Power Analysis for Structural Equation Models: semPower Manual

Morten Moshagen

2022-11-04

Power Analysis for Structural Equation Models

Contact: morten.moshagen@uni-ulm.de

Please cite as:

- Moshagen, M., & Erdfelder, E. (2016). A new strategy for testing structural equation models. *Structural Equation Modeling, 23*, 54–60. https://doi.org/10.1080/10705511.2014.950896

An in-depth tutorial on power analyses in SEM using `semPower` is also provided in the following paper:

- Jobst, L., Bader, M., & Moshagen, M. (in press). A tutorial on power analysis and sample size planning in structural equation modeling. *Psychological Methods*. https://doi.org/10.1037/met0000423

Installation

The `semPower` package can be installed via CRAN. The latest development version and some supporting information is available from github at https://github.com/moshagen/semPower and can be installed as follows

```
install.packages("devtools")
devtools::install_github("moshagen/semPower")
```

Basic functionality is also provided as a shiny app, which can be used online at https://sempower.shinyapps.io/sempower.

Introduction

`semPower` provides a collection of functions to perform power analyses for structural equation models. The main functions of `semPower` are:

- semPower.aPriori: determine required sample size (N), given alpha, beta or power, effect, and df
- semPower.postHoc: determine achieved power (and beta), given alpha, N, effect, and df
- semPower.compromise: determine alpha and beta, given the alpha/beta ratio, N, effect, and df

All functions require the specification of the discrepancy between H0 and H1, i.e. the magnitude of effect in a certain metric. `semPower` understands the following effect-size measures: F0, RMSEA, Mc, GFI, AGFI (see below for details). Alternatively, it is also possible to define the effect by providing the H0 and the H1 covariance matrices (see covariance matrix input). For a simple CFA model, this process is simplified by using the semPower.powerCFA convenience function. Finally, power analyses always require specification of the model degrees of freedom, which can be obtained using the semPower.getDf convenience function.

In addition, `semPower` can be used to create power-plots:

- sempower.powerPlot.byN: Plot achieved power as function of N for a given effect
- sempower.powerPlot.byEffect: Plot achieved power as function of the magnitude of effect for a given sample size

For the impatient reader, this document starts with some brief examples. The remainder of this document provides some notes on the statistical background, the definition of various effect-sizes, and a detailed description of the functions contained in this package.

Examples

Determine required sample size (a priori power analysis)

Use semPower.aPriori to determine the required sample size to detect a certain effect with a certain power. For example, let's determine the required sample size to detect misspecifications of a model (involving df = 100 degrees of freedom) corresponding to RMSEA = .05 with a power of 80% on an alpha error of .05. After loading the `semPower` package, call the function `semPower.aPriori` with arguments `effect = .05`, `effect.measure = 'RMSEA'`, `alpha = .05`, `power = .80`, and `df = 100`.

```
ap <- semPower.aPriori(effect = .05, effect.measure = 'RMSEA',
                       alpha = .05, power = .80, df = 100)
summary(ap)
```

Equivalently, you can use the generic `semPower` function with the additional `type = 'a-priori'` argument:

```
ap <- semPower(type = 'a-priori', effect = .05, effect.measure = 'RMSEA',
               alpha = .05, power = .80, df = 100)
summary(ap)
```

The results show that a sample size of N = 164 yields a power of approximately 80% to reject a wrong model (with df = 100) with an amount of misspecification corresponding to RMSEA = .05 on alpha = .05.

Determine achieved power (post-hoc power analysis)

Use semPower.postHoc to determine the actually achieved power with a certain sample size. For example, let's determine the achieved power with a sample size of N = 1000 to detect misspecifications of a model (involving df = 100 degrees of freedom) corresponding to RMSEA = .05 on an alpha error of .05. Call the function `semPower.postHoc` with arguments `effect = .05`, `effect.measure = 'RMSEA'`, `alpha = .05`, `N = 1000`, and `df = 100`:

```
ph <- semPower.postHoc(effect = .05, effect.measure = 'RMSEA',
                       alpha = .05, N = 1000, df = 100)
summary(ph)
```

Or use the generic `semPower` function with the additional `type = 'post-hoc'` argument:

```
ph <- semPower(type = 'post-hoc', effect = .05, effect.measure = 'RMSEA',
               alpha = .05, N = 1000, df = 100)
summary(ph)
```

The results show that a sample size of N = 1000 is associated with a power larger than > 99.99 % (more precisely, 1 - beta = 1 - 2.903302e-17) to reject a wrong model (with df = 100) with an amount of misspecification corresponding to RMSEA = .05 on alpha = .05.

Determine critical chi-square and implied alpha and beta (compromise power analysis)

Use semPower.compromise to determine the critical chi-square and the implied alpha and beta errors. For example, let's determine the critical chi-square such that the associated alpha and beta errors are equal, assuming sample size of N = 1000, a model involving df = 100 degrees of freedom and misspecifications corresponding to RMSEA = .05. Call the function `semPower.compromise` with arguments `effect = .05`, `effect.measure = 'RMSEA'`, `abratio = 1`, `N = 1000`, and `df = 100`:

```
cp <- semPower.compromise(effect = .05, effect.measure = 'RMSEA',
                          abratio = 1, N = 1000, df = 100)
summary(cp)
```

Or use the generic `semPower` function with the additional `type = 'compromise'` argument:

```
cp <- semPower(type = 'compromise', effect = .05, effect.measure = 'RMSEA',
               abratio = 1, N = 1000, df = 100)
summary(cp)
```

The results show that choosing a critical chi-square of 192.82 is associated with balanced error probabilties alpha = beta = .000000074 when deciding between a correct and a wrong model (with df = 100) with an amount of misspecification corresponding to RMSEA = .05 on alpha = .05 with N = 1000.

Obtaining the model degrees of freedom (df)

Knowledge of the degrees of freedom (df) is required to perform a power analysis. When power refers to the comparison of two explicitly specified, nested models, the resulting df are just the difference between the df of the two models (or equivalently, the number of free parameters removed by the more restrictive model). When power is requested for the comparison of a hypothesized model to the saturated model, the model df are given by

$$df = p \cdot (p+1)/2 - q$$

where p is the number of observed variables and q is the number of free parameters of the hypothesized model. To obtain the latter in a typical SEM, one needs to count (a) loadings, (b) item-residual variances, and (c) covariance/regression parameters between factors and between item residuals.[1]. For instance, consider a correlated two factor CFA model with each factor measured by 4 items and no secondary loadings or residual correlations. Thus, there are (a) $2 \cdot 4$ loadings, (b) 8 item-residual variances, and (c) 1 covariance between the factors, which results in 17 free parameters. The df are thus $8 \cdot 9/2 - 17 = 19$.

If you are unsure or have a more complicated model (or both), `semPower` also includes a convenience function called `semPower.getDf` that determines the df for a model provided in the `lavaan` syntax (note that this function requires the `lavaan` package). For the model sketched above, running

```
# define model using standard lavaan syntax
lavmodel <- '
f1 =~ x1 + x2 + x3 + x4
f2 =~ x5 + x6 + x7 + x8
'
# obtain df
semPower.getDf(lavmodel)
```

[1]Note that factor (residual) variances can be omitted, because each factor needs to be assigned a scale, so one free parameter is lost for each factor anyway. Concerning free parameters, it does not matter whether factors are identified by fixing their variance or by fixing a loading.

gives 19 as result, which matches what we calculated by hand. `semPower.getDf` can also be used to obtain the df in multigroup settings by setting the arguments `nGroups` (and `group.equal` for models involving equality constraints).

```
# configural invariance
semPower.getDf(lavmodel, nGroups = 3)
# metric invariance
semPower.getDf(lavmodel, nGroups = 3, group.equal = c('loadings'))
# scalar invariance
semPower.getDf(lavmodel, nGroups = 3, group.equal = c('loadings', 'intercepts'))
```

Obtain power to detect a difference between two not explicitly specified models

A common scenario is to test two competing models against each other, where a more restrictive model (involving more df) is compared against a less restrictive model (involving less df). When the difference between these models lies just in a single (or few) particular parameter(s), the effect should be determined in terms of the model parameters (see below). If, however, the difference between the models potentially spreads across multiple parameters (as, say, when comparing a 3-factor with a 5-factor model), one approach to power analysis it to define the models in terms of overall fit.

For example, to obtain the required sample size to yield a power of 80% to discriminate a model with an associated RMSEA of .04 and 44 df from a model with an associated RMSEA of .05 and 41 df, define both the `effect` and `df` arguments as vectors (don't define lists!) comprising two elements:

```
ap <- semPower.aPriori(effect = c(.04, .05), effect.measure = 'RMSEA',
                       alpha = .05, power = .80, df = c(44, 41))
summary(ap)
```

which shows that 340 observations are required.

A similar situation often occurs in tests of measurement invariance, where one wants sufficient power to detect whether certain cross-group constraints on the model parameters (such as loadings) are viable. The general syntax is the same as above, but we now also need to set the `N` argument, which gives the number of observations by group in compromise and post-hoc power analysis, and the group weights in a-priori power analysis. For example, the following asks for the required sample size to detect a change in the Mc of .01 in a three-group model, where all groups are equal-sized:

```
ap <- semPower.aPriori(effect = c(.99, .98), effect.measure = 'Mc',
                       alpha = .05, power = .80, df = c(69, 57), N = c(1, 1, 1))
summary(ap)
```

This shows that 858 observations (286 by group) are required for a power of 80%.

Obtain power for factor correlations in a standard CFA model

It is rather common to test whether the correlation between two factors defined in a standard CFA model significantly differs from zero. Suppose that we want sufficient power to detect a correlation between two factors of $|r| \geq .2$. Performing a power analysis in this scenario is complicated by the fact that one needs to translate the hypothesized magnitude of a certain model parameter (i.e., the correlation between two factors) into a measure of effect such as the RMSEA, which is not easily possible.

To simplify this process, `semPower` provides the option to define the effect in terms of the discrepancy between covariance matrices (see next section). When the correlation between two factors in a standard CFA model is of interest, the `semPower.powerCFA` function is a simple shortcut. Note that this function requires the `lavaan` package.

