has been nicely illustrated by the bootstrap. The number of extreme observations will vary between bootstrap samples. The bootstrap samples with fewer extreme values will have larger estimates of ν, since larger values of ν correspond to thinner tails.

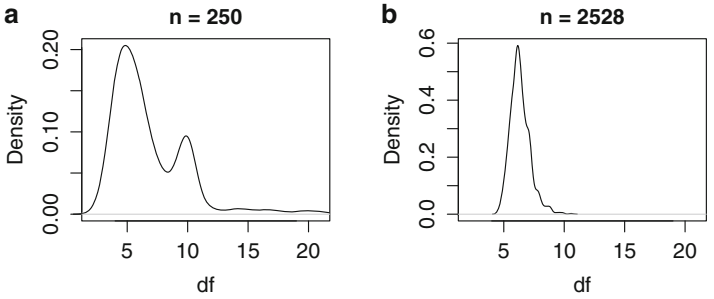

Fig. 6.1. *Kernel density estimates of* 1,000 *bootstrap estimates of df using (**a**) the first* 250 *daily GE returns and (**b**) all* 2,528 *GE returns. The default bandwidth was used in* R*'s* `density` *function to create the estimates.*

However, even with only 250 observations, ν can be estimated accurately enough to show, for example, that for the GE daily returns ν is very likely less than 13, the 98th percentile of the bootstrap distribution of ν. Therefore, the bootstrap provides strong evidence that the normal model corresponding to $\nu = \infty$ is not as satisfactory as a t-model.

By the CLT for the MLE, we know that the MLE is nearly normally distributed for large enough values of n. But this theorem does not tell us how large is large enough. To answer that question, we can use the bootstrap. We have seen here that $n = 250$ is not large enough for near normality of $\widehat{\nu}$, and, though $n = 2{,}528$ is sufficiently large so that the bootstrap distribution is unimodal, there is still some right skewness when $n = 2{,}528$. □

6.3 Bootstrap Confidence Intervals

Besides its use in estimating bias and finding standard errors, the bootstrap is widely used to construct confidence intervals. There are many bootstrap confidence intervals and some are quite sophisticated. We can only describe a few and the reader is pointed to the references in Sect. 6.4 for additional information.

Except in certain simple cases, confidence intervals are based on approximations such as the CLT for the MLE. The bootstrap is based on the approximation of the population's probability distribution using the sample. When a confidence interval uses an approximation, there are two coverage probabilities, the nominal one that is stated and the actual one that is unknown. Only for exact confidence intervals making no use of approximations will the two

probabilities be equal. By the "accuracy" of a confidence interval, we mean the degree of agreement between the nominal and actual coverage probabilities. Even exact confidence intervals such as (A.44) for a normal mean and (A.45) for a normal variance are exact only when the data meet the assumptions exactly, e.g., are exactly normally distributed.

6.3.1 Normal Approximation Interval

Let $\widehat{\theta}$ be an estimate of θ and let $s_{\text{boot}}(\widehat{\theta})$ be the estimate of standard error given by (6.2). Then the normal theory confidence interval for θ is

$$\widehat{\theta} \pm s_{\text{boot}}(\widehat{\theta})\, z_{\alpha/2}, \tag{6.4}$$

where $z_{\alpha/2}$ is the $\alpha/2$-upper quantile of the normal distribution. When $\widehat{\theta}$ is an MLE, this interval is essentially the same as (5.20) except that bootstrap, rather than the Fisher information, is used to find the standard error.

To avoid confusion, it should be emphasized that the normal approximation does not assume that the population is normally distributed but only that $\widehat{\theta}$ is normally distributed by a CLT.

6.3.2 Bootstrap-t Intervals

Often one has available a standard error for $\widehat{\theta}$, for example, from Fisher information. In this case, the bootstrap-t method can be used and, compared to normal approximation confidence intervals, offers the possibility of more accurate confidence intervals, that is, with nominal coverage probability closer to the actual coverage probability. We start by showing how the bootstrap-t method is related to the usual t-based confidence interval for a normal population mean, and then discuss the general theory.

