Brute Force as a Statistical Tool.*

Greg Kochanski
http://kochanski.org/gpk

2005/03/01 12:09:03 UTC

1 Introduction

Several statistical techniques have appeared over the last 50 years that are dramatic departures from classical statistics. These methods are:

- Easier to remember than classical approaches
- Able to handle problems that don't fit the requirements of classical approaches.
- Able to simply handle complex problems where classical approaches would require extensive and careful analysis.
- Able to handle extremely messy problems which would just be inconceivable with classic techniques.
- Intensive consumers of CPU cycles.

Combined with modern fast computers, these techniques are the method of choice for most problems.

The methods go by the names of "simulation", "Monte-Carlo", "Bootstrap Resampling", and "The Jackknife." In essence, all of them involve repeating your calculation hundreds or thousands of times (with a different data set each time), and looking at the range of results it produces. The rules are simple, and the result you want is available almost by inspection.

2 Bootstrap Example - Average

Suppose that you had a statistical procedure for estimating the fraction of lines that follow a iambic meter, in classical Greece in 440BCE. We will have to assume, for the sake of your sanity that it can be written down as a computer program that operates on a set of texts. It computes

$$I(S), \tag{1}$$

where the set of texts is

$$S = \{t_0, t_1, t_2, \ldots\}, \tag{2}$$

and t_i is the i^{th} text. S is your *sample*. It contains all the available information. From it, you want to extrapolate to make statements about the entire language in 440BCE, including texts that have been lost.

Suppose you take all the available texts and run your procedure (Eq.1), and get the answer 0.404. What then? How accurate is your answer? Can you really disprove your arch-competitor's theory of meta-dialectical stress hybridisation, that claims that only 20% were iambic?

Written as a pseudo-program, here's how to find out:

```
start with S
N = the number of texts in S
for attempt between 1 and 1000:
        Pick a new set of N texts from S by sampling with replacement.
        Call this new set s.
        Compute I(s) and add it to a table.

Take the table of I(s) and find the $5^{\mathrm{th}}$ and $95^{\mathrm{th}}$ percentiles.
Those are the lower and upper limits of your confidence interval.
If your arch-competitor is outside the confidence interval,
he or she is wrong (with 90% certainty).
```

This procedure can work with **extremely** complex calculations. You can wrap an entire computer program, no matter how messy, in the bootstrap re-sampling loop.

Essentially, you create a new set of input data by sampling with replacement from your complete data set. You run the analysis on that sampled set. You repeat and repeat. Then, when you have repeated enough, you look at the distribution of answers. What could be simpler?

While there are some limitations and caveats which we will discuss later, it is indeed hard to get much simpler. Let's do a couple of real (but simple) examples:

2.1 Bootstrap

The simplest example is finding a confidence interval for the average of N numbers. (Here, we use samples of an exponentially distributed random variable [Lindgren, 1976, pp.180–182] with mean and standard deviation both one.)

Using the bootstrap approach, we define how to sample the data[1]:

```
def sampler(data):
        # This is sampling with replacement: thus some of
        # the data will not be chosen at all, some will be
        # chosen once, and some will be chosen twice.
        N = len(data) # len(data) is the number of elements in the data set.
        return [ random.choice(data) for i in range(N) ]
```

we define the algorithm to produce the average:

```
def average(data):
        sum = 0.0
        for datum in data:
                sum = sum + datum
        return sum/len(data)
```

and then we do the bootstrap:

```
for j in range(10000):
        print average( sampler( data ) )
```

This prints out a long list of values of the average which we can feed into our favourite statistical package and plot as a histogram. The result is shown in Figure 2.1, left, (black dots), along with the histogram of 1,000,000 independent samples from the underlying population as a comparison (blue curve). Keep in mind that the black dots are derived from a single sample of 20 measurements, which is a realistically small amount of data. The blue curve, on the other hand is derived from 20,000,000 independent measurements, and represents how well you could do with a virtually infinite budget. The right hand plot is equivalent, but was constructed from a different sample of 20 measurements. The difference between the two is just due to the intrinsic sample-to-sample scatter of the measurements.

[1] FYI, this is Python code.

