Bootstrapping Latent Variable Models

Appendix to *Latent Variable Modeling using R: A Step-by-Step Guide*

A. Alexander Beaujean

December 2013

Contents

1 Background

The major assumptions of a LVM are that the manifest variables are multivariate normal, there is a "sufficiently large" sample size, and that the observations are independent of each other. The first assumption is usually the one that causes the most trouble. There are two common ways for a variable to be non-normal: (a) be categorical; or (b) have excess univariate or multivariate skewness or kurtosis. In Chapter 6 of *Latent Variable Modeling using* **R***: A Step-By-Step Guide*, I discussed how to fit LVMs with categorical (specifically dichotomous) indicator variables. When the variables are continuous, but non-normal, it tends to make the standard errors (and, likewise, confidence intervals) too small and the χ^2 fit statistic too large. In this appendix, I just focus on a remedy for the former.

1.1 Confidence Interval Review

Confidence intervals (CI) concern a statistic (e.g., mean, variance), and range from 0% to 100%. The interpretation of a CI is: If we took *a lot* of samples from the same population, and construct X% CIs each time, approximately X% of them will contain the value of the parameter. From a different perspective, it is "... one interval generated by *a procedure* that will give correct intervals 95% of the time" (Antelman, 1997, p. 375, emphasis added).

CIs can be used in and of themselves (e.g., as a measure of the parameter estimates' precision), but they can also be used for other purposes, such as sample size determination and hypothesis testing. For hypothesis testing, the steps usually go something like: (a) determine the value of the parameter that maps onto H_0 (often this is 0.0, but it can be other values); (b) set the type 1 and type 2 error rates, α and β, respectively; (c) gather data; and (d) calculate the $(1 - [\alpha/2])100\%$ CI. If the the CI does not contain null value, reject H_0, otherwise fail to reject it.

When the assumptions for a statistic are not met, this usually causes the standard error to be underestimated. This, in turn, causes the CIs to be narrower than they should be, which can inflate the type 1 error rate. One way to handle the situation when a statistic's assumptions are not met by the data is to use bootstrap resampling to create the confidence intervals.

2 Bootstrapping Confidence Intervals

Bootstrapping

The idea behind **bootstrapping** is to mimic the sampling distribution of a statistic by *resampling with replacement* many, many times. Bootstrapped samples can be used for different purposes, but the most common use is to develop confidence intervals.

2.1 Example: Bootstrapping a Confidence Interval for the Mean

To make the bootstrapping concept and procedures more concrete, I will work through a simple example. Say I am interested in knowing the mean number of landline phone calls received at home after 5 p.m. in a certain geographic region. A sample distribution of this type of data is shown in Figure 1. There are many people who receive 0.0 or 1.0 landline phone calls and very few who receive more than 5.0. To show how far this distribution is from a normal distribution, I overlaid a normal distribution in Figure 1 that has the same mean and standard deviation as the phone call distribution.

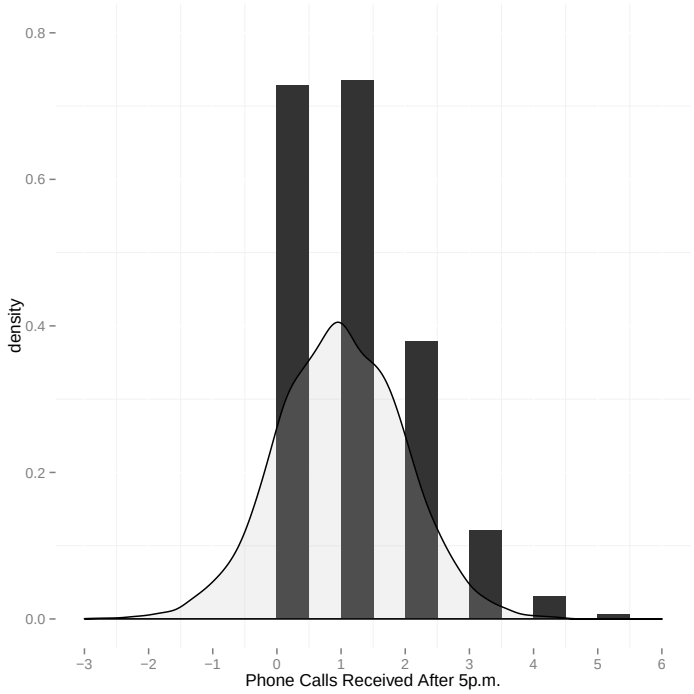

Figure 1 Histogram for bootstrapping example of the number of phone calls received after 5 p.m. The normal distribution overlay has the same mean and standard deviation as the phone call distribution.