`semPower.powerCFA` is quite flexible, but first consider the simplest case of a two factor CFA model with equal loadings on both factors. The argument `phi` defines the population correlation between these factors,

`nIndicator` is a vector indicating the number of indicators for the first and the second factor, respectively, and `loadM` gives the loading for all indicators. Thus, the following defines two factors that are correlated by .2 in the population, one measured by 5 and one measured by 6 indicators, and each loading being equal to .5:

```
# a priori power analysis providing the number of indicators to define
# two factors with correlation of phi and equal loading for all indicators
cfapower <- semPower.powerCFA(type = 'a-priori',
                              phi = .2, nIndicator = c(5, 6), loadM = .5,
                              alpha = .05, power = .95)
summary(cfapower$power)
```

The output shows that 774 observations are required to detect a correlation between the two factors of at least .2 with a power of 95% on alpha = 5%, when the model with the freely estimated correlation is compared against a model that constrains this correlation to zero.

Obviously, the above is a gross simplification, so `semPower.powerCFA` can also be used to sample the loadings for each factor. For example:

```
# a priori power analysisproviding the number of indicators to define
# two factors with correlation of phi, and mean and SD of loadings by factor.
cfapower <- semPower.powerCFA(type = 'a-priori',
                              phi = .2, nIndicator = c(5, 6),
                              loadM = c(.5, .6), loadSD = c(.01, .05),
                              alpha = .05,  power = .95)
summary(cfapower$power)
```

which shows that 672 observations are required to detect a correlation between the first two factors of at least .2 with a power of 95% on alpha = 5%, where the mean loadings on the first and second factor are .5 and .6, respectively (note that the output will slightly differ in each run, because the loadings are sampled).

For more details on `semPower.powerCFA`, see the section further below.

Extended example: Define effect in terms of model parameters

Consider the situation that one is interested in determining whether the observed responses on 8 items reflect two seperate (but correlated) factors or can be described by assuming just a single factor. A suitable model to test this hypothesis would specify two factors and constrain their correlation to 1. When this constrained model fits the data, a single factor is sufficient. Otherwise, two factors are required.

Suppose that a correlation between two factors of r > .9 is considered as implying that these are practically equivalent, so these could be collapsed into a single factor. However, if the correlations between these factors turns out smaller than .9, one might rather prefer a two-factor model. In terms of power analyses, we thus want sufficient power to identify whether the correlation between the factors is $\leq .9$. The misfit associated with a model assuming a correlation of 1 when, in reality, the true correlation is $\leq .9$ is supposed to define the magnitude of effect.

The problem is now that one cannot immediately say how this difference in a model parameter translates to an effect size such as the RMSEA. Instead, any type of power analysis must be performed using the H0 and the H1 covariance matrices, obtained by the help of a suitable sem-package, such as `lavaan`.

First, a suitable model that describes the true situation in the population is required. We need to define the value of each single (non-zero) parameter, so we need to specify each loading, each residual variance, the variance of the factors as well as their covariance. Note that the magnitude of effect (and thus power) does not only depend on the correlation between the factors, but also varies with other parameters of the model, such as the loading magnitude, so some reasonable guess concerning the latter required. Let us assume that the standardized loadings vary between .4 and .8. Importantly, I define that two factors exists that correlate by .9.

```
library(lavaan)

# define (true) population model
model.pop <- '
# define relations between factors and items in terms of loadings
f1 =~ .7*x1 + .7*x2 + .5*x3 + .5*x4
f2 =~ .8*x5 + .6*x6 + .6*x7 + .4*x8
# define unique variances of the items to be equal to 1-loading^2,
# so that the loadings above are in a standardized metric
x1 ~~ .51*x1
x2 ~~ .51*x2
x3 ~~ .75*x3
x4 ~~ .75*x4
x5 ~~ .36*x5
x6 ~~ .64*x6
x7 ~~ .64*x7
x8 ~~ .84*x8
# define variances of f1 and f2 to be 1
f1 ~~ 1*f1
f2 ~~ 1*f2
# define covariance (=correlation, because factor variances are 1)
# between the factors to be .9
f1 ~~ 0.9*f2
'
```

Using the model string above, we can now obtain the associated variance-covariance matrix defining the true state in the population.

```
# population covariance matrix
cov.pop <- fitted(sem(model.pop))$cov
```

It is a good idea to check whether the model string actually defined all parameters as we expected. To verifiy, we can just fit the true model to the population covariance matrix which should yield a perfect fit and give the same parameter estimates we used in the definition of the model string.

```
# sanity check
model.h1 <- '
f1 =~ x1 + x2 + x3 + x4
f2 =~ x5 + x6 + x7 + x8
'
summary(sem(model.h1, sample.cov = cov.pop,
            sample.nobs = 1000, sample.cov.rescale = F),
        stand = T, fit = T)
```

Having obtained the population covariance matrix, we now need to define and estimate the analysis model. The analysis model should make at least one restriction which is factually wrong and thereby defines the effect of interest, in the present case the restriction that the two factors correlate to 1.

```
# define (wrong) analysis model
model.h0 <- '
f1 =~ NA*x1 + x2 + x3 + x4
f2 =~ NA*x5 + x6 + x7 + x8
# define variances of f1 and f2 to be 1
f1 ~~ 1*f1
f2 ~~ 1*f2
# set correlation between the factors to 1
```

```
f1 ~~ 1*f2
'
# fit analysis model to population data; note the sample.nobs are arbitrary
fit.h0 <- sem(model.h0, sample.cov = cov.pop,
              sample.nobs = 1000, sample.cov.rescale = F,
              likelihood='wishart')
```

Any type of power-analysis can now be performed using the model-implied covariance matrix along with the population covariance matrix as arguments. For example, to determine the required sample size use semPower.aPriori with `SigmaHat = cov.h0`, `Sigma = cov.pop`, `alpha = .05`, `power = .80`, and `df = df`.

```
# model implied covariance matrix
cov.h0 <- fitted(fit.h0)$cov
df <- fit.h0@test[[1]]$df
# perform power analysis
ap5 <- semPower.aPriori(SigmaHat = cov.h0, Sigma = cov.pop,
                        alpha = .05, power = .95, df = df)
summary(ap5)
```

The output (which is omitted here) shows that N = 1258 observations are required to detect a correlation between the factors $\leq .9$ with a power of 95% on alpha = .05. The output also shows various effect-sizes as implied by the specified covariance matrices. In the population, fitting a model assuming the two factors correlate by 1 when they actually correlate by .9 leads to a population minimum of the ML-fitting function of F0 = 0.0244, which amounts to model misfit corresponding to RMSEA = 0.035 or SRMR = 0.016.

Extended example: Power to detect invariance constraints in a multiple group model

Models without meanstructures

Suppose we fit a model to two groups and are interested in whether the loadings are invariant across groups. To test this hypothesis, a model with freely estimated loadings would be compared against a model restricting the loadings to be equal across groups (metric invariance model). If the the fit of the latter model is significantly worse, this supports the conclusion that at least one loading systematically differs across groups. In terms of power analyses, the misfit associated with a model assuming equal loadings across groups when, in reality, at least one loading differs, defines the magnitude of effect, which we want to detect with a sufficient power.

Again, we need to setup a model describing the true situation in the population to obtain the magnitude of effect. Because we have a two-group model, the model needs to be specified for both groups. Let's assume we have a two factor model with each factor being indicated by three manifest variables. Let's further assume that the loading of `x2` differs across groups:

```
library(lavaan)

# population model group 1
model.pop.g1 <- '
f1 =~ .8*x1 + .7*x2 + .6*x3
f2 =~ .7*x4 + .6*x5 + .5*x6
x1 ~~ .36*x1
x2 ~~ .51*x2
x3 ~~ .64*x3
x4 ~~ .51*x4
x5 ~~ .64*x5
x6 ~~ .75*x6
f1 ~~ 1*f1
```

```
f2 ~~ 1*f2
f1 ~~ 0.5*f2
'
# population model group 2
model.pop.g2 <- '
f1 =~ .8*x1 + .4*x2 + .6*x3
f2 =~ .7*x4 + .6*x5 + .5*x6
x1 ~~ .36*x1
x2 ~~ .84*x2
x3 ~~ .64*x3
x4 ~~ .51*x4
x5 ~~ .64*x5
x6 ~~ .75*x6
f1 ~~ 1*f1
f2 ~~ 1*f2
f1 ~~ 0.5*f2
'
```

Note that the models are almost identical, with the exception that the loading of `x2` is .7 in the first group, but .4 in the second group. Based on the the model strings above, we can obtain the associated variance-covariance matrices for both groups:

```
# population covariance matrices
cov.pop.g1 <- fitted(sem(model.pop.g1))$cov
cov.pop.g2 <- fitted(sem(model.pop.g2))$cov
```

Next, we define and estimate the analysis model. The model of interest is a metric invariance model, i.e., a multiple group model restricting the loadings to be equal across groups. By fitting the model to the population covariance matrices obtained above (note the `group.equal = 'loadings'` argument in the `lavaan` call) the model-implied covariance matrices for each group is obtained:

```
# define analysis model
model.h0 <- '
f1 =~ x1 + x2 + x3
f2 =~ x4 + x5 + x6
'
# fit analysis model to population data; note the sample.nobs are arbitrary
fit.h0 <- sem(model.h0, sample.cov = list(cov.pop.g1, cov.pop.g2),
              sample.nobs = list(1000, 1000), sample.cov.rescale = F,
              group.equal = 'loadings', likelihood='wishart')
# get model implied covariance matrices
cov.h0.g1 <- fitted(fit.h0)$`Group 1`$cov
cov.h0.g2 <- fitted(fit.h0)$`Group 2`$cov
# df of metric invariance model
df <- fit.h0@test[[1]]$df
# obtain baseline df (no invariance constraints)
fit.bl <- sem(model.h0, sample.cov = list(cov.pop.g1, cov.pop.g2),
              sample.nobs = list(1000, 1000), sample.cov.rescale=F,
              likelihood='wishart')
df.bl <- fit.bl@test[[1]]$df
# difference in df
df.diff <- df - df.bl
```

We also obtain the difference in the degrees of freedom (`df.diff`) between the metric invariance model and the model without invariance constraints (configural invariance model), because the metric invariance model is usually compared to the configural invariance model (rather than to the saturated model). We can now

perform any type of power-analysis by using the population and the model-implied covariance matrices as obtained above. In multiple group models, the arguments `SigmaHat` and `Sigma` are lists containing the associated matrices for each group. In addition, `N` also needs to be a list specifying the sample size for each group. In a priori power analyses, `N` determines the sample size weights for each group, so setting `N = list(1, 1)` assigns the same sample size to both groups.

```
# perform a priori power analysis
ap6 <- semPower.aPriori(SigmaHat = list(cov.h0.g1, cov.h0.g2),
                        Sigma = list(cov.pop.g1, cov.pop.g2),
                        alpha = .05, beta = .20, N = list(1, 1), df = df.diff)

summary(ap6)
```

The output (which is omitted here) shows that N = 778 observations (n = 389 by group) are required to detect the lack of metric invariance with a power of 80% on alpha = .05.

