Intro to Power Analysis Using Simulation Methods

Jessie Sun

14/11/2018

This is a very basic introduction to conducting power analyses based on simulations, using the *lavaan* package.

Types of Power Analysis

1. *A priori power analysis:* What is the smallest sample size we need to have 80% power to detect an effect size of interest (e.g. $\beta = 0.20$, $\beta = 0.50$), at an alpha level of .05?
2. *Sensitivity power analysis:* What is the smallest effect size we can detect with 80% power, given our sample size, at an alpha level of .05?
3. *Post-hoc power analysis:* What power did we have to detect the observed effect size, given the sample size actually used, at an alpha level of .05?

As Daniel Lakens has explained elsewhere, observed power (from a post-hoc power analysis) is a useless statistical concept. Thus, this tutorial will focus on a priori and sensitivity power analyses.

(Note: You can also set different alpha thresholds and power goals, but to keep things simple, let's assume the standard .05 alpha threshold and the goal of 80% power.)

The Model

As shown in Figure 1 below, we will be considering a fairly simple model, with two predictor variables X{1} and X{2}, and one outcome variable, Y. All variables are observed.

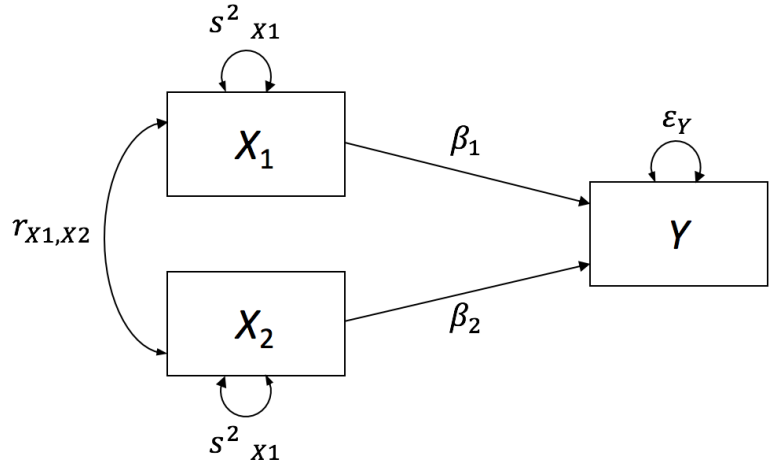

Figure 1. Labelled multiple regression model.

Our Goal

Our goal in this tutorial is to conduct a priori and sensitivity power analyses for the regression path for Y on X_1 (β_1).

Model Assumptions

This model includes the following parameters:

-Variance of X_1 (s^2_{X1})

-Variance of X_2 (s^2_{X2})

-Covariance between X1 and X2 ($cov_{X1,X2}$)

-Regression path for Y on X1 (β_1)

-Regression path for Y on X2 (β_2)

-Residual variance of Y (ϵ_Y)

In this case, to conduct power analysis based on standardized effect sizes, we will:

1. Fix the variance of X_1 and X_2 to 1.
2. Assume that Y has a variance of 1, such that the residual variance ϵY (i.e., the variance not explained by X_1 and X_2) will be equal to 1 - (${\beta_1}^2$ + ${\beta_2}^2$).

Note that since we are using a standardized metric, $cov_{X1,X2}$ is now simply the correlation between X_1 and X_2 (i.e., $r_{X1,X2}$).

To simulate data based on a **population model**, we need to make assumptions about each of the other parameters. Some of these assumptions might be based on existing data (e.g., you might already know how strongly X_1 and X_2 tend to be correlated in the literature), but at other times, they might seem somewhat arbitrary. Because of the potential arbitrariness of our assumptions, it is useful to simulate power under a range of different assumptions (e.g., what if the correlation between X_1 and X_2 was stronger, or if the regression path for β_2 was larger or smaller?).

However, for the purposes of this tutorial, let's assume that X_1 and X_2 are moderately positively correlated ($r = .30$), and that X_2 positively predicts Y to a small extent ($\beta_2 = 0.10$).

Figure 2 illustrates these assumptions.