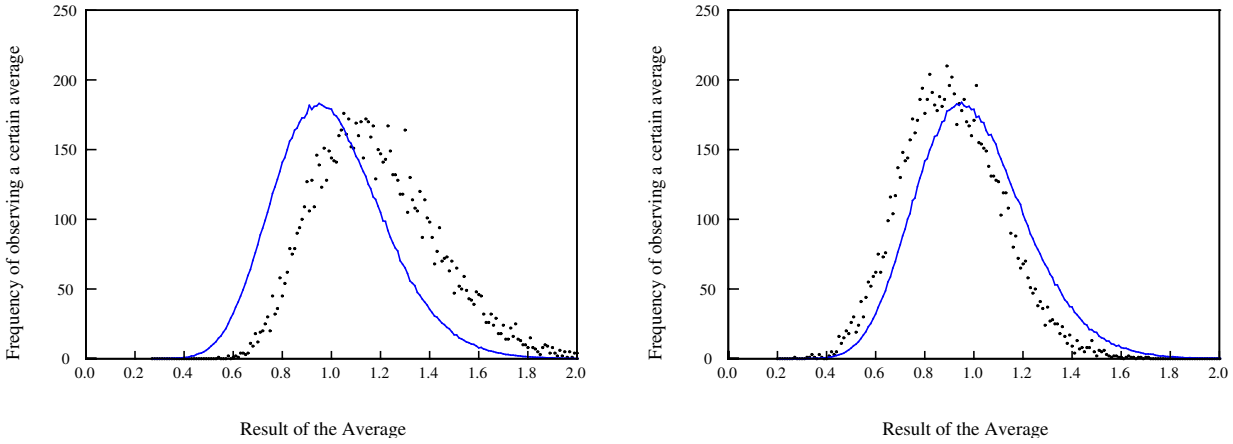

Figure 1: Left: Comparison of a histogram of Bootstrap estimates of the population average which were derived from a single sample (black dots), with an accurate calculation of the histogram of the population average (blue curve). Right: the same, but Bootstrap estimates derived from a different, single sample.

As you can see, the Bootstrap estimate gives a spectacularly good approximation to the correct probability distribution. Consequently, any properties that you can derive from the underlying probability distribution can be derived from a Bootstrap estimate. Specifically, you can get whatever percentiles you need to form your confidence interval[2].

2.2 Classical Statistical Approaches

Admittedly, if these numbers were chosen from a Gaussian distribution, classical statistics would be almost as easy: you take the mean (μ) and standard deviation (σ) of the data, and divide by $N^{1/2}$ to get the standard deviation of the average. Then, you look up the t-distribution with N degrees of freedom [Lindgren, 1976, pp.339–345, 576 (Table III)], and find they are at (*e.g.*) plus and minus 1.75 times the average's standard deviation), then do a little algebra to get the confidence interval to be $\mu - 1.75 \cdot \sigma \cdot N^{-1/2}$ to $\mu + 1.75 \cdot \sigma \cdot N^{-1/2}$. Of course, these answers may not be precise if there is reason to believe that the data is not drawn from a Gaussian[3], and we have hidden much of the complexity by looking up the confidence intervals in a book. The t-distribution is, after all, a complex beast involving both an uncertain mean and variance.

A plot comparing the two approaches is shown in Figure 2.2. The central vertical lines show the population mean, the mean of the actual sample, the classical confidence limits and the histogram of averages derived from bootstrap re-sampling. As you can see, everything is in quite good agreement: the classical mean is right around the peak of the bootstrap curve, and about 5% of the bootstrap samples peek out on either side of the classical confidence intervals, indicating that the bootstrap confidence interval pretty nearly matches the classical confidence interval. (Mind you, the classical confidence interval isn't perfection: it only includes the true mean 90% of the time.)

3 Why does it work?

It works for the trivially simple reason that (absent other knowledge) the data itself is the best possible estimate of the underlying probability distribution from which it came.

[2] In fact, if you need more exotic things like the average of the sine of 12 times the average of the data, you can compute that, too! It takes exactly four more lines of code, and the bootstrap answer is -0.006, which is quite close to the million-sample answer of -0.045.

[3] Although for this particular problem the Central Limit Theorem will normally make the probability distribution of the average quite close to Gaussian (unless the input data has very long tails) so classical statistics will give you an answer that is normally (for this kind of problem) very good.

