22S:166
More on the Bootstrap

Lecture 9
Oct. 05, 2016

Kate Cowles
374 SH, 335-0727
kate-cowles@uiowa.edu

Choosing the number of bootstrap datasets

- approximately 1000 to 2000 is minumum for reasonable performance in most cases
- choosing R = 999 or 1999 facilitates calculation of percentile confidence intervals (see below)

Another version of the function for calculating the statistic for the city data

```
> meanratio
function( df, indices)
{
       #df must be data frame with 2 columns, "x" and "u"
mean( df[indices, "x"]) / mean( df[indices,"u"])

}
```


Running the bootstrap with different settings of R

```
> library(boot)

Attaching package: 'boot'


        The following object(s) are masked _by_ .GlobalEnv :

         city

> data(city)
>
> boot.out <- boot( city, meanratio, R=999)
> boot.out

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = city, statistic = meanratio, R = 999)


Bootstrap Statistics :
    original    bias    std. error
t1* 1.520312 0.0338232    0.218307
>

> boot.out <- boot( city, meanratio, R=999)
> boot.out

ORDINARY NONPARAMETRIC BOOTSTRAP
```

```
Call:
boot(data = city, statistic = meanratio, R = 999)


Bootstrap Statistics :
    original     bias    std. error
t1* 1.520312 0.04969103   0.2316369
>

> boot.out <- boot( city, meanratio, R=1999)
> boot.out

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = city, statistic = meanratio, R = 1999)


Bootstrap Statistics :
    original     bias    std. error
t1* 1.520312 0.04079779   0.2294637

> boot.out <- boot( city, meanratio, R=1999)
> boot.out

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = city, statistic = meanratio, R = 1999)
```



```
Bootstrap Statistics :
    original     bias    std. error
t1* 1.520312 0.03994395    0.222143
```


Interpreting the boot object

Percentile method for confidence intervals

- denote cdf of *bootstrap distribution* of $\hat{\theta}^*$ as
$$C\bar{D}F(t) = Pr_*(\hat{\theta}^* \leq t)$$
- If bootstrap distribution is obtained by simulation then
$$C\bar{D}F(t) \simeq \frac{\#(\hat{\theta}^{*b} \leq t)}{B}$$
- define confidence interval as interval between appropriate quantiles

Bootstrap confidence intervals

- normal
- basic
- percentile
- BCa (adjusted bootstrap percentile)


```
Usage:

     boot.ci(boot.out, conf = 0.95, type = "all",
             index = 1:min(2,length(boot.out$t0)), var.t0 = NULL,
             var.t = NULL, t0 = NULL, t = NULL, L = NULL, h = function(t) t,
             hdot = function(t) rep(1,length(t)), hinv = function(t) t, ...)

Arguments:

boot.out: An object of class '"boot"' containing the output of a
          bootstrap calculation.

    conf: A scalar or vector containing the confidence level(s) of the
          required interval(s).


    type: A vector of character strings representing the type of
          intervals required. The value should be any subset of the
          values 'c("norm","basic", "stud", "perc", "bca")' or simply
          '"all"' which will compute all five types of intervals.

-------------------------------------------------------------------------------


> boot.ci(boot.out)
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 999 bootstrap replicates

CALL :
boot.ci(boot.out = boot.out)

Intervals :
Level      Normal              Basic
95%   ( 1.036,  1.923 )   ( 0.848,  1.786 )
```



```
Level     Percentile            BCa
95%   ( 1.254,  2.192 )   ( 1.264,  2.231 )
Calculations and Intervals on Original Scale
Warning message:
In boot.ci(boot.out = boot.out) :
  bootstrap variances needed for studentized intervals
```


Parametric bootstrap

1. estimate *parametric* mle $\hat{F}$ of unknown F
   - i.e., get mles of parameters
2. Draw a "bootstrap sample" from $\hat{F}$ and calculate statistic of interest on bootstrap sample
   - i.e., simulate data values from parametric model using mles as parameters
   - $Y_1^*, Y_2^*, \ldots, Y_n^* \sim \hat{F}$
   - $\hat{\theta}^* = \hat{\theta}(Y_1^*, Y_2^*, \ldots, Y_n^*)$
3. repeat step 2 independently a large number B of times obtaining bootstrap replications $\hat{\theta}^{*1}, \hat{\theta}^{*2}, \ldots, \hat{\theta}^{*B}$
4. Use bootstrap replications to:
   - estimate standard error of $\hat{\theta}$
   - estimate bias
   - obtain confidence interval

