Bootstrap and Jackknife Methods

Su2016 - Dr. Junvie Pailden

July 25, 2016

Contents

Background

Boostrap 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 a 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 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 generates random samples from the empirical distribution of the sample.

Bootstrap Intro

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

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

Resampling generates a random sample $X_1^*, \ldots, X_n^*$ by sampling with replacment 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 $F_X(x)$.

It can be shown that $F_n(x)$ is a sufficient statistic for $F_X(x)$; that is, all the information about $F_X(x)$ that is contained in the sample is also contained in $F_n(x)$.

The empirical cdf F_n is the cdf of X^*.

Two approximations in bootstrap:

1. The ecdf F_n is an approximation to the cdf F_X.
2. The ecdf F_n^* of the bootstrap replicates is an approximation to the ecdf F_n.

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

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 boostrap sample $x^* = \{x_{i_1}, \ldots, x_{i_n}\}$.

Boostrap Algorithm

Suppose θ is the parameter of interest (θ could be a vector), and $\hat{\theta}$ is an estimator of θ.

The boostrap estimate of the distribution of $\hat{\theta}$ is obtained as 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 $\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B)}$.

Example 1: Bootstrap sample from mixture distribution

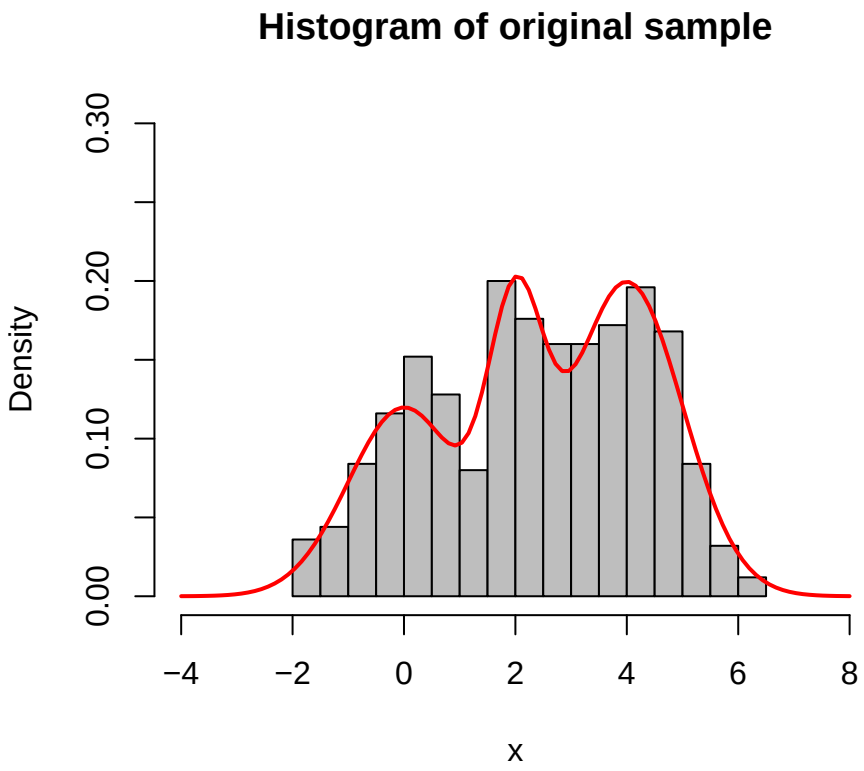
If the original sample is representative of the population, then so will the bootstrap samples.

Suppose we generate a sample of size $n = 500$ from the mixture distribution

$$0.3 \times N(0, 1) + 0.5 \times N(4, 1) + 0.2 \times N(2, .5).$$

We then obtained bootstrap samples (sampling with replacement) from this original sample.

