Appendix I: Using A Monte Carlo Study to Calculate Power and Needed Sample Size

Online Supplemental Appendix for *Latent Variable Models: An Introduction to Factor, Path, and Structural Equation Analysis* (5th Edition)

John C. Loehlin & A. Alexander Beaujean

Contents

In Chapter 2, we discussed assessing power to detect an added path, whether the exact path is known in advance or not. For both situations, we conducted the power analyses using analytically-derived values of the χ^2 statistic which we found using existing tables of values. In some situations, theoretical values for the necessary statistic density functions are not tractable or the they rely on certain assumptions that are not met for a specific model or data (Beasley & Rodgers, 2012).

An alternative to using analytically-derived statistics for power analysis is to use a Monte Carlo (MC) study. MC studies use large samples of randomly-generated data generated from a computer to answer questions of interest (Beaujean, 2017). If simulated correctly, the empirical data generated from MC studies can be used in lieu of analytically-derived statistical values.

MC studies can aid in answering a variety of questions, but we focus only on statistical power and sample size determination in this Appendix. Specifically, we provide a general overview of MC methods as they relate to statistical power and sample size determination for factor, path, and structural equation models. Subsequently, we re-work the power and sample size determination examples from Chapter 2 using the MC methods.

Conducting a Monte Carlo Study

In what follows, we outline the steps for conducting a MC study focusing on statistical power and sample size determination.

1. Determine if a MC study is the best way to examine power or determine the need sample size for a particular study. If the models under investigation are relatively simple and the data are not expected to have any substantial quirks (see Step 5), then the methods we described in Chapter 2 will probably be sufficient.
2. Create two sets of models: (a) population models that will be used to generate the simulated data, and (b) analysis models that will be used to analyze the simulated data. The population and analysis models can be the same or they can be different, such as when examining the effects of a misspecified model.
3. Determine the properties for all the variables and parameters in the population models. This includes specifying the variables' distributions as well as values of all the paths, variances, covariances, and

means/intercepts (if applicable). These values can be specified using unstandardized or standardized values, but it will typically be easier to use standardized values.

4. Examine the values of the model-implied covariance matrices from specified parameter values in the population models from Step 3. This is done as a check to make sure the parameter values are correct before starting the simulation process. The implied values can be calculated by hand using Wright's tracing rules, but most model-fitting programs will do this for you.
5. (Optional) Determine if the simulated data will have any quirks, such as missing values or assumption violations (e.g., non-normal data). Including quirks in the data should only be done if there is an expectation that the resulting data will approximate data from the actual study better than if the quirks are not included.
6. Determine the technical aspects of the simulations. This involves specifying details about the simulation and decision-making processes. Some typical decisions that need to be made concern: (a) desired type 1 error rate (α), which will also determine the confidence intervals' levels; (b) desired power; (c) sample size (n)—or range of sample sizes—of interest; and (d) number of sample datasets to simulate (m).

   There is some debate about what value should be used for m (Skrondal, 2000). Conventional wisdom is to set m to a very large number (e.g., $m \geq 10,000$). If there are just a few population and analysis models, and both sets of models are relatively simple, then the time it takes to simulate a large number of samples with many modern computers is typically minimal (i.e., $<$ 1 hour). If there are many population or analysis models, or the models are complex, then making m large cost a lot of time or computing resources. Thus, preliminary phases of the MC study may need smaller m values in order to gather information across more models; then, use a larger value for m at the final phase.
7. Simulate pilot data using relatively small values for n and m. Then, examine the simulated data to check for data adequacy, such as absence of impossible values and reasonable values of the sample and performance measure statistics (see Step 9). If the data are not being simulated as expected, this may indicate that the simulation program needs to be debugged, the population models' parameter values need to be revisited, or the population or analysis models need to be changed.
8. Simulate the m samples from the population models. Be sure to specify a value for the random seed (i.e., a value used to initiate the data generation process). Most—if not all—modern data simulation programs allow users to input random seed values.
9. Analyze each of the m datasets using the appropriate analysis models. Be sure to save all m sets of parameter estimates and fit statistics for the analyses in Step 10 .
10. Calculate summary and performance statistics. The summary statistics summarize the results from the analysis of the m datasets, while the performance statistics examine the adequacy of the simulation process.

Table I-1 Common Summary and Simulation Performance Statistics for Monte Carlo Studies.

Summary	
Average value of the statistic, $\bar{\hat{\theta}}$	$\bar{\hat{\theta}} = \frac{\sum_{i=1}^{m} \hat{\theta}_i}{m}$
Power	$\frac{\text{\# of datasets where null hypothesis is rejected}}{m}$
Simulation Performance	
Relative parameter bias	$\frac{\bar{\hat{\theta}} - \theta}{\theta}$
Relative standard error bias	$\frac{\hat{\sigma}_\theta - \sigma_\theta}{\sigma_\theta}$
Coverage	$\frac{\text{\# of datasets where the confidence interval contains } \theta}{m}$

Note. θ : parameter value; $\hat{\theta}$: statistic value from single sample; $\bar{\hat{\theta}}$: average value of the statistic; n: sample size; m = number of simulated datasets of size n; σ_θ: SD of the parameter estimate over the m replications; $\hat{\sigma}_\theta$: average of the estimated standard errors for the parameter estimate over the m replications.

