

# Bootstrapping Regression Models in R

An Appendix to *An R Companion to Applied Regression, Second Edition*

John Fox & Sanford Weisberg

last revision: 12 November 2010

## Abstract

*Bootstrapping* is a general approach to statistical inference based on building a sampling distribution for a statistic by resampling from the data at hand. This appendix to Fox and Weisberg (2011) briefly describes the rationale for the bootstrap and explains how to bootstrap regression models using the **boot** package in R.

## 1 Basic Ideas

*Bootstrapping* is a general approach to statistical inference based on building a sampling distribution for a statistic by resampling from the data at hand. The term “bootstrapping,” due to Efron (1979), is an allusion to the expression “pulling oneself up by one’s bootstraps” — in this case, using the sample data as a population from which repeated samples are drawn. At first blush, the approach seems circular, but has been shown to be sound.

Two R packages for bootstrapping are associated with extensive treatments of the subject: Efron and Tibshirani’s (1993) **bootstrap** package, and Davison and Hinkley’s (1997) **boot** package. Of the two, **boot**, programmed by A. J. Canty, is somewhat more capable, and will be used for the examples in this appendix.<sup>1</sup>

There are several forms of the bootstrap, and, additionally, several other resampling methods that are related to it, such as *jackknifing*, *cross-validation*, *randomization tests*, and *permutation tests*. We will stress the *nonparametric bootstrap*.

Suppose that we draw a sample  $\mathbf{S} = \{X_1, X_2, \dots, X_n\}$  from a population  $\mathbf{P} = \{x_1, x_2, \dots, x_N\}$ ; imagine further, at least for the time being, that  $N$  is very much larger than  $n$ , and that  $\mathbf{S}$  is either a simple random sample or an independent random sample from  $\mathbf{P}$ ;<sup>2</sup> We will briefly consider other sampling schemes at the end of the appendix. It will also help initially to think of the elements of the population (and, hence, of the sample) as scalar values, but they could just as easily be vectors (i.e., multivariate).

Now suppose that we are interested in some statistic  $T = t(\mathbf{S})$  as an estimate of the corresponding population parameter  $\theta = t(\mathbf{P})$ . Again,  $\theta$  could be a vector of parameters and  $T$  the corresponding vector of estimates, but for simplicity assume that  $\theta$  is a scalar. A traditional approach to statistical inference is to make assumptions about the structure of the population (e.g., an assumption of normality), and, along with the stipulation of random sampling, to use these assumptions to derive the sampling distribution of  $T$ , on which classical inference is based. In certain

---

<sup>1</sup>The **bootCase** function in the **car** package, introduced in Section 4.3.7 of the *R Companion* implements the case-resampling bootstrap (described below). The Appendix provides a broader treatment than in the text of bootstrapping in R.

<sup>2</sup>Alternatively,  $\mathbf{P}$  could be an infinite population, specified, for example, by a probability distribution function.

instances, the exact distribution of  $T$  may be intractable, and so we instead derive its asymptotic distribution. This familiar approach has two potentially important deficiencies:

1. If the assumptions about the population are wrong, then the corresponding sampling distribution of the statistic may be seriously inaccurate. On the other hand, if asymptotic results are relied upon, these may not hold to the required level of accuracy in a relatively small sample.
2. The approach requires sufficient mathematical prowess to derive the sampling distribution of the statistic of interest. In some cases, such a derivation may be prohibitively difficult.

In contrast, the nonparametric bootstrap allows us to estimate the sampling distribution of a statistic empirically without making assumptions about the form of the population, and without deriving the sampling distribution explicitly. The essential idea of the nonparametric bootstrap is as follows: We proceed to draw a sample of size  $n$  from among the elements of  $\mathbf{S}$ , sampling with replacement. Call the resulting *bootstrap sample*  $\mathbf{S}_1^* = \{X_{11}^*, X_{12}^*, \dots, X_{1n}^*\}$ . It is necessary to sample with replacement, because we would otherwise simply reproduce the original sample  $\mathbf{S}$ . In effect, we are treating the sample  $\mathbf{S}$  as an estimate of the population  $\mathbf{P}$ ; that is, each element  $X_i$  of  $\mathbf{S}$  is selected for the bootstrap sample with probability  $1/n$ , mimicking the original selection of the sample  $\mathbf{S}$  from the population  $\mathbf{P}$ . We repeat this procedure a large number of times,  $R$ , selecting many bootstrap samples; the  $b$ th such bootstrap sample is denoted  $\mathbf{S}_b^* = \{X_{b1}^*, X_{b2}^*, \dots, X_{bn}^*\}$ .

The key bootstrap analogy is therefore as follows:

**The population is to the sample  
as  
the sample is to the bootstrap samples.**

Next, we compute the statistic  $T$  for each of the bootstrap samples; that is  $T_b^* = t(\mathbf{S}_b^*)$ . Then the distribution of  $T_b^*$  around the original estimate  $T$  is analogous to the sampling distribution of the estimator  $T$  around the population parameter  $\theta$ . For example, the average of the bootstrapped statistics,

$$\bar{T}^* = \widehat{E}^*(T^*) = \frac{\sum_{b=1}^R T_b^*}{R}$$

estimates the expectation of the bootstrapped statistics; then  $\widehat{B}^* = \bar{T}^* - T$  is an estimate of the bias of  $T$ , that is,  $T - \theta$ . Similarly, the estimated bootstrap variance of  $T^*$ ,

$$\widehat{\text{Var}}^*(T^*) = \frac{\sum_{b=1}^R (T_b^* - \bar{T}^*)^2}{R - 1}$$

estimates the sampling variance of  $T$ .

The random selection of bootstrap samples is not an essential aspect of the nonparametric bootstrap: At least in principle, we could enumerate *all* bootstrap samples of size  $n$ . Then we could calculate  $E^*(T^*)$  and  $\text{Var}^*(T^*)$  exactly, rather than having to estimate them. The number

of bootstrap samples, however, is astronomically large unless  $n$  is tiny.<sup>3</sup> There are, therefore, two sources of error in bootstrap inference: (1) the error induced by using a particular sample  $\mathbf{S}$  to represent the population; and (2) the sampling error produced by failing to enumerate all bootstrap samples. The latter source of error can be controlled by making the number of bootstrap replications  $R$  sufficiently large.