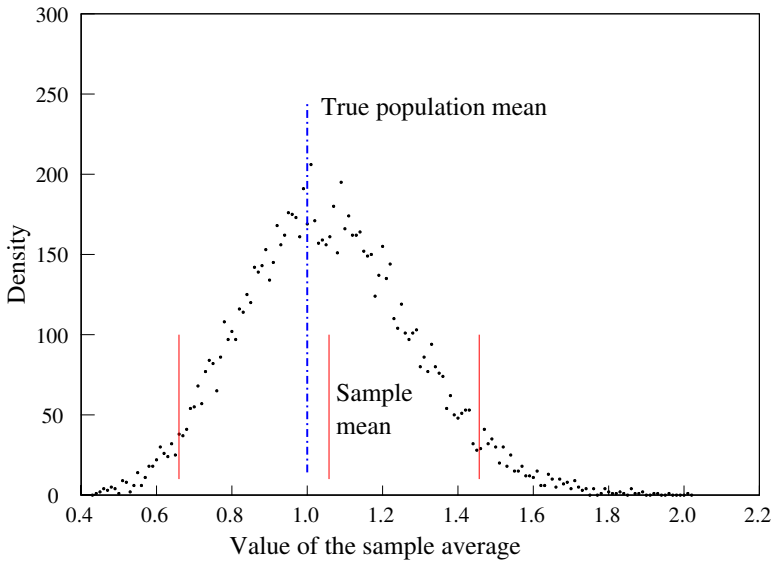

Figure 2: Comparison of Bootstrap histogram with Gaussian Confidence Intervals

The Bootstrap is not the necessarily the most efficient technique: classical statistical techniques do make the best possible use use of the data ***if*** the data is known to be Gaussian. However, Mother Nature is not at all required to produce Gaussian data, and often doesn't. The Bootstrap wins when the data is non-Gaussian or the model is too complex for classical statistics to be computed.

4 Limitations of the Bootstrap

The limitations of the Bootstrap are straightforward:

1. You need enough data. The mathematical proofs of the accuracy of Bootstrap estimates are valid only in the limit of very large samples. This limit is hard to precisely quantify, because it depends on the function/algorithm that you are bootstrapping and the statistical properties of the data (which is unknown).

   Loosely speaking, your algorithm has to produce enough distinct values to "fill up" the histogram.

   For well-behaved algorithms, like the average, the Bootstrap can produce good results with as few as 15 samples. Possibly ten, if you are willing to live dangerously. Most rules of thumb in the literature indicate $N \geq 20$ or $N \geq 50$ [Chernick, 1999, p.150–151].

   The trouble is, for small samples, there are a relatively few distinct bootstrap samples[4] that can be drawn, so one starts to get identical samples. Consequently, estimates of the variances will tend to be too small.

   For example, there are $B = C(2 \cdot N - 1, N)$ possible bootstrap samples when you have N data[5]. That means that there are at most 92,378 distinct samples for 10 data, or (at most) 92,378 possible outcomes from Equation 1. These numbers are not really *that* big, considering that you may want hundreds or thousands of samples to measure things like covariances.

   One would like $B = 100$ or more samples to allow a bootstrap estimate of the bias of some estimator, or to estimate its variance. To measure the covariance of a M-parameter model, you need at least a few times M^2 distinct samples, which can become large from a complex model.

[4] *I.e.* the number of distinct results possible from the sampling with replacement procedure.

[5] $C(2 \cdot N - 1, N)$ is the number of distinct combinations of $2 \cdot N - 1$ objects chosen N at a time. $C(m, n) = \frac{m!}{n!(m-n)!}$, where $n! = n \cdot (n-1) \cdot (n-2) \ldots \cdot 3 \cdot 2 \cdot 1$ is n-factorial.

2. You need an algorithm that is not sensitive to duplicate data. The sampling-with-replacement procedure, which is the first step in the Bootstrap, will produce samples where lots of the data are used more than once. For instance, about a quarter of the points will be used twice.

