**Nuances of Bootstrapping**

Most applied statisticians and data scientists understand that bootstrapping is a method that mimics repeated sampling by drawing some number of new samples (with replacement) from the original sample in order to perform inference. However, it can be difficult to understand output from the software that carries out the bootstrapping without a more nuanced understanding of how uncertainty is quantified from bootstrap samples.

To demonstrate the possible sources of confusion, start with the data described in Efron and Tibshirani’s (1993) text on bootstrapping (page 19). We have 15 paired observations of student LSAT scores and GPAs. We want to estimate the correlation between LSAT and GPA scores. The data are the following:

| **student** | **lsat** | **gpa** |
| --- | --- | --- |
| 1 | 576 | 3.39 |
| 2 | 635 | 3.30 |
| 3 | 558 | 2.81 |
| 4 | 578 | 3.03 |
| 5 | 666 | 3.44 |
| 6 | 580 | 3.07 |
| 7 | 555 | 3.00 |
| 8 | 661 | 3.43 |
| 9 | 651 | 3.36 |
| 10 | 605 | 3.13 |
| 11 | 653 | 3.12 |
| 12 | 575 | 2.74 |
| 13 | 545 | 2.76 |
| 14 | 572 | 2.88 |
| 15 | 594 | 2.96 |

The correlation turns out to be 0.776. For reasons we’ll explore, we want to use the nonparametric bootstrap to get a confidence interval around our estimate of \(r\). We do so using the boot package in R. This requires the following steps:

1. Define a function that returns the statistic we want.
2. Use the boot function to get R bootstrap replicates of the statistic.
3. Use the boot.ci function to get the confidence intervals.

For step 1, the following function is created:

get\_r <- function(data, indices, x, y) {

d <- data[indices, ]

r <- round(as.numeric(cor(d[x], d[y])), 3)

r

}

Steps 2 and 3 are performed as follows:

set.seed(12345)

boot\_out <- boot(

tbl,

x = "lsat",

y = "gpa",

R = 500,

statistic = get\_r

)

boot.ci(boot\_out)

## Warning in boot.ci(boot\_out): bootstrap variances needed for studentized

## intervals

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

## Based on 500 bootstrap replicates

##

## CALL :

## boot.ci(boot.out = boot\_out)

##

## Intervals :

## Level Normal Basic

## 95% ( 0.5247, 1.0368 ) ( 0.5900, 1.0911 )

##

## Level Percentile BCa

## 95% ( 0.4609, 0.9620 ) ( 0.3948, 0.9443 )

## Calculations and Intervals on Original Scale

## Some BCa intervals may be unstable

Looking at the boot.ci output, the following questions come up:

1. Why are there multiple CIs? How are they calculated?
2. What are the bootstrap variances needed for studentized intervals?
3. What does it mean that the calculations and intervals are on the original scale?
4. Why are some BCa intervals unstable?

To understand this output, let’s review statistical inference, confidence intervals, and the bootstrap.

**Statistical Inference**

The usual test statistic for determining if \(r \neq 0\) is:

\[  
t = \frac{r}{SE\_r}  
\]

where

\[  
SE\_r = \sqrt{\frac{1-r^2}{n-2}}  
\]

In our case:

\[  
SE\_r = \sqrt{\frac{1-r^2}{n-2}}  
= \sqrt{\frac{1-0.776^2}{15-2}}  
= 0.175  
\]

Dividing \(r\) by \(SE\_r\) yields our \(t\) statistic

\[  
t = \frac{r}{SE\_r} = \frac{0.776}{0.175} = 4.434  
\]

We compare this to a \(t\) distribution with \(n-2 = 13\) degrees of freedom and easily find it to be significant.

In words: If the null hypothesis were true, and we repeatedly draw samples of size \(n\), and we calculate \(r\) each time, then the probability that we would observe an estimate of \(|r| = 0.776\) or larger is less than 5%.

An important caveat. The above formula for the standard error is only correct when \(r = 0\). The closer we get to \(\pm 1\), the less correct it is.