## 2 Bootstrap Confidence Intervals

There are several approaches to constructing bootstrap confidence intervals. The *normal-theory interval* assumes that the statistic  $T$  is normally distributed (which is often approximately the case for statistics in sufficiently large samples), and uses the bootstrap estimate of sampling variance, and perhaps of bias, to construct a  $100(1 - \alpha)$ -percent confidence interval of the form

$$\theta = (T - \widehat{B}^*) \pm z_{1-\alpha/2} \widehat{\text{SE}}^*(T^*)$$

Here,  $\widehat{\text{SE}}^*(T^*) = \sqrt{\widehat{\text{Var}}^*(T^*)}$  is the bootstrap estimate of the standard error of  $T$ , and  $z_{1-\alpha/2}$  is the  $1 - \alpha/2$  quantile of the standard-normal distribution (e.g., 1.96 for a 95-percent confidence interval, where  $\alpha = .05$ ).

An alternative approach, called the *bootstrap percentile interval*, is to use the empirical quantiles of  $T_b^*$  to form a confidence interval for  $\theta$ :

$$T_{(\text{lower})}^* < \theta < T_{(\text{upper})}^*$$

where  $T_{(1)}^*, T_{(2)}^*, \dots, T_{(R)}^*$  are the ordered bootstrap replicates of the statistic; lower =  $[(R + 1)\alpha/2]$ ; upper =  $[(R + 1)(1 - \alpha/2)]$ ; and the square brackets indicate rounding to the nearest integer. For example, if  $\alpha = .05$ , corresponding to a 95-percent confidence interval, and  $R = 999$ , then lower = 25 and upper = 975.

Although they do not artificially assume normality, percentile confidence intervals often do not perform well. So-called *bias-corrected*, *accelerated* (or  $BC_a$ ) *percentile intervals* are preferable. To find the  $BC_a$  interval for  $\theta$ :

- Calculate

$$z = \Phi^{-1} \left[ \frac{\sum_{b=1}^R \mathbb{1}(T_b^* \leq T)}{R + 1} \right]$$

where  $\Phi^{-1}(\cdot)$  is the standard-normal quantile function, and  $\#(T_b^* \leq T)/(R + 1)$  is the (adjusted) proportion of bootstrap replicates at or below the original-sample estimate  $T$  of  $\theta$ . If the bootstrap sampling distribution is symmetric, and if  $T$  is unbiased, then this proportion will be close to .5, and the correction factor  $z$  will be close to 0.

- Let  $T_{(-i)}$  represent the value of  $T$  produced when the  $i$ th observation is deleted from the sample;<sup>4</sup> there are  $n$  of these quantities. Let  $\bar{T}$  represent the average of the  $T_{(-i)}$ ; that is

<sup>3</sup>If we distinguish the order of elements in the bootstrap samples and treat all of the elements of the original sample as distinct (even when some have the same values) then there are  $n^n$  bootstrap samples, each occurring with probability  $1/n^n$ .

<sup>4</sup>The  $T_{(-i)}$  are called the *jackknife values* of the statistic  $T$ . Although we will not pursue the subject here, the jackknife values can also be used as an alternative to the bootstrap to find a nonparametric confidence interval for  $\theta$ .

$\bar{T} = \sum_{i=1}^n T_{(-i)}/n$ . Then calculate

$$a = \frac{\sum_{i=1}^n (\bar{T} - T_{(-i)})^3}{6 \left[ \sum_{i=1}^n (T_{(-i)} - \bar{T})^2 \right]^{\frac{3}{2}}}$$

- With the correction factors  $z$  and  $a$  in hand, compute

$$a_1 = \Phi \left[ z + \frac{z - z_{1-\alpha/2}}{1 - a(z - z_{1-\alpha/2})} \right]$$

$$a_2 = \Phi \left[ z + \frac{z + z_{1-\alpha/2}}{1 - a(z + z_{1-\alpha/2})} \right]$$

where  $\Phi(\cdot)$  is the standard-normal cumulative distribution function. The values  $a_1$  and  $a_2$  are used to locate the endpoints of the corrected percentile confidence interval:

$$T_{(\text{lower}^*)}^* < \theta < T_{(\text{upper}^*)}^*$$

where  $\text{lower}^* = [Ra_1]$  and  $\text{upper}^* = [Ra_2]$ . When the correction factors  $a$  and  $z$  are both 0,  $a_1 = \Phi(-z_{1-\alpha/2}) = \Phi(z_{\alpha/2}) = \alpha/2$ , and  $a_2 = \Phi(z_{1-\alpha/2}) = 1 - \alpha/2$ , which corresponds to the (uncorrected) percentile interval.

To obtain sufficiently accurate 95-percent bootstrap percentile or  $BC_a$  confidence intervals, the number of bootstrap samples,  $R$ , should be on the order of 1000 or more; for normal-theory bootstrap intervals we can get away with a smaller value of  $R$ , say, on the order of 100 or more, because all we need to do is estimate the standard error of the statistic.

### 3 Bootstrapping Regressions

Recall (from Chapters 1 and 6 in the text) Duncan's regression of prestige on income and education for 45 occupations. In the Appendix on robust regression, we refit this regression using an  $M$ -estimator with the Huber weight function (via the `rlm` function in the **MASS** package):

```
> library(car)      # for Duncan data and (later) dataEllipse
> library(MASS)     # for rlm
> mod.duncan.hub <- rlm(prestige ~ income + education, data=Duncan)
> summary(mod.duncan.hub)
```

```
Call: rlm(formula = prestige ~ income + education, data = Duncan)
```

```
Residuals:
```

```
   Min       1Q   Median       3Q      Max
-30.12  -6.89   1.29   4.59  38.60
```

```
Coefficients:
```

```
              Value Std. Error t value
(Intercept) -7.111   3.881    -1.832
income        0.701   0.109     6.452
education     0.485   0.089     5.438
```

```
Residual standard error: 9.89 on 42 degrees of freedom
```

The coefficient standard errors reported by `rlm` rely on asymptotic approximations, and may not be trustworthy in a sample of size 45. Let us turn, therefore, to the bootstrap.