Confidence Intervals for a Population Mean

Suppose we wish to construct a confidence interval for the population mean based on a random sample. One starts with the so-called "t-statistic,"[3] which is

$$t = \frac{\mu - \overline{Y}}{s/\sqrt{n}}. \tag{6.5}$$

The denominator of t, $s/\sqrt{n}$, is just the standard error of the mean, so that the denominator estimates the standard deviation of the numerator.

[3] Actually, t is not quite a statistic since it depends on the unknown μ, whereas a statistic, by definition, is something that depends only on the sample, not on unknown parameters. However, the term "t-statistic" is so widespread that we will use it here.

If we are sampling from a normally distributed population, then the probability distribution of t is known to be the t-distribution with $n-1$ degrees of freedom. Using the notation of Sect. 5.5.2, we denote by $t_{\alpha/2,n-1}$ the $\alpha/2$ upper t-value, that is, the $\alpha/2$-upper quantile of this distribution. Thus, t in (6.5) has probability $\alpha/2$ of exceeding $t_{\alpha/2,n-1}$. Because of the symmetry of the t-distribution, the probability is also $\alpha/2$ that t is less than $-t_{\alpha/2,n-1}$.

Therefore, for normally distributed data, the probability is $1-\alpha$ that

$$-t_{\alpha/2,n-1} \le t \le t_{\alpha/2,n-1}. \tag{6.6}$$

Substituting (6.5) into (6.6), after a bit of algebra we find that

$$1-\alpha = P\left\{\overline{Y} - t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} \le \mu \le \overline{Y} + t_{\alpha/2,n-1}\frac{s}{\sqrt{n}}\right\}, \tag{6.7}$$

which shows that

$$\overline{Y} \pm \frac{s}{\sqrt{n}} t_{\alpha/2,n-1}$$

is a $1-\alpha$ confidence interval for μ, assuming normally distributed data. This is the confidence interval given by Eq. (A.44). Note that in (6.7) the random variables are $\overline{Y}$ and s, and μ is fixed.

What if we are not sampling from a normal distribution? In that case, the distribution of t defined by (6.5) is *not* the t-distribution, but rather some other distribution that is not known to us. There are two problems. First, we do not know the distribution of the population. Second, even if the population distribution were known, it is a difficult, usually intractable, probability calculation to get the distribution of the t-statistic from the distribution of the population. This calculation has only been done for normal populations. Considering the difficulty of these two problems, can we still get a confidence interval? The answer is "yes, by resampling."

We start with a large number, say B, of resamples from the original sample. Let $\overline{Y}_{\text{boot},b}$ and $s_{\text{boot},b}$ be the sample mean and standard deviation of the bth resample, $b = 1, \ldots, B$, and let $\overline{Y}$ be the mean of the original sample. Define

$$t_{\text{boot},b} = \frac{\overline{Y} - \overline{Y}_{\text{boot},b}}{s_{\text{boot},b}/\sqrt{n}}. \tag{6.8}$$

Notice that $t_{\text{boot},b}$ is defined in the same way as t except for two changes. First, $\overline{Y}$ and s in t are replaced by $\overline{Y}_{\text{boot},b}$ and $s_{\text{boot},b}$ in $t_{\text{boot},b}$. Second, μ in t is replaced by $\overline{Y}$ in $t_{\text{boot},b}$. The last point is a bit subtle, and uses the principle stated at the end of Sect. 6.1—a resample is taken using the original sample as the population. Thus, for the resample, the population mean is $\overline{Y}$!

Because the resamples are independent of each other, the collection $t_{\text{boot},1}$, $t_{\text{boot},2}, \ldots$ can be treated as a random sample from the distribution of the t-statistic. After B values of $t_{\text{boot},b}$ have been calculated, one from each resample, we find the $\alpha/2$-lower and -upper quantiles of these $t_{\text{boot},b}$ values. Call these percentiles t_L and t_U.

If the original population is skewed, then there is no reason to suspect that the $\alpha/2$-lower quantile is minus the $\alpha/2$-upper quantile as happens for symmetric populations such as the t-distribution. In other words, we do not necessarily expect that $t_L = -t_U$, but this causes us no problem since the bootstrap allows us to estimate t_L and t_U without assuming any relationship between them. Now we replace $-t_{\alpha/2,n-1}$ and $t_{\alpha/2,n-1}$ in the confidence interval (6.7) by t_L and t_U, respectively. Finally, the bootstrap confidence interval for μ is

$$\left(\overline{Y} + t_L \frac{s}{\sqrt{n}},\ \overline{Y} + t_U \frac{s}{\sqrt{n}}\right). \tag{6.9}$$

In (6.9), $\overline{Y}$ and s are the mean and standard deviation of the original sample, and only t_L and t_U are calculated from the B bootstrap resamples.