Alternatively, `semPower` can also compute power in multiple group settings when the effect contribution of each group to the total effect of interest is provided. In the present example, F0 is 0.01102 and 0.01979 in the first and second group, respectively. Computing a post-hoc power analysis with `effect = list(0.01102, 0.01979)`, `effect.measure = 'F0'`, and `N = list(389, 389)` thus replicates the above result in showing that the power on alpha = .05 is approximately 80%.

```
# perform post-hoc power analysis
ph6 <- semPower.postHoc(effect = list(0.01102, 0.01979), effect.measure = 'F0',
                        alpha = .05, N = list(389, 389), df = df.diff)
summary(ph6)
```

Models including meanstructures

The process described in the previous section is highly similar when power to detect invariance constrains concerning means (i.e., intercepts or latent means) is of concern. The main difference is that the model strings need to define intercepts as well as latent means and the population and implied means need to be obtained as well.

For illustration, we add information on means to the models defined above. Specifically, we define all intercepts and latent means to be zero in the first group, whereas the intercept of `x4` is .5 is the second group, which also has a latent mean on the first factor `f1` of 1:

```
library(lavaan)

# population model group 1
model.pop.g1 <- '
f1 =~ .8*x1 + .7*x2 + .6*x3
f2 =~ .7*x4 + .6*x5 + .5*x6
x1 ~~ .36*x1
x2 ~~ .51*x2
x3 ~~ .64*x3
x4 ~~ .51*x4
x5 ~~ .64*x5
x6 ~~ .75*x6
f1 ~~ 1*f1
f2 ~~ 1*f2
f1 ~~ 0.5*f2
x1 ~ 0*1
x2 ~ 0*1
x3 ~ 0*1
x4 ~ 0*1
```

```
x5 ~ 0*1
x6 ~ 0*1
f1 ~ 0*1
f2 ~ 0*1
'
# population model group 2
model.pop.g2 <- '
f1 =~ .8*x1 + .4*x2 + .6*x3
f2 =~ .7*x4 + .6*x5 + .5*x6
x1 ~~ .36*x1
x2 ~~ .84*x2
x3 ~~ .64*x3
x4 ~~ .51*x4
x5 ~~ .64*x5
x6 ~~ .75*x6
f1 ~~ 1*f1
f2 ~~ 1*f2
f1 ~~ 0.5*f2
x1 ~ 0*1
x2 ~ 0*1
x3 ~ 0*1
x4 ~ .5*1 # no inv
x5 ~ 0*1
x6 ~ 0*1
f1 ~ 1*1  #no inv
f2 ~ 0*1
'
```

We then obtain both the model-implied covariance matrices and the model-implied means:

```
# population covariance matrices
cov.pop.g1 <- fitted(sem(model.pop.g1))$cov
cov.pop.g2 <- fitted(sem(model.pop.g2))$cov
# population means
mu.pop.g1 <- fitted(sem(model.pop.g1))$mean
mu.pop.g2 <- fitted(sem(model.pop.g2))$mean
```

We next fit the hypothesized model (here assuming scalar invariance) to the population data, obtain the associated model-implied covariance matrices and mean vectors, and again store the df of the models.

```
# define analysis model
model.h0 <- '
f1 =~ x1 + x2 + x3
f2 =~ x4 + x5 + x6
'
# fit analysis model to population data; note the sample.nobs are arbitrary
fit.h0 <- sem(model.h0,
              sample.cov = list(cov.pop.g1, cov.pop.g2),
              sample.mean = list(mu.pop.g1, mu.pop.g2),
              sample.nobs = list(1000, 1000), sample.cov.rescale = F,
              group.equal = c('loadings', 'intercepts'), likelihood='wishart')
# get model implied covariance matrices
cov.h0.g1 <- fitted(fit.h0)$`Group 1`$cov
cov.h0.g2 <- fitted(fit.h0)$`Group 2`$cov
# get model implied means
```

```
mu.h0.g1 <- fitted(fit.h0)$`Group 1`$mean
mu.h0.g2 <- fitted(fit.h0)$`Group 2`$mean
# df of metric invariance model
df <- fit.h0@test[[1]]$df
# obtain baseline df (no invariance constraints)
fit.bl <- sem(model.h0,
              sample.cov = list(cov.pop.g1, cov.pop.g2),
              sample.mean = list(mu.pop.g1, mu.pop.g2),
              sample.nobs = list(1000, 1000), sample.cov.rescale = F, likelihood='wishart')
df.bl <- fit.bl@test[[1]]$df
# difference in df
df.diff <- df - df.bl
```

Finally, we can plug all matrices and vectors in the `semPower.aPriori` command:

```
#  apriori power using cov matrices and mean vectors
ap6b <- semPower.aPriori(SigmaHat = list(cov.h0.g1, cov.h0.g2),
                         Sigma = list(cov.pop.g1, cov.pop.g2),
                         muHat = list(mu.h0.g1, mu.h0.g2),
                         mu = list(mu.pop.g1, mu.pop.g2),
                         alpha = .05, beta = .20, N = list(1, 1), df = df.diff)
summary(ap6b)
```

Background

The statistical evaluation of mathematical models often proceeds by considering a test-statistic that expresses the discrepancy between the observed data and the data as implied by the fitted model. In SEM, the relevant test statistic for a sample of size N is given by $T = \hat{F}(N-1)$. $\hat{F}$ denotes the minimized sample value of the chosen discrepancy function (such as Maximum-Likelihood) and thereby indicates lack of fit of the model to the sample data. Thus, T permits a likelihood-ratio test of the null hypothesis (H0) that the model is correct. If the hypothesized model holds in the population, T can be shown to follow asymptotically a central χ^2(df) distribution with $df = .5 \cdot p(p+1) - q$ degrees of freedom, where p is the number of manifest variables and q denotes the number of free parameters. This is why T is often referred to as "chi-square model test statistic" – a convention which is followed here.

Given an observed value for the chi-square test statistic, a null hypothesis significance test can be performed to evaluate whether this value is larger than would be expected by chance alone. The usual test proceeds as follows: Given a certain specified alpha-error level (typically alpha = .05), a critical chi-square value is obtained from the asymptotic central χ^2(df) distribution. If the observed value for the chi-square test statistic exceeds the critical value, the null hypothesis that the model fits the data is rejected. Otherwise, H0 is retained. A finding of the observed statistic exceeding the critical value (implying an upper-tail probability that falls below the specified alpha level) thus leads to the statistical decision that the discrepancy between the hypothesized and the actual population covariance matrix is too large to be attributable to sampling error only. Accordingly, a statistically significant chi-square test statistic provides evidence against the validity of the hypothesized model.

When testing statistical hypotheses using this framework, two types of decision errors can occur: The alpha error (or type-I error) of incorrectly rejecting a true H0 (a correct model) and the beta error (or type-II error) of incorrectly retaining a false H0 (an incorrect model). Statistical power is defined as the complement of the beta-error probability (1 - beta) and thus gives the probability to reject an incorrect model.

If the H0 is false, the chi-square test statistic is no longer central χ^2(df) distributed, but can be shown to follow a noncentral χ^2(df, λ) distribution with non-centrality parameter λ and expected value df + λ (MacCallum, Browne, & Sugawara, 1996). The non-centrality parameter λ shifts the expected value of the

non-central χ^2(df, λ) distribution to the right of the corresponding central distribution. Having determined the critical value associated with the desired alpha probability from the central χ^2(df) distribution, the beta-error probability can be computed by constructing the corresponding non-central χ^2(df, λ)distribution with a certain non-centrality parameter λ and obtaining the area (i.e., the integral) of this distribution to the left of the critical value Correspondingly, statistical power is the area of the non-central χ^2(df, λ) distribution to the right of the critical value. The general situation is illustrated in Figure 1.

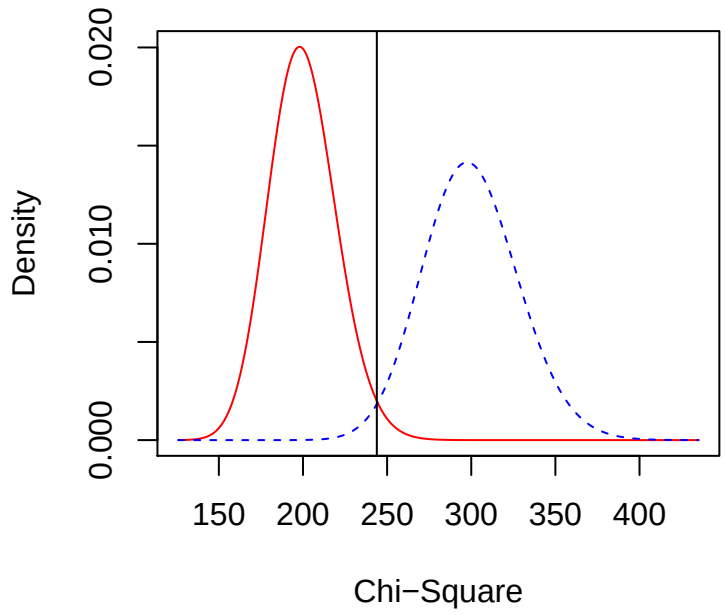


Figure 1: Central (left) and non-central (right) chi-square distributions

Figure 1 depicts a central (solid) χ^2(df) and a non-central (dashed) χ^2(df, λ) distribution with df = 200 and $\lambda = 100$. The area of the central distribution χ^2(df) to the right of the critical value reflects the alpha error. The solid line indicates a critical value of 234, which corresponds to alpha = .05. The dotted line shows a critical value of 244 obtained from compromise power analyses, which is associated with alpha = .018. The area of χ^2(df, λ) distribution to the left of the critical value is the beta-error probability (beta = .006 and beta = .018 for critical values of 234 and 244, respectively). Statistical power is defined as 1 - beta, that is, the area under the noncentral χ^2(df, λ) distribution to the right of the critical value.