   Averages are fine: duplicate data doesn't cause any trouble. However, an algorithm that calculates the mode could be severely affected by duplicate data. Worst of all would be an algorithm that explicitly looks for unique data. Applying Bootstrap re-sampling to an algorithm that tries to estimate the number of important people in ancient Greece by counting how many people appear in more than one text would be a disaster.

3. You need an algorithm whose output is not controlled by a single datum. For instance, using Bootstrap re-sampling to estimate the likely range of the sample maximum or minimum is a bad idea. Most of the time, when you re-sample, you will get the same answer: the largest datum in the re-sampled data is just the largest datum in the sample. Almost all the time, the max of the re-sampled data will be one or another of the top three data in the sample. The output histogram of the Bootstrap process will be just a few isolated spikes, and computing percentiles on it is a hopelessly silly task.

   The bootstrap estimate will always be biased low (for the maximum) or high (for the minimum) because the bootstrap sample sometimes won't include the largest datum, and it will never include points above the largest datum. (See Angus [1993].)

4. You need an algorithm that is not too sensitive to extrema, or you need an underlying probability distribution with steep tails. After all, if you only have 20 or 30 samples, it is unlikely that any of them will come from the extreme tails of the underlying probability distribution.

   The Bootstrap procedure effectively throws away the tails of the underlying distribution. For instance, if you have a Gaussian contaminated by 1% of gross errors, you probably won't see one of the gross errors until you have 100 or more samples in your data set. Without a large error in your data, the Bootstrap has no way to estimate a correctly large variance.

   Using a robust estimators like a trimmed mean (§6) can help here. Such robust statistical techniques are designed to be insensitive to a few wild data.

5. Finally, your algorithm cannot depend on the order of the data, as the re-sampling process breaks any sequences. However, it is often possible to get around this limitation by sampling a whole block of data at a time. The algorithm can then see order inside each block. In linguistics, it often makes sense to treat a document or a paragraph as a block.

5 The limits of Bootstrap: the Median

Averages are easy. Medians are a harder case, both for the Bootstrap and for classical statistics. Not harder for the Bootstrap in the sense of requiring complicated calculations and human effort, but harder in the sense of requiring much more data and CPU time to get a useful result. For instance, we can convert the Bootstrap calculation of the confidence interval of an average into a calculation of the confidence interval for a median by changing just five lines of code.

However, we will find that we need at least 30 times as many measurements in the sample (i.e. about 600) to get as good a result for the median as we got for the average with just 20 measurements. The reason is that the median almost violates point 3 above. To compute a median, you sort the data, then take the single central value (if you have an odd number of points), or take the average of the two central values. Either way, the few central data have a dis-proportionally strong effect on the result.

Equally importantly, a median doesn't have as many possible outcomes as an average, so it flirts with violation of point 1. If there are N data in a sample, then a median can only produce only $2 \cdot N$ (if N is even) or just N (if N is odd) distinct values under Bootstrap re-sampling. You see this effect as the strong spikiness of the histograms in Figure 5. In the figure, you see that while both of the Bootstrap estimates are roughly correct, in the sense that they won't give *wildly* inaccurate confidence intervals, neither 60 nor even 400 samples will give you results that are as reliable as you would get for an average with only 20 samples.

Aside from the median, Bootstrap usually works.