The bootstrap has solved both problems mentioned above. One does not need to know the population distribution since we can estimate it by the sample. A sample isn't a probability distribution. What is being done is creating a probability distribution, called the *empirical distribution*, from the sample by giving each observation in the sample probability $1/n$ where n is the sample size. Moreover, one doesn't need to calculate the distribution of the t-statistic using probability theory. Instead we can simulate from the empirical distribution.

Confidence Interval for a General Parameter

The method of constructing a t-confidence interval for μ can be generalized to other parameters. Let $\widehat{\theta}$ and $s(\widehat{\theta})$ be the estimate of θ and its standard error calculated from the sample. Let $\widehat{\theta}^*_b$ and $s_b(\widehat{\theta})$ be the same quantities from the bth bootstrap sample. Then the bth bootstrap t-statistic is

$$t_{\text{boot},b} = \frac{\widehat{\theta} - \widehat{\theta}^*_b}{s_b(\widehat{\theta})}. \tag{6.10}$$

As when estimating a population mean, let t_L and t_U be the $\alpha/2$-lower and $\alpha/2$-upper sample quantiles of these t-statistics. Then the confidence interval for θ is

$$\left(\widehat{\theta} + t_L s(\widehat{\theta}),\ \widehat{\theta} + t_U s(\widehat{\theta})\right)$$

since

$$1 - \alpha \approx P\left\{t_l \le \frac{\widehat{\theta} - \widehat{\theta}^*_b}{s_b(\widehat{\theta})} \le t_U\right\} \tag{6.11}$$

$$\approx P\left\{t_l \le \frac{\theta - \widehat{\theta}}{s(\widehat{\theta})} \le t_U\right\} \tag{6.12}$$

$$= P\left\{\widehat{\theta} + t_L s(\widehat{\theta}) \le \theta \le \widehat{\theta} + t_U s(\widehat{\theta})\right\}.$$

The approximation in (6.11) is due to Monte Carlo error and can be made small by choosing B large. The approximation in (6.12) is from the bootstrap approximation of the population's distribution by the empirical distribution. The error of the second approximation is independent of B and becomes small only as the sample size n becomes large. Though one generally has no control over the sample size, fortunately, sample sizes are often large in financial engineering.

6.3.3 Basic Bootstrap Interval

Let q_L and q_U be the $\alpha/2$-lower and -upper sample quantiles of $\widehat{\theta}_1^*, \ldots, \widehat{\theta}_B^*$. The fraction of bootstrap estimates that satisfy

$$q_L \leq \widehat{\theta}_b^* \leq q_U \tag{6.13}$$

is $1-\alpha$. But (6.13) is algebraically equivalent to

$$\widehat{\theta} - q_U \leq \widehat{\theta} - \widehat{\theta}_b^* \leq \widehat{\theta} - q_L, \tag{6.14}$$

so that $\widehat{\theta} - q_U$ and $\widehat{\theta} - q_L$ are lower and upper quantiles for the distribution of $\widehat{\theta} - \widehat{\theta}_b^*$. The basic bootstrap interval uses them as lower and upper quantiles for the distribution of $\theta - \widehat{\theta}$. Using the bootstrap approximation, it is assumed that

$$\widehat{\theta} - q_U \leq \theta - \widehat{\theta} \leq \widehat{\theta} - q_L \tag{6.15}$$

will occur in a fraction $1-\alpha$ of samples. Adding $\widehat{\theta}$ to each term in (6.15) gives $2\widehat{\theta} - q_U \leq \theta \leq 2\widehat{\theta} - q_L$, so that

$$(2\widehat{\theta} - q_U, 2\widehat{\theta} - q_L) \tag{6.16}$$

is a confidence interval for θ. Interval (6.16) is sometimes called the *basic bootstrap interval.*

6.3.4 Percentile Confidence Intervals

There are several bootstrap confidence intervals based on the so-called percentile method. Only one, the basic percentile interval, in discussed here in detail.

