A primer to bootstrapping; and an overview of `doBootstrap`

Desmond C. Ong
Department of Psychology, Stanford University

August 20, 2014

Abstract

This primer is targeted mainly at psychologists in light of the current paradigm shift in psychological statistics away from Null Hypothesis Statistical Testing towards better statistics. I shall focus here on one particular tool: the bootstrap. This set of notes will be divided into two parts: the first part will contain a short primer to non-parametric statistics, and in particular, the bootstrap. The second part will contain an overview of several R functions that I wrote to assist in calculating bootstrapped statistics. The link to the code can be found at : `https://github.com/desmond-ong/doBootstrap`

0.1 Moving towards the "new statistics"

Disclaimer: This section will be rewritten to be more "formal" in the future. This is the "short and sweet" version. For more information, I would suggest reading the new guidelines adopted in January 2014 by Psychological Science (Cumming, 2014).

There is an ongoing paradigm shift in the psychological sciences towards better statistical practices. One major component is to move away from p-values and Null Hypothesis Statistical Testing (NHST). Cumming (2014) recommends (not an exhaustive list):

- Not using p-values at all. Don't use Null Hypothesis Statistical Testing (NHST)
- When reporting statistical tests, reporting effect sizes (ES) and Confidence Intervals (CIs).
- In graphs, ploting error bars as 95% CIs instead of Standard Errors (SEs). (Note that a quick way to obtain 95% CIs is to multiply the SEs by 1.96. But this has a hidden assumption that the underlying statistic is normally distributed!)

One of the suggested alternatives is to move towards using non-parametric statistics. This set of notes is a primer to one of the more powerful tools that we can easily use in psychological research: the bootstrap.

1 The Bootstrap

1.1 Motivating non-parametric statistics

Most of the statistics that we are familiar with are parametric statistics. These statistics make assumptions about the distribution of the underlying property, and often simplify calculations and make closed-form solutions possible. We often assume that the distribution can be described with a number of *parameters* (hence, "parametric"). For example, we can say that a variable X is distributed normally about 10 with standard deviation 1. This allows us to compute the probability that we draw a sample that is above 12, Pr(X>12), and so on.

Many of these statistics make strong assumptions about the distribution. The most common are that the distribution is normal, it's symmetric, the error term is independent of x (homoscedasticity), and there are no outliers. If these assumptions are violated, then parametric statistics would give unreliable results.

One way around this is to use non-parametric methods that make no assumptions about the underlying distribution. A common class of non-parametric methods are resampling methods, which allows the estimation of population variables via continual resampling of the empirical sample. These methods often have less statistical power than parametric methods, but are much more applicable in a wider variety of situations. One drawback is that they tend to be computationally expensive, but that has become much less of a problem in today's world.

1.2 The bootstrap

The population is to the sample what the sample is to the bootstrapped sample

Bootstrapping is a way of estimating statistical parameters (e.g. the population mean and its confidence interval) from the sample by means of *resampling with replacement*. Like other non-parametric approaches, bootstrapping does not make any assumptions about the distribution of the sample (e.g. whether it's normally distributed and hence can be characterized by mean and variance parameters). The major assumption behind bootstrapping is that the sample distribution is a good approximation to the population distribution, i.e. that the sample is representative of the population.

Here's a simple example. Consider a situation as in Figure 1, where a sample of five observations is drawn from a larger population, and there is a variable of interest. In this case, let's assume this to be the weight of each ball, clockwise from top left {4, 5, 7, 2, 4}. From this sample, we can easily calculate the mean (4.4), the standard deviation (1.82), the standard error (0.812), multiply the SE by 1.96 to get the width of the CI (1.59). Thus, we might report the mean as 4.4 [2.81, 5.99].

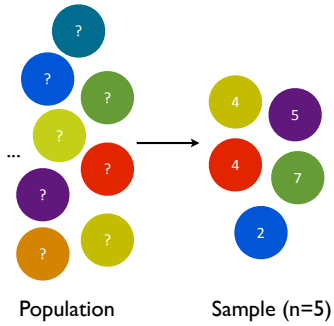

Figure 1: Taking a sample of n=5 from the population

But there were many hidden assumptions in performing those simple calculations! For one, we had assumed that the distribution of weights was normal. For this particular sample, we might say that that is a pretty bad assumption (well, the sample size is too small to really say for sure). Another assumption we made was that the confidence intervals are symmetric: this might not be the case either (the sample is right-skewed, but again, the sample size is too small to generalize).

