22S:166
Introduction to the Bootstrap

Lecture 11
September 29, 2017

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

Resources

- Efron, B. (1982) *The Jackknife, the Bootstrap, and Other Resampling Plans.* Number 38 in CBMS-NSF Regional Conference Series in Applied Mathematics. Philadelphia: SIAM.
- Efron, B. and Tibshirani, R.J. (1993) *An Introduction to the Bootstrap.* New York: Chapman & Hall.
- Davison, A.c. and Hinkley, D.V. (1997) *Bootstrap Methods and their Application*, New York: Cambridge University Press.
- materials listed under Web Resources

Review concepts

- suppose we have one sample of n data values: $y_1, \ldots, y_n$
- sample values considered outcomes of i.i.d. random variables $Y_1, \ldots, Y_n$
- probability density function (pdf) or probability mass function (pmf) f
- cumulative distribution function (cdf) F
- sample will be used to make inference
  - about population characteristic θ
  - using statistic T whose value in sample is t
- questions of interest regarding T
  - bias?
  - standard error?
  - quantiles?
  - how to compute confidence limits for θ?

  - likely values under a null hypothesis of interest?

Two classes of statistical methods

- parametric
  - particular mathematical model for behavior of random variables Y_j
  - pdf or pmf f is completely determined by values of unknown parameters ψ
  - quantity of interest in statistical analysis θ is a component or function of ψ
- nonparametric
  - uses only the fact the Y_js are i.i.d.
  - no mathematical model for their distribution
  - (may be useful to do a nonparameteric analysis even if a reasonable parametric model exists)
    - ∗ to assess sensitivity of conclusions to assumptions of parametric model

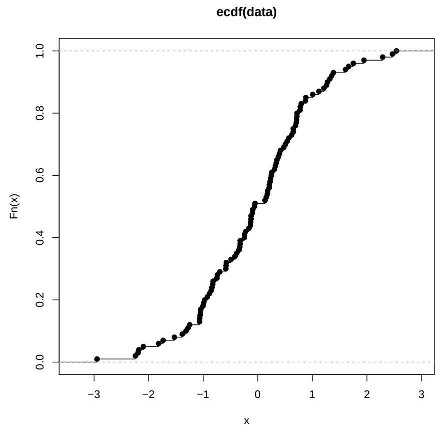
The empirical distribution

- puts probability mass $\frac{1}{n}$ at each sample value y_j
- empirical distribution function (edf) or $\hat{F}$
  - nonparametric mle of F
  - sample proportion $\hat{F}(y) = \frac{\#\{y_j \leq y\}}{n}$
    - ∗ where # denotes the number of items in a set
- edf plays role of fitted model when no mathematical form is assumed for F

Example of edf

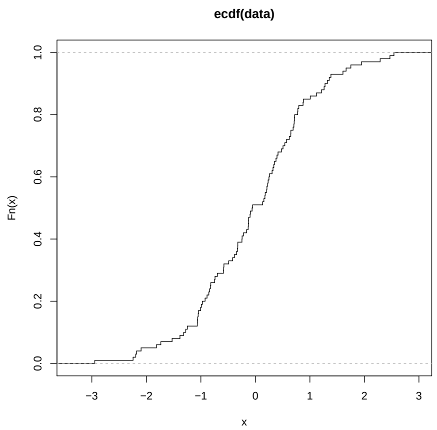
```
> data <- rnorm(100)
> Fndata <- ecdf(data)
> Fndata
Empirical CDF
Call: ecdf(data)
 x[1:100] = -2.946, -2.2434, -2.1946,  ...,
> plot(Fndata)

> plot(Fndata,verticals=T, do.points=FALSE)
```



Example for the nonparametric bootstrap: City population data

- for each of $n = 49$ U.S. cities, two data values
  - u_j = population in 1920 (in 1000s)
  - x_j = population in 1930 (in 1000s)
- population of interest is all U.S. cities



- the 49 cities are assumed to be a simple random sample from this population
- define (U,X) as pair of population values for a randomly selected city
- then if we knew $\theta = \frac{E(X)}{E(U)}$ and the total 1920 population for the U.S., we could estimate the total 1930 population of U.S.