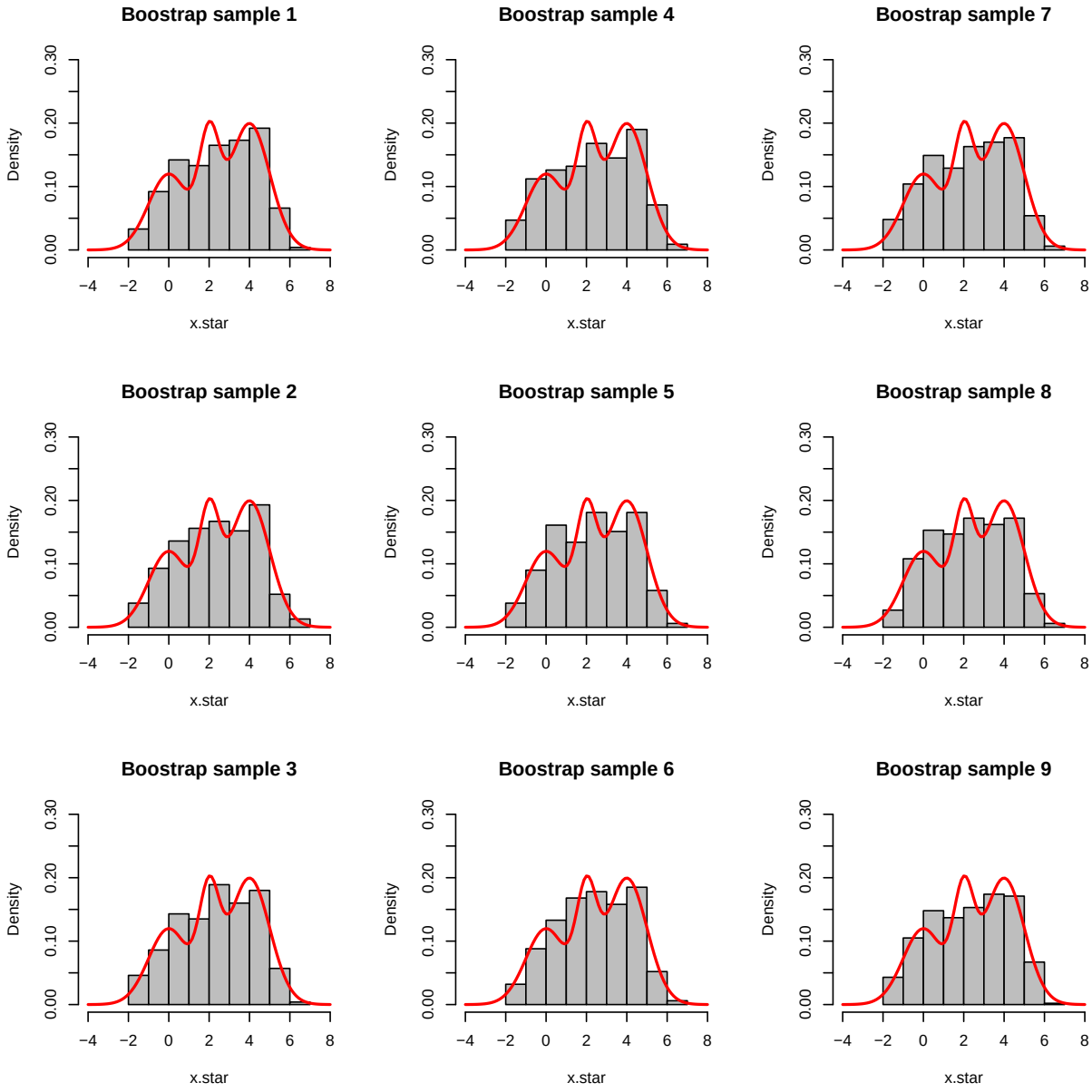
```r
set.seed(2016) # set initial seed number

# number of samples from the mixture distribution
n <-  500
# sample n random unif(0,1)
u <- runif(n)
# variable to store the samples from the mixture distribution
x <- rep(NA,n)
# Sampling from the mixture
for(i in 1:n){
    if(u[i] < .3){
        x[i] <- rnorm(1,0,1)
    } else if(u[i] < .8) {
        x[i] <- rnorm(1,4,1)
    } else {
        x[i] <- rnorm(1,2,.5)
    }
}

# density histogram of the generated samples

hist(x, prob=TRUE, breaks = 15, col="gray", ylim = c(0,0.3), xlim = c(-4,8),
     main = "Histogram of original sample")
mixture.f <- function(x) .3*dnorm(x,0,1) + .5*dnorm(x,4,1) + .2*dnorm(x,2,.5)
curve(mixture.f, add=TRUE,  col = "red", lwd = 2 )
```



Note that the original sample represent the population quite well.

We then obtained $B = 9$ bootstrap samples (sampling with replacement) each of size $ m = 500$ from this original sample.

```
## bootstrap sample

m <- 1000  # bootstrap sample size

par(mfcol = c(3,3))

t <- 1 # initialize the while loop

# we generate 9 bootstrap samples

set.seed(1234)
while (t <= 9) {
  x.star <- sample(x, size = m, replace = TRUE)
  hist(x.star, prob=TRUE, breaks = 10, col="gray", ylim = c(0,0.3),
       xlim = c(-4,8), main = paste0("Boostrap sample ", t))
  curve(mixture.f, add=TRUE,  col = "red", lwd = 2 )
  t <- t + 1 # next iteration
}
```



The $B = 9$ bootstrap samples also reasonably represent the original mixture density.

Example 2: Bootstrap samples from Poisson distribution

> If the original sample **do not** represent of the population well (possibly bias or small sample size), then so will the bootstrap samples.

Suppose we generate a sample $\{x\}$ from a Poisson population with parameter $\lambda = 2$.

```r
set.seed(17)
x <- rpois(10, lambda=2)
# original sample, note the absence of 0 and 4 values
x
```

```
##  [1] 1 5 2 3 2 2 1 1 3 1
```

```r
# original sample empirical mass function
table(x)/10
```

```
## x
##   1   2   3   5
## 0.4 0.3 0.2 0.1
```

Resampling from $\{x\}$ we select 1, 2, 3, or 5 with prob'y 0.4, 0.3, 0.2, and 0.1, respectively, so the cdf F_{X^*} of a randomly selected replicate is exactly the ecdf $F_n(x)$:

$$F_{X^*}(x) = F_n(x) = \begin{cases} 0, & x < 1; \\ 0.4, & 1 \le x < 2; \\ 0.7, & 2 \le x < 3; \\ 0.9, & 3 \le x < 5; \\ 1, & x \ge 5. \end{cases}$$

Since, 0 and 4 are not present in the original sample then it is impossible to generate these values in the bootstrap sample.