Pretend I surveyed $n = 30$ random individuals in the region of interest and asked them the number of landline phone calls they received the previous evening. The results from my survey are shown in Table 1 in the *Original* column. The mean and standard error (SE) of the sample are 0.97 and 0.21, respectively. The 95% CI is then (0.61 - 1.32).

I calculated the 95% confidence interval via: $\bar{x} \pm 1.96 \times se$

Since the sample size is not large and does not follow a normal distribution, there are likely some suspicions about the accuracy of the SE and the CI, which were formed based on the normality assumption of the phone calls variable. Consequently, I can use bootstrap resampling as an alternative way to form confidence intervals.

For the bootstrap resampling, say I select a single value, randomly, from the original 30 observations *and then replace* the value back into the sample data. Then, I repeat this procedure 29 more times for a total of $n = 30$ values. These 30 "new" values constitute a *bootstrap sample*. The key is that after each selection of a value from the original dataset, I replaced the value so that I could possibly select it again. For most bootstrapping applications, this whole process is repeated many times, typically $1000 \leq B \leq 2000$. Columns 2-4 in Table 1 present some bootstrap samples of the phone call data.

Since I am interested in the CI for the mean in my example, I can calculate the mean for each of the B bootstrap samples. An example distribution of bootstrapped means for $B = 1000$ is shown in Figure 2. While the original values (see Figure 1) did not look like they came from a normal distribution, the distribution of the B bootstrapped means in Figure 2 look much

Table 1 Frequency Count of Original Phone Call Data with Bootstrapped Samples.

Phone Calls	Original	X_1^*	X_2^*	...	X_B^*
0	14	14	15	...	12
1	8	7	7	...	5
2	3	3	3	...	4
3	5	6	5	...	9
$\bar{x}$	0.97	1.03	0.93	...	1.33

$\bar{x}$: the mean value.

closer to a normally distributed variable. Usually, the distribution of a statistic's bootstrapped samples will mimic the statistic's actual sampling distribution if B is large enough.

The B means from the B bootstrap samples can now act as an *empirical* sampling distribution for the mean. Thus, if I rank order the B means, I can find the value that is at the 2.5%-ile and the value that is at the 97.5%-ile and use them to create a new 95% CI. I did this in Figure 2 via the two vertical lines at both ends of the distribution. This method is known as the *percentile method* for obtaining a CI using bootstrap resampling.

The relationship between the sample statistics and the statistics from the bootstrap samples approximate the relationship between the sample statistics and population parameters. Thus, the difference between the mean from the original sample and the mean of the bootstrapped means $\bar{x}_B^*$ (or the standard error in the original sample and standard deviation of the bootstrapped means, $s_{\bar{x}^*}$) are indications of bootstrapping version of bias.

Dividing the bias estimate by the value from the original sample would give a relative measure of bias.

For the $B = 1000$ bootstrapped samples used for Figure 2, $\bar{x}_B^* = 0.96$ and $s_{\bar{x}^*} = 0.19$. An estimate of the parameter bias is then:

$$\bar{x}_B^* - \bar{x} = 0.963 - 0.967 = -0.004\,.$$

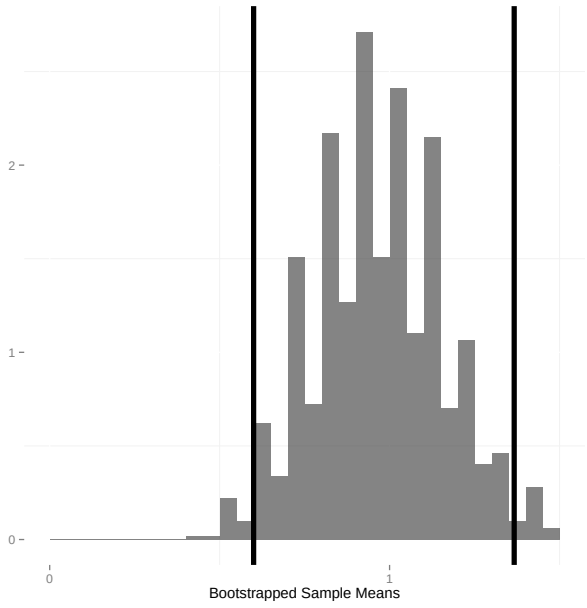

Figure 2 Distribution of $B = 1000$ bootstrapped means. The two vertical line represent the 95% confidence interval.