- want to estimate θ without assuming any parametric model for X and U
- sample-based statistic is $T = \frac{\bar{X}}{\bar{U}}$
- observations 1 to 10 of this dataset are included with the `boot` package for R

```
> library(boot)
> data(city)
> city
     u   x
1  138 143
2   93 104
3   61  69
4  179 260
5   48  75
6   37  63
7   29  50
8   23  48
9   30 111
10   2  50
```

The non-parametric bootstrap

- goal: to get an idea of the sampling distribution of the statistic T under repeated sampling from the population of interest
- basic idea: our sample data gives us all the information we have about the whole population
- steps:
  1. calculate statistic of interest (call it $\hat{\theta}$) from dataset as a whole
  2. fit edf $\hat{F}$
  3. Draw a "bootstrap sample" from $\hat{F}$ and calculate statistic of interest on bootstrap sample
     - i.e., draw a sample of size n from original dataset **with replacement**
     - $Y_1^*, Y_2^*, \ldots, Y_n^* \sim \hat{F}$
     - $\hat{\theta}^* = \hat{\theta}(Y_1^*, Y_2^*, \ldots, Y_n^*)$

4. repeat step 2 independently a large number B of times obtaining bootstrap replications $\hat{\theta}^{*1}, \hat{\theta}^{*2}, \ldots, \hat{\theta}^{*B}$
5. Use bootstrap replications to:
   - estimate standard error of $\hat{\theta}$
   - estimate bias
   - obtain confidence interval

Using the R `sample` function to draw bootstrap samples

```
sample                package:base               R Documentation

Random Samples and Permutations

Description:

     'sample' takes a sample of the specified size from the
     elements of 'x' using either with or without replacement.

Usage:

     sample(x, size, replace = FALSE, prob = NULL)

Arguments:

       x: Either a (numeric, complex, character or logical)
          vector of more than one element from which to choose,
          or a positive integer.

    size: non-negative integer giving the number of items to
          choose.

 replace: Should sampling be with replacement?

    prob: A vector of probability weights for obtaining the
```

```
          elements of the vector being sampled.


> x <- seq(1:25)
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
     19 20 21 22 23 24 25
> sample(x, 25)
 [1]  2 20  3  9  6  8 15 10 23  1 19 25 12 21 14  4 13 24
     17  5 11 18  7 22 16
> sample(x, 25, replace = TRUE)
 [1]  4  6 16 11 21 17  6 12  5  8 15 19 23 16 15 20 18 19
     21  5 25  7  8 20  3

> mindex <- sample(1:10, replace=T)
> mindex
 [1]  4  9  1  9 10  9  6  5  3  3
> city[mindex, ]
      u   x
4   179 260
9    30 111
1   138 143
9.1  30 111
10    2  50
9.2  30 111
6    37  63
5    48  75
3    61  69
```

```
3.1  61  69
```

Bias correction using the bootstrap

- notation
  - θ – true and unknown population quantity value
  - $\hat{\theta}$ – estimate of θ based on sample data
  - $\hat{\theta}^{*b}$ – estimate of θ from b-th bootstrap sample

Bias correction continued

- So in a sense:
  - $\hat{\theta}^{*}$s are to $\hat{\theta}$ as $\hat{\theta}$ is to θ
- bootstrap esimate of bias
  - Note: bias $= E_F(\hat{\theta} - \theta)$

$$\begin{aligned} \widehat{bias}_{boot} &= \frac{1}{B}\left(\sum_{b=1}^{B} \hat{\theta}^{*b} - \hat{\theta}\right) \\ &= \hat{\theta}^{*\cdot} - \hat{\theta} \end{aligned}$$

- So bias-corrected point estimate is

$$\begin{aligned} \tilde{\theta} &= \hat{\theta} - \left(\hat{\theta}^{*\cdot} - \hat{\theta}\right) \\ &= 2\hat{\theta} - \hat{\theta}^{*\cdot} \end{aligned}$$