There are two general ways to bootstrap a regression like this: We can treat the predictors as *random*, potentially changing from sample to sample, or as *fixed*. We will deal with each case in turn, and then compare the two approaches. For reasons that should become clear in the subsequent sections, random- $x$  resampling is also called *case resampling*, and fixed- $x$  resampling is also called *model-based resampling*.

### 3.1 Random- $x$ Resampling

Broadening the scope of the discussion, assume that we want to fit a regression model with response variable  $y$  and predictors  $x_1, x_2, \dots, x_k$ . We have a sample of  $n$  observations  $\mathbf{z}'_i = (y_i, x_{i1}, x_{i2}, \dots, x_{ik})$ ,  $i = 1, \dots, n$ . In random- $x$  resampling, we simply select  $R$  bootstrap samples of the  $\mathbf{z}'_i$ , fitting the model and saving the coefficients from each bootstrap sample. We can then construct confidence intervals for the regression coefficients using the methods described in the preceding section.

Although it is not hard to program bootstrap calculations directly in R, it is more convenient to use the `boot` function in the `boot` package.<sup>5</sup> This function takes several arguments, the first three of which are required:

- **data**: A data vector, matrix, or data frame to which bootstrap resampling is to be applied. Each element of the data vector, or each row of the matrix or data frame, is treated as an observation.
- **statistic**: A function that returns the (possibly vector-valued) statistic to be bootstrapped. The first two arguments of this function must specify the original (sample) data set and an *index vector*, giving the indices of the observations included in the current bootstrap sample.
- **R**: the number of bootstrap replications.

Some of the other arguments of `boot` are described below. For complete information, examine the on-line help for `boot`.

To use `boot`, we therefore have to write a function that returns the **statistic** to be bootstrapped. For example, for the Huber  $M$ -estimator of Duncan's regression,

```
> boot.huber <- function(data, indices, maxit=20){
+   data <- data[indices, ] # select obs. in bootstrap sample
+   mod <- rlm(prestige ~ income + education, data=data, maxit=maxit)
+   coefficients(mod) # return coefficient vector
+ }
```

The function `boot.huber` takes a third argument, `maxiter`, which sets the maximum number of iterations to perform in each  $M$ -estimation; we included this provision because we found that the default of 20 iterations in `rlm` is not always sufficient. The `boot` function is able to pass additional arguments through to the function specified in its **statistic** argument:

```
> set.seed(12345) # for reproducibility
> library(boot)
> system.time(duncan.boot <- boot(Duncan, boot.huber, 1999, maxit=200))
```

---

<sup>5</sup>As mentioned, the `bootCase` function in the `car` package also implements random- $x$  resampling.

```
user  system elapsed
24.39   0.00  24.52
```

```
> duncan.boot
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP
```

```
Call:
```

```
boot(data = Duncan, statistic = boot.huber, R = 1999, maxit = 200)
```

```
Bootstrap Statistics :
```

	original	bias	std. error
t1*	-7.1107	0.139653	3.1001
t2*	0.7014	-0.012739	0.1785
t3*	0.4854	0.006993	0.1390

We ran `boot` within a call to the function `system.time` to provide a sense of how long a bootstrapping operation like this takes — in this case, generating  $R = 1999$  bootstrap replicates of the regression coefficients. The first number returned by `system.time` is the CPU (processing) time for the operation, in seconds, while the third number is the total elapsed time. Here, both CPU and elapsed time are less than half a minute.<sup>6</sup> Although this is a small problem, the time spent depends more upon the number of bootstrap samples than upon the sample size. There are two ways to think about waiting 24 seconds for the bootstrapping to take place: On the one hand, we tend to be spoiled by the essentially instantaneous response that R usually provides, and by this standard, 24 seconds seems a long time; on the other hand, bootstrapping is not an exploratory procedure, and 24 seconds is a trivial proportion of the time typically spent on a statistical investigation.

The `boot` function returns an object, here `boot.duncan`, of class "boot". Printing the object gives the original sample value for each component of the bootstrapped statistic, along with the bootstrap estimates of bias (i.e., the difference  $\bar{T}^* - T$  between the average bootstrapped value of the statistic and its original-sample value), and the bootstrap estimates of standard error  $[\widehat{SE}^*(T^*)]$ . As explained in the previous section, these values may be used to construct normal-theory confidence intervals for the regression coefficients (see below). Notice that the bootstrap standard errors of the `income` and `education` coefficients are substantially larger than the asymptotic estimates reported by `rlm`.

In addition to a `print` method, the `boot` package contains several useful functions for examining and manipulating `boot` objects. The `boot.array` function, for example, returns an  $R \times n$  matrix; the entry in row  $b$ , column  $i$  indicates how many times the  $i$ th observation appears in the  $b$ th bootstrap sample:

---

<sup>6</sup>These calculations were performed with R version 2.9.0 under Windows Vista, running on a 32-bit system with a 2.4GHz quad-core processor and the maximum 4 GB of memory.

```

> duncan.array <- boot.array(duncan.boot)
> duncan.array[1:2, ]

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
[1,]    0    2    1    2    0    0    1    0    2    1    3    2    3    1
[2,]    2    2    1    1    2    0    0    0    0    2    2    1    0    2
      [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
[1,]     2     0     1     1     1     2     1     1     0     1     0     1
[2,]     1     3     1     0     0     0     0     0     0     2     2     1
      [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38]
[1,]     3     2     0     0     1     1     1     0     0     2     1     0
[2,]     0     1     0     2     0     2     1     2     3     0     2     1
      [,39] [,40] [,41] [,42] [,43] [,44] [,45]
[1,]     0     0     2     2     1     0     0
[2,]     1     1     1     1     1     0     1

```

Thus, for example, observations 1 appears twice in the second bootstrap sample, but not at all in the first sample.