**Confidence Intervals**

We can see why the standard error formula above becomes less correct the further we get from zero by considering the 95% confidence interval for our estimate. The usual formula you see for a confidence interval is the estimate plus or minus the 97.5th percentile of the normal or \(t\) distribution times the standard error. In this case, the \(t\)-based formula would be:

\[  
\text{95% CI} = r \pm t\_{df = 13} SE\_r  
\]

If we were to sample 15 students repeatedly from the population and calculate this confidence interval each time, the interval should include the true population value 95% of the time. So what happens if we use the standard formula for the confidence interval?

\[  
\begin{align}  
\text{95% CI} &= r \pm t\_{df = 13}SE\_r \\  
&= 0.776 \pm 2.16\times 0.175 \\  
&= [0.398, 1.154]  
\end{align}  
\]

Recall that correlations are bounded in the range \([-1, +1]\), but our 95% confidence interval contains values greater than one!

Alternatives:

* Use Fisher’s \(z\)-transformation. This is what your software will usually do, but it doesn’t work for most other statistics.
* Use the bootstrap. While not necessary for the correlation coefficient, its advantage is that it can be used for almost any statistic.

The next sections review the nonparametric and parametric bootstrap.

**Nonparametric Bootstrap**

We do not know the true population distribution of LSAT and GPA scores. What we have instead is our sample. Just like we can use our sample mean as an estimate of the population mean, we can use our sample distribution as an estimate of the population distribution.

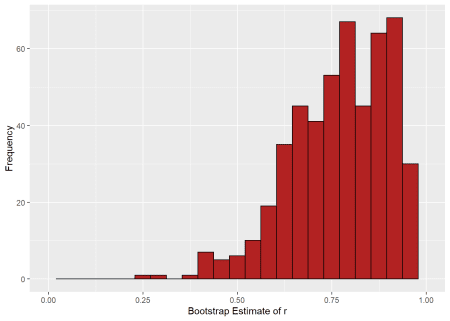
In the absence of supplementary information about the population (e.g. that it follows a specific distribution like bivariate normal), the *empirical distribution* from our sample contains as much information about the population distribution as we can get. If statistical inference is typically defined by repeated sampling from a population, and our sample provides a good estimate of the population distribution, we can conduct inferential tasks by repeatedly sampling from our *sample*.

(Nonparametric) bootstrapping thus works as follows for a sample of size *N*:

1. Draw a random sample of size *N* with replacement from our sample, which is the first *bootstrap sample*.
2. Estimate the statistic of interest using the bootstrap sample.
3. Draw a new random sample of size *N* with replacement, which is the second bootstrap sample.
4. Estimate the statistic of interest using the new bootstrap sample.
5. Repeat \(k\) times.
6. Use the distribution of estimates across the \(k\) bootstrap samples as the sampling distribution.

Note that the sampling is done with replacement. As an aside, most results from traditional statistics are based on the assumption of random sampling with replacement. Usually, the population we sample from is large enough that we do not bother noting the “with replacement” part. If the sample is large relative to the population, and sampling *without* replacement is used, we would typically be advised to use a *finite population correction*. This is just to say that the “with replacement” requirement is a standard part of the definition of *random sampling*.

Let’s take our data as an example. We will draw 500 bootstrap samples, each of size \(n = 15\) chosen with replacement from our original data. The distribution across repeated samples is:



Note a few things about this distribution.

1. The distribution is definitely *not* normal.
2. The mean estimate of \(r\) across the 500 bootstrap samples is 0.771. The difference between the mean of the bootstrap estimates \((\mathbb{E}(r\_b) = 0.771)\) and the original sample estimate \((r = 0.776)\) is the *bias*.
3. The bootstrap standard error is the standard deviation of the bootstrap sampling distribution. Here the value is 0.131, which is much smaller than our earlier estimate of 0.175. This is because 0.175 was based on a formula that is only valid when \(r = 0\) and does not account for the fact that a correlation’s values are bounded at -1 and +1.

The non-normality of the sampling distribution means that, if we divide \(r\) by the bootstrap standard error, we will not get a statistic that is distributed standard normal or \(t\). Instead, we decide that it is a better idea to summarize our uncertainty using a confidence interval. Yet we also want to make sure that our confidence intervals are bounded within the \([-1, +1]\) range, so the usual formula will not work.