Measures of Effect

To define the discrepancy between the H0 and the H1 model, any non-centrality based measure of effect can be used. `semPower` understands the measures detailed below.

F0

F_0 is the population minimum of the Maximum-Likelihood fitting function, defined as

$$F_0 = \log|\Sigma| - \log|\hat{\Sigma}| + tr(\hat{\Sigma}\Sigma) - p$$

where Σ is the $p \cdot p$ population covariance matrix, $\hat{\Sigma}$ the $p \cdot p$ model-implied covariance matrix, and p the number of observed variables. If the model is correct, $\hat{\Sigma} = \Sigma$ and $F_0 = 0$. Otherwise, $F_0 > 0$ with higher values expressing a larger discrepancy (misfit) of the model to the data.

If fitting a model to some sample data of size N, the estimated minimum of the fit-function, $\hat{F}$, is used to construct an asymptotically χ^2-distributed model test-statistic, commonly simply called the chi-squared model test:

$$\chi^2 = \hat{F}(N-1)$$

Note that $\hat{F}$ is a biased estimate of F_0. If $F_0 = 0$, the expected value of $\hat{F}$ is df, i.e. the model degree of freedom. For a model with q free parameters, the df are given by

$$df = \frac{p(p+1)}{2} - q$$

RMSEA

The Root-Mean-Squared Error of Approximation (RMSEA; Browne & Cudeck, 1992; Steiger & Lind, 1980) scales F_0 by the model degrees of freedom:

$$RMSEA = \sqrt{(F_0/df)}$$

so that the RMSEA is bounded by zero and lower values indicate better fit. The implied F_0 is:

$$F_0 = df \cdot RMSEA^2$$

Given that $F0$ is scaled by the df, defining an effect in terms of the RMSEA requires specification of the degrees of freedom.

Mc

Mc (McDonald, 1989) is a transformation of F_0 on the interval 0-1 with higher values indicating better fit.

$$Mc = e^{-.5F_0}$$

$$F_0 = -2\ln Mc$$

GFI

The Goodness-of-Fit Index (GFI; Jöreskog & Sörbom, 1984; Steiger, 1990) scales F_0 on the interval 0-1 with higher values indicating better fit:

$$GFI = \frac{p}{p + 2F_0}$$

$$F_0 = \frac{p(1 - GFI)}{2GFI}$$

As the GFI depends on the number of observed variables (p), this number need to be provided when defining an effect in terms of the GFI.

AGFI

The Adjusted Goodness-of-Fit Index (AGFI; Jöreskog & Sörbom, 1984; Steiger, 1990) modifies the GFI by including a penalty for the number pf free parameters, as measured (somewhat indirectly) by the model degrees of freedom:

$$AGFI = 1 - \frac{p(p+1)}{2df}\left(1 - \frac{p}{p + 2F_0}\right)$$
$$F_0 = \frac{p(1 - AGFI)df}{p(p+1) - 2df(1 - AGFI)}$$

Specifying an effect in terms of the AGFI requires specification of both the number of observed variables (p) and the model degrees of freedom (df).

Measures not based on non-centrality

Fit-indices that are not based on non-centrality have no straightforward relation to F0 and are thus not well suited for power-analyses. However, if defining the effect through H0 and H1 covariance matrices (see covariance matrix input), such measures can at least be computed.

SRMR

The Standardized-Root-Mean-Square Residual (SRMR) is a measure of the (root of the) average (squared) difference between the (standardized) model-implied and population covariance matrices, so that it ranges from 0 to 1 with lower values indicating better fit. Let E_0 be the difference between the model-implied and the population covariance matrix, $E_0 = \Sigma - \hat{\Sigma}$, $vech$ denote the vectorization transformation, and Q be a diagonal matrix of dimension $.5p(p+1)$ containing the inverse of the product of standard deviations of observed variables i and j. Then, the SRMR can be defined as

$$SRMR = \sqrt{\frac{1}{.5p(p+1)} vech(E_0) \cdot Q \cdot vech(E_0)'}$$

The relation of the residual matrix E_0 to F_0 is complex and depends on the model-implied covariance matrix, so the SRMR is not well suited to define an effect in terms of F_0 (based on ML estimation):

$$F_0 = -\ln|I + \hat{\Sigma}^{-.5} E_0 \hat{\Sigma}^{-.5}|$$

CFI

The Comparative Fit Index (CFI) is an incremental index expressing the proportionate reduction of misfit associated with the hypothesized model (F_{0H}) in relation to the null-model (F_{0N}), defined as a model that constraints all covariances to zero. In the population, the CFI ranges from 0 to 1 with higher values indicating better fit.

$$CFI = \frac{F_{0N} - F_{0H}}{F_{0N}}$$

Whereas it is simple to obtain $F0$ from the CFI, this requires knowledge of F_{0N}, which is rather difficult to determine a priori:

$$F_0 = F_{0N} - CFI \cdot F_{0N}$$

Power Analyses

Performing a power analysis generally requires the specification of a measure and magnitude of effect that is to be detected and a provision of the model degrees of freedom. Further arguments are required depending on the type of power analysis. This section assumes that the effect is specified in terms of one of the measures described above, see below for a description of how to define an effect by resorting on model-implied and population covariance matrices.

A-Priori: Determine required N, given alpha, beta, effect, and df

The purpose of a-priori power analyses is to determine the required sample size to detect an effect, given a specified alpha error. In the language of structural equation modeling, an a-priori power analysis asks: How many observations do I need to falsify my model with a probability of X% (power), if it is actually wrong (to the extent as defined by the chosen effect)?

Performing an a-priori power analyses requires the specification of an alpha error, the desired power (or, equivalently the acceptable beta error), the type and magnitude of effect, and the model df. Depending on the chosen effect-size measure, it may also be required to define the number of observed variables.

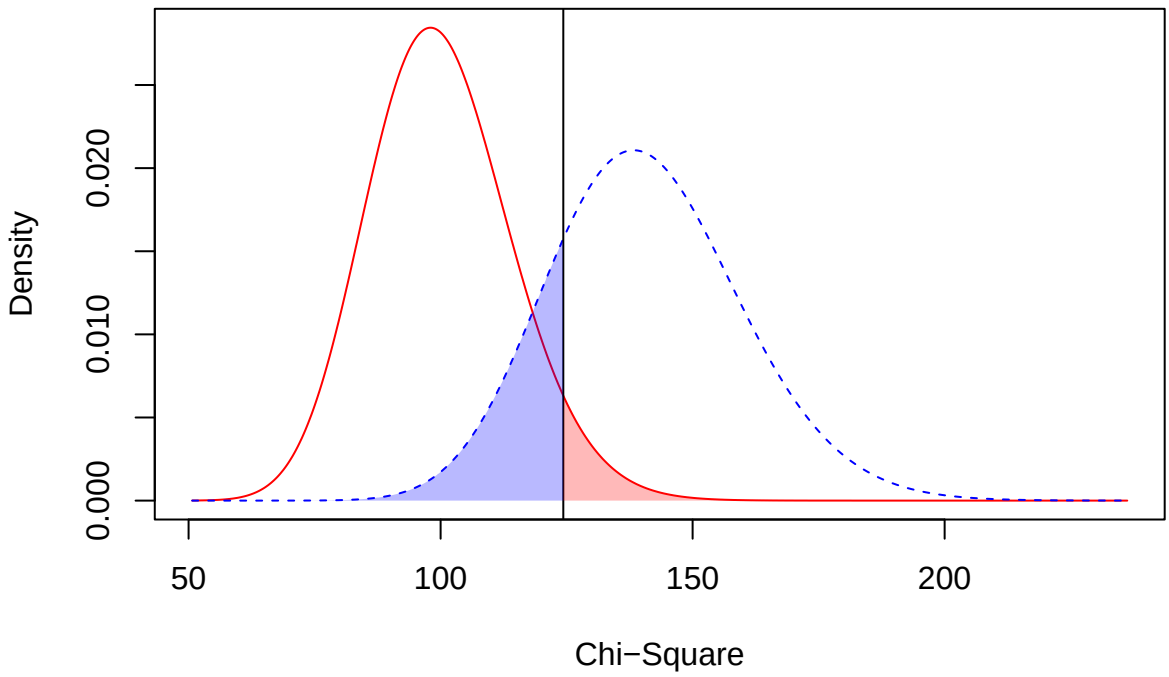
Suppose, we want the required sample size to detect misspecifications of a model (involving df = 100 degrees of freedom) with a power of 80% on an alpha error of .05., where the amount of misfit corresponds to RMSEA = .05. After loading the `semPower` package, we call the function `semPower.aPriori` with arguments `effect = .08`, `effect.measure = 'RMSEA'`, `alpha = .05`, `power = .80`, and `df = 100`, and store the results in a list called `ap1`.

```
library(semPower)
#>
#> ### Welcome to semPower 1.3.0 ###
#>
#> See https://github.com/moshagen/semPower for quick examples and a detailed manual.
#>
#> Please cite as:
#> Moshagen, M., & Erdfelder, E. (2016). A new strategy for testing structural equation models.
#> Structural Equation Modeling, 23, 54-60. doi: 10.1080/10705511.2014.950896
ap1 <- semPower.aPriori(effect = .05, effect.measure = 'RMSEA',
                        alpha = .05, power = .80, df = 100)
```

Calling the `summary` method on `ap1` prints the results and a figure of the associated central and non-central chi-squared distributions.

```
summary(ap1)
#>
#>  semPower: A-priori power analysis
#>
#>  F0                          0.250000
#>  RMSEA                       0.050000
#>  Mc                          0.882497
#>
```

```
#>  df                          100
#>  Required Num Observations 164
#>
#>  Critical Chi-Square       124.3421
#>  NCP                       40.75000
#>  Alpha                     0.050000
#>  Beta                      0.197211
#>  Power (1-beta)            0.802789
#>  Implied Alpha/Beta Ratio  0.253535
```