Likewise, an estimate of the standard error bias is

$$s_{\bar{x}^*} - s_{\bar{x}} = 0.19 - 0.21 = -0.01\,.$$

For the *mean*, parameter and standard error bias tends to be small, but this is not necessarily the case with other statistics. Consequently, an alternative to the percentile method for calculating bootstrapped confidence intervals is one that attempts to correct for bias: the bias-corrected and accelerated (*BCa*) bootstrap. In a rather complex way, it adjusts for both bias and skewness in the bootstrap samples' distribution.

The `boot` package in **R** provides the `boot()` function, which can bootstrap just about any statistic.Once the bootstrap samples are created, the `boot.ci()` function will estimate the desired confidence intervals. The price for such a general function is that its use is not very intuitive. To use it requires writing a function that will calculate the statistic of interest and also allows the statistic to be calculated over multiple datasets (i.e, the B bootstrap samples). Below I provide syntax for creating bootstrap samples of the mean for the phone call data, as well as calculating the CI from the bootstrapped samples.

`boot()`

`boot.ci()`

```
1  # bootstrapping the phone call data
2  mean.boot <- function(data,r){return(mean(data[r]))}
3  phone.boot <- boot(data=phone.data, statistic=mean.boot, R=1000)
4  boot.ci(phone.boot, conf = 0.95, type = c("perc", "bca"))
```

Line 2, contains a function to calculate the mean. As there are multiple bootstrapped datasets, I had to allow the *data* variable to be indexed (i.e., `[r]`). In line 3, I use the `boot()` function with three arguments: (a) the dataset (`data`); (b) the statistic function (`statistic`); and (c) the number of bootstrap samples (`R`). In line 4, I use the `boot.ci()` function to estimate the 95% CI using the percentile and BCa methods.

The `boot()` function has a `parallel` argument if your computer has those capabilities.

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = phone.boot, conf = 0.95, type = c("perc",
##     "bca"))
##
## Intervals :
## Level     Percentile            BCa
## 95%   ( 0.6,  1.4 )   ( 0.6,  1.4 )
## Calculations and Intervals on Original Scale
```

As there was little bias in the parameter estimates, the percentile and Bca CIs are identical out to the one digit after the decimal.

3 Example: Latent Variable Model with Non-Normal Data

For this example, I simulated ($n = 300$) observations for six variables with non-negligible skewness and kurtosis.[1] Histograms of all the variables are shown in Figure 3.

[1] **R** syntax for how I simulated the data is available on the book's website.

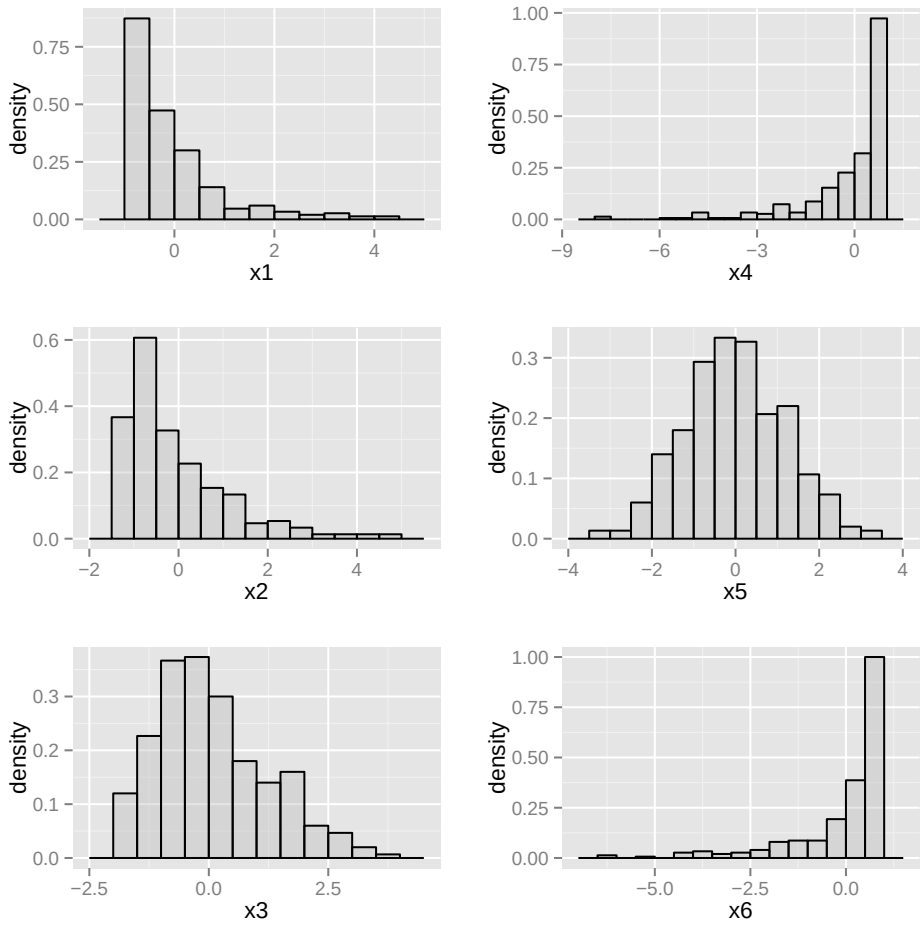

Figure 3 Histograms of the six simulated variables.

I hypothesized there were two LVs in data, with $X_1 - X_3$ loading on the first and $X_4 - X_6$ loading on the second. I then fit the model to the data in `lavaan`. To create the bootstrapped samples and calculate the CIs from them, I have to add two additional arguments when fitting a model. The first is `se="boot"`, which tells `lavaan` to use bootstrapping for the standard errors. The second argument is `bootstrap=1000`, which indicates I want 1000 bootstrap resamples (although the number does not have to be 1000).