Although the original sample points were selected at random, there number was not enough to cover the entire support (non-negative integers) of the Poisson distribution. We can say that this is sample does not represent the population.

Note that if F_n is not close to F_X then the distribution of the replicates will not be close to F_X.

Resampling from x a large number of replicates produces a good estimate of F_n but not a good estimator of F_X because the bootstrap samples will never include 0 and 4.

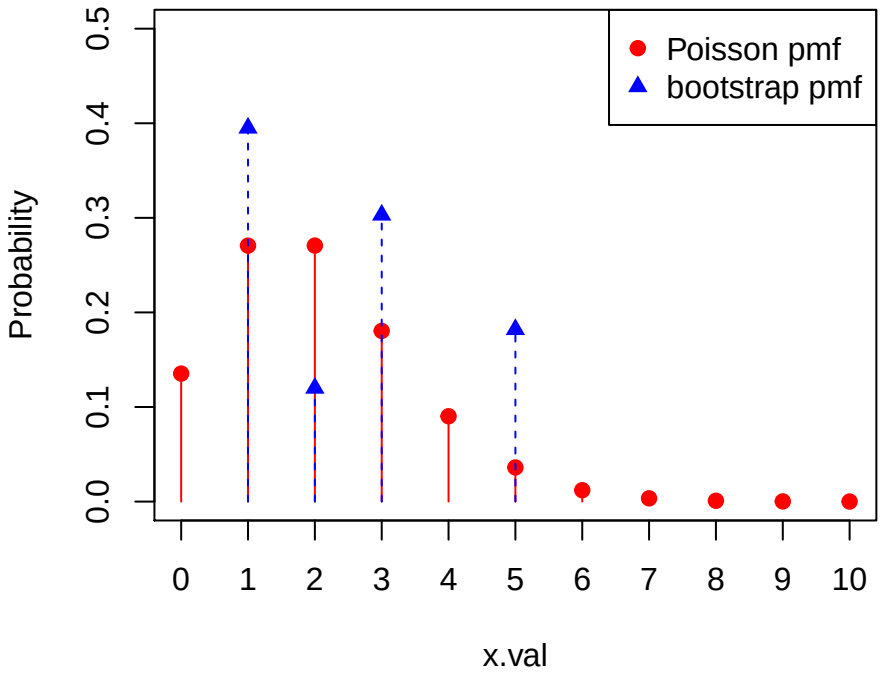
```r
x.uniq <- unique(x)
prob0 <- as.data.frame(table(x))[,2]/length(x)  # bootstrap empirical prob'y mass values
data.frame(x.samp = x.uniq, pmf = prob0 )
```

```
##   x.samp pmf
## 1      1 0.4
## 2      5 0.3
## 3      2 0.2
## 4      3 0.1
```

```r
m <- 1000  # bootstrap sample size
set.seed(1234)
x.star <- sample(x.uniq, size = m, replace = TRUE,
                 prob = prob0)
# pmf f_n
table(x.star)/m
```

```
## x.star
##     1     2     3     5
## 0.395 0.182 0.120 0.303
```

```
x.val <- seq(0, 10)
plot(x.val, dpois(x.val, lambda = 2), ylim = c(0,0.5), col = "red",
     xaxt = "n", pch = 19, ylab = "Probability")
axis(side = 1, at = x.val,labels = T)
lines(x.val, dpois(x.val, lambda = 2), type = "h", col = "red")
points(x.uniq, table(x.star)/m, col = "blue", pch = 17)
lines(x.uniq, table(x.star)/m, type = "h", col = "blue", lty = 2 )
legend("topright", legend = c("Poisson pmf", "bootstrap pmf"),
       pch = c(19,17), col = c("red", "blue"))
```



Bootstrap Estimation of Standard Error

We are interested in the population parameter θ - θ can be the mean, median, correlation, skewness, slope, etc, of a probability distribution.

Suppose the statistic $\hat{\theta}$ is an estimator of θ.

The main motivation for using bootstrap samples is so that we can estimate the sampling distribution of the statistic $\hat{\theta}$ even though we only have one original sample from the population.

Recall the procedure for generating bootstrap samples given the original sample $x = \{x_1, \ldots, x_n\}$:

For each bootstrap replicate, indexed $b = 1, \ldots, B$:

1. Generate sample $x^{*(b)} = \{x_1^*, \ldots, x_n^*\}$ by sampling with replacement from the observed sample $\{x_1, \ldots, x_n\}$.
2. Compute the b^{th} replicate $\hat{\theta}^{(b)}$ from the b^{th} bootstrap sample.