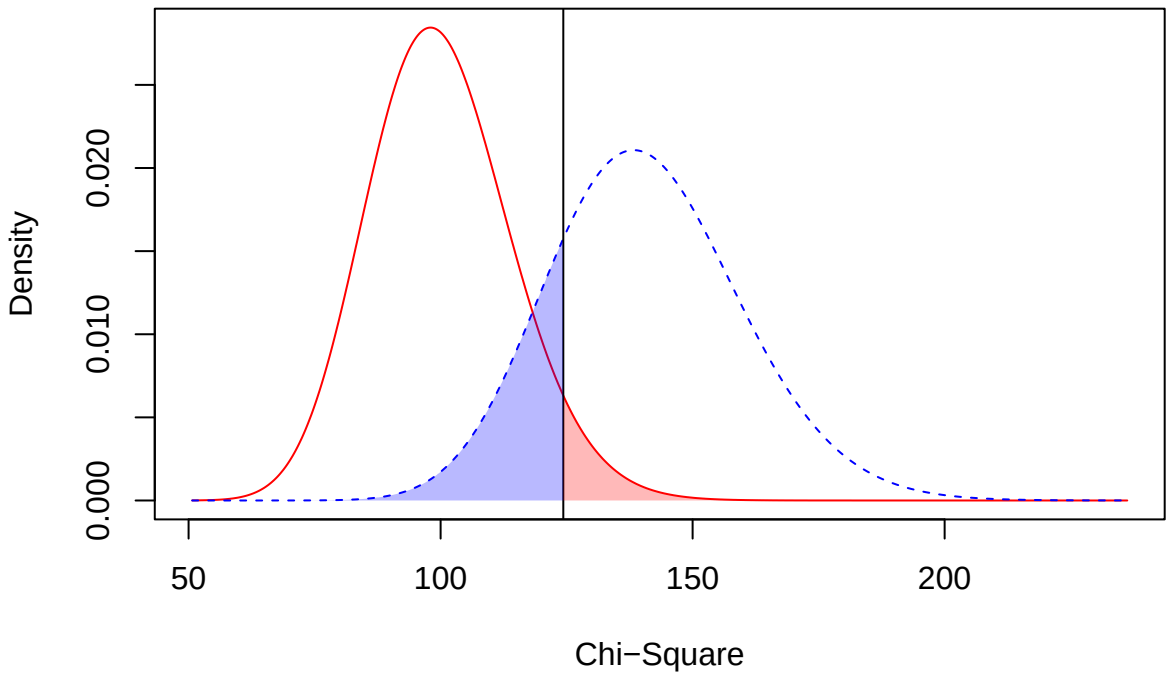


This shows that $N = 164$ yields a power of approximately 80% to detect the specified effect. The output further shows the `Critical Chi-Square`, the non-centrality parameter (`NCP`), and the ratio between the error probabilities (`Implied Alpha/Beta ratio`). In this example, the ratio between alpha and beta is 0.25, showing that committing an beta error is four-times as likely as committing an alpha error. This is obviously a consequence of the chosen input parameters, since a power (1 - beta) of .80 implies an beta error of .20, which is four times the chosen alpha error of .05.

`semPower` also converts the chosen effect into other effect size measures, here into F0 and Mc: An RMSEA of .05 (based on df = 100) corresponds to F0 = 0.250 and Mc = .882. If we would also be interested in obtaining the associated GFI and AGFI, the number of variables needs to be provided. Let's assume our model involves 20 observed variables. We repeat the call above, this time including the argument `p = 20`:

```
ap2 <- semPower.aPriori(effect = .05, effect.measure = 'RMSEA',
                        alpha = .05, power = .80, df = 100, p = 20)
summary(ap2)
#>
#>  semPower: A-priori power analysis
#>
#>  F0                        0.250000
#>  RMSEA                     0.050000
#>  Mc                        0.882497
#>  GFI                       0.975610
#>  AGFI                      0.948780
```

```
#>
#>  df                          100
#>  Required Num Observations 164
#>
#>  Critical Chi-Square       124.3421
#>  NCP                       40.75000
#>  Alpha                     0.050000
#>  Beta                      0.197211
#>  Power (1-beta)            0.802789
#>  Implied Alpha/Beta Ratio  0.253535
```



We now also get the GFI and AGFI equivalents of RMSEA = .05, assuming df = 100 and p = 20.

Instead of specifying the desired power, it is of course also possible to specify the accepted beta error. For example, calling

```
ap3 <- semPower.aPriori(effect = .05, effect.measure = 'RMSEA',
                        alpha = .05, beta = .20, df = 100, p = 20)
```

gives the same output as above.

If you are interested in how power changes for a range of sample sizes, it is useful to request a power plot, as described in detail below.

Post-hoc: Determine achieved power, given alpha, N, effect, and df

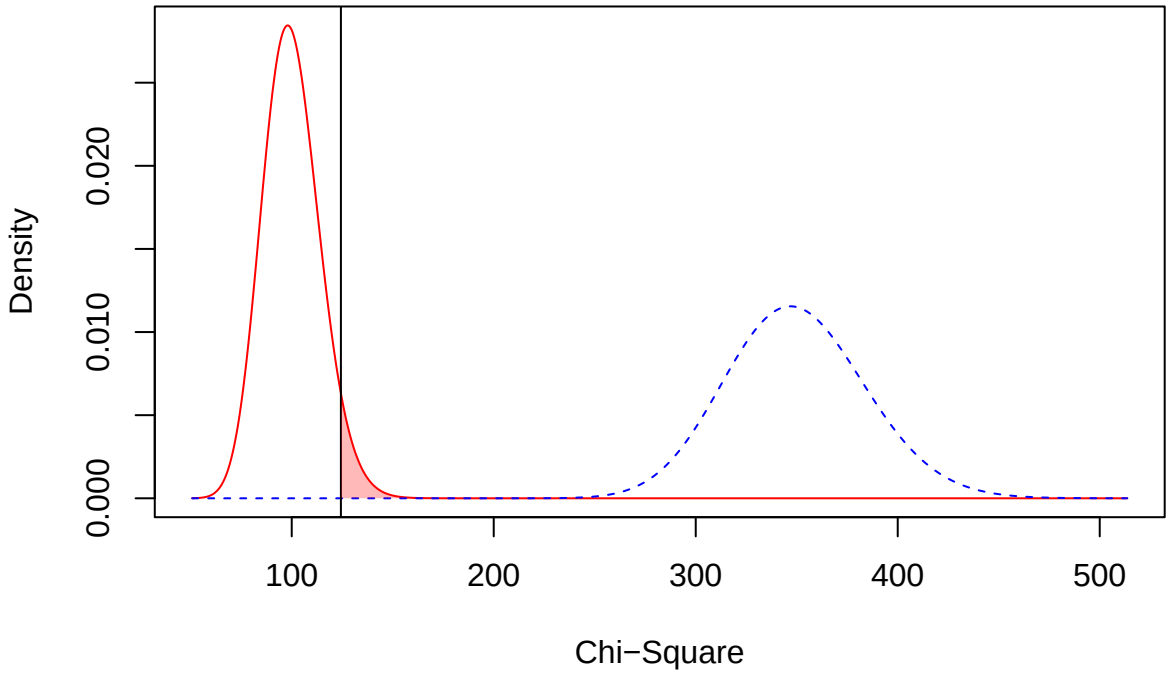
The purpose of post-hoc power analyses is to determine the actually achieved power to detect a specified effect with given sample size on a certain alpha error. In the language of structural equation modeling, a post-hoc power analysis asks: With my sample at hand, how large is the probability (power) to falsify my model if it is actually wrong (at least to the extent as defined by the chosen effect)?

Performing a post-hoc power analyses requires the specification of an alpha error, the sample size, the type and magnitude of effect, and the model df. Again, depending on the chosen effect-size measure, it may also

be required to define the number of observed variables.

Suppose, we want the achieved power with a sample size of $N = 1000$ to detect misspecifications of a model (involving $df = 100$ degrees of freedom) on an alpha error of .05., where the amount of misfit corresponds to RMSEA = .05. We call the function `semPower.postHoc` with arguments `effect = .08`, `effect.measure = 'RMSEA'`, `alpha = .05`, `N = 1000`, and `df = 100`, and store the results in a list called `ph1`.

```
ph1 <- semPower.postHoc(effect = .05, effect.measure = 'RMSEA',
                        alpha = .05, N = 1000, df = 100)
summary(ph1)
#>
#>  semPower: Post-hoc power analysis
#>
#>  F0                          0.250000
#>  RMSEA                       0.050000
#>  Mc                          0.882497
#>
#>  df                          100
#>  Num Observations            1000
#>  NCP                         249.7500
#>
#>  Critical Chi-Square         124.3421
#>  Alpha                       0.050000
#>  Beta                        2.903302e-17
#>  Power (1-beta)              > 0.9999
#>  Implied Alpha/Beta Ratio 1.722177e+15
```



Calling the `summary` method on `ph1` shows that the power is very high (`power > .9999`). The associated error probabilities are provided in higher precision. Specifically, the beta error is `beta = 2.903302e-17` which translates into $2.9 \cdot 10^{-17} = 0.000000000000000029$. In practice, you would almost never miss a model with an actual RMSEA >= .05 (or F0 >= 0.25 or Mc <= .882) under these conditions. The implied alpha/beta ratio is 1.722177e+15, showing that committing an alpha error is about two quadrillion (10^{15}) times as likely as committing a beta error.

If you are interested in how power changes for a range of different magnitudes of effect (say, for RMSEA ranging from .01 to .15), it is useful to request a power plot, as described in detail below.