We could try to take a bootstrapped sample[1] by **resampling with replacement** from the original sample. This means that each observation in the sample has the same probability to be sampled; and because it is with replacement, the observation has the same probability to be picked subsequently as well.

[1]There are problems with bootstrapping small samples (some textbooks say the sample size should be larger than 8). I chose 5 because it's easy to illustrate here.

Figure 2 shows an example of 3 bootstrapped samples. We could then calculate the means for each of the bootstrapped samples (5, 4.4, 3.6). This process of resampling and recalculating the statistic is repeated for some number of iterations, e.g., 5,000 iterations. Now we can sort these bootstrapped estimates, extract the median, the 2.5th and 97.5th percentile of these estimates, to obtain an overall estimate and confidence interval of the population mean.

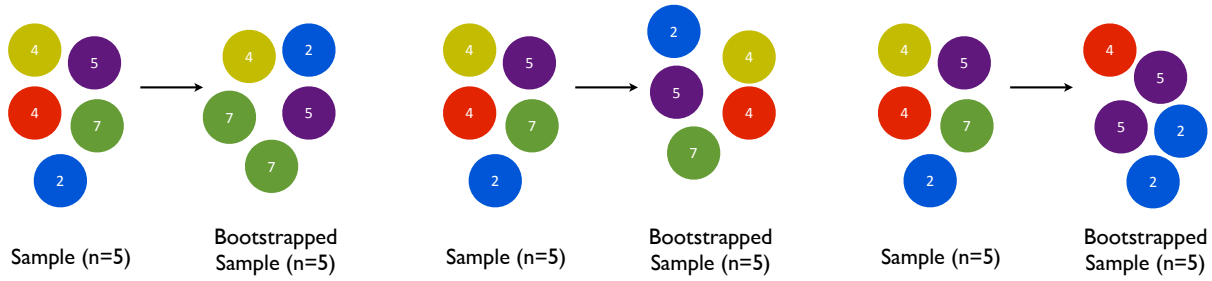

Figure 2: Taking 3 bootstrapped samples of n=5 from the original sample (n=5). Notice that the middle bootstrapped sample reproduces the original sample exactly. Counter to a naive intuition, resampling the original sample exactly is NOT more likely (and in fact, is just as likely) as drawing a "skewed" sample such as the one on the right.

Performing a bootstrap with 5,000 iterations, I obtained a mean of 4.4 [3, 6] (compared with the earlier parametric calculation of 4.4 [2.81, 5.99]) Notice that the bootstrapped estimate's CI is not symmetric. The difference in this small, made-up sample isn't astounding, but I hope to have illustrated how general this technique can be. In fact, if you look at this process schematically (Figure 3), you can see why the crucial assumption is just that the original sample is representative of the population. The bootstrapped samples are drawn from the original sample (just like the original sample was drawn from the population), and hence for the bootstrap to be reliable, the original sample has to be representative of the population.

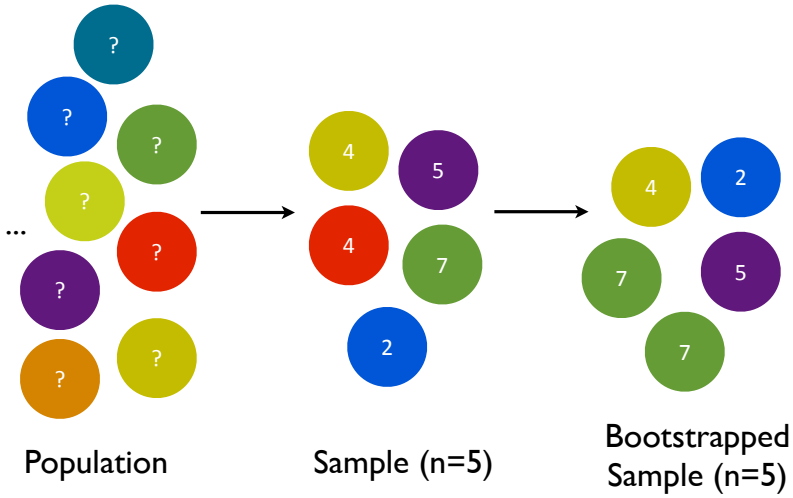

Figure 3: The population is to the sample what the sample is to the bootstrapped sample