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

$$\hat{se}(\hat{\theta}^*) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{\theta}^{(b)} - \bar{\hat{\theta}}^*\right)^2},$$

where $\bar{\hat{\theta}}^* = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^{(b)}$.

According to Efron and Tibshirani (1993), the number of replicates needed for good estimates of standard error is not large; $B = 50$ is not usually large enough, and rarely is $B > 200$ necessary.

Much larger B will be needed for confidence interval estimation.

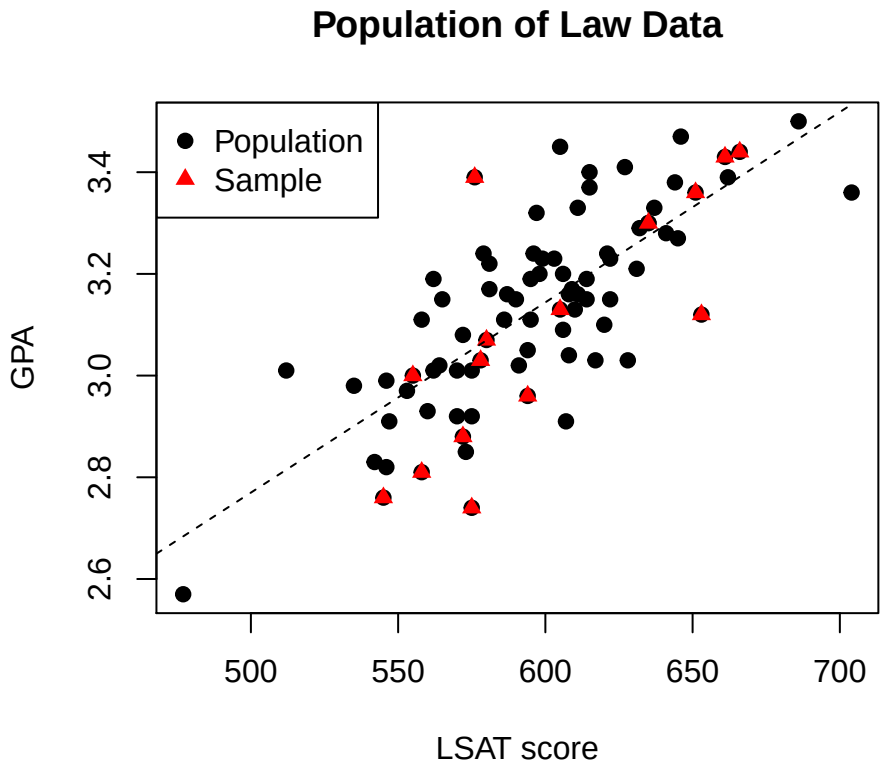
Example 3: Estimating the std. error of the correlation coefficient

The law school data set `law` in the `bootstrap` package contains LSAT (average score on law school admission test score) and GPA (average undergraduate gpa) for 15 law schools.

This data set is a random sample from a population of 82 law schools in `law82` (boostrap).

> Goal: Estimate the correlation between LSAT and GPA scores, and compute the boostrap estimate of the standard error of the sample correlation.

```
library(bootstrap)    # needed to get the law data
plot(law82$LSAT, law82$GPA, xlab = "LSAT score", ylab = "GPA", main = "Population of Law Data", pch = 19
points(law$LSAT, law$GPA/100, col = "red", pch = 17)
abline(lm(law82$GPA ~ law82$LSAT ), lty = 2)
legend("topleft", c("Population", "Sample"), pch = c(19, 17), col = c("black", "red"))
```