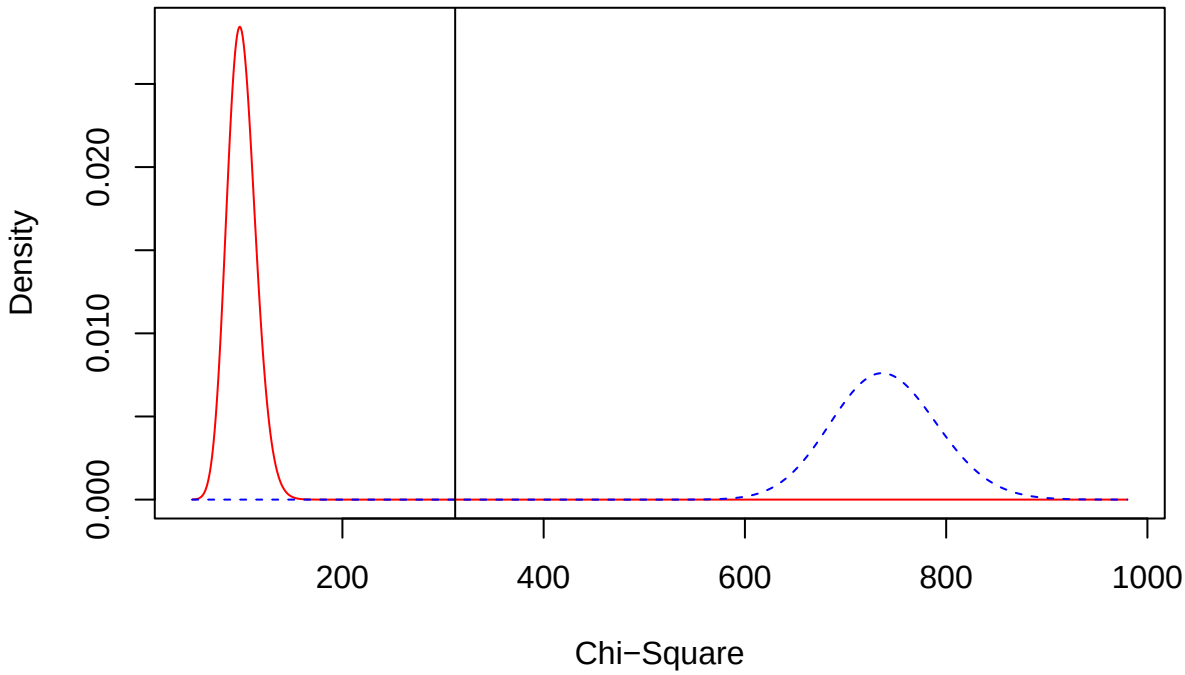
Compromise: Determine alpha and beta, given the alpha/beta ratio, N, effect, and df

The purpose of compromise power analyses is to determine alpha and beta (and the associated critical value of the chi-squared test-statistic) given a specified effect, a certain sample size, and a desired ratio between alpha and beta (Moshagen & Erdfelder, 2016). In the language of structural equation modeling, a compromise analysis asks: With my sample at hand, how should I select a critical value for the chi-square model-test to obtain proportionate alpha and beta errors in deciding whether my model is rather aligned with the hypothesis of perfect fit or with the hypothesis of unacceptable degree of misfit (as defined by the chosen effect)?

Performing a compromise power analyses requires the specification of a desired ratio between alpha and beta (which defaults to 1), the sample size, the type and magnitude of effect (defining the H1 model representing an unacceptable degree of misfit), and the model df. Again, depending on the chosen effect-size measure, it may also be required to define the number of observed variables.

Suppose, we want to determine the critical chi-square and the associated alpha and beta errors, forcing them to be equal (i.e., an ratio of 1). Our model involves 100 df, our sample size is $N = 1000$, and we define the H1 model representing an unacceptable degree of misfit as a model associated with an RMSEA of at least .08. Thus, we call the function `semPower.compromise` with arguments `effect = .08`, `effect.measure = 'RMSEA'`, `abratio = 1`, `N = 1000`, and `df = 100`, store the results in a list called `cp1`, and call the `summary` method.

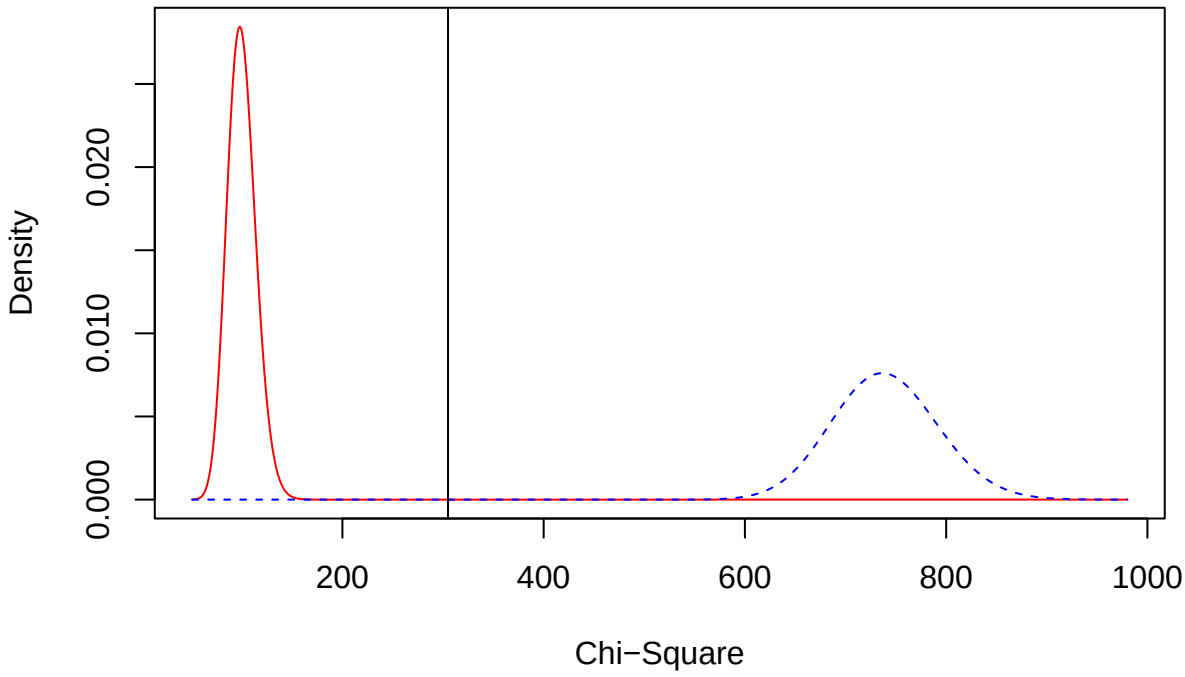
```
cp1 <- semPower.compromise(effect = .08, effect.measure = 'RMSEA',
                           abratio = 1, N = 1000, df = 100)
summary(cp1)
#>
#>  semPower: Compromise power analysis
#>
#>  F0                        0.640000
#>  RMSEA                     0.080000
#>  Mc                        0.726149
#>
#>  df                        100
#>  Num Observations          1000
#>  Desired Alpha/Beta Ratio 1.000000
#>
#>  Critical Chi-Square       312.0477
#>  Implied Alpha             1.212986e-23
#>  Implied Beta              1.212986e-23
#>  Implied Power (1-beta)    > 0.9999
#>  Actual Alpha/Beta Ratio   1.000000
```



The results show that choosing a `Critical Chi-Square = 312` is associated with balanced error probabilities, `alpha = 1.212986e-23` and `beta = 1.212986e-23`. As requested, both error probabilities as just as large. In addition, committing either error is highly unlikely: an error of `1.212986e-23` translates into $1.2 \cdot 10^{-23} = 0.000000000000000000000012$. In practice, one almost never would make a wrong decision.

If, for some reason, you rather prefer the error probabilities to differ (for example because you consider falsely accepting an incorrect model to be 100 times as bad as falsely rejecting an correct model), you can achieve this by changing the `abratio` argument accordingly. For example, requesting the alpha error to be 100 times as large as the beta error proceeds by setting `abratio = 100`.

```
cp2 <- semPower.compromise(effect = .08, effect.measure = 'RMSEA',
                           abratio = 100, N = 1000, df = 100)
summary(cp2)
#>
#>  semPower: Compromise power analysis
#>
#>  F0                        0.640000
#>  RMSEA                     0.080000
#>  Mc                        0.726149
#>
#>  df                        100
#>  Num Observations          1000
#>  Desired Alpha/Beta Ratio 100.000000
#>
#>  Critical Chi-Square       304.9642
#>  Implied Alpha             1.373729e-22
#>  Implied Beta              1.373729e-24
#>  Implied Power (1-beta)    > 0.9999
#>  Actual Alpha/Beta Ratio   100.000000
```



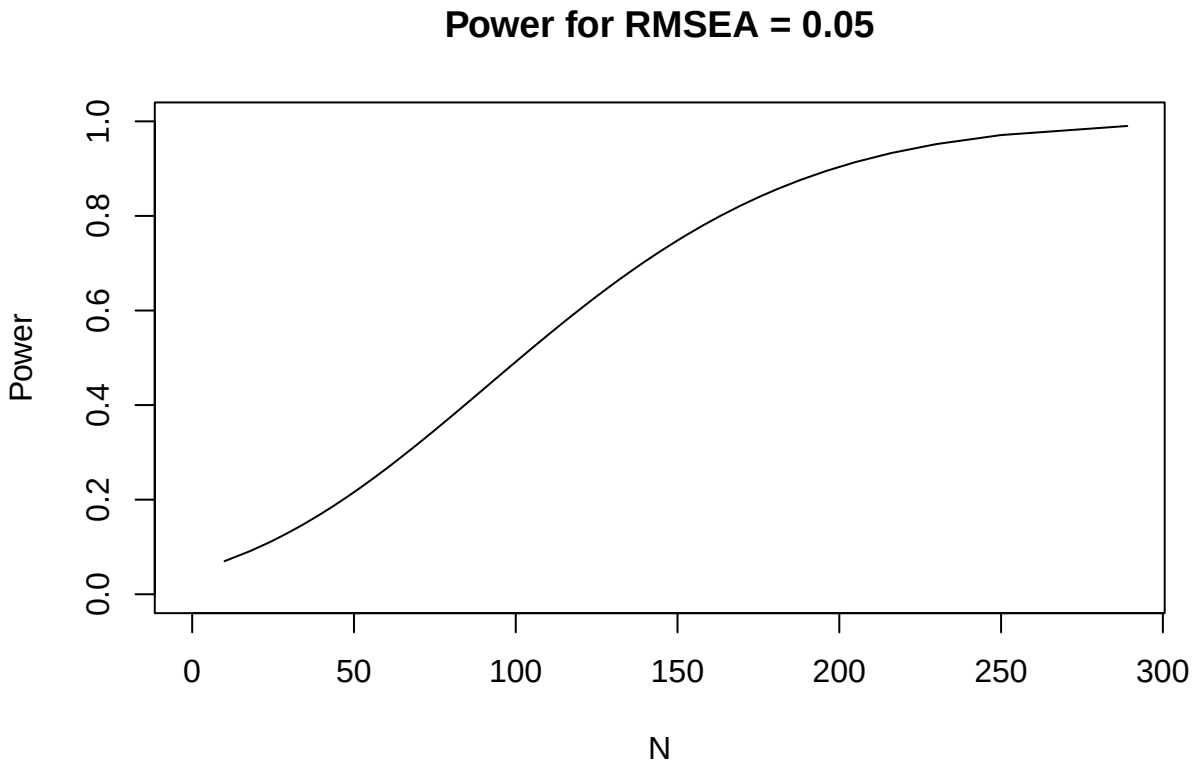
Power Plots

Power plots show the implied power as function of some other variable. `semPower` provides two different types of power plots. You can either plot the achieved power to detect a certain effect over a range of different sample sizes (`semPower.powerPlot.byN`). Alternatively, you can plot the achieved power with a given N to detect a range of different effect size magnitudes (`semPower.powerPlot.byEffect`).