Plotting a `boot` object draws a histogram and normal quantile-comparison plot of the bootstrap replications for one of the coefficients, selected via the `index` argument:

```

> plot(duncan.boot, index=2) # income coef.
> plot(duncan.boot, index=3) # education coef.

```

The resulting graphs are shown in Figure 1. The bootstrap distributions of the `income` and `education` coefficients are both reasonably symmetric but somewhat heavy-tailed, and appear to have more than one mode.

We next use the `dataEllipse` function from the `car` package to examine the joint distribution of the bootstrapped `income` and `education` coefficients. The function draws a scatterplot of the pairs of coefficients, with concentration ellipses superimposed (Figure 2):

```

> dataEllipse(duncan.boot$t[, 2], duncan.boot$t[, 3],
+   xlab="income coefficient", ylab="education coefficient",
+   cex=0.3, levels=c(.5, .95, .99), robust=TRUE)

```

The bootstrap replicates of the regression coefficients are stored in the `t` component of the `boot` object `duncan.boot`; this component is a matrix containing one row for each bootstrap replication and one column for each coefficient. The negative correlation of the `income` and `education` coefficients reflects the *positive* correlation between the two predictors.

The `boot.ci` function calculates several kinds of bootstrap confidence intervals, including the bias-corrected normal-theory, percentile, and  $BC_a$  intervals described in the previous section:

```

# income coefficient:
> boot.ci(duncan.boot, index=2, type=c("norm", "perc", "bca"))

```

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
Based on 1999 bootstrap replicates

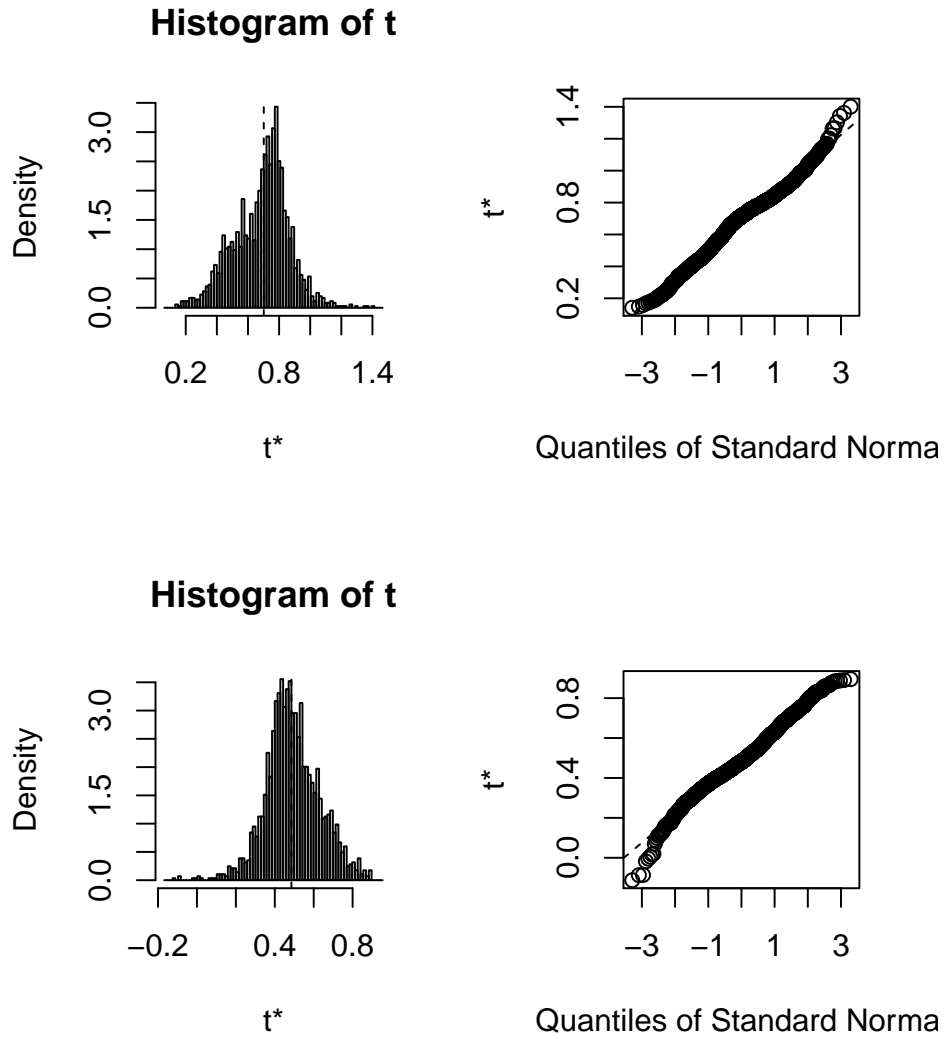


Figure 1: Histograms and normal quantile-comparison plots for the bootstrap replications of the `income` (top) and `education` (bottom) coefficients in the Huber regression for Duncan's occupational-prestige data. The broken vertical line in each histogram shows the location of the regression coefficient for the model fit to the original sample.

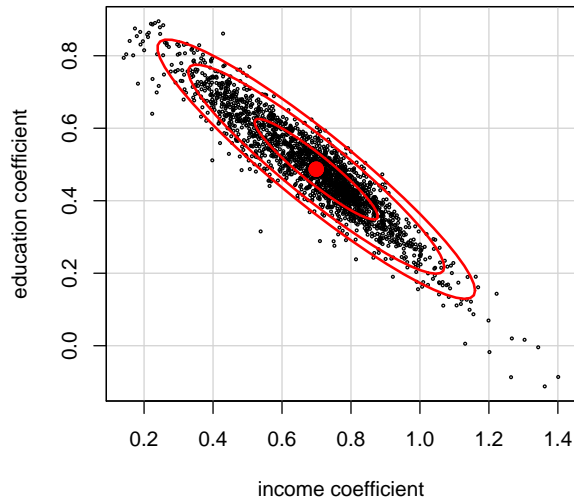


Figure 2: Scatterplot of bootstrap replications of the `income` and `education` coefficients from the Huber regression for Duncan's occupational-prestige data. The concentration ellipses are drawn at the 50, 90, and 99-percent levels using a robust estimate of the covariance matrix of the coefficients.