Table I-2 Criteria for Performance Statistics from Monte Carlo Simulations.

Measure	Suggested criteria
Relative parameter estimate bias	Absolute value $\leq$ .10 for all parameters
Relative standard error bias	Absolute value $\leq$ .10 for all parameters & absolute value $\leq$ .05 for parameters of interest
Coverage	.91 $-$ .98 for all parameters
Power	Close to desired power level without going below

Note. Criteria taken from Muthén and Muthén (2002, pp. 605–606).

Table I-1 has some common summary and performance measures for MC power/sample size studies, while Table I-2 has some criteria to judge the performance measures.

11. If the performance measures' values (calculated in Step 10) are acceptable, examine the power for the parameters or fit measures of interest. If the performance measures values are not acceptable, increase the *m* and repeat Steps 8–10.
12. Repeat Steps 8–11 using a different random seed. If the results from the different random seeds converge, no need for further simulations of the current scenario. If they do not converge, this may indicate that there is a problem with the data simulation process. Re-check the models and repeat Steps 8–11 using different random seeds and larger values of *m*.

Power and Sample Size Determination Examples

The examples in this Appendix come from the *Power to detect an added path* section of Chapter 2. Those examples all used the model shown in Fig. 2.10, which we repeat for convenience as Fig. I.1. We do not discuss the method for assessing power in the *Overall power to reject a model* section, which used the RMSEA to estimate the power of a test of poor fit. That is because this approach does not require any specific parameter values for any model, so there is no need for a MC study. The only feature of the model under study that is relevant in that approach is the *df* (MacCallum, Browne, & Sugawara, 1996), so all models with the same *df* and *n* have the same level of power.

We implement the simulations for the MC studies using the *lavaan* (Rosseel, 2012) and *simsem* (Pornprasertmanit, Miller, & Schoemann, 2012) packages in R. We do not discuss the details of specifying and fitting a confirmatory factor model in lavaan since we do so in the extended examples of Chapters 2 and 3. Readers should review those sections if they want to refresh themselves on this information.

After installing the simsem package, it can be loaded into R using the `library()` function. Loading the simsem package will also load the lavaan package, so there is no need to load it separately.

```
library(simsem)
```

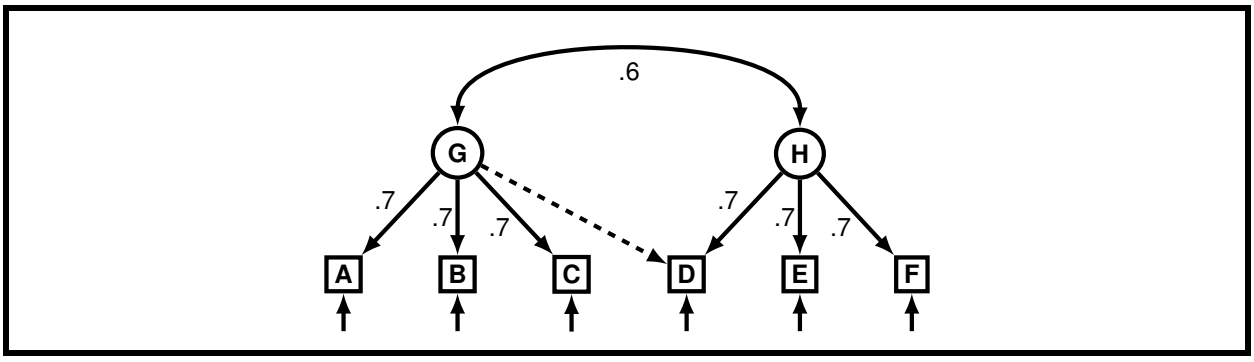

Figure I.1 Simple factor analysis model for power example. Dashed arrow (- ->) additional path to be detected.

Power to detect a known added path

For this example, we want to assess the power to detect an added path to a model when we know in advance what path to add. Specifically, we want to know the power for detecting the dashed path in Fig. I.1 when its population value is .30, $\alpha = .05$, and $n = 500$.

In Chapter 2, we calculated power for this example by evaluating a model without the G- ->D path using the correlations implied from a model with the G- ->D path. We then examined the resulting χ^2 value using values from a noncentral χ^2 distribution with a single *df*.

An alternative to the single *df* test of model fit for a specific path is to use the Wald test (see *Hierarchical* χ^2 *Tests* section in Chapter 2). From this perspective, the significance test is for a single parameter instead of an entire model. Thus, for the power analysis we are concerned about wanting to know the probability of rejecting the null hypothesis that the G- ->D path's value in Fig. I.1 is zero when it is really .30.

The general approach for MC studies using the Wald test is to simulate data from a population model that includes the parameter of interest and then analyze the data using a correctly-specified analysis model. Readers may wonder: if the population and analysis models' structures are the same, then won't the analysis model always provide a good fit of the data and result in always rejecting the null hypothesis that the parameter's value is zero?