Before turning to different methods for obtaining bootstrap confidence intervals, for completeness the next section describes the parametric bootstrap.

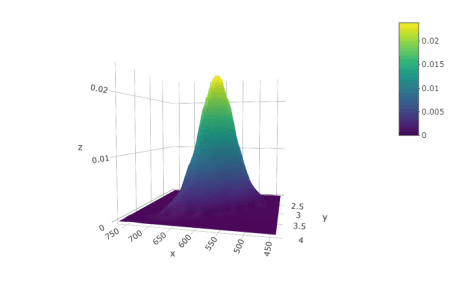
**Parametric Bootstrap**

The prior section noted that, in the absence of supplementary information about the population, the empirical distribution from our sample contains as much information about the population distribution as we can get.

An example of supplementary information that may improve our estimates would be that we know the LSAT and GPA scores are distributed bivariate normal. If we are willing to make this assumption, we can use our sample to estimate the distribution parameters. Based on our sample, we find:

\[  
\begin{pmatrix}  
\text{LSAT} \\  
\text{GPA}  
\end{pmatrix}\sim N\left(\begin{pmatrix}  
600.27 \\  
3.09  
\end{pmatrix},\begin{pmatrix}  
1746.78 & 7.90 \\  
7.90 & 0.06  
\end{pmatrix}\right).  
\]

The distribution looks like the following:



We can draw 500 random samples of size 15 from this specific bivariate normal distribution and calculate the correlation between the two variables for each.

get\_cor <- function(iteration, n) {

dta <- MASS::mvrnorm(

n,

mu = c(600, 3),

Sigma = matrix(c(1747, 7.9, 7.9, .06), 2)

) %>%

as.data.frame()

tibble(

iteration = iteration,

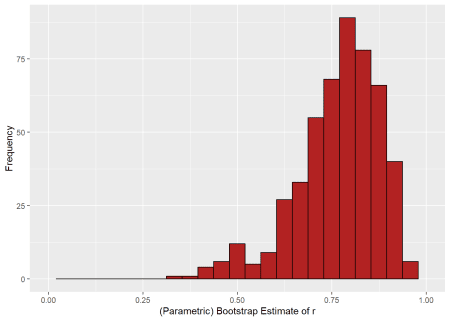
r = cor(dta$V1, dta$V2)

)

}

par\_boot\_tbl <- map\_dfr(1:500, ~ get\_cor(.x, 15))

The distribution of the correlation estimates across the 500 samples represents our parametric bootstrap sampling distribution. It looks like the following:



The average correlation across the 500 samples was 0.767, and the standard deviation (our estimate of the standard error) was 0.111.

This is smaller than our non-parametric bootstrap estimate of the standard error, 0.131, which is reflective of the fact that our knowledge of the population distribution gives us more information. This in turn reduces sampling variability.

Of course, we often will not feel comfortable saying that the population distribution follows a well-defined shape, and hence we will typically default to the non-parametric version of the bootstrap.

**Bootstrap Confidence Intervals**

Recall that the usual formula for estimating a confidence around a statistic \(\theta\) is something like:

\[  
\text{95% CI} = \theta \pm t\_{df,1-\alpha/2} SE\_{\theta}  
\]

We saw that using the textbook standard error estimate for a correlation led us astray because we ended up with an interval outside of the range of plausible values. There are a variety of alternative approaches to calculating confidence intervals based on the bootstrap.

**Standard Normal Interval**

The first approach starts with the usual formula for calculating a confidence interval, using the normal distribution value of 1.96 as the multiplier of the standard error. However, there are two differences. First, we use our bootstrap estimate of the standard error in the formula. Second, we make an adjustment for the estimated bias, -0.005:

In our example, we get

\[  
\begin{align}  
\text{95% CI} &= r – \text{bias} \pm 1.96 \times SE\_r \\  
&= 0.776 + .005 \pm 1.96 \times 0.131 \\  
&= [0.524, 1.038]  
\end{align}  
\]

This matches R’s output (given our hand calculations did some rounding along the way).

boot.ci(boot\_out, type = "norm")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

## Based on 500 bootstrap replicates

##

## CALL :

## boot.ci(boot.out = boot\_out, type = "norm")

##

## Intervals :

## Level Normal

## 95% ( 0.5247, 1.0368 )

## Calculations and Intervals on Original Scale

Problems:

* If a normal approximation were valid, we probably don’t need to bootstrap.
* We still have a CI outside the appropriate range.