CALL :

```
boot.ci(boot.out = duncan.boot, type = c("norm", "perc", "bca"),
        index = 2)
```

Intervals :

Level	Normal	Percentile	BCa
95%	( 0.3643, 1.0641 )	( 0.3160, 1.0192 )	( 0.2210, 0.9415 )

Calculations and Intervals on Original Scale

```
> # education coefficient:
```

```
> boot.ci(duncan.boot, index=3, type=c("norm", "perc", "bca"))
```

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

Based on 1999 bootstrap replicates

CALL :

```
boot.ci(boot.out = duncan.boot, type = c("norm", "perc", "bca"),
        index = 3)
```

Intervals :

Level	Normal	Percentile	BCa
95%	( 0.2061, 0.7508 )	( 0.2210, 0.7795 )	( 0.2743, 0.8311 )

Calculations and Intervals on Original Scale

The `index` argument to `boot.ci` selects the coefficient for which confidence intervals are to be constructed, while `type` specifies the type of intervals; the confidence level for the intervals is set via

the `conf` argument, which defaults to 0.95 (for 95-percent intervals). In this example, the normal-theory and percentile intervals are reasonably similar to each other, but the more trustworthy  $BC_a$  intervals are somewhat different.

The (unfortunately named) `jack.after.boot` function displays a diagnostic *jackknife-after-bootstrap* plot. This plot shows the sensitivity of the statistic and of the percentiles of its bootstrapped distribution to deletion of individual observations. An illustration, for the coefficients of `income` and `education`, appears in Figure 3, which is produced by the following commands:

```
> par(mfcol=c(2, 1))
> jack.after.boot(duncan.boot, index=2, main="(a) income coefficient")
> jack.after.boot(duncan.boot, index=3, main="(b) education coefficient")
```

The horizontal axis of the graph, labeled “standardized jackknife value,” is a measure of the influence of each observation on the coefficient. The observation indices corresponding to the points in the graph are shown near the bottom of the plot. Thus observations 6 and 16 serve to decrease the `income` coefficient and increase the `education` coefficient. As is familiar from Chapters 1 and 6 of the text, these are the occupations minister and railroad-conductor.

The horizontal broken lines on the plot are quantiles of the bootstrap distribution of each coefficient, centered at the value of the coefficient for the original sample. By default the .05, .10, .16, .50, .84, .90, and .95 quantiles are plotted. The points connected by solid lines show the quantiles estimated only from bootstrap samples in which each observation in turn did not appear. Therefore, deleting the occupation minister or conductor makes the bootstrap distributions for the coefficients slightly less dispersed. On the other hand, removing occupation 27 (railroad-engineer) or 30 (plumber) makes the bootstrap distributions somewhat more variable. From our earlier work on Duncan’s data (see, in particular, Chapter 6), we recognize railroad-engineer as a high-leverage but in-line occupation; it is unclear to us, however, why the occupation plumber should stand out in this manner.

### 3.2 Fixed- $x$ Resampling

The observations in Duncan’s occupational-prestige study are meant to represent a larger population of occupations (i.e., all of the Census occupations), but they are not literally sampled from that population. It therefore makes some sense to think of these occupations, and hence the pairs of `income` and `education` values used in the regression, as fixed with respect to replication of the study. The response values, however, are random, because of the error component of the model. There are other circumstances in which it is even more compelling to treat the predictors in a study as fixed — for example, in a designed experiment where the values of the predictors are set by the experimenter.

How can we generate bootstrap replications when the model matrix  $\mathbf{X}$  is fixed? One way to proceed is to treat the fitted values  $\hat{y}_i$  from the model as giving the expectation of the response for the bootstrap samples. Attaching a random error to each  $\hat{y}_i$  produces a fixed- $x$  bootstrap sample,  $\mathbf{y}_b^* = \{y_{bi}^*\}$ . The errors could be generated parametrically from a normal distribution with mean 0 and variance  $s^2$  (the estimated error variance in the regression), if we are willing to assume that the errors are normally distributed, or nonparametrically, by resampling residuals from the original regression.<sup>7</sup> We would then regress the bootstrapped values  $\mathbf{y}_b^*$  on the fixed  $\mathbf{X}$  matrix to obtain bootstrap replications of the regression coefficients.

---

<sup>7</sup>A slightly better approach would be to take account of the leverages of the observations, because the residuals from a regression have variances that depend upon the leverages, even when the errors are homoscedastic (see Chapter 6 of the text). In ordinary least-squares regression, we can simply resample *modified residuals*  $e_i/\sqrt{1-h_i}$ , where  $e_i$