The answer is "no." Since the data are generated randomly from the population model, there will be samples where the null hypothesis is not rejected using the specified α and *n* values—even though we know it should be. In fact, this is the very definition of power for a Wald test using a MC study: the proportion of the *m* samples where the false null hypothesis is rejected (see Table I-1). In our example, this translates to knowing how often we will fail to reject the null hypothesis that the G- ->D path is zero when the true effect size is .30, $n = 500$, and $\alpha =: 05$.

Specifying the population model using lavaan syntax requires setting all the parameter values. This is done through the labeling operator $(*)$. Remember, the population model requires specifying all parameter values. So, we not only need to provide values for factor loadings and correlations, but also the latent variables' variances and the residual error variances. Fig. I.2 has a version of Fig. I.1 that shows all parameter values when G- ->D path set to .30 (The values in Fig. I.2 were set to standardize the manifest and latent variables.)

```
# population model with G-->D loading set to .3
pop.model.3 <- '
# loadings
G =~ .7*A + .7*B + .7*C + .3*D
H =~ .7*D + .7*E + .7*F
# factor correlation
G~~.6*H
# factor variances
G~~1*G
H~~1*H
'
```

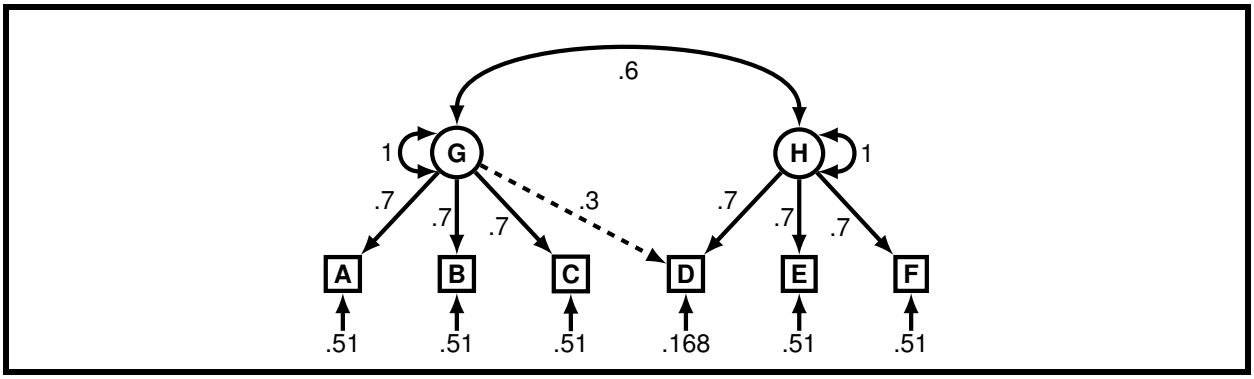

Figure I.2 Fig. I.1 with all parameter values specified when G- ->D path set to .3. Error variances set to make all manifest variables' variances equal to one.

```
# residual variances
A~~.51*A
B~~.51*B
C~~.51*C
D~~.168*D
E~~.51*E
F~~.51*F
'
```

The analysis model is the same as the population model, but excludes the parameter values. We also excluded syntax related to variances since those are estimated by default in lavaan.

```
# correclty-specified analysis model
analysis.model <- '
G =~ A + B + C + D
H =~ D + E + F
G~~H
'
```

To make sure the parameters in the population model were specified correctly, we fit the population model and examined the model-implied covariances. First, we fit the model using the `cfa()` function, but used the `do.fit=FALSE` argument. This argument makes the parameter estimate values equal to the fixed values (or default starting values if a parameter is not fixed).

```
pop.fit.3 <- cfa(pop.model.3, do.fit=FALSE)
```

Second, we use the `fitted()` function; it returns the model-implied covariances and means (if applicable). Since we specified a model with standardized variables, the covariances will be correlations.

```
fitted(pop.fit.3)
```

The correlation values match those below the diagonal of Table 2-14. Thus, it appears that we specified the population model correctly.

To simulate data from the population model, we use the `sim()` function. It will complete Steps 8–11. By default, variables are simulated to follow a normal distribution.

The main arguments for the `sim()` function are:

- *nRep*: the number of datasets to simulate (*m*)
- *generate*: the population model
- *model*: the analysis model
- *n*: sample size (*n*)
- *lavaanfun*: the lavaan function to use for the analysis given as a string (e.g., “cfa”, “sem”, “growth”)
- *seed*: the value of the random seed

Two other arguments for the `sim()` function can also be useful. First, setting `std.lv=TRUE` will fit a standardized latent variable model to the data. Otherwise, the latent variables will be identified using the first indictor variable. Second, using the `multicore=TRUE` argument distributes the simulation and analysis across multiple processors on a computer. If a computer has multiple cores available, this can greatly speed up the time it takes to complete the MC power study. The default option is to use the maximum number of processors in the computer; to specify a different number, use the `numProc` argument.