We generally won’t use this method.

**Studentized (t) Intervals**

Recall that, when we calculate a \(t\)-statistic, we mean-center the original statistic and divide by the sample estimate of the standard error. That is,

\[  
t = \frac{\hat{\theta} – \theta}{\widehat{SE}\_{\theta}}  
\]

where \(\hat{\theta}\) is the sample estimate of the statistic, \(\theta\) is the “true” population value (which we get from our null hypothesis), and \(\widehat{SE}\_{\theta}\) is the sample estimate of the standard error.

There is an analog to this process for bootstrap samples. In the bootstrap world, we can convert each bootstrap sample into a \(t\)-score as follows:

\[  
t = \frac{\tilde{\theta} – \hat{\theta}}{\widehat{SE}\_{\hat{\theta}}}  
\]

Here \(\tilde{\theta}\) is the statistic estimated from a single bootstrap sample, and \(\hat{\theta}\) is the estimate from the original (non-bootstrap) sample.

But where does \(\widehat{SE}\_{\hat{\theta}}\) come from?

Just like for a \(t\)-test, where we estimated the standard error using our one sample, we estimate the standard error separately for each bootstrap sample. That is, we need an estimate of the *bootstrap sample variance*. (Recall the message from the R output above).

*If* we are lucky enough to have a formula for a sample standard error, we use that in each sample. For the mean, each bootstrap sample would return:

1. The bootstrap sample mean, \(\frac{1}{n}\sum(s\_{bi})\)
2. The bootstrap sample variance: \(\frac{s^2\_b}{n}\).

We don’t have such a formula that works for any correlation, so we need another means to estimate the variance. Alternatively, there is the *nested bootstrap*.

Nested bootstrap algorithm:

1. Draw a bootstrap sample.
2. Estimate the statistic.
3. Bootstrap the bootstrap sample, using the variance of estimates across the bootstrapped estimates as the estimate of the variance.
4. Save the bootstrap estimate of the statistic and the nested bootstrap estimate of the variance.
5. For each bootstrap sample, estimate \(t = \frac{\tilde{\theta} – \hat{\theta}}{\widehat{SE}\_{\hat{\theta}}}\)

We now have the information we need to calculate the studentized confidence interval. The formula for the studentized bootstrap confidence interval is:

\[  
95\% \text{ CI} = [\hat{\theta} – sq\_{1-\alpha/2}, \hat{\theta} – sq\_{\alpha/2}]  
\]

The terms are:

1. \(\hat{\theta}\): Our sample statistic (without performing the bootstrap)
2. \(s\): Our bootstrap estimate of the standard error (the standard deviation of bootstrap estimates, *not* the nested bootstrap part)
3. \(q\_{1-\alpha/2}\): For \(\alpha = .05\), the 97.5th percentile of our bootstrap \(t\) estimates.
4. \(q\_{\alpha/2}\): For \(\alpha = .05\), the 2.5th percentile of our bootstrap \(t\) estimates.

For each bootstrap sample, we calculated a \(t\) statistic. The \(q\_{1-\alpha/2}\) and \(q\_{\alpha/2}\) are identified by taking the appropriate quantile of these \(t\) estimates. This is akin to creating our own table of \(t\)-statistics, rather than using the typical tables for the \(t\) distribution you’d find in text books.

What does this look like in R? We need a second function for bootstrapping inside our bootstrap. The following will work.

get\_r\_var <- function(x, y, data, indices, its) {

d <- data[indices, ]

r <- cor(d[x], d[y]) %>%

as.numeric() %>%

round(3)

n <- nrow(d)

v <- boot(

x = x,

y = y,

R = its,

data = d,

statistic = get\_r

) %>%

pluck("t") %>%

var(na.rm = TRUE)

c(r, v)

}

boot\_t\_out <- boot(

x = "lsat", y = "gpa", its = 200,

R = 1000, data = tbl, statistic = get\_r\_var

)