Determine power as function of N for a given effect

The function `semPower.powerPlot.byN` creates a plot showing the achieved power to detect a given effect on a given alpha error over a range of sample sizes. However, because it is difficult to specify diagnostic sample sizes for a given effect, the `semPower.powerPlot.byN` instead asks to provide the desired power range. For example, suppose we are interested in how the power to detect an effect corresponding to RMSEA = .05 changes as function of N. We are interested in a power ranging from .05 to .99 (note that the power cannot become smaller than alpha). This is achieved by setting the arguments `power.min = .05` and `power.max = .99`. In addition, as in any a-priori power analysis, the type and magnitude of effect, the *df*, and the alpha error need to be defined: `effect = .05`, `effect.measure = 'RMSEA'`,`alpha = .05`, `df = 100`.

```
semPower.powerPlot.byN(effect = .05, effect.measure = 'RMSEA',
                       alpha = .05, df = 100, power.min = .05, power.max = .99)
```

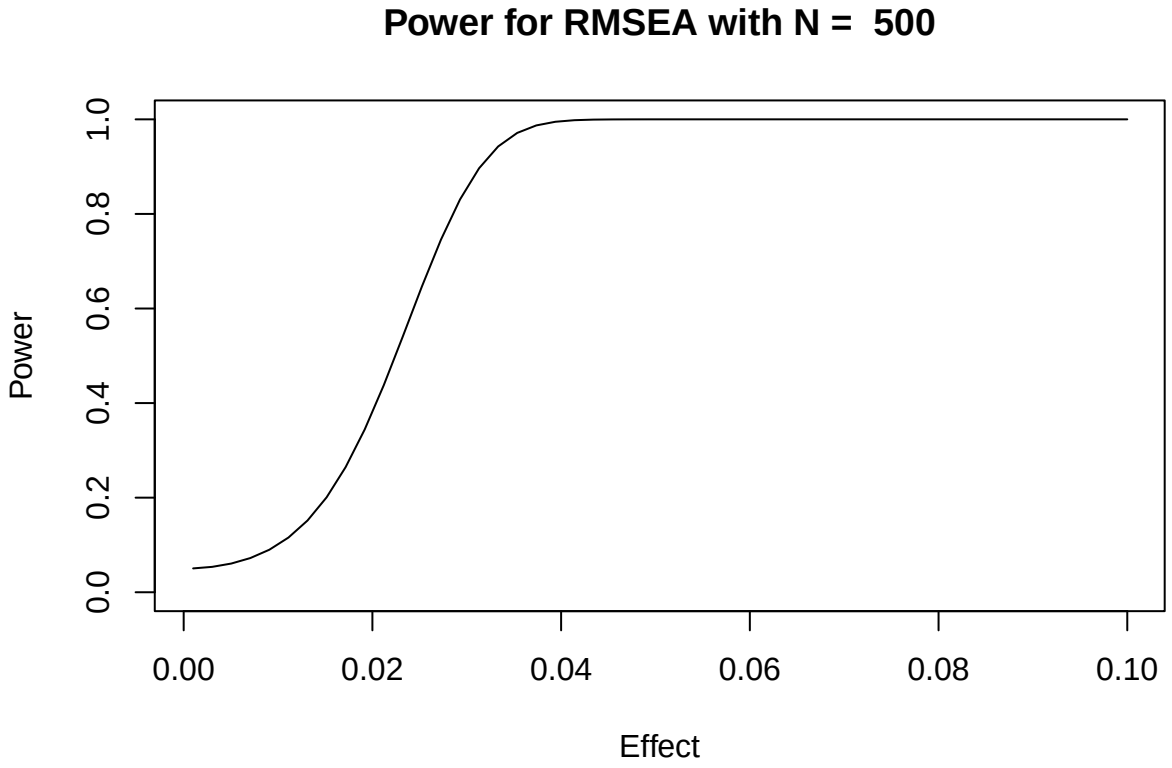


This shows that a model with an associated RMSEA = .05 is rejected with a very high power when $N > 250$, whereas power is small when $N < 100$.

Determine power as function of the magnitude of effect for a given N

The function `semPower.powerPlot.byEffect` creates a plot showing the achieved power at a given sample size over a range of effect sizes. For example, suppose we are interested in how the power at $N = 500$ changes as function of effect size magnitude, corresponding to an RMSEA ranging from .001 to .10. This is achieved by setting the arguments `effect.measure = 'RMSEA'`, `effect.min = .001` and `effect.max = .10`. In addition, as in any post-hoc power analysis, the sample size, the df, and the alpha error need to be defined: `effect = .05`, `effect.measure = 'RMSEA'`, `alpha = .05`, `df = 100`.

```
semPower.powerPlot.byEffect(effect.measure = 'RMSEA', alpha = .05, N = 500,
                            df = 100, effect.min = .001, effect.max = .10)
```



This shows that with $N = 500$, a model with an associated RMSEA $> .04$ is detected with a very high power, whereas power for RMSEA $< .03$ is rather modest.

Covariance Matrix input

The previous sections assumed that the magnitude of effect is determined by defining a certain effect size metric (such as F0 or RMSEA) and a certain magnitude of effect. Alternatively, the effect can also be determined by specifying the population (Σ) and the model-implied ($\hat{\Sigma}$) covariance matrices directly. To determine the associated effect in terms of F0, `semPower` just plugs these matrices in the ML-fitting function:

$$F_0 = \log|\Sigma| - \log|\hat{\Sigma}| + tr(\hat{\Sigma}\Sigma) - p$$

Suppose, Σ and $\hat{\Sigma}$ have been defined previously in the script and are referred to by the variables `Sigma` and `SigmaHat` Then, any of the power-analysis functions is called setting the `Sigma` and `SigmaHat` arguments accordingly (and omitting the `effect` and `effect.measures` arguments). This could look as follows:

```
semPower.aPriori(alpha = .05, power = .80, df = 100,
                 Sigma = Sigma, SigmaHat = SigmaHat)
semPower.postHoc(alpha = .05, N = 1000, df = 100,
                 Sigma = Sigma, SigmaHat = SigmaHat)
semPower.compromise(abratio = 1, N = 1000, df = 100,
                    Sigma = Sigma, SigmaHat = SigmaHat)
semPower.powerPlot.byN(alpha = .05, df = 100, power.min = .05, power.max = .99,
                       Sigma = Sigma, SigmaHat = SigmaHat)
```

This feature is particularly useful when used in conjunction with some other SEM software, which is used to generate the population and the model-implied covariance matrix (note that a proper definition of the latter usually requires fitting the model to the population data). For illustration, let's consider how this could be

done using `lavaan`. Note that the particular situation illustrated below is more conveniently implemented in the `sempower.powerCFA` function described further below. Also note that `sempower.powerCFA` function is sometimes handy to simplify the process of obtaining the relevant covariance matrices.

```
library(lavaan)

# define (true) population model
model.pop <- '
f1 =~ .8*x1 + .7*x2 + .6*x3
f2 =~ .7*x4 + .6*x5 + .5*x6
f1 ~~ 1*f1
f2 ~~ 1*f2
f1 ~~ 0.5*f2
'
# define (wrong) H0 model
model.h0 <- '
f1 =~ x1 + x2 + x3
f2 =~ x4 + x5 + x6
f1 ~~ 0*f2
'
```

After loading the `lavaan` package, two lavaan model strings are defined. The first model string (`model.pop`) is used to define the population covariance matrix, so all model parameters are defined (at least implicitly, consult the lavaan documentation for defaults). Here, we define a model comprising two latent factors (each with variance of 1), that are correlated by .5. Each latent factor is measured through three observed indicators (x1 to x6) with (unstandardized) loadings ranging from .5 to .8. The second model string (`model.h0`) is used to define the hypothesized (wrong) H0 model. To obtain the model-implied covariance matrix, we need to fit this model to the population data; so this is basically an ordinary CFA model string with many free parameters. Note that this model contraints the correlation between the two latent factors to zero. When fitting this model to the population data defined previously, the model is thus wrong, since we defined the correlation between these factors in the population to be .50.

Having defined the model strings, we proceed by actually obtaining the relevant covariance matrices.

```
# get population covariance matrix; equivalent to a perfectly fitting model
cov.pop <- fitted(sem(model.pop))$cov

# get covariance matrix as implied by H0 model
fit.h0 <- sem(model.h0, sample.cov = cov.pop,
              sample.nobs = 1000, sample.cov.rescale = F,
              likelihood='wishart')
df <- fit.h0@test[[1]]$df
cov.h0 <- fitted(fit.h0)$cov
```

`cov.pop` is now the covariance matrix in the population (Σ). To obtain the model implied covariance matrix ($\hat{\Sigma}$), we need to fit our hypothesized, wrong, H0 model (`model.h0`) to the population data (`cov.pop`). The model-implied covariance matrix then can be obtained by calling `cov.h0 <- fitted(fit.h0)$cov`.