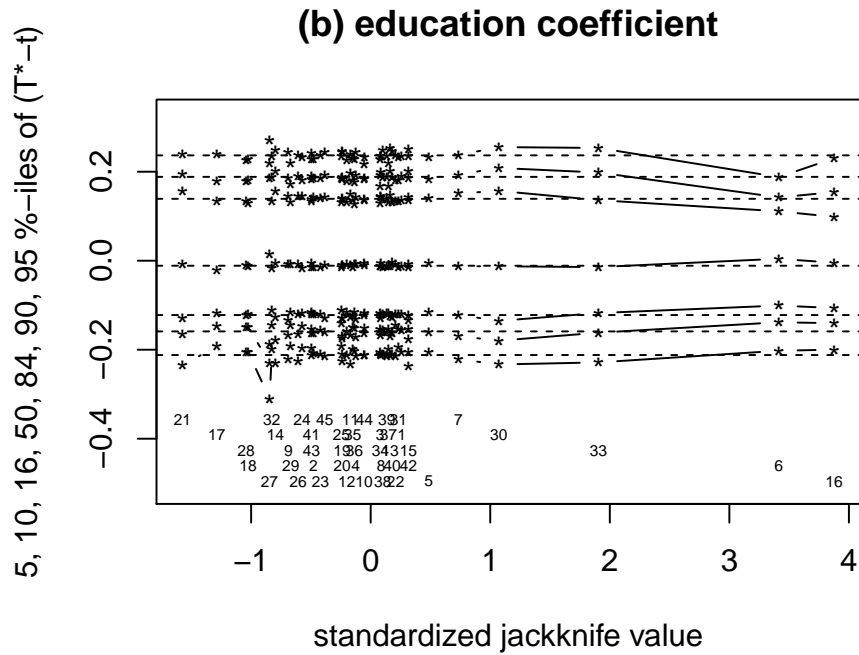
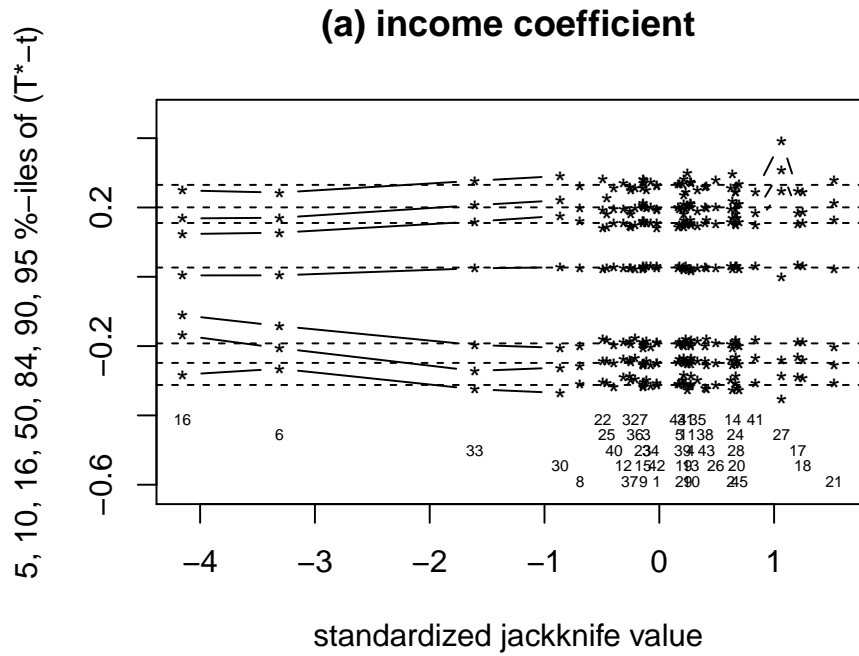


Figure 3: Jackknife-after-bootstrap plot for the income (a) and education (b) coefficients in the Huber regression for Duncan's occupational-prestige data.

Applying this procedure to the Huber regression for Duncan's data proceeds as follows:

```
> fit <- fitted(mod.duncan.hub)
> e <- residuals(mod.duncan.hub)
> X <- model.matrix(mod.duncan.hub)
> boot.huber.fixed <- function(data, indices, maxit=20){
+   y <- fit + e[indices]
+   mod <- rlm(y ~ X - 1, maxit=maxit) # constant is already in X
+   coefficients(mod)
+ }
> set.seed(54321) # for reproducibility
> (duncan.fix.boot <- boot(Duncan, boot.huber.fixed, 1999, maxit=200))
```

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:

```
boot(data = Duncan, statistic = boot.huber.fixed, R = 1999, maxit = 200)
```

Bootstrap Statistics :

	original	bias	std. error
t1*	-7.1107	-0.1258645	3.78048
t2*	0.7014	-0.0007888	0.11194
t3*	0.4854	0.0017297	0.09092

We calculated fitted values, residuals, and the model matrix outside of the function `boot.huber.fixed` so that these quantities are not unnecessarily, and consequently inefficiently, recomputed for each bootstrap sample. In this example, the bootstrap coefficient standard errors for fixed- $x$  resampling are smaller than for random- $x$  resampling, and indeed the former are quite similar to the estimated asymptotic standard errors for the Huber estimator.

Examining the jackknife-after-bootstrap plot for the fixed- $x$  resampling results (Figure 4) provides some insight into the properties of the method:

```
> par(mfcol=c(2, 1))
> jack.after.boot(duncan.fix.boot, index=2, main="(a) income coefficient")
> jack.after.boot(duncan.fix.boot, index=3, main="(b) education coefficient")
```

The quantile traces in Figure 4 are much less variable than in Figure 3 for random- $x$  resampling, because in fixed- $x$  resampling residuals are decoupled from the original observations. In effect, fixed- $x$  resampling enforces the assumption that the errors are identically distributed by resampling residuals from a common distribution. Consequently, if the model is incorrectly specified — for example, if there is unmodeled nonlinearity, non-constant error variance, or outliers — these characteristics will not carry over into the resampled data sets. For this reason, it may be preferable to perform random- $x$  resampling even when it makes sense to think of the model matrix as fixed.

---

and  $h_i$  are, respectively, the residual and hat-value for the  $i$ th observation. In robust regression a similar, if more complex, adjustment can be made. See Davison and Hinkley (1997, sec. 6.5).