We now have our bootstrap estimates of \(t\), and we can use the quantiles of the distribution to plug into the formula. We find that \(q\_{1-\alpha/2} = 8.137\) and that \(q\_{\alpha/2} = -1.6\). Substituting:

\[  
\begin{align}  
\text{95% CI} &= [\hat{\theta} – sq\_{1-\alpha/2}, \hat{\theta} – sq\_{\alpha/2}] \\  
&= [0.776 – 0.142 \times 8.137, 0.776 – 0.142 \times -1.6] \\  
&= [-0.383,1.004]  
\end{align}  
\]

Checking our by-hand calculations, the studentized confidence interval from boot.ci is:

boot.ci(boot\_t\_out, type = "stud")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

## Based on 1000 bootstrap replicates

##

## CALL :

## boot.ci(boot.out = boot\_t\_out, type = "stud")

##

## Intervals :

## Level Studentized

## 95% (-0.3828, 1.0039 )

## Calculations and Intervals on Original Scale

Problems:

* The nested bootstrap part is computationally intensive, even for simple problems like this.
* May still produce estimates outside the range of plausible values.
* Here our sample was small, so the variance of each bootstrap estimate was large.
* Has been found to be erratic in practice.

**Basic Bootstrap Confidence Interval**

Another way of writing a confidence interval:

\[  
1-\alpha = P(q\_{\alpha/2} \leq \theta \leq q\_{1-\alpha/2})  
\]

In non-bootstrap confidence intervals, \(\theta\) is a fixed value while the lower and upper limits vary by sample. In the basic bootstrap, we flip what is random in the probability statement. Define \(\tilde{\theta}\) as a statistic estimated from a bootstrap sample. We can write

\[  
1-\alpha = P(q\_{\alpha/2} \leq \tilde{\theta} \leq q\_{1-\alpha/2})  
\]

Recall that the bias of a statistic is the difference between its expected value (mean) across many samples and the true population value:

\[  
\text{bias} = \mathbb{E}(\hat{\theta}) – \theta  
\]

We estimate this using our bootstrap samples, \(\mathbb{E}(\tilde{\theta}) – \hat{\theta}\), where \(\hat{\theta}\) is the estimate from the original sample (before bootstrapping).

We can add in the bias-correction term to each side of our inequality as follows.

\[  
\begin{align}  
1-\alpha &= P(q\_{\alpha/2} \leq \tilde{\theta} \leq q\_{1-\alpha/2}) \\  
&= P(q\_{\alpha/2} – \hat{\theta} \leq \tilde{\theta} – \hat{\theta} \leq q\_{1-\alpha/2} – \hat{\theta})  
\end{align}  
\]

Some more algebra eventually leads to:

\[  
1-\alpha = P(2\hat{\theta} – q\_{1-\alpha/2} \leq \theta \leq 2\hat{\theta} – q\_{\alpha/2} )  
\]

The right-hand side is our formula for the basic bootstrap confidence interval.

Because we started out with \(\tilde{\theta}\) as the random variable, we can use our bootstrap quantiles for the values of \(q\_{1-\alpha/2}\) and \(q\_{\alpha/2}\). To do so, arrange the estimates in order from lowest to highest, then use a percentile function to find the value at the 2.5th and 97.5th percentiles (given two-tailed \(\alpha = .05\)). We find that \(q\_{1-\alpha/2} = 0.962\) and that \(q\_{\alpha/2} = 0.461\). Substituting into the inequality:

\[  
\begin{align}  
1-\alpha &= P(2\hat{r} – q\_{1-\alpha/2} \leq r \leq 2\hat{r} – q\_{\alpha/2} ) \\  
&= P(2(0.776) – 0.962) \leq r \leq 2(0.776) – 0.461) \\  
&= P(0.59 \leq r \leq 1.091)  
\end{align}  
\]

The basic bootstrap interval is \([0.59, 1.091]\).

To confirm:

boot.ci(boot\_out, type = "basic")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

## Based on 500 bootstrap replicates

##

## CALL :

## boot.ci(boot.out = boot\_out, type = "basic")

##

## Intervals :

## Level Basic

## 95% ( 0.5900, 1.0911 )

## Calculations and Intervals on Original Scale

But we’re still outside the range we want.