Some points to note:

- Bootstrapping is essentially a (pseudo-)random simulation. That means that if I repeat my bootstrapping calculation, I might get a slightly different result. This shouldn't matter too much, and

performing more iterations should help to stabilize your results. If you wanted to reproduce the same result every time, consider using a `seed` in your program – if you run a simulation multiple times with the same seed, it should give you the same answer every time: that's partly why computer simulations are pseudo-random rather than truly random.

- How many observations should we resample? A good suggestion is the original sample size.
- How many iterations should we run? Well, in the past, processing power was a big limitation, but nowadays, it isn't really. The field has to come to a consensus to a conventional number, but in the meantime, I would suggest something like 5,000 (which my statistics lecturer proposed too when I learnt bootstrapping). (Technically, one would keep increasing the iteration size until the simulation converges, i.e. when the results from a run doesn't change if you add more iterations.)

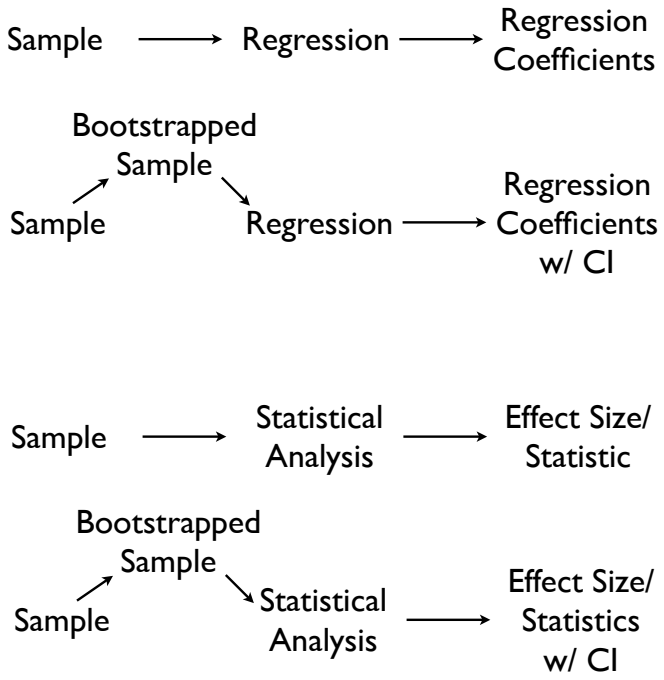

Figure 4: Illustrating the generalizability of the bootstrap

The bootstrap process is very generalizable (Figure 4). If you would run a regression on a sample to get out regression coefficients of interest, you could bootstrap the regression to estimate confidence intervals on the coefficients. If you had any statistical analysis of interest that would calculate an effect size or a statistic, you can bootstrap it to get out confidence intervals on that statistic.

1.3 Other resampling methods

Other methods that you might want to look into are jackknifing, split-half correlations, permutation tests.

The basic version of jackknifing involves removing one observation from the sample ("delete-1 jackknifing") and calculating the quantity of interest (e.g. the mean), before replacing that observation. This process is repeated one for each observation. Thus, for a sample of size N, there will be a total of N subsets of size (N-1) each; the statistic is calculated for each of these N subsets and averaged to give an estimate. There are other generalizations like the delete-m jackknife where m is specified.