Using boot package for parametric bootstrap

Usage:

```
     boot(data, statistic, R, sim="ordinary", stype="i",
          strata=rep(1,n), L=NULL, m=0, weights=NULL,
          ran.gen=function(d, p) d, mle=NULL, ...)

     sim: A character string indicating the type of simulation
          required.  Possible values are '"ordinary"' (the default),
          '"parametric"', '"balanced"', '"permutation"', or
          '"antithetic"'.  Importance resampling is specified by
          including importance weights; the type of importance
          resampling must still be specified but may only be
          '"ordinary"' or '"balanced"' in this case.
 ran.gen: This function is used only when 'sim' is '"parametric"' when
          it describes how random values are to be generated.  It
          should be a function of two arguments.  The first argument
          should be the observed data and the second argument consists
          of any other information needed (e.g. parameter estimates).
          The second argument may be a list, allowing any number of
          items to be passed to 'ran.gen'.  The returned value should
          be a simulated data set of the same form as the observed data
          which will be passed to statistic to get a bootstrap
          replicate.  It is important that the returned value be of the
          same shape and type as the original dataset.  If 'ran.gen' is
          not specified, the default is a function which returns the
          original 'data'  in which case all simulation should be
          included as part of 'statistic'.  Use of 'sim="parametric"'
          with a suitable 'ran.gen' allows the user to implement any
          types of nonparametric resampling which are not supported
          directly.
```



```
     mle: The second argument to be passed to 'ran.gen'.  Typically
          these will be maximum likelihood estimates of the parameters.
           For efficiency 'mle' is often a list containing all of the
          objects needed by 'ran.gen' which can be calculated  using
          the original data set only.
```


Example: assuming population distribution is normal

Suppose we are using the trimmed mean as a measure of center using continuous data.

```
> x <- rcauchy(25)

> trimmed.mean <- function(x) {mean(x, trim=0.25) }


ran.gen.normal <- function(d,p)
{
rnorm( length(d), mean = p$xbar, sd = p$s)
}

boot.normal.out <- boot( data = x, statistic = trimmed.mean,
R=999, sim="parametric", ran.gen = ran.gen.normal,
mle = list( xbar = mean(x), sd = sqrt(var(x))) )

> boot.normal.out

PARAMETRIC BOOTSTRAP


Call:
boot(data = x, statistic = trimmed.mean, R = 999, sim = "parametric",
    ran.gen = ran.gen.normal, mle = list(xbar = mean(x), sd = sqrt(var(x))))


Bootstrap Statistics :
      original    bias    std. error
t1* 0.04053538 0.1625267    0.526352
```


>

For Cauchy data

Since mean and variance do not exist for Cauchy distribution, choice of measures of center and spread for simulating data are somewhat arbitrary.

```
> ran.gen.cauchy <- function(d, p )
{
rcauchy(length(d), location = p$med, scale = p$sc)
}
```

```
> boot.cauchy.out <-boot(data=x, statistic=trimmed.mean, R=999,
sim="parametric",ran.gen = ran.gen.cauchy,
mle = list( med = median(x), sc = IQR(x)/2) )

> boot.cauchy.out

PARAMETRIC BOOTSTRAP


Call:
boot(data = x, statistic = trimmed.mean, R = 999, sim = "parametric",
    ran.gen = ran.gen.cauchy, mle = list(med = median(x), sc = IQR(x)/2))


Bootstrap Statistics :
      original      bias    std. error
```



```
t1* 0.04053538 -0.03116794   0.3955436
```


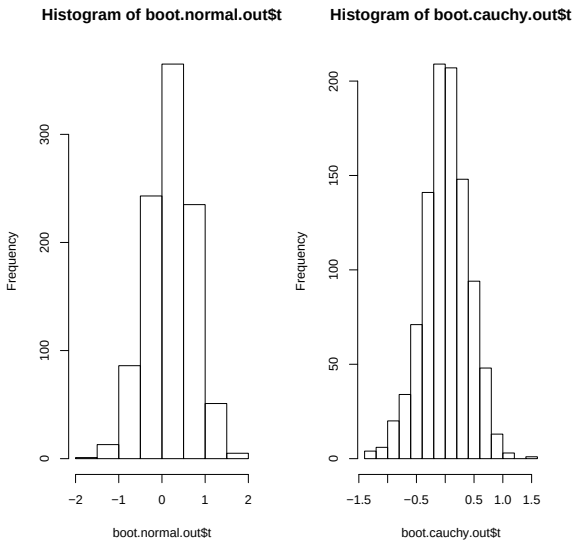

Figure 1: Histograms of bootstrap statistics