We simulated $m = 10,000$ samples of size $n = 500$ using the previously specified population and analysis models. We identified the model by standardizing the latent variables, used the multicore functionality, and set the random seed to 45686. There is nothing special about that random seed value; readers can use whatever value they wish. Using a different value will produce slightly different results than the ones we report, but they should be relatively close.

```
specific.power.30 <- sim(nRep=10000, generate=pop.model.3, model=analysis.model, n =500, lavaanfun = "cfa",
        seed=45686, std.lv=TRUE,  multicore=TRUE)
```

The simulations and analyses took approximately 256 seconds for the simulations and analyses using an iMac with a 3.3 GHz Intel Core i7 processor, 4 cores, and 16 GB of memory.

To extract the summary and performance statistics, use the `summaryParam()` function with the `detail=TRUE` argument. By default, $\alpha = .05$, but this can be changed using the `alpha` argument.

```
summaryParam(specific.power.30,detail=TRUE)
```

For this simulation, the relative parameter estimate bias and relative standard error bias values were both $< .05$ for all parameters estimated, and the coverage values were all between .94–.96. This indicates the simulation process worked well.

The power to detect the dashed path was .9856. This means that, using $\alpha = .05$, the false null hypotheses that the G- ->D path's value is zero was rejected in 9856 of the 10,000 simulated data sets. This matches what we noted in Chapter 2, i.e., we have somewhere between a 90%–99% probability of rejecting the hypothesis that the value of the G- ->D path is zero when it is really .3 in the population.

When we repeated the simulations using a different random seed (52578), the results were quite similar. The performance statistics were all within the desired ranges and the power to detect the G- ->D path was .9862.

Power to detect an unknown added path

This example is similar to the previous one except we do not know in advance exactly which path will be added to the model. This translates to calculating the power to reject a misspecified (false) model. The population model is same as before (i.e., Fig. I.2), as are the $n = 500$ and $\alpha = .05$. What differs is the analysis model. Specifically, the analysis model is a version of Fig. I.1 without the G- ->D path. The misspecified model is shown in Fig. I.3.

```
# misspecified analysis model
analysis.model.mis <- '
G =~ A + B + C
H =~ D + E + F
G~~H
'
```

The syntax for the MC simulations is the same as from the previous example except the misspecified model is used for the analysis model.

```
model.power.30.mis <- sim(nRep=10000, generate=pop.model.3, model=analysis.model.mis, n =500,
        lavaanfun = "cfa", seed=45686, std.lv=TRUE, multicore=TRUE)
```

Because the null hypothesis in this example relates to a model, we need to use a measure of model fit as the criterion for rejecting the null hypothesis. Following Chapter 2, we use the χ^2 statistic. To use this statistic, we have to specify a threshold at which we will reject the model.

Using a MC study, the χ^2 threshold can be determined two ways. The first involves using an analytically-determined value, which is the same method we used in Chapter 2. Thus we reject the model for any data set that produces a $\chi^2 > 14.93$.

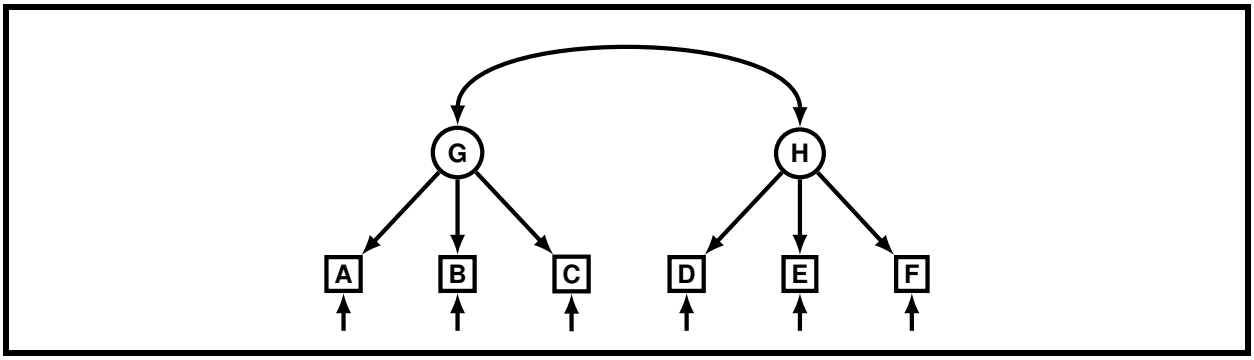

Figure I.3 A misspecified version of the model in Fig. I.3 with no G- ->D path.