**Percentile Confidence Intervals**

Here’s an easy solution. Line up the bootstrap estimates from lowest to highest, then take the 2.5th and 97.5th percentile.

quantile(boot\_out$t, probs = c(.025, .975), type = 6)

## 2.5% 97.5%

## 0.460775 0.962000

Compare:

boot.ci(boot\_out, type = "perc")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

## Based on 500 bootstrap replicates

##

## CALL :

## boot.ci(boot.out = boot\_out, type = "perc")

##

## Intervals :

## Level Percentile

## 95% ( 0.4609, 0.9620 )

## Calculations and Intervals on Original Scale

(The slight difference is due to boot using its own quantile function.)

Looks like we have a winner. Our confidence interval will necessarily be limited to the range of plausible values. But let’s look at one other.

**Bias Corrected and Accelerated (BCa) Confidence Intervals**

BCa intervals require estimating two terms: a bias term and an acceleration term.

Bias is by now a familiar concept, though the calculation for the BCa interval is a little different. For BCa confidence intervals, estimate the bias correction term, \(\hat{z}\_0\), as follows:

\[  
\hat{z}\_0 = \Phi^{-1}\left(\frac{\#\{\hat{\theta}^\*\_b < \hat{\theta}\}}{B}\right)  
\]

where \(\#\) is the counting operator. The formula looks complicated but can be thought of as estimating something close to the median bias transformed into normal deviates (\(\Phi^{-1}\) is the inverse standard normal cdf).

The acceleration term is estimated as follows:

\[  
\hat{a} = \frac{\sum^n\_{i=1}(\hat{\theta}\_{(\cdot)} – \hat{\theta}\_{(i)})}{6\{\sum^n\_{i=1}(\hat{\theta}\_{(\cdot)} – \hat{\theta}\_{(i)})^2\}^{3/2}}  
\]

where \(\hat{\theta}\_{(\cdot)}\) is the mean of the bootstrap estimates and \(\hat{\theta}\_{(i)}\) the estimate after deleting the \(i\)th case. The process of estimating a statistic \(n\) times, each time dropping one of the \(i \in N\) observations, is known as the *jackknife* estimate.

The purpose of the acceleration term is to account for situations in which the standard error of an estimator changes depending on the true population value. This is exactly what happens with the correlation (the SE estimator we provided at the start of the post only works when \(r = 0\)). An equivalent way of thinking about this is that it accounts for skew in the sampling distribution, like what we have seen in the prior histograms.

Armed with our bias correction and acceleration term, we now estimate the quantiles we will use for establishing the confidence limits.

\[  
\alpha\_1 = \Phi\left(\hat{z}\_0 + \frac{\hat{z}\_0 + z^{(\alpha)}}{1-\hat{a}(\hat{z}\_0 + z^{(\alpha)}) } \right)  
\]

\[  
\alpha\_2 = \Phi\left(\hat{z}\_0 + \frac{\hat{z}\_0 + z^{(1 – \alpha)}}{1-\hat{a}(\hat{z}\_0 + z^{(1-\alpha)}) } \right)  
\]

where \(\alpha\) is our Type-I error rate, usually .05.

Our confidence limits are:

\[  
\\  
95\% \text{ CI} = [\hat{\theta}^{\*(\alpha\_1)}, \hat{\theta}^{\*(\alpha\_2)}]  
\]

Based on the formulas above, it should be obvious that \(a\_1\) and \(a\_2\) reduces to the percentile intervals when the bias and acceleration terms are zero. The effect of the bias and acceleration corrections is to change the percentiles we use to establish our limits.

If we perform all of the above calculations, we get the following:

boot.ci(boot\_out, type = "bca")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

## Based on 500 bootstrap replicates

##

## CALL :

## boot.ci(boot.out = boot\_out, type = "bca")

##

## Intervals :

## Level BCa

## 95% ( 0.3948, 0.9443 )

## Calculations and Intervals on Original Scale

## Some BCa intervals may be unstable

We get a warning message that BCa intervals may be unstable. This is because the accuracy of the bias and acceleration terms require a large number of bootstrap samples and, especially when using the jackknife to get the acceleration parameter, this can be computationally intensive. If so, there is another type of confidence interval known as the ABC interval that provides a satisfactory approximation to BCa intervals that is less computationally demanding. Type ?boot::abc.ci at the command line for how to implement this in R.

