semmcci: Monte Carlo Confidence Intervals

Ivan Jacob Agaloos Pesigan

Installation

You can install the CRAN release of `semmcci` with:

```
install.packages("semmcci")
```

You can install the development version of `semmcci` from GitHub with:

```
install.packages("remotes")
remotes::install_github("jeksterslab/semmcci")
```

Documentation

See GitHub Pages for package documentation.

Description

In the Monte Carlo method, a sampling distribution of parameter estimates is generated from the multivariate normal distribution using the parameter estimates and the sampling variance-covariance matrix. Confidence intervals for defined parameters are generated by obtaining percentiles corresponding to $100(1 - \alpha)\%$ from the generated sampling distribution, where α is the significance level.

Monte Carlo confidence intervals for free and defined parameters in models fitted in the structural equation modeling package `lavaan` can be generated using the `semmcci` package. The package has two main functions, namely, `MC()` and `MCStd()`. The output of `lavaan` is passed as the first argument to the `MC()` function to generate Monte Carlo confidence intervals. Monte Carlo confidence intervals for the standardized estimates can also be generated by passing the output of the `MC()` function to the `MCStd()` function.

Example

A common application of the Monte Carlo method is to generate confidence intervals for the indirect effect. In the simple mediation model, variable `X` has an effect on variable `Y`, through a mediating variable `M`. This mediating or indirect effect is a product of path coefficients from the fitted model.

```
library(semmcci)
library(lavaan)
```

Data

```
n <- 1000
X <- rnorm(n = n)
M <- 0.50 * X + rnorm(n = n)
Y <- 0.25 * X + 0.50 * M + rnorm(n = n)
data <- data.frame(X, M, Y)
```

Model Specification

The indirect effect is defined by the product of the slopes of paths `X` to `M` labeled as `a` and `M` to `Y` labeled as `b`. In this example, we are interested in the confidence intervals of `indirect` defined as the product of `a` and `b` using the `:=` operator in the `lavaan` model syntax.

```
model <- "
  Y ~ cp * X + b * M
  M ~ a * X
  indirect := a * b
  direct := cp
  total := cp + (a * b)
"
```

Model Fitting

We can now fit the model using the `sem()` function from `lavaan`.

```
fit <- sem(data = data, model = model)
```

Monte Carlo Confidence Intervals

The `fit` `lavaan` object can then be passed to the `MC()` function to generate Monte Carlo confidence intervals.

```
MC(fit, R = 20000L, alpha = c(0.001, 0.01, 0.05))

#> Monte Carlo Confidence Intervals
#>             est      se     R   0.05%    0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.1717 0.03582 20000 0.05741 0.07807 0.1016 0.2417 0.2630 0.2893
#> b        0.5512 0.03331 20000 0.43985 0.46538 0.4869 0.6163 0.6375 0.6647
#> a        0.4595 0.03084 20000 0.35783 0.38026 0.3991 0.5204 0.5393 0.5596
#> Y~~Y     1.0387 0.04658 20000 0.88502 0.91552 0.9469 1.1294 1.1574 1.1913
#> M~~M     0.9289 0.04164 20000 0.79415 0.82246 0.8473 1.0108 1.0352 1.0671
#> indirect 0.2533 0.02292 20000 0.18326 0.19659 0.2099 0.3000 0.3162 0.3337
#> direct   0.1717 0.03582 20000 0.05741 0.07807 0.1016 0.2417 0.2630 0.2893
```

```
#> total    0.4250 0.03681 20000 0.30624 0.33150 0.3526 0.4972 0.5198 0.5414
```

Standardized Monte Carlo Confidence Intervals

Standardized Monte Carlo Confidence intervals can be generated by passing the result of the `MC()` function to `MCStd()`.

Note: We recommend setting `fixed.x = FALSE` when generating standardized estimates and confidence intervals to model the variances and covariances of the predictors if they are assumed to be random.

```
fit <- sem(data = data, model = model, fixed.x = FALSE)
unstd <- MC(fit, R = 20000L, alpha = c(0.001, 0.01, 0.05))
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.1389 0.0290 20000 0.0489 0.0659 0.0820 0.1952 0.2116 0.2305
#> b        0.4800 0.0262 20000 0.3919 0.4111 0.4278 0.5311 0.5460 0.5630
#> a        0.4270 0.0259 20000 0.3383 0.3589 0.3752 0.4771 0.4933 0.5115
#> Y~~Y     0.6933 0.0243 20000 0.6096 0.6282 0.6439 0.7395 0.7532 0.7668
#> M~~M     0.8176 0.0221 20000 0.7383 0.7566 0.7724 0.8593 0.8712 0.8855
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2050 0.0171 20000 0.1523 0.1629 0.1719 0.2388 0.2499 0.2633
#> direct   0.1389 0.0290 20000 0.0489 0.0659 0.0820 0.1952 0.2116 0.2305
#> total    0.3439 0.0281 20000 0.2527 0.2701 0.2882 0.3982 0.4150 0.4328
```

References

MacKinnon, D. P., Lockwood, C. M., & Williams, J. (2004). Confidence limits for the indirect effect: Distribution of the product and resampling methods. *Multivariate Behavioral Research*, *39*(1), 99–128. https://doi.org/10.1207/s15327906mbr3901_4

Preacher, K. J., & Selig, J. P. (2012). Advantages of Monte Carlo confidence intervals for indirect effects. *Communication Methods and Measures*, *6*(2), 77–98. https://doi.org/10.1080/19312458.2012.679848

R Core Team. (2022). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/

Tofighi, D., & Kelley, K. (2019). Indirect effects in sequential mediation models: Evaluating methods for hypothesis testing and confidence interval formation. *Multivariate Behavioral Research*, *55*(2), 188–210. https://doi.org/10.1080/00273171.2019.1618545

Tofighi, D., & MacKinnon, D. P. (2015). Monte Carlo confidence intervals for complex functions of indirect effects. *Structural Equation Modeling: A Multidisciplinary Journal*, *23*(2), 194–205. https://doi.org/10.1080/10705511.2015.1057284