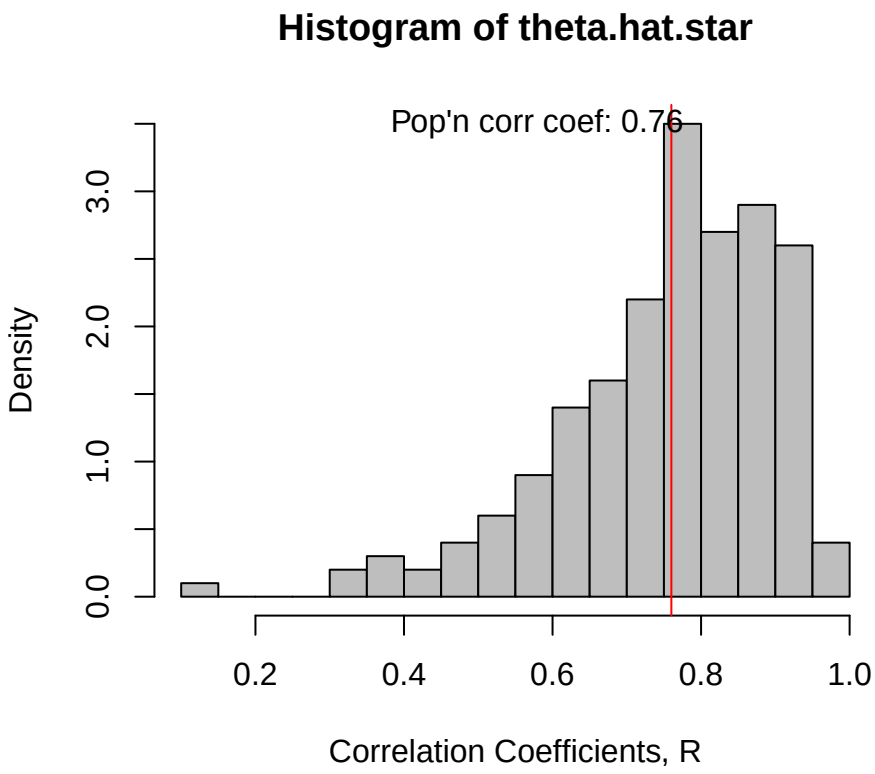



```
# pop'n correlation of 82 law schools
theta <- cor(law82$LSAT, law82$GPA)

# sample correlation
theta.hat <- cor(law$LSAT, law$GPA)

data.frame(popn.rcoef = theta, samp_rcoef = theta.hat)
```

```
##   popn.rcoef samp_rcoef
## 1  0.7599979  0.7763745
```

```
my.corr <- function(n){
  # randomly select the indices
  i <- sample(1:n, size = n, replace = TRUE) # sample with replacement for the bootstrap
  LSAT <- law$LSAT[i]       #i is a vector of indices
  GPA <- law$GPA[i]
  cor(LSAT, GPA)
}
#set up the bootstrap
B <- 200          #number of replicates
n <- nrow(law)      #sample size

# correlation coefficients computed from generated bootstrap samples
set.seed(17)
theta.hat.star <- replicate(B, expr = my.corr(n))
```

```r
par(mfcol = c(1,1))
hist(theta.hat.star, prob = TRUE, breaks=20, col="gray", xlab = "Correlation Coefficients, R")
abline(v = theta, col="red")
text(theta - 0.18, 3.5, paste0("Pop'n corr coef: ", round(theta, digits = 2) ))
```




```r
# bootstrap estimate of standard error of R
se.theta.hat <- sd(theta.hat.star)
se.theta.hat
```

```
## [1] 0.1433895
```

```r
# The normal theory estimate for standard error of
# R is 0.1434.
```

We can also use the popular `boot` function in the `boot` package to generate bootstrap samples.

```r
r <- function(x, i) {
        #want correlation of columns 1 and 2
        cor(x[i,1], x[i,2])
    }
library(boot)       #for boot function
set.seed(17)
# statistic in this case is the correlation coef r above
# R is the number of bootstrap samples
```

```
obj <- boot(data = law, statistic = r, R = 200)
obj
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = law, statistic = r, R = 200)
##
##
## Bootstrap Statistics :
##      original      bias    std. error
## t1* 0.7763745 -0.02494292   0.1434171
```

```
y <- obj$t   # extract the computed values of correlation coefficients
sd(y)
```

```
## [1] 0.1434171
```

Bootstrap Estimation of Bias

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

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

Every statistic is an unbiased estimator of its expected value.

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 $\hat{\theta} = \hat{\theta}(x)$ and there are B independent and identically distributed estimators of $\hat{\theta}^{(b)}$.

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

$$\hat{bias}(\hat{\theta}) = \hat{\bar{\theta}}^* - \hat{\theta},$$

where $\bar{\hat{\theta}}^* = \frac{1}{B}\sum_{b=1}^{B} \hat{\theta}^{(b)}$ and $\hat{\theta} = \hat{\theta}(x)$ is the estimate computed from the original observed sample.

```
# sample estimate for n=15
theta.hat <- cor(law$LSAT, law$GPA)
#set up the bootstrap
B <- 2000          #larger for estimating the bias
n <- nrow(law)
# bootstrap estimate of correlation coefficients
set.seed(17)
theta.hat.star <- replicate(B, expr = my.corr(n))
# estimated bias of correlation coef using bootstrap method
bias.theta.hat <- mean(theta.hat.star) - theta.hat
bias.theta.hat
```

```
## [1] -0.001915937
```

To sum up the correlation coefficient analysis of the `law82` data using the sample `law` of size `n=15`, we have the following:

```
data.frame(corr = theta.hat, std.error = se.theta.hat, bias = bias.theta.hat)
```

```
##        corr std.error         bias
## 1 0.7763745 0.1433895 -0.001915937
```

Example 4: Boostraping Regression Coefficients

The following example computes bootstrapped estimates of bias and standard error of three model regression coefficients (intercept, car weight, displacement) in the linear regression of miles per gallon (`mpg`) on car weight (`wt`) and displacement (`disp`). The data source is `mtcars`.

```
#library(boot)
# function to obtain regression weights
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data=d)
  return(coef(fit))
 }
# bootstrapping with 500 replications
set.seed(17)
results <- boot(data = mtcars, statistic = bs,
                R = 500, formula = mpg ~ wt + disp)
# view results
# t1* corresponds to the y-intercept
# t2* corresponds to slope coef of weight
# t3* corresponds to slope coef of displacement
results
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = mtcars, statistic = bs, R = 500, formula = mpg ~
##     wt + disp)
##
##
## Bootstrap Statistics :
##        original        bias    std. error
## t1* 34.96055404  0.0223127105  2.47113931
## t2* -3.35082533 -0.0098442788  1.16120353
## t3* -0.01772474 -0.0002745367  0.00886513
```

The Jackknife

The *jackknife* is another resampling method for estimating bias and standard error.