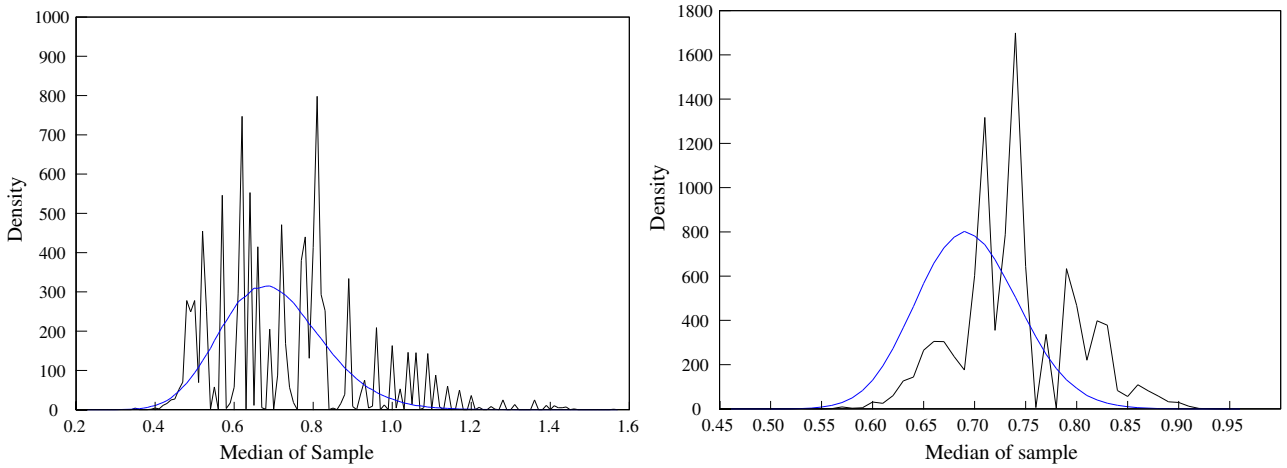

Figure 3: Left: Comparison of a histogram of Bootstrap estimates of the population median which were derived from a single sample of 60 data (black dots), with an accurate calculation of the histogram of the population average (blue curve). Right: the same, but Bootstrap estimates derived from a single sample of 400 data.

6 Robust estimators.

The reason you'd use a median is that it is a robust estimator. Specifically, if your data has a few wild points in it, using a median instead of an average can be a very good idea: wild points that will mess up an average will barely disturb a median.

However, there are other robust estimators that are far more amenable to bootstrap resampling than the median. One example is the trimmed mean. A trimmed mean is a very simple estimator: you trim off the hightest few points and the lowest few points, then you average the remaining points. It has the useful property that you can move at least one point to any value and only change the output by only a small amount.

Trimming off one point from each edge of the histogram makes the estimator able to survive one wild datum, and leaves you with an estimator that is highly efficient on Gaussian data[6]. Trimming many points gives an estimator that is robust against many wild points, but is somewhat less efficient. If one trims off all but the central datum (pair of data), one is left with the median.

Figure 6 shows a bootstrap histogram for a 40%-trimmed mean, where 40% of the data is trimmed off the top, 40% trimmed off the bottom, and then an average is taken of the remaining 20%. This estimator is quite close to the median in terms of robustness, but it behaves very well with bootstrap resampling. As can be seen, the bootstrap histogram is generally close to the "true" distribution of trimmed means, despite using one one-millionth of the data. One could compute useful error bars and confidence intervals from the histogram.

7 How do I know I'm in trouble?

As we've seen, inspecting the histogram is a very sensible procedure. If you don't have a densly filled-in histogram, deriving percentiles and error bars is risky.

In addition, you can bootstrap the bootstrap, and see if the results change wildly. It is possible to do two layers of Bootstrap re-sampling, and this is often useful to help convince yourself that the Bootstrap procedure is working well. You take your original data, choose a same-sized sample (sampling with replacement) from the data. Then, you pretend that this first-level sample is your real data. You apply your Bootstrap analysis and get some result (*e.g.* a confidence interval). You then pick another first-level sample and repeat the Bootstrap analysis. If the results generally agree, especially if you repeat a few times, you have some reason to believe that everything is OK. If they disagree, you know you have trouble.

[6] It will be slightly less efficient than the mean, in the sense that (if the data is really Gaussian), the standard deviation of the output will be slightly larger, but only slightly.