R code for the City Data

```
> library(boot)
> help(boot, package="boot")
-----------------------------------------------------------------------------
boot                  package:boot                  R Documentation

Bootstrap Resampling

Description:

     Generate ’R’ bootstrap replicates of a statistic applied to data.
     Both parametric and nonparametric resampling are possible.  For
     the nonparametric bootstrap, possible resampling methods are the
     ordinary bootstrap, the  balanced bootstrap, antithetic
     resampling, and permutation. For nonparametric multi-sample
     problems stratified resampling is used.   This is specified by
     including a vector of strata in the call to boot. Importance
     resampling weights may be specified.

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, ...)

Arguments:

    data: The data as a vector, matrix or data frame.  If it is a
          matrix or data frame then each row is considered as one
          multivariate observation.

statistic: A function which when applied to data returns a vector
          containing the statistic(s) of interest.  When
          ’sim="parametric"’, the first argument to ’statistic’ must be
          the data.  For each replicate a simulated dataset returned by
          ’ran.gen’ will be passed.  In all other cases ’statistic’
          must take at least two arguments.  The first argument passed
          will always be the original data. The second will be a vector
          of indices, frequencies or weights which define the bootstrap
          sample. Further, if predictions are required, then a third
          argument is required which would be a vector of the random
          indices used to generate the bootstrap predictions.  Any
```

```
          further arguments can be passed to ’statistic’  through the
          ’...{}’ argument.

       R: The number of bootstrap replicates.  Usually this will be a
          single positive integer.  For importance resampling, some
          resamples may use one set of weights and others use a
          different set of weights. In this case ’R’ would be a vector
          of integers where each component gives the number of
          resamples from each of the rows of weights.

     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.

   stype: A character string indicating what the second argument of
          statistic represents. Possible values of stype are ’"i"’
          (indices - the default), ’"f"’ (frequencies), or ’"w"’
          (weights).

Details:

     The statistic to be bootstrapped can be as simple or complicated
     as desired as long as its arguments correspond to the dataset and
     (for a nonparametric bootstrap) a vector of indices, frequencies
     or weights.  ’statistic’ is treated as a black box by the ’boot’
     function and is not checked to ensure that these conditions are
     met.


Value:

     The returned value is an object of class ’"boot"’, containing the
     following  components :

      t0: The observed value of ’statistic’ applied to ’data’.

       t: A matrix with ’R’ rows each of which is a bootstrap replicate
          of ’statistic’.

       R: The value of ’R’ as passed to ’boot’.
```

```
    data: The 'data' as passed to 'boot'.

    seed: The value of '.Random.seed' when 'boot' was called.

statistic: The function 'statistic' as passed to 'boot'.

     sim: Simulation type used.

   stype: Statistic type as passed to 'boot'.

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

Example of nonparametric bootstrap with boot package:

# define "statistic" function
> meanratio <-
function( mydat, indices )
{
if (!(is.matrix( mydat)  && ncol(mydat) == 2 && length(indices)==
nrow(mydat) ))
{
    stop("invalid arguments")
}

mean( mydat[indices,2]) / mean(mydat[indices,1])
}

# call boot function
> boot.out <- boot( as.matrix(city), meanratio, 999)

# summarize results

> boot.out

ORDINARY NONPARAMETRIC BOOTSTRAP

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


Bootstrap Statistics :
    original     bias    std. error
t1* 1.520312 0.04051090   0.2263570

----------------------------------------------------------------------------
```

```
# bootstrap c.i.

> help(boot.ci, package="boot")

boot.ci                 package:boot                 R Documentation

Nonparametric Bootstrap Confidence Intervals

Description:

     This function generates 5 different types of equi-tailed two-sided
     nonparametric confidence intervals.  These are the first order
     normal  approximation, the basic bootstrap interval, the
     studentized bootstrap  interval, the bootstrap percentile
     interval, and the adjusted bootstrap  percentile (BCa) interval.
     All or a subset of these intervals can be  generated.

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
```