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} obs x_i,

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

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

Suppose the parameter $\theta = t(F)$ is a function of the distribution F . Let F_n be the ecdf of a random sample from F. The "plug-in" estimate of θ is $\hat{\theta} = t(F_n)$.

> A *plug-in* statistic $\hat{\theta}$ is **smooth** in the sense that small changes in the data correspond to small changes in $\hat{\theta}$.

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

$$\hat{bias}_{jack} = (n-1)(\bar{\hat{\theta}}_{(\cdot)} - \hat{\theta}),$$

where $\bar{\hat{\theta}}_{(\cdot)} = \frac{1}{n}\sum_{i=1}^{n} \hat{\theta}_{(i)}$ is the mean of the estimates from the leave-one-out samples.

Why do we need to multiply by $n-1$? To see this, note that 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 $\hat{\theta}_{(i)}$ with sample size $n-1$; so $bias(\hat{\theta}_{(i)}) = -\sigma_X^2/(n-1)$. Thus,

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

***

The Jackknife Estimate of Standard Error

A jackknife estimate of standard error for smooth statistics $\hat{\theta}$ is

$$\hat{se}_{jack} = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}\left(\hat{\theta}_{(i)} - \bar{\hat{\theta}}_{(\cdot)}\right)^2}.$$

To see this, consider the plug-in estimate of the standard error of the mean, $\hat{se}(\bar{X})$. When X is continuous r.v. and Y is uniformly distributed on the sample $x_1, \ldots, x_n$, then the plug-in estimate of variance of the sample is

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

Example 5: Analysis of ratio estimate using jackknife samples

The `patch (bootstrap)` data 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 $\bar{Y}/\bar{Z}$, where $Y = new - old$ and $Z = old - placebo$.

```
data(patch)
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
```

The sample estimate from the original data is

```
n <- nrow(patch)
y <- patch$y
z <- patch$z
theta.hat <- mean(y) / mean(z)
theta.hat
```

```
## [1] -0.0713061
```

Compute the jacknife estimate of bias, standard error and coefficient of variation of $\hat{\theta}$ for the `patch` data.

```
#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])
    # z[-i] removes the ith entry in vector z

#jackknife estimate of bias
bias <- (n - 1) * (mean(theta.jack) - theta.hat)

#jackknife estimate of standard error
se <- sqrt((n-1) *
       mean((theta.jack - mean(theta.jack))^2))

#coefficient of variation
cv <- bias/se
data.frame(theta.hat, bias, se, cv)
```

```
##    theta.hat        bias        se       cv
## 1 -0.0713061 0.008002488 0.1055278 0.075833
```

When the Jackknife Fails

The jackknife can fail when the statistic $\hat{\theta}$ is not "smooth" in the sense that $\hat{\theta}$ is not sensitive to changes in the data. For example, the median is not a smooth function since changing the outlying values will not necessarily change the value of the median of the data.

```
n <- 10
set.seed(123)
x <- sample(1:100, size = n)

# jackknife estimate of se
Mjack <- numeric(n)
for( i in 1:n)
  Mjack[i] <- median(x[-i]) # leave one out
Mbar <- mean(Mjack)

#bootstrap estimate of se using 50 bootstrap sampples
Mboot <- replicate(50, expr = {
   y <- sample(x, size = n, replace = TRUE)
   median(y) })

rbind(x, jk.Medians = Mjack, boot.Medians = c(Mboot[1:9],"... 41 more"))
```

```
##              [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## x            "29" "79" "41" "86" "91" "5"  "50" "83" "51" "42"
## jk.Medians   "51" "50" "51" "50" "50" "51" "51" "50" "50" "51"
## boot.Medians "46" "50" "46" "79" "79" "51" "81" "65" "79" "... 41 more"
```

```r
# Compute the standard errors
se.jack = sqrt((n-1)/n * sum((Mjack - Mbar)^2))
se.boot = sd(Mboot)
data.frame(se.jack, se.boot)
```

```
##   se.jack  se.boot
## 1     1.5 12.32453
```

Notice the wide discrepancy between the bootstrap estimate and the jackknife estimate. The jackknife fails because the median is not smooth.

Example 6: Jackknife-after-Bootstrap

Bootstrap estimates of standard error and bias are random variables.

If we are interested in the variance of these estimates, one idea is to try the jackknife.