We are now in the position to use the obtained covariance matrix in power analyses. Let's do a post-hoc power-analysis assuming $N = 1000$ and alpha =.05 by calling `semPower.postHoc` with the arguments `SigmaHat = cov.h0` and `Sigma = cov.pop`.

```
ph4 <- semPower.postHoc(SigmaHat = cov.h0, Sigma = cov.pop, alpha = .05, N = 1000, df = df)
summary(ph4)
```

The output (which is omitted here) indicates that fitting the hypothesized model to the population data is associated with a discrepancy of F0 = 0.081 (or RMSEA = .095 or SRMR = .139 or ...) and that the power

to reject the H1 model is very high, 1 - beta = 1 - 1.347826e-08.

It is instructive to compare the expected chi-square (calculated via the obtained F0) with the chi-square model test statistics as reported by `lavaan`:

```
fitmeasures(res.h0, 'chisq')
ph4$fmin * (ph4$N-1)
```

`fitmeasures(res.h0, 'chisq')` prints the model chi-square test as obtained by `lavaan` when fitting `model.h1` to the population data (`cov.pop`, see above). The line `ph4$fmin * (ph4$N-1)` computes the expected chi-square, i.e. F0 multiplied by (N - 1). Obviously, both values match (= 81.52).

Power for factor correlations in a standard CFA model

Given that obtaining the relevant covariance matrix to define an effect of interest can be rather cumbersome, `semPower` also provides the `semPower.powerCFA` function that simplifies this process for the common case that the correlation between two factors in a standard CFA model is of interest. In addition, this function can also be used as a helper function to obtain implied covariance matrices that can be plugged into the more general sempower commands (see below for an example). Note that this function requires the `lavaan` package.

Generally, `semPower.powerCFA` requires that a certain CFA model is defined in terms of the number of factors, loadings, and (population) correlations between the factors. One of these correlations defines the effect of interest. `semPower.powerCFA` then obtains the true population covariance matrix, fits a model contraining the chosen correlation to be zero, obtains the associated model-implied covariance matrix, and finally performs a power analysis on these matrices.

In this context, there are two relevant comparison models. By default (`comparison = 'restricted'`), the model restricting the correlation to zero is compared against the model that freely estimates this correlation. Alternatively (`comparison = 'saturated'`), one might also use the saturated model as relevant comparison model.

In the simplest case, only the number of indicators by factor (`nIndicator = c(3, 6)`), the factor correlation (`phi = .3`), and a single loading (`loadM = .6`) is provided to define a two factor CFA:

```
cfapower <- semPower.powerCFA(type = 'a-priori', comparison = 'restricted',
                              phi = .3, nIndicator = c(3, 6), loadM = .6,
                              alpha = .05, power = .80)
```

The result `cfapower` is a list comprising the results of the power analysis, the population and model-implied covariance matrices, and several lavaan model strings. To view the power output, call:

```
summary(cfapower$power)
```

To peek into the lavaan model strings, use:

```
# population model
cfapower$modelPop
# (incorrect) analysis model
cfapower$modelAna
```

When using sampled loadings (see below), view the generated loading matrix using

```
cfapower$lambda
```

It is a good idea to check whether everything was set up as intended by fitting the true model to the population covariance matrix:

```
summary(lavaan::sem(cfapower$modelTrue, sample.cov = cfapower$Sigma,
                    sample.nobs = 1000, likelihood = 'wishart', sample.cov.rescale = FALSE),
        stand = TRUE)
```

You can of course also plug the the population and model-implied covariance matrices into a power analysis:

```
ph <- semPower.aPriori(SigmaHat = cfapower$SigmaHat, Sigma = cfapower$Sigma,
                       df = 1, alpha = .05, beta = .05)
summary(ph)
```

Instead of just defining two factors with equal loadings, one may also define more than two factors with varying loadings. Then, `phi` becomes a factor correlation matrix. For example, let's define three factors with the following correlations:

```
# define correlation matrix between 3 factors
phi <- matrix(c(
  c(1.0, 0.3, 0.7),
  c(0.3, 1.0, 0.4),
  c(0.7, 0.4, 1.0)
), byrow = T, ncol=3)
```

Given that there are now more than two factors and thus more than a single factor correlation, we need to define on which of the factor correlations power analysis should be performed using the `nullCor` argument. For example, `nullCor = c(1, 2)` defines that we are interested in the correlation between the first and the second factor.

Having defined the factor correlations, we also need to define the number of indicators via `nIndicator` by factor and the factor loadings. Concerning the latter, we can have equal loadings for all indicators, define specific loadings for each indicator, or sample the loadings from a normal distribution with given mean and standard deviation or from a uniform distribution providing minimum and maximum loading:

```
# sample loadings from normal distribution using mean and sd; same for all factors
cfapower <- semPower.powerCFA(type = 'post-hoc',
                              phi = phi, nullCor = c(1, 2),
                              nIndicator = c(6, 5, 4), loadM = .5, loadSD = .1,
                              alpha = .05, N = 250)

# sample loadings from normal distribution using mean and sd for each factor
cfapower <- semPower.powerCFA(type = 'post-hoc',
                              phi = phi, nullCor = c(1, 2),
                              nIndicator = c(3, 6, 5),
                              loadM = c(.5, .6, .7), loadSD = c(.1, .05, 0),
                              alpha = .05, N = 250)

# sample loadings from uniform using min-max; same for all factors
cfapower <- semPower.powerCFA(type = 'post-hoc',
                              phi = phi, nullCor = c(1, 2),
                              nIndicator = c(6, 5, 4), loadMinMax = c(.3, .8),
                              alpha = .05, N = 250)

# sample loadings from uniform using min-max for each factor
loadMinMax <- list(
  c(.4, .6),
  c(.5, .8),
  c(.3, .7)
)
```

```
cfapower <- semPower.powerCFA(type = 'post-hoc',
                              phi = phi, nullCor = c(1, 2),
                              nIndicator = c(6, 5, 4), loadMinMax = loadMinMax,
                              alpha = .05, N = 250)
```

Among other things, `semPower.powerCFA` also returns the model-implied covariance matrices. We can therefore use this function as a helper function to obtain the relevant matrices, when we are actually interested in power concerning a parameter other than the factor correlation. For example, consider the case of a latent regression involving two predictors and the interest lies on the required sample to detect an increment in the R^2 of $\geq .01$ with a sufficient power. Given that `semPower.powerCFA` only computes power for the correlation between two factors, we cannot immediately use this function for these types of questions. However, we can use `semPower.powerCFA` to define factors with plausible intercorrelations and use the thereby obtained covariance matrix as input for any power analysis.

```
# define correlation matrix
# factors 1 and 2 act as predictors, 3 is the criterion
phi <- matrix(c(
  c(1.0, 0.5, 0.3),
  c(0.5, 1.0, 0.276),
  c(0.3, 0.276, 1.0)
), byrow = T, ncol=3)
# perform dummy power analysis
dummypower <- semPower.powerCFA(type = 'post-hoc',
                                phi = phi, nullCor = c(1, 2),
                                nIndicator = c(5, 4, 3),
                                loadM = .6,
                                alpha = .05, N = 1000)
# store implied covariance matrix
Sigma <- dummypower$Sigma
```

In the above, we first define the correlation matrix between three factors. Then we use the `semPower.powerCFA` command to define the factors in terms of the number of indicators and the loadings. Here, we are not interested in the computed power, but rather just want the generated covariance matrix of the observed variables, which we store the the variable `Sigma`.

By fitting two regression models to Sigma, we can confirm that the factor correlations realized above (phi) indeed correspond to an R^2 increase of about .01 in a latent regression model predicting the third factor:

```
# peek into model string as reference
dummypower$modelTrue

# define regression model
m <- '
f1 =~ x1 + x2 + x3 + x4 + x5
f2 =~ x6 + x7 + x8 + x9
f3 =~ x10 + x11 + x12
f3 ~ f1 + f2
'
# define restricted regression model
mr <- '
f1 =~ x1 + x2 + x3 + x4 + x5
f2 =~ x6 + x7 + x8 + x9
f3 =~ x10 + x11 + x12
f3 ~ f1 + 0*f2
'
```

```
# fit both models to Sigma and store R^2 concerning f3
res <- sem(m, sample.cov = Sigma, sample.nobs = 1000, sample.cov.rescale = FALSE)
par <- parameterestimates(res, standardized = TRUE)
R2 <- 1 - par[par$lhs == 'f3' & par$op == '~~' & par$rhs == 'f3', 'std.all']

resr <- sem(mr, sample.cov = Sigma, sample.nobs = 1000, sample.cov.rescale = FALSE)
par <- parameterestimates(resr, standardized = TRUE)
R2r <- 1 - par[par$lhs == 'f3' & par$op == '~~' & par$rhs == 'f3', 'std.all']

# verify that R^2 increase is about .01
R2 - R2r
```

Using the model-implied covariance matrix associated with the restricted regression model, we are now in the position to perform any type of power analysis. For example, to obtain the required number of observations to detect an R^2 increase of $\geq .01$ in the situation defined above with a power of 80% on alpha = .05, we can use

```
# obtain model-implied covariance matrix
SigmaHat <- fitted(resr)$cov

# do power analysis
ap <- semPower.aPriori(alpha = .05, power = .80, df = 1,
                       Sigma = Sigma, SigmaHat = SigmaHat)
summary(ap)
```

which shows that 977 observations are required.

References

- Browne, M. W., & Cudeck, R. (1992). Alternative ways of assessing model fit. *Sociological Methods & Research, 21*, 230–258.
- Jöreskog, K. G., & Sörbom, D. (1984). *LISREL VI user's guide* (3rd ed.). Mooresville: Scientific Software.
- McDonald, R. P. (1989). An index of goodness-of-fit based on noncentrality. *Journal of Classification, 6*, 97–103.
- 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.
- Moshagen, M., & Erdfelder, E. (2016). A new strategy for testing structural equation models. *Structural Equation Modeling, 23*, 54–60.
- Steiger, J. H. (1990). Structural model evaluation and modification: An interval estimation approach. *Multivariate Behavioral Research, 25*, 173–180.
- Steiger, J. H., & Lind, J. C. (1980). *Statistically based tests for the number of common factors.* Presented at the Annual meeting of the Psychometric Society, Iowa City.