**Transformations**

What does it mean that calculations and intervals are on the original scale?

There are sometimes advantages to transforming a statistic so that it is on a different scale. An example is the correlation coefficient. We mentioned briefly above that the usual way of performing inference is to use the Fisher-\(z\) transformation.

\[  
z = \frac{1}{2}\text{ln}\left(\frac{1+r}{1-r} \right)  
\]

This transformation *is* normally distributed with standard error \(\frac{1}{\sqrt{N – 3}}\), so we can construct confidence intervals the usual way and then reverse-transform the limits using the function’s inverse. For Fisher-\(z\), the inverse of the transformation function is:

\[  
r = \frac{\text{exp}(2z) – 1}{\text{exp}(2z) + 1}  
\]

If we prefer to work with the transformed statistic, we can include the transformation function and its inverse in the boot.ci function. Define the transformations:

fisher\_z <- function(r) .5 \* log((1 + r) / (1 - r))

inv\_fisher\_z <- function(z) (exp(2 \* z) - 1) / (exp(2 \* z) + 1)

We can use these functions within a call to boot.ci. What we get in return will depend on which functions we specify.

* If only the transformation function is applied, the confidence intervals are on the transformed scale.
* If the transformation and the inverse transformation functions are applied, the confidence intervals are calculated on the transformed scale but returned on the original scale.

Recall that not specifying either function returns:

boot.ci(

boot\_out,

type = c("norm", "basic", "perc", "bca")

)

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

## Based on 500 bootstrap replicates

##

## CALL :

## boot.ci(boot.out = boot\_out, type = c("norm", "basic", "perc",

## "bca"))

##

## Intervals :

## Level Normal Basic

## 95% ( 0.5247, 1.0368 ) ( 0.5900, 1.0911 )

##

## Level Percentile BCa

## 95% ( 0.4609, 0.9620 ) ( 0.3948, 0.9443 )

## Calculations and Intervals on Original Scale

## Some BCa intervals may be unstable

Specifying the transformation only returns:

boot.ci(

boot\_out,

h = fisher\_z,

type = c("norm", "basic", "perc", "bca")

)

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

## Based on 500 bootstrap replicates

##

## CALL :

## boot.ci(boot.out = boot\_out, type = c("norm", "basic", "perc",

## "bca"), h = fisher\_z)

##

## Intervals :

## Level Normal Basic

## 95% ( 0.212, 1.688 ) ( 0.098, 1.572 )

##

## Level Percentile BCa

## 95% ( 0.498, 1.972 ) ( 0.417, 1.777 )

## Calculations and Intervals on Transformed Scale

## Some BCa intervals may be unstable

Specifying the transformation and its inverse returns the following:

boot.ci(

boot\_out,

h = fisher\_z,

hinv = inv\_fisher\_z,

type = c("norm", "basic", "perc", "bca")

)

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

## Based on 500 bootstrap replicates

##

## CALL :

## boot.ci(boot.out = boot\_out, type = c("norm", "basic", "perc",

## "bca"), h = fisher\_z, hinv = inv\_fisher\_z)

##

## Intervals :

## Level Normal Basic

## 95% ( 0.2091, 0.9340 ) ( 0.0981, 0.9173 )

##

## Level Percentile BCa

## 95% ( 0.4609, 0.9620 ) ( 0.3948, 0.9443 )

## Calculations on Transformed Scale; Intervals on Original Scale

## Some BCa intervals may be unstable

**Conclusion**

It is hoped that this post clarifies the output from boot::boot.ci, and in particular facilitates understanding the messages the function produces. We saw that the percentile and BCa methods were the only ones considered here that were guaranteed to return a confidence interval that respected the statistic’s sampling space. It turns out that there are theoretical grounds to prefer BCa in general. It is “second-order accurate”, meaning that it converges faster to the correct coverage. Unless you have a reason to do otherwise, make sure to perform a sufficient number of bootstrap replicates (a few thousand is usually not too computationally intensive) and go with reporting BCa intervals.