Recall that $\hat{se}(\hat{\theta})$ is the sample standard deviation of B bootstrap replicates of $\hat{\theta}$. If we leave out the i^{th} obs, the algorithm for estimation of standard error is to resample B replicates from the $n-1$ remaining observations - for each i .

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 the 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.

The jackknife estimate of standard error is computed by

$$\begin{aligned}\hat{se}(\hat{\theta}) &= \hat{se}_{jack}(\hat{se}_{B(1)}, \ldots, \hat{se}_{B(n)}) \\ &= \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}\left(\hat{se}_{B(i)} - \bar{\hat{se}}_{B(\cdot)}\right)^2},\end{aligned}$$

where

$$\begin{aligned}\bar{\hat{se}}_{B(\cdot)} &= \frac{1}{n}\sum_{i=1}^{n}\hat{se}_{B(i)} \\ \hat{se}_{B(i)} &= \sqrt{\frac{1}{B(i)}\sum_{j\in J(i)}^{n}\left(\hat{\theta}_{(j)} - \bar{\hat{\theta}}_{(J(\cdot))}\right)^2} \\ \bar{\hat{\theta}}_{(J(\cdot))} &= \frac{1}{B(i)}\sum_{j\in J(i)}^{n}\hat{\theta}_{(j)}\end{aligned}$$

Note that $\bar{\hat{se}}_{B(\cdot)}$ is the average boostrap SE estimates from all the leave-x_i-out jackknife samples.

Similarly, $\bar{\hat{\theta}}_{(J(\cdot))}$ is the sample mean of the estimates from all the leave-x_i-out jackknife samples.

```r
# initialize
data(patch, package = "bootstrap")
n <- nrow(patch)
y <- patch$y
z <- patch$z
B <- 2000
theta.b <- numeric(B)
# set up storage for the sampled indices
indices <- matrix(0, nrow = B, ncol = n)
# jackknife-after-bootstrap step 1: run the bootstrap
set.seed(17)
for (b in 1:B) {
    i <- sample(1:n, size = n, replace = TRUE)
    y <- patch$y[i]
    z <- patch$z[i]
    theta.b[b] <- mean(y) / mean(z)
    #save the indices for the jackknife
    indices[b, ] <- i
}

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

se.boot <- sd(theta.b)
se.jack.boot <- sqrt((n-1) * mean((se.jack -
          mean(se.jack))^2))
data.frame(se.boot,se.jack.boot)
```

```
##     se.boot se.jack.boot
## 1 0.1000559   0.02271112
```

Bootstrap Confidence Intervals

Many variations exists for the bootstrap confidence intervals.

- The Standard Normal Boostrap Confidence Intervals
- The Basic Bootstrap Confidence Interval
- The Percentile Bootstrap Confidence Interval
- The Bootstrap t interval

The Standard Normal Bootstrap Confidence Interval

If $\hat{\theta}$ is an unbiased estimator of parameter θ, then an approximate $100(1-\alpha)\%$ confidence interval for θ is the Z-interval

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

where $z_{\alpha/2} = \Phi^{-1}(1-\alpha/2)$.

Assumptions:

1. Distribution of $\hat{\theta}$ is normal or $\hat{\theta}$ is a sample mean and the sample size is large.
2. $\hat{\theta}$ is unbiased for θ.

The Percentile Bootstrap Confidence Interval

A bootstrap percentile interval uses the empirical distribution of the bootstrap replciates as the reference distribution.

The quantiles of the empirical distribution are estimators of the quantiles of the sampling distribution of $\hat{\theta}$, so that these quantiles may match the true distribution better when the distribution of $\hat{\theta}$ is not normal.

Suppose that $\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B)}$ are the bootstrap replicates of the statistic $\hat{\theta}$.

From the ecdf of the replicates, compute the $\alpha/2$ quantile $\hat{\theta}_{\alpha/2}$, and the $1-\alpha/2$ quantile $\hat{\theta}_{1-\alpha/2}$.

The percentile interval has some theoretical advatages over the standard normal interval and somewhat better coverage performance.

The Basic Bootstrap Confidence Interval

In bootstrap, the distribution of $\hat{\theta}$ is unknown but quantiles can be estimated.

Compute the α quantiles of $\hat{\theta}_\alpha$ from the ecdf of the replicates $\hat{\theta}^*$.

Denote the α quantile of $\hat{\theta}^* - \hat{\theta}$ by b_α.

Then $\hat{b}_\alpha = \hat{\theta}_\alpha - \hat{\theta}$ is an estimator of b_α.