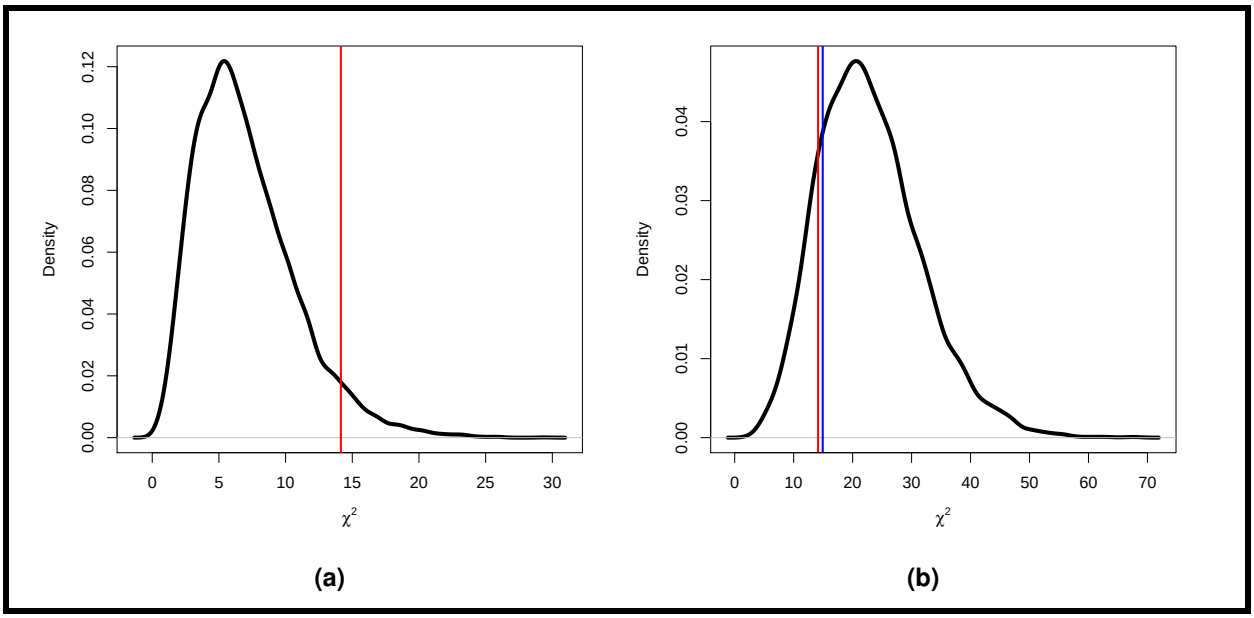

Figure I.4 Density plots of empirical χ^2 values from fitting data generated from Fig. I.2 that are correctly-specified (a), and misspecified (b). Vertical red line is the empirical rejection threshold ($\chi^2 = 14.14$) and vertical blue line is the analytical rejection threshold ($\chi^2 = 14.93$).

The second way involves using an empirically-determined value calculated from simulated data. This requires two steps. First, simulating data from the population model and analyzing it using a correctly-specified analysis model. This is the same procedure we used in the first example, only now we are interested in the distribution of the χ^2 statistics instead of the how frequently we rejected a false null hypothesis for a specific parameter. A graph of this empirical χ^2 distribution is shown in Fig. I.4(a).

The second step in this process involves finding a χ^2 from the empirically-derived χ^2 distribution to represent extreme cases. This value then can be used as a rejection threshold when fitting the misspecified model. Since $\alpha = .05$, a logical value to represent extreme χ^2 values would be the value at the 95th percentile. Using that threshold value with the correctly specified model would result in falsely rejecting 5% of the true models, which is the definition of a type 1 error.

To find the χ^2 value at the 95th percentile, we use the results from the first example, which we stored in the R object named *specific.power.30*. To find the specific χ^2 value, use the `getCutoff()` function with the `usedFit="chisq"` argument and the `alpha = 0.05` argument. Because large χ^2 values indicate poor fit, the `getCutoff()` function is programmed to select the percentile from the right tail of the distribution (e.g., the 95th percentile for the `alpha = 0.05` argument). Because the value will be used later, we stored the value in the R object named *chi.cutoff*.

```
chi.cutoff <- getCutoff(specific.power.30, alpha = 0.05, usedFit="chisq")
```

The threshold value was $\chi^2 = 14.14$, which is shown as a vertical red line in Fig. I.4(a). Approximately 5% of the χ^2 values calculated from fitting the correctly-specified model are to the right of the red line in this figure.

The `getPowerFit()` function finds the power of rejecting the analysis model using a specified criterion. The threshold for model rejection is specified using the *cutoff* argument. For the analytically-derived value, use `cutoff = c(chisq=14.93)`; for the empirically derived value, use `cutoff = cutoff.chi`

```
# power of rejecting the misspecified (false) analysis model using an analytically-derived chi-square value
getPowerFit(model.power.30.mis, cutoff = c(chisq=14.93))
# power of rejecting the misspecified (false) analysis model using an empirically-derived chi-square value
getPowerFit(model.power.30.mis, cutoff=chi.cutoff)
```

The power to reject the false analysis model using the analytical χ^2 threshold is .82, and the power using the empirical threshold is slightly more at .84. This is shown in Fig. I.4(b), where the area to the right of the vertical lines is power.

Instead of the χ^2 measure of model fit, one could use other fit measures. The `sim()` function calculates the AIC, BIC, CFI, RMSEA, SRMR and TLI for every model fit. (For descriptions of these fit statistics, see Appendix D.) To find power based on those statistics, apply the methods discussed for the empirically-derived χ^2 except do not supply the `usedFit` argument in the `getCutoff()` function. The resulting output will provide power estimates using each of the aforementioned model fit measures.

```
mult.cutoff <- getCutoff(specific.power.30, alpha = 0.05)
getPowerFit(model.power.30.mis, cutoff = mult.cutoff)
```

Determining needed sample size

We saw in Chapter 2 that if the population value of the G- ->D path in Fig. I.1 was .10 instead of .30, the power was substantially reduced. This made using $n = 500$ result in an underpowered study. In such situations, we want to find the minimum *n* that will provide adequate power. This requires simulating data using multiple values of *n* and then homing in on the value that meets the minimum power threshold.