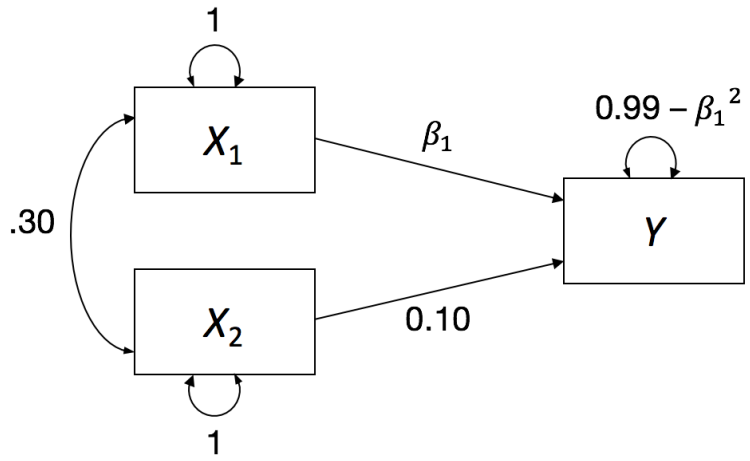

Figure 2. Labelled multiple regression model with model assumptions.

A Priori Power Analysis

Let's get started with an a priori power analysis. What is the smallest sample size we need to have 80% power to detect an effect size of $\beta_1 = 0.20$, at an alpha level of .05?

First, we need to load the *lavaan* package

```
library(lavaan)
```

```
## This is lavaan 0.5-22
```

```
## lavaan is BETA software! Please report any bugs.
```

Next, we need to specify the population model, based on the assumptions in Figure 2, plus our effect size of interest ($\beta_1 = 0.20$). This is the model that, at the population level, we assume is generating the data that we might see in any given dataset.

Basic *lavaan* notation: a double ~~ denotes variances and covariances, whereas a single ~ denotes a regression path.

```
popmod1 <- '
# variances of X1 and X2 are fixed at 1
x1~~1*x1
x2~~1*x2

# correlation between X1 and X2 is assumed to be .30
x1~~.3*x2

# regression path for Y on X1 is assumed to be .10
y~.10*x1

# regression path of interest, Y on X2, is assumed to be .20
y~.20*x2

# residual variance of Y is 1 - (.1^2 + .2^2) = .95
y~~.95*y
'
```

We also need to create another *lavaan* model, without those population-level assumptions.

```
fitmod <- '
# variances of X1 and X2
x1~~x1
x2~~x2

# correlation between X1 and X2
x1~~x2

# regression path for Y on X2
y~x1

# regression path of interest, Y on X2
y~x2

# residual variance of Y
y~~y
'
```

To see the logic of the simulation process, let's first just simulate one dataset based on the population model, popmod1.

```
set.seed(20181102)  # setting a seed for reproducibility of the example
data <- simulateData(popmod1, sample.nobs = 500)  # assume 500 participants for now
```

Now, we're going to fit our model (fitmod) to this dataset.

```
fit <- sem(model = fitmod, data=data, fixed.x=F)
```

Here are the parameter estimates. The parameter of interest, y ~ x1, is in row 4. As you can see, this parameter was statistically significant ($p = .012$) in this simulation based on one dataset.

```
parameterEstimates(fit)  # see all parameters
```

```
##   lhs op rhs   est    se      z pvalue ci.lower ci.upper
## 1  x1 ~~  x1 0.951 0.060 15.811  0.000    0.833    1.068
## 2  x2 ~~  x2 1.046 0.066 15.811  0.000    0.916    1.176
## 3  x1 ~~  x2 0.310 0.047  6.634  0.000    0.218    0.401
## 4   y  ~  x1 0.110 0.050  2.192  0.028    0.012    0.209
## 5   y  ~  x2 0.168 0.048  3.514  0.000    0.074    0.262
## 6   y ~~   y 1.084 0.069 15.811  0.000    0.950    1.218
```

```
parameterEstimates(fit)[4,]  # isolating the row with the parameter of interest
```

```
##   lhs op rhs  est   se     z pvalue ci.lower ci.upper
## 4   y  ~  x1 0.11 0.05 2.192  0.028    0.012    0.209
```

However, to estimate power, we need to simulate many datasets. Then, we can obtain the % of datasets in which the parameter of interest is statistically significant. This is our power estimate.

So, let's go ahead and simulate 1000 datasets, still assuming a sample size of 500.

```
results <- NULL  # create empty object to store results

for (i in 1:1000){  # simulating 1000 datasets
  data <- simulateData(popmod1, sample.nobs = 500)  # each dataset contains 500 participants
  fit <- sem(model = fitmod, data=data, fixed.x=F)  # fit the model to each dataset
  results <- rbind(results,parameterEstimates(fit)[4,])  # save the row for y ~ x1 for each dataset
}

# Count the proportion of significant parameter estimates out of 1000 datasets
paste0('Estimated power = ',mean(results$pvalue < .05))
```

```
## [1] "Estimated power = 0.609"
```

As you can see, power in this example was pretty subpar; we were only able to detect the effect of interest in 60.9% of those 1,000 simulations.