As in Sect. 6.3.3, let q_L and q_U be the $\alpha/2$-lower and -upper sample quantiles of $\widehat{\theta}_1^*, \ldots, \widehat{\theta}_B^*$. The basic percentile confidence interval is simply

$$(q_L,\ q_U). \tag{6.17}$$

By (6.13), the proportion of $\widehat{\theta}_b^*$-values in this interval is $1-\alpha$. This interval can be justified by assuming that $\widehat{\theta}^*$ is distributed symmetrically about $\widehat{\theta}$. This assumption implies that for some $C > 0$, $q_L = \widehat{\theta} - C$ and $q_U = \widehat{\theta} + C$.

Then $2\widehat{\theta} - q_U = q_L$ and $2\widehat{\theta} - q_L = q_U$, so the basic bootstrap interval (6.16) coincides with the basic percentile interval (6.17).

What if $\widehat{\theta}^*$ is not distributed symmetrically about $\widehat{\theta}$? Fortunately, not all is lost. As discussed in Sect. 4.6, often random variables can be transformed to have a symmetric distribution. So, now assume only that for some monotonically increasing function g, $g(\widehat{\theta}^*)$ is symmetrically distributed about $g(\widehat{\theta})$. As we will now see, this weaker assumption is all that is needed to justify the basic percentile interval. Because g is monotonically strictly increasing and quantiles are transformation-respecting,[4] $g(q_L)$ and $g(q_U)$ are lower- and upper-$\alpha/2$ quantiles of $g(\widehat{\theta}_1^*), \ldots, g(\widehat{\theta}_B^*)$, and the basic percentile confidence interval for $g(\theta)$ is

$$\{g(q_L),\ g(q_U)\}. \tag{6.18}$$

Now, if (6.18) has coverage probability $(1-\alpha)$ for $g(\theta)$, then, since g is monotonically increasing, (6.17) has coverage probability $(1-\alpha)$ for θ. This justifies the percentile interval, at least if one is willing to assume the existence of a transformation to symmetry. Note that it is only assumed that such a g exists, not that it is known. No knowledge of g is necessary, since g is not used to construct the percentile interval.

The basic percentile method is simple, but it is not considered very accurate, except for large sample sizes. There are two problems with the percentile method. The first is an assumption of unbiasedness. The basic percentile interval assumes not only that $g(\widehat{\theta}^*)$ is distributed symmetrically, but also that it is symmetric about $g(\widehat{\theta})$ rather than $g(\widehat{\theta})$ plus some bias. Most estimators satisfy a CLT, e.g., the CLTs for sample quantiles and for the MLE in Sects. 4.3.1 and 5.10, respectively. Therefore, bias becomes negligible in large enough samples, but in practice the sample size might not be sufficiently large and bias can cause the nominal and actual coverage probabilities to differ.

The second problem is that $\widehat{\theta}$ may have a nonconstant variance, a problem called heteroskedasticity. If $\widehat{\theta}$ is the MLE, then the variance of $\widehat{\theta}$ is, at least approximately, the inverse of Fisher information and the Fisher information need not be constant—it often depends on θ.

More sophisticated percentile methods can correct for bias and heteroskedasticity. The BC_a and ABC (approximate bootstrap confidence) percentile intervals are improved percentile intervals in common use. In the name "BC_a," "BC" means "bias-corrected" and "a" means "accelerated," which refers to the rate at which the variance changes with θ. The BC_a method automatically estimates both the bias and the rate of change of the variance and then makes suitable adjustments. The theory behind the BC*a* and ABC intervals is beyond the scope of this book, but is discussed in references found in Sect. 6.4. Both the BC_a and ABC methods have been implemented in statistical software such as `R`. In `R`'s `bootstrap` package, the functions `bcanon()`, `abcpar()`, and `abcnon()` implement the nonparametric BC_a, parametric ABC, and nonparametric ABC intervals, respectively.

[4] See Appendix A.2.2.