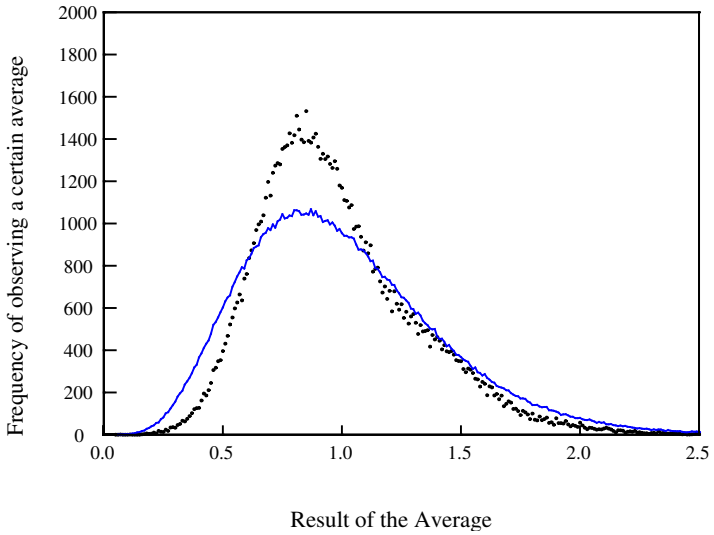

Figure 4: Boostrap histogram for a trimmed mean. This 40%-trimmed mean is nearly as robust as a median, but behaves well under bootstrap resampling. Here, we show a bootstrap histogram derived from one sample of 30 data. The histogram shows the distribution of trimmed means of bootstrap samples. The solid blue line shows a Monte-Carlo distribution of trimmed means of 1,000,000 independent samples of 30 observations each.

You can also use this idea of boostrapping as a sanity check on other statistical procedures. Just create a bootstrap sample, run your analysis again, and if your conclusions don't change, you can sleep more easily.

8 Improving the Bootstrap

Without going into detail here, I'll just note that it is possible to combine the ideas behind Bootstrap and Monte-Carlo estimates, and they can work well together.

On one extreme, the bootstrap uses a very sparse probability distribution, made only of the data you have. It is as real as possible, and makes no assumptions about the data, other than assuming you have enough data so that it is representative of the underlying probability distribution. On the other hand, you have Monte Carlo techniques that assume the underlying probability distributions are known (which is sometimes a risky and incorrect assumption), but are never short of data. Monte Carlo techniques, after all, create samples as required.

One way to combine the two is as follows:

1. Develop some reasonable model for the underlying distribution.
2. Divide it up into regions around the data you have.
3. Choose a Bootstrap sample from the data.
4. Rather than returning each datum, replace each datum with a sample from the corresponding region of the model distribution.
5. Do your data analysis.
6. Repeat stems 3–5 thousands of times.

To give a simple example, assume we have four data, 0, 3, 2, 5, and we want to estimate the confidence limits on the median.

1. We decide on a plausible model distribution: a four-humped distribution. Each hump is a Gaussian (centred on a datum) with standard deviation 0.5. Note that we don't need to settle on a Gaussian. We could equally well choose a Gaussian with mean 2.5 and standard deviation 2.6. However, a distribution with four narrow humps is more conservative, being closer to the Bootstrap distribution.

2. We divide the distribution into four regions, one for each datum. In this case, we can simply take each Gaussian hump separately. Alternatively, we could divide the distribution into regions $(-\infty, 1]$, $[1, 2.5]$, $[2.5, 4]$, and $[4, \infty)$.

3. One takes a bootstrap sample of the data: 0, 3, 3, 5.

4. Then, one takes four samples from the regions that correspond to the sample of the data. We get -0.232 from the Gaussian centred around 0, 3.741 from the Gaussian around 3, 2.7786 from the same Gaussian, and 5.53 from the Gaussian around 5.

5. Take the median of the samples: $\frac{1}{2} \cdot (3.741 + 2.7786) = 3.2598$.

6. Repeat thousands of times to build a histogram.

Note that this answer will depend on your assumptions for the model distribution, but it won't depend on the distribution as strongly as if you went whole-hog into a Monte Carlo approach.

9 Recommended Reading

Davison and Hinkley have written "Bootstrap Methods and their Application," which is rather a good book [Davison and Hinkley, 1999]. Chernick's book is also good [Chernick, 1999].

References

J. E. Angus. Asymptotic theory of bootstrapping the extremes. *Commun. Statist. Theory Methods*, 22:15–30, 1993.

Michael R. Chernick. *Bootstrap Methods, A Practitioner's Guide*. Wiley Interscience, New York, Chichester, Singapore, Toronto, 1999. ISBN 0-471-34912-7.

A. C. Davison and D. V. Hinkley. *Bootstrap Methods and their Applications*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, New York, Melbourne, Madrid, 1999. ISBN 0-521-57391-2 or 0-521-57471-4.

Bernard W. Lindgren. *Statistical Theory*. MacMillan, third edition, 1976. ISBN 0-02-370830-1.