Now, let's simulate some results based on different sample sizes. It's a good idea to start with rough increments (e.g., $N = 600, 800, 1000$) and fewer datasets (e.g., 100) to save computing time. Then once you know which ballpark to aim for, re-run the simulations based on narrower increments (e.g., increments of $N = 10$) and more datasets (e.g., 1000) to get a more exact estimate of sample requirements.

```
# now that we are trying a few different sample sizes, we need to create a list to store these results
powerlist <- list()

# extending the for() loop with a loop for different sample sizes
for(j in seq(600,1200,by=200)){  # trying sample sizes of 600, 800, 1000, 1200
  results <- NULL
  for (i in 1:100){  # starting with 100 datasets for each sample size
    data <- simulateData(popmod1, sample.nobs = j)  # j corresponds to the sample size
    fit <- sem(model = fitmod, data=data, fixed.x=F)
    results <- rbind(results,parameterEstimates(fit)[4,])  # row for y ~ x1
    powerlist[[j]] <- mean(results$pvalue < .05)
```

```
  }
}

# Convert the power list into a table
library(plyr)
powertable <- ldply(powerlist)
names(powertable)[1] <- c('power')

# Add a column for the sample size
powertable$N <- seq(600,1200,by=200)

# Here are all the power estimates:
powertable
```

```
##   power    N
## 1  0.70  600
## 2  0.75  800
## 3  0.89 1000
## 4  0.92 1200
```

```
# Conclusion:
paste0('The smallest sample size that provided at least 80% power was N = ',
       powertable[which(powertable$power>.80),'N'][1])
```

```
## [1] "The smallest sample size that provided at least 80% power was N = 1000"
```

Based on 100 simulations, it seems like the ballpark for 80% power is between 800 and 900 participants. So now let's re-run the simulations, but in increments of N = 10, and with 1000 simulations each.

```
# create a list to store these results for different sample sizes
powerlist <- list()

# extending the for() loop with a loop for different sample sizes
for(j in seq(800,900,by=10)){  # trying sample sizes between 800 to 900, in increments of 10
  results <- NULL
  for (i in 1:1000){  # now simulating 1000 datasets for each sample size
    data <- simulateData(popmod1, sample.nobs = j)  # j corresponds to the sample size
    fit <- sem(model = fitmod, data=data, fixed.x=F)
    results <- rbind(results,parameterEstimates(fit)[4,])  # row for y ~ x1
    powerlist[[j]] <- mean(results$pvalue < .05)
  }
}

# Convert the power list into a table
powertable <- ldply(powerlist)
names(powertable)[1] <- c('power')

# Add a column for the sample size
powertable$N <- seq(800,900,by=10)

# Here are all the power estimates:
powertable
```

```
##    power   N
## 1  0.770 800
## 2  0.797 810
```

```
## 3  0.781 820
## 4  0.812 830
## 5  0.812 840
## 6  0.805 850
## 7  0.814 860
## 8  0.841 870
## 9  0.820 880
## 10 0.835 890
## 11 0.838 900
```

```
# Conclusion:
paste0('The smallest sample size that provided at least 80% power was N = ',
       powertable[which(powertable$power>.80),'N'][1])
```

```
## [1] "The smallest sample size that provided at least 80% power was N = 830"
```

Try it Out: Alternative Effect Sizes

In this example, we have assumed that we are interested in detecting an effect size at least as great as $\beta_1 = 0.20$. But of course, you might be interested in detecting smaller larger effect sizes. Can you try adapting the population model, and re-run the simulations for different effect sizes?

Sensitivity Power Analysis

Instead of specifying an **effect size of interest**, you might already have a **predefined sample size** (e.g., due to time and financial constraints, or already-collected data). In this case, you can ask the question: What is the smallest effect size we can detect with at least 80% power, given our sample size? Let's assume that we have a sample size of $N = 500$.