Example 6.3. Confidence interval for a quantile-based tail-weight parameter

It was mentioned in Sect. 5.8 that a quantile-based parameter quantifying tail weight can be defined as the ratio of two scale parameters:

$$\frac{s(p_1, 1-p_1)}{s(p_2, 1-p_2)}, \tag{6.19}$$

where

$$s(p_1, p_2) = \frac{F^{-1}(p_2) - F^{-1}(p_1)}{a},$$

a is a positive constant that does not affect the ratio (6.19) and so can be ignored, and $0 < p_1 < p_2 < 1/2$. We will call (6.19) `quKurt`. Finding a confidence interval for `quKurt` can be a daunting task without the bootstrap, but with the bootstrap it is simple. In this example, BC_a confidence intervals will be found for `quKurt`. The parameter is computed from a sample `y` by this `R` function, which has default values $p_1 = 0.025$ and $p_2 = 0.25$:

```
quKurt = function(y, p1 = 0.025, p2 = 0.25)
{
Q = quantile(y, c(p1, p2, 1 - p2, 1 - p1))
(Q[4] - Q[1]) / (Q[3] - Q[2])
}
```

The BC_a intervals are found with the `bcanon()` function in the `bootstrap` package using $B = 5{,}000$. The seed of the random number generator was fixed so that these results can be reproduced.

```
bmw = read.csv("bmw.csv")
library("bootstrap")
set.seed("5640")
bca_kurt = bcanon(bmwRet[ ,2], 5000, quKurt)
bca_kurt$confpoints
```

By default, the output gives four pairs of confidence limits.

```
> bca_kurt$confpoints
     alpha bca point
[1,] 0.025      4.07
[2,] 0.050      4.10
[3,] 0.100      4.14
[4,] 0.160      4.18
[5,] 0.840      4.41
[6,] 0.900      4.45
[7,] 0.950      4.50
[8,] 0.975      4.54
```

The results above show, for example, that the 90 % BC_a confidence interval is (4.10, 4.50). For reference, any normal distribution has `quKurt` equal 2.91, so these data have heavier than Gaussian tails, at least as measured by `quKurt`. □

Example 6.4. Confidence interval for the ratio of two quantile-based tail-weight parameters

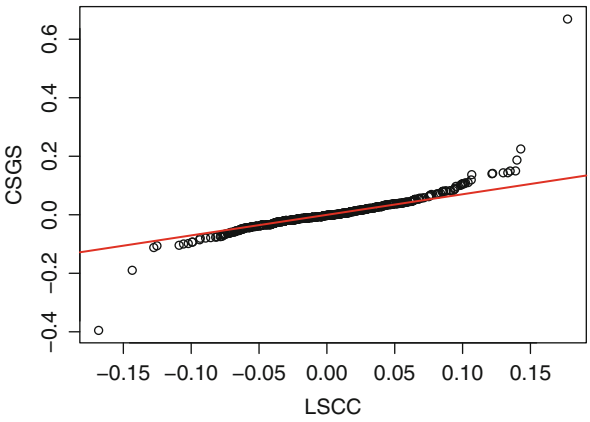

Fig. 6.2. *QQ plot of returns on two stocks in the* `midcapD.ts` *data set. The reference line goes through the first and third quartiles.*

This example uses the data set `midcapD.ts.csv` of returns on midcap stocks. Two of the stocks in this data set are LSCC and CSGS. From Fig. 6.2, which is a QQ plot comparing the returns from these two companies, it appears that LSCC returns have lighter tails than CSGS returns. The values of `quKurt` are 2.91 and 4.13 for LSCC and GSGS, respectively, and the ratio of the two values is 0.704. This is further evidence that LSCC returns have the lesser tail weight. A BC_a confidence interval for the ratio of `quKurt` for LSCC and CSGS is found with the following `R` program.

```
midcapD.ts = read.csv("midcapD.ts.csv")
attach(midcapD.ts)
quKurt = function(y, p1 = 0.025, p2 = 0.25)
{
   Q = quantile(y, c(p1, p2, 1 - p2, 1 - p1))
   as.numeric((Q[4] - Q[1]) / (Q[3] - Q[2]))
}
compareQuKurt = function(x, p1 = 0.025, p2 = 0.25, xdata)
{
   quKurt(xdata[x,1], p1, p2) / quKurt(xdata[x,2], p1, p2)
}
quKurt(LSCC)
quKurt(CSGS)
xdata = cbind(LSCC, CSGS)
compareQuKurt(1:n, xdata = xdata)
library("bootstrap")
```

```
17 set.seed("5640")
18 bca_kurt = bcanon((1:n), 5000, compareQuKurt, xdata = xdata)
19 bca_kurt$confpoints
```