Before discussing how to simulate data using multiple values of *n*, we need to re-specify the population model using a value of .10 for the dashed path. This means we also need to change the residual variance of the D variable. This model is shown in Fig. I.5.

```
# population model with G-->D loading set to .1
pop.model.1 <- '
G =~ .7*A + .7*B + .7*C + .1*D
H =~ .7*D + .7*E + .7*F
G~~.6*H

A~~.51*A
B~~.51*B
C~~.51*C
D~~.416*D
E~~.51*E
F~~.51*F
G~~1*G
H~~1*H
'
```

To simulate data under a variety of sample sizes requires using the sim`()` function slightly differently than in the previous examples. Specifically, set the `nRep` argument to `NULL` and provide a range of values for the `n` argument. The range of *n*s can be specified using the sequence function, `seq()`. For example, to examine power for values from $n = 500$ to $n = 4000$ increasing by increments of 50, use `seq(500,4000,50)` for the `n` argument. This produces one simulation with $n = 500$, one with $n = 550$, and so on (i.e., $m = 1$ for each *n*).

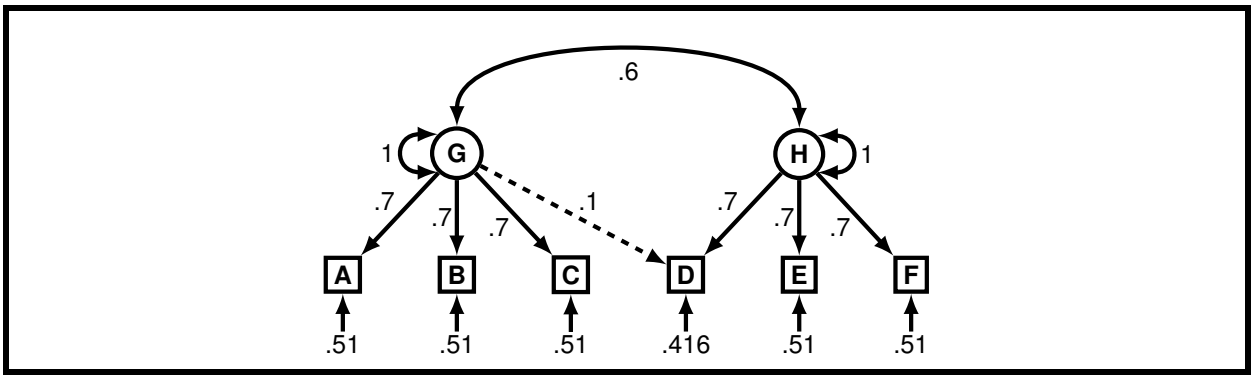

Figure I.5 Fig. I.1 with all parameter values specified when G- ->D path set to .1. Error variances set to make all manifest variables' variances equal to one.

To increase the value of m at each sample size, wrap the `seq()` function inside the replicate function, `rep()`. For example, to repeat the 500–4000 sequence 50 times (i.e., $m = 50$ for each n), use: `rep(seq(500,4000,50), 50)`. We did not set $m = 10,000$ because it would make the number of simulations quite large. (There are 71 values between 500 and 4000 increasing in increments of 50; simulating $m = 10,000$ dataset for each n would require simulating and analyzing $71 \times 10,000 = 710,000$ datasets!) After finding an approximate sample size to use, the simulation can be re-done using a single value for n and a larger value for m (and then re-done using a different random seed).

```
# known path with multiple sample sizes
specific.power.10.ns <- sim(nRep=NULL, n = rep(seq(500,4000,50),50), generate=pop.model.1,
        model=analysis.model, seed=45686, lavaanfun = "cfa", multicore=TRUE, std.lv=TRUE)

# unknown path with multiple sample sizes
model.power.10.ns <- sim(nRep=NULL, n = rep(seq(500,4000,50),50), generate=pop.model.1,
        model=analysis.model.mis, seed=45686, lavaanfun = "cfa", multicore=TRUE, std.lv=TRUE)
```

After completing the simulations, the resulting power estimations can be examined graphically and numerically. The graphical method involves plotting a power curve (i.e., a graph of the power estimate as a function of sample size). The `plotPower()` function creates power curves for Wald test situations (i.e., hypotheses concerning single parameters). To plot curves for only the parameters of interest, use the `powerParam` argument and specify the parameters in lavaan syntax. To make interpretation of power curves easier, we suggest plotting a horizontal line at the value of desired power, which for this example is .80. The resulting power curve—along with a horizontal line at.80—is shown in Fig. I.6(a).

```
# power curve for G-->D path
plotPower(specific.power.10.ns, powerParam="G=~D")
# dashed horizontal line at .80
abline(h=.8, lwd=2, lty=2)
```

A similar plot can be calculated for rejecting the entire model in the unknown path situation using the `plotPowerFit()` function. Instead of specifying the parameters of interest, specify the fit measures of interest and their cutoff values using the `cutoff` argument. The power curve for the unknown path situation is shown in Fig. I.6(b).

