Simulating large-scale assessment data using the R package simsem

Alexander M. Schoemann[1] Terrence D. Jorgensen[2]

[1]East Carolina University

[2]University of Amsterdam

NCME 2023

Outline

- `simsem` basics
- `simsem` functions
- Useful features for Large Scale Assessments

Why Use the `simsem` Package?

Flexibility, special features, and *some* automated results

- Generate data using
  - `lavaan`, `OpenMx`, or any custom function
  - a fitted model (estimates treated as parameters)
- Analyze data using
  - `lavaan`, `OpenMx`, or any custom function
  - a fitted model
- Impose missing data using
  - MCAR mechanisms (arbitrary % or planned missing-data designs)
  - MAR mechanisms (specify a logit model)
- Analyze missing data using FIML or multiple imputation

Why Use the simsem Package?

- Fixed or random exogenous predictors
- Fixed or random parameters
- Non-normal latent or observed variable generation
- Built-in options for parallel processing
- Automatically compute bias, coverage, and rejection probability
- Automatically find minimum N for given power

See list of features (and comparison of approaches) along with many `simsem` examples on the Vignettes web page

simsem Functions and Arguments

3 functions at the core of `simsem` work-flow:

- `generate()` a single sample set using arguments:
  - `model=` `SimSem` or `MxModel` object, `lavaan` syntax or `parTable()`,
  - alternatively, treat `realData=` as a population
  - `n=` the sample size(s)
- `analyze()` a single sample using arguments:
  - `model=` a `SimSem` or `MxModel` object
  - fitted to simulated or real sample `data=`
- `sim()` runs a Monte Carlo simulation
  - calls `generate()` using the `generate=` argument
    - alternatively, use `rawData=` as a population
    - or `rawData=` list of already generated samples
  - `model=` argument calls `lavaan` function, custom function, or sends a `SimSem` or `MxModel` object to `analyze()`
  - `nRep=` Monte Carlo sample size
  - `n=` size(s) of simulated samples

`simsem` for LSA

`lavaan` can analyze LSA data with the following features:

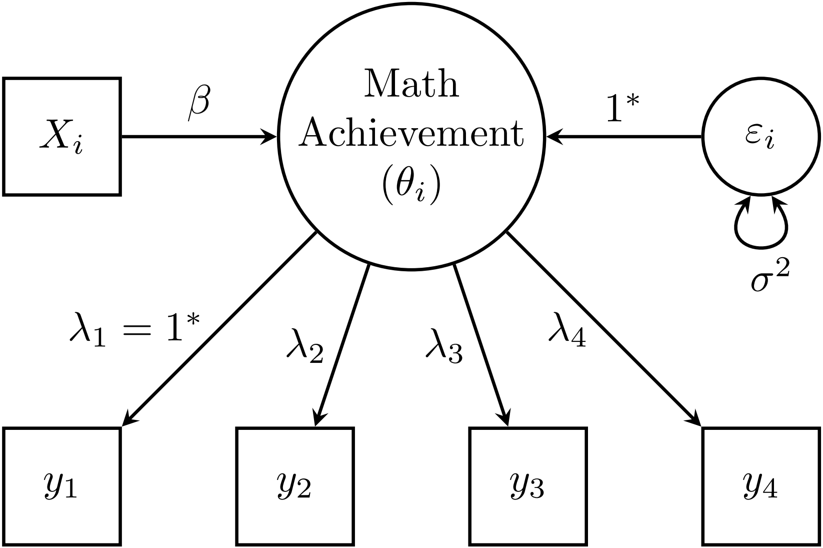
- probability weights, specified by passing a variable name to the `sampling.weights=` argument
- cluster-robust *SE*s and test statistics, specified by passing a variable name to the `cluster=` argument
- incomplete data using full-information maximum likelihood (FIML)

Strata require using the `lavaan.survey` package

- FIML not implemented, only multiple imputation
- no threshold models for binary/ordinal data
- need to write a custom data-analysis function for `sim(model=)`

The `sim()` function can use **any other software** (provided it can be accessed via R) to `generate=` data or to fit an analysis `model=`

Example model



Generate Data

For our first example we will generate data based on a stratified sample with normally distributed indicators

- This example will use `lavaan` syntax
- This example appears in Ch. 1 of our vignette

Generate Data

```
pop.mod.norm <-'
  Math =~ 1?y1
  Math =~ 0.887?y2
  Math =~ 1.151?y3
  Math =~ 0.546?y4

  Math ~ 0.3?X
  X ~~ 1*X
  Math ~~ 0.91*Math

  y1 ~~ 1?y1
  y2 ~~ 0.9?y2
  y3 ~~ 1.1?y3
  y4 ~~ 1?y4

## Intercepts vary across 5 strata (groups)
  y1 ~ c(0.47, -1, -0.22, -0.08, -0.22)*1
  y2 ~ c(-0.2, -0.64, 0.9, -0.09, -0.35)*1
  y3 ~ c(0.93, 0.41, 0.29, -0.22, 0.4)*1
  y4 ~ c(0.09, -0.55, -0.03, 0.59, -0.99)*1
'
```