```
# specification for the model with two latent variables
sample.model <- '
f1 =~ x1 + x2 + x3
f2 =~ x4 + x5 + x6
'
sample.fit <- cfa(model=sample.model, data=sample.data, se="boot", bootstrap=1000)
summary(sample.fit)
```

```
## lavaan (0.5-15) converged normally after  45 iterations
##
##   Number of observations                           300
##
##   Estimator                                         ML
##   Minimum Function Test Statistic               11.590
##   Degrees of freedom                                 8
##   P-value (Chi-square)                           0.170
```

```
##
## Parameter estimates:
##
##   Information                                 Observed
##   Standard Errors                            Bootstrap
##   Number of requested bootstrap draws             1000
##   Number of successful bootstrap draws             946
```

The summary results look very similar to the regular output, only now the *Standard Errors* line says that they were estimated using bootstrapping. Moreover, the two lines below the *Standard Errors* line indicate the number of bootstrap samples were requested and how many of those requested were successfully drawn.

To obtain the bootstrapped confidence intervals, I have to use the `parameterEstimates()` function with the `boot.ci.type` argument. If `boot.ci.type="perc"`, it will return the percentile confidence intervals and if `boot.ci.type="bca.simple"`, it will return a simplified BCa confidence interval that does not correct for skew.

`parameterEstimates()`

```
parameterEstimates(sample.fit, level = 0.95, boot.ci.type="perc")
```

```
##    lhs op rhs   est    se      z pvalue ci.lower ci.upper
## 1   f1 =~  x1 1.000 0.000     NA     NA    1.000    1.000
## 2   f1 =~  x2 1.421 0.459  3.097  0.002    0.841    2.643
## 3   f1 =~  x3 1.613 1.092  1.477  0.140    0.856    4.932
## 4   f2 =~  x4 1.000 0.000     NA     NA    1.000    1.000
## 5   f2 =~  x5 0.373 0.090  4.138  0.000    0.190    0.550
## 6   f2 =~  x6 0.719 0.193  3.720  0.000    0.390    1.131
## 7   x1 ~~  x1 0.943 0.257  3.678  0.000    0.521    1.517
## 8   x2 ~~  x2 0.992 0.227  4.377  0.000    0.519    1.439
## 9   x3 ~~  x3 0.943 0.246  3.832  0.000    0.313    1.264
## 10  x4 ~~  x4 1.028 0.422  2.438  0.015   -0.030    1.621
## 11  x5 ~~  x5 1.148 0.114 10.044  0.000    0.930    1.364
## 12  x6 ~~  x6 0.763 0.200  3.817  0.000    0.310    1.114
## 13  f1 ~~  f1 0.207 0.120  1.724  0.085    0.038    0.496
## 14  f2 ~~  f2 1.643 0.550  2.986  0.003    0.793    2.841
## 15  f1 ~~  f2 0.146 0.061  2.395  0.017    0.015    0.262
```