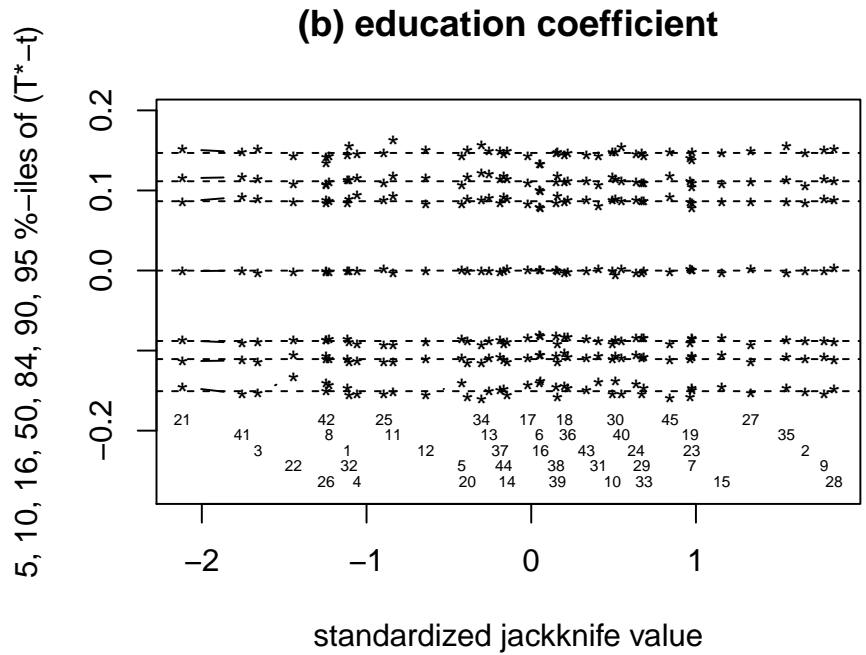
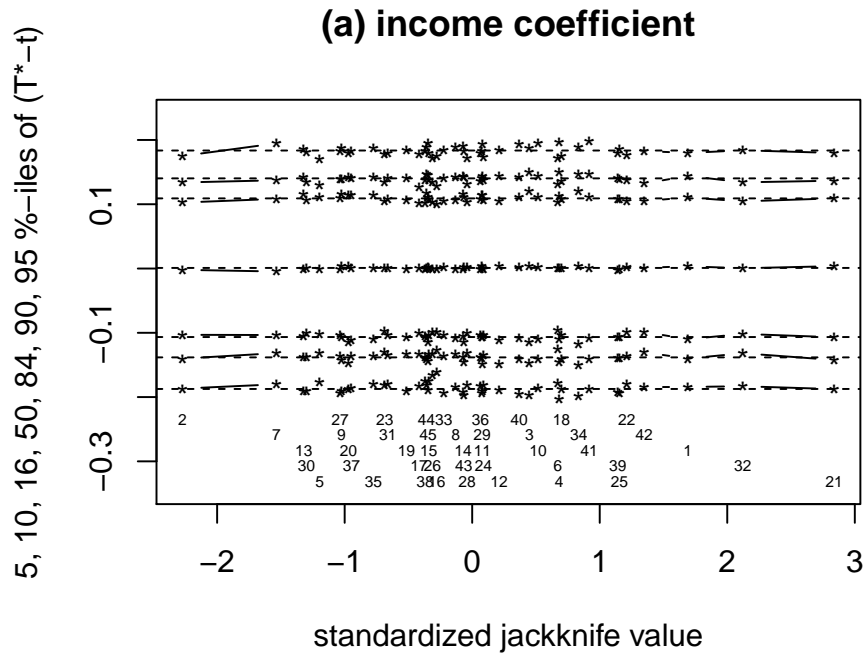


Figure 4: Jackknife-after-bootstrap plot for the income (a) and education (b) coefficients in the Huber regression for Duncan’s occupational-prestige data, using fixed- $x$  resampling.

## 4 Bootstrap Hypothesis Tests

It is also possible to use the bootstrap to construct an empirical sampling distribution for a test statistic. Imagine, for example, that in Duncan's regression, we want to use the robust-regression estimator to test the hypothesis that the `income` and `education` coefficients are the same,  $H_0: \beta_1 = \beta_2$ . This hypothesis arguably makes some sense, because both predictors are scaled as percentages. We could test the hypothesis with the Wald statistic

$$z = \frac{b_1 - b_2}{\left[ (0, 1, -1) \widehat{\text{Var}}(\mathbf{b}) \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix} \right]^{1/2}}$$

where  $\mathbf{b}$  is the vector of estimated regression coefficients;  $b_1$  and  $b_2$  are respectively the `income` and `education` coefficients; and  $\widehat{\text{Var}}(\mathbf{b})$  is the estimated asymptotic covariance matrix of the coefficients. Applying this formula to the data:

```
> b <- coefficients(mod.duncan.hub)
> sumry <- summary(mod.duncan.hub) # to obtain coef. cov. matrix
> V <- sumry$cov.unscaled * sumry$stddev^2 # coef. cov. matrix
> L <- c(0, 1, -1) # hypothesis matrix
> b1 <- b[2] - b[3]
> (t <- b1/sqrt(L %*% V %*% L)) # test statistic

      [,1]
[1,] 1.174

> 2*pnorm(t, lower.tail=FALSE) # p-value

      [,1]
[1,] 0.2404
```

If we can trust the asymptotic normality of  $\mathbf{b}$  and its asymptotic covariance matrix, then  $z$  is distributed as a standard normal variable under the null hypothesis. The test statistic  $z = 1.174$  (two-tail  $p = .240$ ) therefore suggests that the difference between  $\beta_1$  and  $\beta_2$  is not statistically significant. In a small sample such as this, however, we are more comfortable relying on the bootstrap.