The function `compareQuKurt()` (lines 8–11) computes a `quKurt` ratio. The function `bcanon()` is designed to bootstrap a vector, but this example has bivariate data in a matrix with two columns. To bootstrap multivariate data, there is a trick given in `R`'s help for `bcanon()`—bootstrap the integers 1 to n where n is the sample size. This is done at line 18. The resamples of $1, \ldots, n$ allow one to resample the rows of the data vector. Thus, in this example `bcanon()` draws a random sample with replacement from $1, \ldots, n$ and selects the rows of `xdata` corresponding to these indices to create a resample.

The 95 % confidence interval for the `quKurt` ratio is 0.587 to 0.897, so with 95 % confidence it can be concluded that LSCC has a smaller value of `quKurt`.

```
> bca_kurt$confpoints
     alpha bca point
[1,] 0.025     0.587
[2,] 0.050     0.607
[3,] 0.100     0.634
[4,] 0.160     0.653
[5,] 0.840     0.811
[6,] 0.900     0.833
[7,] 0.950     0.864
[8,] 0.975     0.897
```

□

6.4 Bibliographic Notes

Efron (1979) introduced the name "bootstrap" and did much to popularize resampling methods. Efron and Tibshirani (1993), Davison and Hinkley (1997), Good (2005), and Chernick (2007) are introductions to the bootstrap that discuss many topics not treated here, including the theory behind the BC_a and ABC methods for confidence intervals. The `R` package `bootstrap` is described by its authors as "functions for Efron and Tibshirani (1993)" and the package contains the data sets used in that book. The `R` package `boot` is a more recent set of resampling functions and data sets to accompany Davison and Hinkley (1997).

6.5 `R` Lab

6.5.1 BMW Returns

This lab uses a data set containing 6146 daily returns on BMW stock from January 3, 1973 to July 23, 1996. Run the following code to fit a skewed t-distribution to the returns and check the fit with a QQ plot.

```
1  library("fGarch")
2  bmwRet = read.csv("bmwRet.csv")
3  n = dim(bmwRet)[1]
4
5  kurt = kurtosis(bmwRet[ ,2], method = "moment")
6  skew = skewness(bmwRet[,2], method = "moment")
7  fit_skewt = sstdFit(bmwRet[ ,2])
8
9  q.grid = (1:n) / (n+1)
10 qqplot(bmwRet[ ,2], qsstd(q.grid, fit_skewt$estimate[1],
11    fit_skewt$estimate[2],
12    fit_skewt$estimate[3], fit_skewt$estimate[4]),
13    ylab = "skewed-t quantiles" )
```

The function `qsstd()` is in the `fGarch` package loaded at line 1. The required package `timeDate` is also loaded and the function `kurtosis()` is in `timeDate`.

Problem 1 *What is the MLE of ν? Does the t-distribution with this value of ν have a finite skewness and kurtosis?*

Since the kurtosis coefficient based on the fourth central moment is infinite for some distributions, as in Sect. 6.4 we will define a quantile-based kurtosis:

$$\text{quKurt(F)} = \frac{F^{-1}(1-p_1) - F^{-1}(p_1)}{F^{-1}(1-p_2) - F^{-1}(p_2)},$$

where F is a CDF and $0 < p_1 < p_2 < 1/2$. Typically, p_1 is close to zero so that the numerator is sensitive to tail weight and p_2 is much larger and measures dispersion in the center of the distribution. Because the numerator and denominator of quKurt are each the difference between two quantiles, they are location-free and therefore scale parameters. Moreover, because quKurt is a ratio of two scale parameters, it is scale-free and therefore a shape parameter. A typical example would be $p_1 = 0.025$ and $p_2 = 0.25$. `quKurt` is estimated by replacing the population quantiles by sample quantiles.