```
parameterEstimates(sample.fit, level = 0.95, boot.ci.type="bca.simple")
```

```
##    lhs op rhs   est    se      z pvalue ci.lower ci.upper
## 1   f1 =~  x1 1.000 0.000     NA     NA    1.000    1.000
## 2   f1 =~  x2 1.421 0.459  3.097  0.002    0.826    2.615
## 3   f1 =~  x3 1.613 1.092  1.477  0.140    0.827    4.409
## 4   f2 =~  x4 1.000 0.000     NA     NA    1.000    1.000
## 5   f2 =~  x5 0.373 0.090  4.138  0.000    0.195    0.559
## 6   f2 =~  x6 0.719 0.193  3.720  0.000    0.394    1.138
## 7   x1 ~~  x1 0.943 0.257  3.678  0.000    0.552    1.615
## 8   x2 ~~  x2 0.992 0.227  4.377  0.000    0.544    1.447
## 9   x3 ~~  x3 0.943 0.246  3.832  0.000    0.423    1.305
## 10  x4 ~~  x4 1.028 0.422  2.438  0.015    0.228    1.709
## 11  x5 ~~  x5 1.148 0.114 10.044  0.000    0.933    1.377
## 12  x6 ~~  x6 0.763 0.200  3.817  0.000    0.351    1.124
## 13  f1 ~~  f1 0.207 0.120  1.724  0.085    0.046    0.523
## 14  f2 ~~  f2 1.643 0.550  2.986  0.003    0.815    2.929
## 15  f1 ~~  f2 0.146 0.061  2.395  0.017    0.052    0.317
```

4 Writing the Results

When writing the results after using bootstrap resampling for the confidence intervals, follow the guidelines already specified for that particular model. In addition, give the reason for using the bootstrapping (i.e., what statistical assumptions were not met with the data), the type of bootstrapped confidence interval used, and the number of bootstrap samples used.

5 Exercises

1 Use the abortion question data in the `ltm` package.

- 1.a Conduct an item factor analysis with one latent variable using the `DWLS` estimator and obtain 1000 bootstrap samples.
- 1.b Calculate the 95% confidence intervals for all parameters using the percentile method.
- 1.c Calculate the 95% confidence intervals for all parameters using the simplified BCa method.

6 References & Further Readings

Antelman, G. (1997). *Elementary Bayesian statistics.* Cheltenham, UK: Edward Elgar.

Beasley, W. H., & Rodgers, J. L. (2012). Bootstrapping and Monte Carlo methods. In H. Cooper, P. M. Camic, D. L. Long, A. T. Panter, D. Rindskopf, & K. J. Sher (Eds.), *APA handbook of research methods in psychology, Vol 2: Research designs: Quantitative, qualitative, neuropsychological, and biological* (pp. 407–425). Washington, DC: American Psychological Association.

Carpenter, J., & Bithell, J. (2000). Bootstrap confidence intervals: When, which, what? A practical guide for medical statisticians. *Statistics in Medicine*, *19*, 1141-1164. doi: 10.1002/(SICI)1097-0258(20000515)19:9<1141::AID-SIM479>3.0.CO;2-F

Efron, B., & Tibshirani, R. J. (1994). *An introduction to the bootstrap.* London: Chapman Hall.

Holzinger, K., & Swineford, F. (1937). The bi-factor method. *Psychometrika*, *2*, 41-54. doi: 10.1007/BF02287965

West, S. G., Finch, J. F., & Curran, P. J. (1995). Structural equation models with non-normal variables: Problems and remedies. In R. A. Hoyle (Ed.), *Structural equation modeling: Concepts, issues and applications* (p. 56-75). Newbury Park, CA: Sage.

7 Exercise Answers

Exercise 1.a

```
library(ltm)
data(Abortion)
# rename items
names(Abortion) <- paste("I",1:4,sep="")

library(lavaan)
abortion.model <- '
LV =~ I1 + I2 + I3 + I4
'

abortionBoot.fit <- cfa(abortion.model, data=Abortion, std.lv=TRUE,
ordered=paste("I",seq(1:4), sep=""), se="boot", bootstrap=1000, estimator="DWLS")
```

Exercise 1.b

```
parameterEstimates(abortionBoot.fit, level = 0.95, boot.ci.type="perc")
```

Exercise 1.c

```
parameterEstimates(abortionBoot.fit, level = 0.95, boot.ci.type="bca.simple")
```