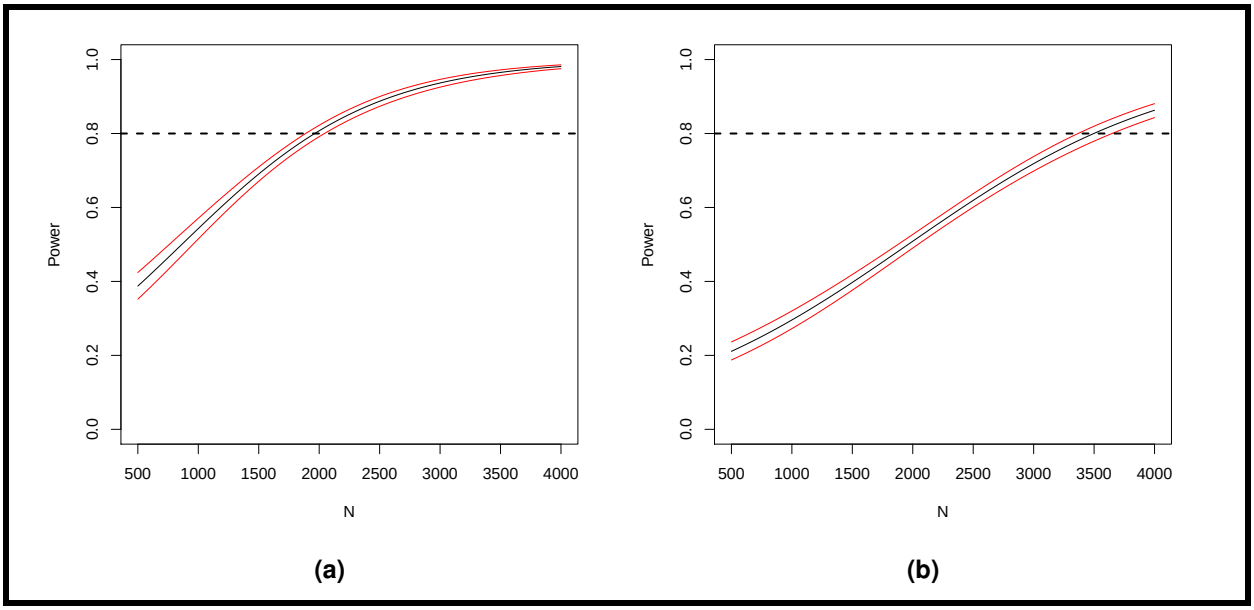

Figure I.6 Power curve and 95% confidence interval for G- ->D path in Fig. I.1 when parameter is set to .1 and parameter is known (a) and unknown (b).

```
# power curve for unknown path
plotPowerFit(model.power.10.ns, cutoff = c(chisq = 14.14))
# dashed horizontal line at .80
abline(h=.8, lwd=2, lty=2)
```

The numerical method of examining the results from simulating data with multiple sample sizes involves finding the smallest value of *n* that provides sufficient power. For the known parameter situation, the `getPower()` function will do this; for the unknown parameter situation, use the `getPowerFit()` function. Both require specifying the `nVal` argument with a value of *n*. As with creating power plots, specifying the `powerParam` or `cutoff` arguments will limit the returned power estimates to only contain the indicated parameters or fit measures, respectively.

```
getPower(specific.power.10.ns, nVal=1964, powerParam="G=~D")
getPowerFit(model.power.10.ns, nVal=3470, cutoff= c(chisq=14.14))
```

Achieving a power of .80 for rejecting the null hypothesis that the G-->D is zero when is it really .1 requires a *n* in excess of 1900. Achieving a power of .80 for rejecting a model without the G-->D path requires a *n* of almost 3500.

Appendix I Notes

Parameter accuracy. An alternative to assessing power is to examine parameter accuracy (Kelley & Lai, 2011, Lai & Kelley, 2011). From this perspective, the question of interest is the width of a parameter's or fit statistic's confidence interval instead of whether they are different from a specific value. This approach can be implemented using MC studies (e.g., Beaujean, 2014b) or using analytical values (e.g., Kelley, 2007).

Data quirks. Beaujean (2014a, 2014b) and Muthén and Muthén (2002) provide some examples of MC power studies that include missing data or manifest variables that are distinctly non-normal.

Other types of models. The MC approach to power analysis is very general and can be applied to any type of statistical model. For example, Meuleman and Billiet (2009) and Mathieu, Aguinis, Culpepper, and Chen (2012) provide examples for multilevel models, Thoemmes, MacKinnon, and Reiser (2010) and Schoemann, Boulton, and Short (2017) provide examples for complex mediation analysis, Schoemann, Miller, Pornprasertmanit, and Wu (2014) provide an example with a planned missingness design, and Væth and Skovlund (2004) provide examples for logistic and Cox regressions.