An upper and lower confidence limit, respectively, for $100(1-\alpha)\%$ CI for θ are

$$\hat{\theta} - \hat{b}_{\alpha/2} = \hat{\theta} - (\hat{\theta}_{\alpha/2} - \hat{\theta}) = 2\hat{\theta} - \hat{\theta}_{\alpha/2},$$
$$\hat{\theta} - \hat{b}_{1-\alpha/2} = \hat{\theta} - (\hat{\theta}_{1-\alpha/2} - \hat{\theta}) = 2\hat{\theta} - \hat{\theta}_{1-\alpha/2}.$$

Example 7: Patch Data, Bootstrap Confidence Intervals

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

y <- patch$y
z <- patch$z
dat <- cbind(y, z)

set.seed(1234)
# generate 200 bootstrap samples and compute theta.boot for each one
boot.obj <- boot(dat, statistic = theta.boot, R = 200)
boot.obj
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = dat, statistic = theta.boot, R = 200)
##
##
## Bootstrap Statistics :
##       original        bias    std. error
## t1* -0.0713061 -0.0008960259    0.101522
```

```r
boot.ci(boot.obj, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 200 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.obj, type = "basic")
##
## Intervals :
## Level      Basic
## 95%   (-0.2994,  0.0952 )
## Calculations and Intervals on Original Scale
## Some basic intervals may be unstable
```

```r
boot.ci(boot.obj, type = "norm")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 200 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.obj, type = "norm")
##
## Intervals :
```

```
## Level      Normal
## 95%   (-0.2694,  0.1286 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(boot.obj, type = "perc")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 200 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.obj, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   (-0.2378,  0.1568 )
## Calculations and Intervals on Original Scale
## Some percentile intervals may be unstable
```

Or you can use the output and manually compute the CI's

```
#calculations for bootstrap confidence intervals
alpha <- c(.025, .975)
#normal
print(boot.obj$t0 + qnorm(alpha) * sd(boot.obj$t))
```

```
## [1] -0.2702855  0.1276733
```

```
#basic
print(2*boot.obj$t0 -
    quantile(boot.obj$t, rev(alpha), type=1))
```

```
##       97.5%        2.5%
## -0.29242406  0.09518404
```

```
#percentile
print(quantile(boot.obj$t, alpha, type=6))
```

```
##       2.5%      97.5%
## -0.2377962  0.1567819
```

Example 8: Law data, Bootstrap Confidence Intervals for Correlation

```
set.seed(1234)
boot.obj <- boot(law, R = 200,
    statistic = function(x, i){cor(x[i,1], x[i,2])})
boot.ci(boot.obj,type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 200 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.obj, type = "basic")
##
## Intervals :
## Level      Basic
## 95%   ( 0.5869,  1.0434 )
## Calculations and Intervals on Original Scale
## Some basic intervals may be unstable
```

```
boot.ci(boot.obj,type = "norm")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 200 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.obj, type = "norm")
##
## Intervals :
## Level      Normal
## 95%   ( 0.5299,  1.0097 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(boot.obj,type = "perc")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 200 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.obj, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 0.5093,  0.9659 )
## Calculations and Intervals on Original Scale
## Some percentile intervals may be unstable
```

(Optional) The Bootstrap t interval

Even if the dist'n of $\hat{\theta}$ is normal and $\hat{\theta}$ is unbiased for θ, the normal distribution is not exactly correct for the Z statistic in standard normal boostrap confidence interval, because we estimate $se(\hat{\theta})$.

Nor can we claim that it is a Studen t statistic, because the distribution of $\hat{se}(\hat{\theta})$ is unknown.

The bootstrap t interval does not use a Studen t dist'n as the reference dist'n.

Instead, the sampling dist'n of "t-type" statistic is generated by resampling.

The $100(1-\alpha)\%$ bootstrap t confidence interval is

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

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

Suppose $x = \{x_1, \ldots, x_n\}$ is an observed sample.

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 $\hat{se}(\hat{\theta}^{(b)})$ (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, $t^{(b)} = \frac{\hat{\theta}^{(b)} - \hat{\theta}}{\hat{se}(\hat{\theta}^{(b)})}$.
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 $\hat{se}(\hat{\theta})$, the sample standard deviation of the replicates $\hat{\theta}^{(b)}$.
5. Compute the confidence limits

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

A disadvante to the bootstrap t interval is that typically the estimates of standard error $\hat{se}(\hat{\theta}^{(b)})$ must be obtained by bootstrap. This a bootstrap nested inside a bootstrap.

If $B = 1000$, for example, the bootstrap t confidence interval method takes approx 1000 longer to implement than any of the other methods.

Example 9: Patch Data, Bootstrap t Confidence Intervals

```
boot.t.ci <- function(x, B = 500, R = 200, 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
    # use the quantile() function to compute the limits of the conf intervals
    # if alpha = 0.05, then quantile() computes the 2.5% and 97.5% percentile of the computed t.stats
    Qt <- quantile(t.stats, c(alpha/2, 1-alpha/2),
           type = 1)
    names(Qt) <- rev(names(Qt))
    CI <- rev(stat0 - Qt * se0)
  return(CI)
}

# (Bootstrap t confidence interval
#  for patch ratio statistic)

dat <- cbind(patch$y, patch$z)
# statistic
stat <- function(dat) {
    mean(dat[, 1]) / mean(dat[, 2]) }

set.seed(1234)
ci <- boot.t.ci(dat, statistic = stat, B=500, R=200)
print(ci)
```

```
##       2.5%      97.5%
## -0.2778919  0.4516199
```