Problem 2 *Write an* `R` *program to plot* `quKurt` *for the t-distribution as a function of ν. Use $p_1 = 0.025$ and $p_2 = 0.25$. Let ν take values from 1 to 10, incremented by 0.25. If you want to get fancy while labeling the axes,* `xlab=expression(nu)` *in the call to* `plot` *will put a "ν" on the x-axis.*

Run the following code, which defines a function to compute quKurt and bootstraps this function on the BMW returns. Note that p_1 and p_2 are given default values that are used in the bootstrap and that both model-free and model-based bootstrap samples are taken.

```
quKurt = function(y, p1 = 0.025, p2 = 0.25)
{
    Q = quantile(y, c(p1, p2, 1 - p2, 1 - p1))
    k = (Q[4] - Q[1]) / (Q[3] - Q[2])
    k
}
nboot = 5000
ModelFree_kurt  = rep(0, nboot)
ModelBased_kurt = rep(0, nboot)
set.seed("5640")
for (i in 1:nboot)
{
    samp_ModelFree = sample(bmwRet[,2], n, replace = TRUE)
    samp_ModelBased = rsstd(n, fit_skewt$estimate[1],
      fit_skewt$estimate[2],
      fit_skewt$estimate[3], fit_skewt$estimate[4])
    ModelFree_kurt[i] = quKurt(samp_ModelFree)
    ModelBased_kurt[i] = quKurt(samp_ModelBased)
}
```

Problem 3 *Plot KDEs of* `ModelFree_kurt` *and* `ModelBased_kurt`*. Also, plot side-by-side boxplots of the two samples. Describe any major differences between the model-based and model-free results. Include the plots with your work.*

Problem 4 *Find 90 % percentile method bootstrap confidence intervals for quKurt using the model-based and model-free bootstraps.*

Problem 5 *BC_a confidence intervals can be constructed using the function* `bcanon()` *in* `R`*'s* `bootstrap` *package. Find a 90 % BC_a confidence interval for quKurt. Use 5,000 resamples. Compare the BC_a interval to the model-free percentile interval from Problem 4.*

6.5.2 Simulation Study: Bootstrapping the Kurtosis

The sample kurtosis is highly variable because it is based on the 4th moment. As a result, it is challenging to construct an accurate confidence interval for the kurtosis. In this section, five bootstrap confidence intervals for the kurtosis will be compared. The comparisons will be on widths of the intervals, where smaller is better, and actual coverage probabilities, where closer to nominal is better.

Run the following code. Warning: this simulation experiment takes a while to run, e.g., 5 to 10 minutes, and will have only moderate accuracy. To increase the accuracy, you might wish to increase `niter` and `nboot` and run the experiment over a longer period, even overnight.

```
library(bootstrap)
Kurtosis = function(x) mean(((x - mean(x)) / sd(x))^4)
set.seed(3751)
niter = 500
nboot = 400
n = 50
nu = 10
trueKurtosis =  3 + 6 / (nu - 4)
correct = matrix(nrow = niter, ncol = 5)
width   = matrix(nrow = niter, ncol = 5)
error = matrix(nrow = niter, ncol = 1)
t1 = proc.time()
for (i in 1:niter){
   y = rt(n,nu)
   int1 = bootstrap(y, Kurtosis, nboott = nboot,
      nbootsd = 50)$confpoints[c(3, 9)]
   width[i,1] = int1[2] - int1[1]
   correct[i,1] = as.numeric((int1[1] < trueKurtosis) &
      (trueKurtosis < int1[2]))
   int2 = bcanon(y, nboot, Kurtosis)$confpoints[c(1, 8), 2]
   width[i,2] = int2[2] - int2[1]
   correct[i,2] = as.numeric((int2[1] < trueKurtosis) &
      (trueKurtosis < int2[2]))
   boot = bootstrap(y, nboot, Kurtosis)$thetastar
   int3 = Kurtosis(y) + 1.96 * c(-1, 1) * sd(boot)
   width[i,3] = int3[2] - int3[1]
   correct[i,3] = as.numeric((int3[1] < trueKurtosis) &
      (trueKurtosis < int3[2]))
   int4 = quantile(boot, c(0.025, 0.975))
   width[i,4] = int4[2] - int4[1]
   correct[i,4] = as.numeric((int4[1] < trueKurtosis) &
      (trueKurtosis < int4[2]))
   int5 = 2*Kurtosis(y) - quantile(boot, c(0.975, 0.025))
   width[i,5] = int5[2] - int5[1]
   correct[i,5] = as.numeric((int5[1] < trueKurtosis) &
      (trueKurtosis < int5[2]))
   error[i] = mean(boot) - Kurtosis(y)
}
t2 = proc.time()
(t2 - t1)/60
colMeans(width)
colMeans(correct)
options(digits = 3)
mean(error)
mean(error^2)
```