Using random- $x$  resampling, we generated  $R = 999$  resampled values of  $z$ :

```
> boot.test <- function(data, indices, maxit=20){
+   data <- data[indices, ]
+   mod <- rlm(prestige ~ income + education, data=data, maxit=maxit)
+   sumry <- summary(mod)
+   V.star <- sumry$cov.unscaled * sumry$stddev^2
+   b.star <- coefficients(mod)
+   b1.star <- b.star[2] - b.star[3]
+   t <- (b1.star - b1)/sqrt(L %*% V.star %*% L)
+   t
+ }
> set.seed(2468) # for reproducibility
> duncan.test.boot <- boot(Duncan, boot.test, 999, maxit=200)
> summary(duncan.test.boot$t)
```

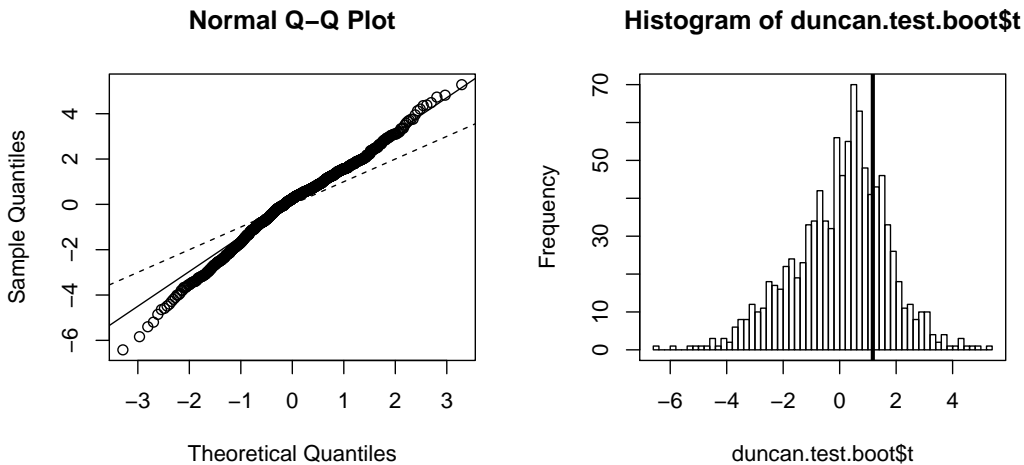


Figure 5: Distribution of the bootstrapped test statistic  $z^*$  for the hypothesis that the coefficients of `income` and `education` are equal. The broken line in the normal quantile-comparison plot is for a standard-normal distribution; the solid line is fit to the data. The vertical line in the histogram is at  $z = 1.174$ , the observed value of the test statistic in the original sample.

```
V1
Min.   :-6.4184
1st Qu.: -0.9284
Median :  0.2535
Mean    :  0.0402
3rd Qu.:  1.1406
Max.    :  5.2829
```

The proper bootstrap analogy is to test the bootstrapped difference in regression coefficients,  $b_{b1}^* - b_{b2}^*$  for the  $b$ th bootstrap sample, against the ‘hypothesized value’  $b_1 - b_2$  (that is, the difference in the original sample), *not* against a difference of 0. The summary of the distribution of the bootstrapped test statistics  $z^*$  suggests that the observed value  $z = 1.174$  is not unusually large. Let us look a little more closely by plotting the distribution of  $z^*$  (Figure 5):

```
> par(mfrow=c(1, 2))
> qqnorm(duncan.test.boot$t)
> qqline(duncan.test.boot$t)
> abline(0, 1, lty=2)
> hist(duncan.test.boot$t, breaks=50)
> box()
> abline(v=t, lwd=3)
```

The distribution of  $z^*$  is not far from normal, but it is much more variable than the standard normal distribution; the position of  $z = 1.174$ , indicated by the vertical line in the histogram, shows that this is not a rare value under the null hypothesis. We can find a  $p$ -value for the bootstrap test by computing the (adjusted) two-tail proportion of  $z^*$ ’s beyond  $|z|$ ,

$$p = 2 \times \frac{1 + \#_{b=1}^R(z^* > z)}{R + 1}$$

```
> 2 * (1 + sum(duncan.test.boot$t > as.vector(t)))/1000
```

```
[1] 0.49
```

The nonparametric  $p$ -value of .49 is much larger than the value  $p = .24$  obtained from the standard-normal distribution.<sup>8</sup>

## 5 Concluding Remarks

Extending random- $x$  resampling to other sorts of parametric regression models, such as generalized linear models, is straightforward. In many instances, however, fixed- $x$  resampling requires special treatment, as does resampling for nonparametric regression.

The discussion in the preceding sections assumes independent random sampling, but bootstrap methods can easily be adapted to other sampling schemes. For example, in stratified sampling, bootstrap resampling is simply performed within strata, building up a bootstrap sample much as the original sample was composed from subsamples for the strata. Likewise, in a cluster sample, we resample clusters rather than individual observations. If the elements of the sample were selected with unequal probability, then so must the elements of each bootstrap sample.

The essential point is to preserve the analogy between the selection of the original sample from the population and the selection of each bootstrap sample from the original sample. Indeed, one of the attractions of the bootstrap is that it can provide correct statistical inference for complex sampling designs which often are handled by ad-hoc methods.<sup>9</sup> The software in the **boot** package can accommodate these complications; see, in particular, the **stype** and **strata** arguments to the **boot** function.

## 6 Complementary Reading and References

Bootstrapping is discussed in Fox (2008, chap. 21) and Weisberg (2005, sec. 4.6). As mentioned, Efron and Tibshirani (1993) and Davison and Hinkley (1997) provide book-length treatments of the subject. See Stine (1990) for a briefer presentation of bootstrapping.

## References

- Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap Methods and their Application*. 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. J. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, New York.
- Fox, J. (2008). *Applied Regression Analysis and Generalized Linear Models*. Sage, Thousand Oaks, CA, second edition.

---

<sup>8</sup>The slightly awkward expression `as.vector(t)` changes `t` from a  $1 \times 1$  matrix into a one-element vector, allowing the computation to proceed.

<sup>9</sup>The **survey** package for R (Lumley, 2010) has extensive facilities for statistical inference in complex sample surveys.

- Fox, J. and Weisberg, S. (2011). *An R Companion to Applied Regression*. Sage, Thousand Oaks, CA, second edition.
- Lumley, T. (2010). *Complex Surveys: A Guide to Analysis Using R*. John Wiley & Sons, Hoboken, NJ.
- Stine, R. (1990). An introduction to bootstrap methods: examples and ideas. In Fox, J. and Long, J. S., editors, *Modern Methods of Data Analysis*, pages 325–373. Sage, Newbury Park, CA.
- Weisberg, S. (2005). *Applied Linear Regression*. John Wiley & Sons, Hoboken, NJ, third edition.