This time, we'll need to generate a series of population models with different effect sizes. Just as before, it's a good idea to start with larger increments (e.g., effect sizes in increments of .10, from .20 to .80) and fewer simulations, then refine the simulations in a second round.

```
# First, we need to create a template for the population model
popmodtemplate <- '
# variances of X1 and X2 are fixed at 1
x1~~1*x1
x2~~1*x2

# correlation between X1 and X2 is assumed to be .30
x1~~.3*x2

# regression path for Y on X1 is assumed to be .10
y~.10*x1

# regression path of interest, Y on X2, will be varied
y~beta2*x2  # we will be substituting different effect sizes into beta2

# residual variance of Y
y~~resy*y  # we will be substituting different effect sizes into res, depending on beta2
'

# Use this template to generate a series of population model
# syntaxes with varying sizes of beta2
```

```
popmodlist <- list()

for(i in seq(0.20,0.80,by=0.10)){
  popmodlist[[paste0(i)]] <- gsub('beta2',i,popmodtemplate)
  popmodlist[[paste0(i)]] <- gsub('resy',paste0(1-(.1^2+i^2)),popmodlist[[paste0(i)]])
}
```

Now that we have our initial list of population models, we can run the first round of simulations to get our ballpark estimates.

```
# create a list to store power estimates for each effect size
powerlist <- list()

# this time, the outside for() loop is for each of the different population models/effect sizes
for(j in names(popmodlist)){
  results <- NULL
  for (i in 1:100){  # starting with 100 datasets for each effect size
    data <- simulateData(popmodlist[[j]],  # for a given population model
                         sample.nobs = 500)  # assuming a sample size of 500
    fit <- sem(model = fitmod, data=data, fixed.x=F)  # fitmod is the same as for the a priori power an
    results <- rbind(results,parameterEstimates(fit)[4,])  # row for y ~ x1
    powerlist[[j]] <- mean(results$pvalue < .05)
  }
}

# Convert the power list into a table
powertable <- ldply(powerlist)
names(powertable) <- c('beta2','power')

# Here are all the power estimates:
powertable
```

```
##   beta2 power
## 1   0.2  0.55
## 2   0.3  0.68
## 3   0.4  0.67
## 4   0.5  0.70
## 5   0.6  0.77
## 6   0.7  0.84
## 7   0.8  0.96
```

```
# Conclusion:
paste0('The smallest effect size that could be detected with at least 80% power was beta = ',
       powertable[which(powertable$power>.80),'beta2'][1])
```

```
## [1] "The smallest effect size that could be detected with at least 80% power was beta = 0.7"
```

Now that we know the ballpark is between $\beta_2 = 0.60$ and $\beta_2 = 0.70$, we can repeat the process with narrower increments, and more simulations:

```
# create population models
popmodlist <- list()

for(i in seq(0.60,0.70,by=0.01)){
  popmodlist[[paste0(i)]] <- gsub('beta2',i,popmodtemplate)
  popmodlist[[paste0(i)]] <- gsub('resy',paste0(1-(.1^2+i^2)),popmodlist[[paste0(i)]])
```

```
}

# create a list to store power estimates for each effect size
powerlist <- list()

# this time, the outside for() loop is for each of the different population models/effect sizes
for(j in names(popmodlist)){
  results <- NULL
  for (i in 1:1000){  # 1000 simulations for each model
    data <- simulateData(popmodlist[[j]],  # for a given population model
                         sample.nobs = 500)  # assuming a sample size of 500
    fit <- sem(model = fitmod, data=data, fixed.x=F)  # fitmod is the same as for the a priori power an
    results <- rbind(results,parameterEstimates(fit)[4,])  # row for y ~ x1
    powerlist[[j]] <- mean(results$pvalue < .05)
  }
}

# Convert the power list into a table
powertable <- ldply(powerlist)
names(powertable) <- c('beta2','power')

# Here are all the power estimates:
powertable
```

```
##    beta2 power
## 1    0.6 0.747
## 2   0.61 0.787
## 3   0.62 0.780
## 4   0.63 0.801
## 5   0.64 0.767
## 6   0.65 0.800
## 7   0.66 0.824
## 8   0.67 0.814
## 9   0.68 0.806
## 10  0.69 0.857
## 11   0.7 0.836
```

```
# Conclusion:
paste0('The smallest effect size that could be detected with at least 80% power was beta = ',
       powertable[which(powertable$power>.80),'beta2'][1])
```

```
## [1] "The smallest effect size that could be detected with at least 80% power was beta = 0.63"
```

Try it Out: Alternative Sample Sizes

In this example, we have assumed that we only have 500 participants. Can you try adapting the code to run sensitivity power analyses assuming a different sample size (e.g., 300 participants)?

Contact

Feel free to use and adapt for teaching and research purposes (with attribution), and please get in touch (jesun@ucdavis.edu) if you spot any errors!