2 doBootstrap.R

First, using what we've already discussed in the previous section, we see that bootstrapping is actually conceptually quite easy to grasp – and to code as well. Here's some very basic pseudo-code in R that serves as a template for writing your own bootstrap function.

```
d0 <- "read_in_dataset"
bootstrappedCoefficients <- numeric(numberOfIterations)
for (j in 1:numberOfIterations) {
   bootstrappedDataset <- d0[sample(nrow(d0), nrow(d0), replace = TRUE, prob=NULL), ]
   bootstrappedCoefficients[j] <- "calculate_statistic(bootstrappedDataset)"
}

quantile(bootstrappedCoefficients, .500) # median
quantile(bootstrappedCoefficients, .025) # 2.5th percentile
quantile(bootstrappedCoefficients, .975) # 97.5th percentile
```

Basically all the action happens in the resampling loop. This also serves as the backbone behind `doBootstrap`. Thus, if you ever need to modify `doBootstrap` or make your own function to calculate your statistic of interest, you'll know how to.

`doBootstrap.R` consists of two main functions, `doBoot` and `doBootRegression`, as well as some helper functions.

2.1 doBoot

`doBoot` is the basic function that calculates descriptive statistics from one or two samples

```
doBoot(x, y=NULL, mediator=NULL, whichTest = NULL, customFunction = NULL,
                   numberOfIterations = 5000,
                   confidenceInterval=.95, na.rm=TRUE)
# Input:
#   x = first data vector. Feed in a VECTOR!
#   y = second data vector (not necessary if you're doing statistics on one data vector)
#   mediator = mediator data vector (if you want to do mediation)
#   whichTest = the test you want to do, as a string.
#           If not provided, program will prompt for a test.
#   numberOfIterations = number of bootstrapping iterations. Default is 5000
#   confidenceInterval = specifies the percentile of the CI. Default is .95
#   na.rm = remove NAs?
```

Currently the tests supported are:

- "mean" (single vector test)
- "correlation" (correlation between two vectors, paired)
- "difference, unpaired" (difference between two vectors, unpaired)
- "difference, paired" (difference between two vectors, paired)
- "cohen, unpaired" (basically unpaired difference / pooled standard dev) * not bias corrected nor accelerated
- "cohen, paired" (basically paired difference / standard deviation) * not bias corrected nor accelerated

- "mediation" (calculates a $x \rightarrow y$, $x \rightarrow med \rightarrow y$ mediation, using Benoit's bm.bootmed code) basically just feeds into Benoit's code for now.
- "custom function": allows you to supply your own function, such that the desired statistic is `customFunction(x)` or `customFunction(x,y)` if `y` is supplied.
  Warning: `doBoot` doesn't test if you have supplied the correct number of arguments. It'll just try to call it.

Example usage:

- `doBoot(x)` – will prompt you to choose the test
- `doBoot(x, whichTest = "mean")`
- `doBoot(x, y, whichTest = "custom function", customFunction=cor)` : if customFunction is not provided, `doBoot` will prompt for it.
- `doBoot(x, y, med, whichTest = "mediation")`

2.2 doBootRegression

`doBootRegression` is the function you call if you want to bootstrap regressions. It returns all the (fixed-effect) coefficients, and for fixed-effect only models, the R^2 statistic as well.

```
doBootRegression(dataset, formula, mixedEffects = FALSE, numberOfIterations = 5000,
                  confidenceInterval=.95, na.rm=TRUE)
# Input:
#   dataset = input dataset
#   formula = the regression formula of interest. For e.g., "y ~ x" or "y ~ x + (1|z)"
#   mixedEffects = whether your formula has mixed effects. Default FALSE.
#       If true, will use lmer, otherwise, lm.
#       For lmer, will output only fixed effect coefficients
#       For lm, will output coefficients, and R^2
#   numberOfIterations = number of bootstrapping iterations. Default is 5000
#   confidenceInterval = specifies the percentile of the CI. Default is .95
#   na.rm = remove NAs?
```

Currently `doBootRegression` supports:

- Fixed-effects only multiple linear regression (i.e. with `lm`).
- Random-effects multiple linear regression (i.e. with `lmer`).

Example usage:

- `doBootRegression(d0, income ~ age + education)`
- `doBootRegression(d0, income ~ age + education + (1|subjID), mixedEffects = TRUE)`

References

Cumming, G. (2014). The New Statistics: Why and How. *Psychological Science*, *25*(1), 7-29