Monte Carlo power studies using other packages and programs. Hallgren (2013) provides examples of implementing MC power studies using R's native functions; Muthén and Muthén (2002) provides examples using Mplus; Paxton, Curran, Bollen, Kirby, and Chen (2001) provide examples using EQS; and Lee (2015) provides examples using Mplus, LISREL, and CALIS. The *simsem* package's website (http://simsem.org) provides additional examples of the simsem package using OpenMx and LISREL matrices.

Random seeds. Producing random numbers from computers—which are precise and deterministic—is a complex endeavor. In fact, it really cannot be done. That is why computer-generated data are usually called *pseudo-random*. Random seeds are values (typically integers) fed into a pseudo-random number generator to initiate the simulation process (Marsaglia, 2003). For more information, see Gentle (2003), Ross (2002), Press, Teukolsky, and Vetterling (2007).

References

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.

Beaujean, A. A. (2014a). *Latent variable modeling using R: A step-by-step guide*. New York, NY: Routledge/Taylor and Francis.

Beaujean, A. A. (2014b). Sample size determination for regression models using Monte Carlo methods in **R**. *Practical Assessment, Research, and Evaluation*, *19*(12), 1-16. Retrieved from `http://pareonline.net/getvn.asp?v=19&n=12`

Beaujean, A. A. (2017). Simulating data for clinical research: A tutorial. *The Journal of Psychoeducational Assessment*, Advance online publication. doi:10.1177/0734282917690302.

Gentle, J. E. (2003). *Random number generation and Monte Carlo methods* (2nd ed.). New York, NY: Springer.

Hallgren, K. A. (2013). Conducting simulation studies in the R programming environment. *Tutorials in Quantitative Methods for Psychology*, *9*, 43-60.

Kelley, K. (2007). Methods for the behavioral, educational, and social sciences: An R package. *Behavior Research Methods*, *39*, 979-984.

Kelley, K., & Lai, K. (2011). Accuracy in parameter estimation for the Root Mean Square Error of Approximation: Sample size planning for narrow confidence intervals. *Multivariate Behavioral Research*, *46*, 1-32.

Lai, K., & Kelley, K. (2011). Accuracy in parameter estimation for targeted effects in structural equation modeling: Sample size planning for narrow confidence intervals. *Psychological Methods*, *16*, 127-148.

Lee, S. (2015). Implementing a simulation study using multiple software packages for structural equation modeling. *SAGE Open*, *5*.

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods*, *1*, 130-149.

Marsaglia, G. (2003). Seeds for random number generators. *Communications of the ACM*, *46*, 90-93.

Mathieu, J. E., Aguinis, H., Culpepper, S. A., & Chen, G. (2012). Understanding and estimating the power to detect cross-level interaction effects in multilevel modeling. *Journal of Applied Psychology*, *97*, 951-966.

Meuleman, B., & Billiet, J. (2009). A Monte Carlo sample size study: How many countries are needed for accurate multilevel SEM? *Survey Research Methods*, *3*(1), 45-58. Retrieved from `http://ojs.ub.uni-konstanz.de/srm/article/view/666`

Muthén, L. K., & Muthén, B. O. (2002). How to use a Monte Carlo study to decide on sample size and determine power. *Structural Equation Modeling: A Multidisciplinary Journal*, *9*, 599-620.

Paxton, P., Curran, P. J., Bollen, K. A., Kirby, J., & Chen, F. (2001). Monte Carlo experiments: Design and implementation. *Structural Equation Modeling: A Multidisciplinary Journal*, *8*, 287 - 312.

Pornprasertmanit, S., Miller, P., & Schoemann, A. (2012). *simsem: SIMulated Structural Equation Modeling* [Computer software]. Retrieved from `http://CRAN.R-project.org/package=simsem`.

Press, W. H., Teukolsky, S. A., & Vetterling, W. T. (2007). *Numerical recipes: The art of scientific computing* (3rd ed.). Cambridge university press.

Ross, S. (2002). *Simulation* (3rd ed.). Orlando, FL: Academic Press.

Rosseel, Y. (2012). lavaan: An **R** package for structural equation modeling. *Journal of Statistical Software*, *48*(2), 1-36. Retrieved from `http://www.jstatsoft.org/v48/i02/`

Schoemann, A. M., Boulton, A. J., & Short, S. D. (2017). Determining power and sample size for simple and complex mediation models. *Social Psychological and Personality Science*, 1948550617715068.

Schoemann, A. M., Miller, P., Pornprasertmanit, S., & Wu, W. (2014). Using Monte Carlo simulations to determine power and sample size for planned missing designs. *International Journal of Behavioral Development*, *38*, 471-479.

Skrondal, A. (2000). Design and analysis of Monte Carlo experiments: Attacking the conventional wisdom. *Multivariate Behavioral Research*, *35*, 137-167.

Thoemmes, F., MacKinnon, D. P., & Reiser, M. R. (2010). Power analysis for complex mediational designs using Monte Carlo methods. *Structural Equation Modeling: A Multidisciplinary Journal*, *17*, 510-534.

Væth, M., & Skovlund, E. (2004). A simple approach to power and sample size calculations in logistic regression and Cox regression models. *Statistics in Medicine*, *23*, 1781-1792.