Generate Data

```
(dat.strat   <- generate(model = pop.mod.norm, n = c(4, 3, 6, 3, 5)))
```

```
##              y1          y2         y3          y4           X group
## 1  -1.345886114 -3.63350123 -2.9261717 -1.41885263 -1.39362861     1
## 2   0.375690436 -1.28657814 -0.1746688 -0.05552251  0.81310890     1
## 3   1.150926478  0.11276272 -0.3058755  1.14535676  0.35333109     1
## 4  -0.937650240 -1.42818776  0.6006028 -1.58661073  1.34848323     1
## 5  -0.004012185 -0.38632054  1.6710407 -0.33885701  1.93844232     2
## 6  -2.030393210 -1.23488644 -0.4884841 -0.67670048 -0.02792024     2
## 7  -0.565226984 -2.12176357 -0.7375503 -1.14480156  1.11973044     2
## 8  -0.125053794  1.36717232  0.6691177 -0.09382128  1.35193683     3
## 9  -0.517816078  0.17603247 -1.1011250 -0.50129847  1.69570154     3
## 10  0.418985503  2.77343138  1.8810291  0.04701581  0.18111756     3
## 11  1.264339300  0.09974456 -0.5141094 -1.19951226  0.94944697     3
## 12  0.826409117  1.30124666  1.2033360  0.14398790 -0.94815075     3
## 13  0.953197113  2.34542159  1.9969393  1.53216606 -0.16890222     3
## 14  0.373251495  1.59026487  1.3452098 -0.61189951  0.35596044     4
## 15 -2.024190497  0.53828676 -0.9886752 -0.14996824 -0.38869066     4
## 16 -1.222754428 -0.90789354 -3.4865884  0.92652258 -0.44073259     4
## 17  0.315091626  0.45842338 -1.0823535 -1.04877394  0.32235479     5
## 18  0.602620419 -1.54495013  1.6251355 -0.64470240 -2.33025257     5
## 19 -0.184830195 -0.14320568 -0.2998273 -1.74556897 -0.60891771     5
## 20  0.415370560 -1.31046857 -0.4452117 -1.87579650 -3.06590590     5
## 21  0.072963564 -0.79539561  0.9433763 -1.19641716  0.41491223     5
```

Analyze Data

```
library(lavaan.survey)
mod <- '
  Math =~ 1?y1
  Math =~ 0.887?y2
  Math =~ 1.151?y3
  Math =~ 0.546?y4

  Math ~ 0.3?X

  y1 ~~ 1?y1
  y2 ~~ 0.9?y2
  y3 ~~ 1.1?y3
  y4 ~~ 1?y4
'
fit.naive <- sem(mod, data = dat.strat, meanstructure=TRUE)
myDesign <- svydesign(strata = ~group, ids = ~1, weights = ~1,
                      data = dat.strat)
fit.svy <- lavaan.survey(fit.naive, survey.design = myDesign)
summary(fit.svy) # robust SEs, robust test in "Scaled" column
```

Missing Data Features in `simsem`

- Missing data patterns and mechanisms can be specified via the `miss` function
  - Global MCAR missingness can also be specified in the `sim` function
  - MCAR, MAR, planned missingness, and longitudinal attrition are automated in the function
- Example with MCAR missing for y1 and y2, and MAR missing for y3 (with y 4 as an auxiliary variable)
- The missing data object created by the `miss` function can then be passed to the `sim(miss=)` argument

Specify Missing-Data

This example appears in Ch. 2 of our vignette

```
missMech <- '## MCAR for first 2 indicators
  y1 ~ p(.1)  # 10% chance of missing data
  y2 ~ -2     # logit-scale intercept implies 12% chance
## MAR for third indicator, explained by fourth indicator
  y3 ~ p(.1) + -0.2*y4    # 22% decrease in odds
'
myMissingnessModel <- miss(logit = missMech)
```

Include Missingness in Simulation

This example appears in Ch. 2 of our vignette

```
mis2L <- sim(seed = 12345, nRep = 10, n = 100, # Level-2 N
             generate = gen2L,      # function to generate
        miss = myMissingnessModel, # NEW: impose missing mechanism(
             model = mod.clus,      # lavaan script to analyze
             lavaanfun = "sem",     # use sem() function
             cluster = "schoolID", # argument passed to sem()
             silent = TRUE)
```

Other simsem functionality

- Analyze missing data with multiple imputations
  - Automated in the `miss()` function or write a custom function
- Custom functions for data transformation, analysis, and processing of results
  - Plausible values can be generated with a custom function for data transformation and analyzed with a separate custom function

Find these annotated examples in our vignette:

- Jorgensen, T. D., & Schoemann, A. M. (2023). *How the R package simsem can be useful in simulation studies of large-scale assessments*. Retrieved from https://psyarxiv.com/pxsrt

Thank you!

- Materials from today available in our vignette on the PsyArXiv:
  - https://psyarxiv.com/pxsrt
- Many more examples are available at simsem.org
- email: schoemanna@ecu.edu