Problem 6 *Which five bootstrap intervals are being used here?*

Problem 7 *What is the value of B here?*

Problem 8 *How many simulations are used?*

Problem 9 *What are the estimates of bias?*

Problem 10 *What is the estimated MSE?*

Problem 11 *Estimate the actual coverage probability of the BC_a and bootstrap-t intervals. (Because this is a simulation experiment, it is subject to Monte Carlo errors, so the coverage probability is only estimated.)*

Problem 12 *Find a 95 % confidence interval for the actual coverage probability of the BC_a interval?*

Problem 13 *Which interval is most accurate? Would you consider any of the intervals as highly accurate?*

Problem 14 *How much clock time did the entire simulation take?*

As mentioned, kurtosis is difficult to estimate because it is based on the 4th moment and a quantile-based measure of tailweight might be a better alternative. The next problem investigates this conjecture.

Problem 15 *Repeat the simulation experiment with kurtosis replaced by* `quKurt()` *defined is Sect. 6.5.1 Which interval is most accurate now? Would you consider any of the intervals as highly accurate?*

6.6 Exercises

1. To estimate the risk of a stock, a sample of 50 log returns was taken and s was 0.31. To get a confidence interval for σ, 10,000 resamples were taken. Let $s_{b,\text{boot}}$ be the sample standard deviation of the bth resample. The 10,000 values of $s_{b,\text{boot}}/s$ were sorted and the table below contains selected values of $s_{b,\text{boot}}/s$ ranked from smallest to largest (so rank 1 is the smallest and so forth).

Rank	Value of $s_{b,\text{boot}}/s$
250	0.52
500	0.71
1,000	0.85
9,000	1.34
9,500	1.67
9,750	2.19

Find a 90 % confidence interval for σ.

2. In the following R program, resampling was used to estimate the bias and variance of the sample correlation between the variables in the vectors x and y.

```
samplecor = cor(x, y)
n = length(x)
nboot = 5000
resamplecor = rep(0, nboot)
for (b in (1:nboot))
{
   ind = sample(1:n, replace = TRUE)
   resamplecor[b] = cor(x[ind], y[ind])
}
samplecor
mean(resamplecor)
sd(resamplecor)
```

The output is

```
> n
[1] 20
> samplecor
[1] 0.69119
> mean(resamplecor)
[1] 0.68431
> sd(resamplecor)
[1] 0.11293
```

(a) Estimate the bias of the sample correlation coefficient.
(b) Estimate the standard deviation of the sample correlation coefficient.
(c) Estimate the MSE of the sample correlation coefficient.
(d) What fraction of the MSE is due to bias? How serious is the bias? Should something be done to reduce the bias? Explain your answer.

3. The following R code was used to bootstrap the sample standard deviation.

```
( code to read the variable x )
sampleSD = sd(x)
n = length(x)
nboot = 15000
resampleSD = rep(0, nboot)
```

```
for (b in (1:nboot))
{
   resampleSD[b] = sd(sample(x, replace = TRUE))
}
options(digits = 4)
sampleSD
mean(resampleSD)
sd(resampleSD)
```

The output is

```
> sampleSD
[1] 1.323
> mean(resampleSD)
[1] 1.283
> sd(resampleSD)
[1] 0.2386
```

(a) Estimate the bias of the sample standard deviation of `x`.
(b) Estimate the mean squared error of the sample standard deviation of `x`.

References

Chernick, M. R. (2007) *Bootstrap Methods: A Guide for Practitioners and Researchers,* 2nd ed., Wiley-Interscience, Hoboken, NJ.

Davison, A. C., and Hinkley, D. V. (1997) *Bootstrap Methods and Their Applications*, Cambridge University Press, Cambridge.

Efron, B. (1979) Bootstrap methods: Another look at the jackknife. *Annals of Statistics*, **7**, 1–26.

Efron, B., and Tibshirani, R. (1993) *An Introduction to the Bootstrap*, Chapman & Hall, New York.

Good, P. I. (2005) *Resampling Methods: A Practical Guide to Data Analysis,* 3rd ed., Birkhauser, Boston.