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       0.2597 0.0357 20000 0.1420 0.1655 0.1895 0.3293 0.3529 0.3785
#> b        0.4915 0.0333 20000 0.3860 0.4061 0.4265 0.5570 0.5766 0.5986
#> a        0.4825 0.0298 20000 0.3819 0.4052 0.4249 0.5413 0.5600 0.5807
#> Y~~Y     1.0653 0.0478 20000 0.9118 0.9432 0.9713 1.1591 1.1881 1.2254
#> M~~M     0.9525 0.0426 20000 0.8139 0.8445 0.8689 1.0360 1.0661 1.0989
#> indirect 0.2371 0.0218 20000 0.1699 0.1826 0.1961 0.2814 0.2960 0.3145
#> direct   0.2597 0.0357 20000 0.1420 0.1655 0.1895 0.3293 0.3529 0.3785
```

```
#> total    0.4969 0.0349 20000 0.3806 0.4065 0.4279 0.5653 0.5884 0.6114
```

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       0.2148 0.0289 20000 0.1158 0.1393 0.1577 0.2711 0.2889 0.3080
#> b        0.4314 0.0271 20000 0.3368 0.3601 0.3774 0.4843 0.4988 0.5166
#> a        0.4547 0.0251 20000 0.3716 0.3894 0.4044 0.5025 0.5171 0.5326
#> Y~~Y     0.6835 0.0243 20000 0.6024 0.6195 0.6346 0.7296 0.7451 0.7607
#> M~~M     0.7932 0.0228 20000 0.7163 0.7326 0.7475 0.8365 0.8484 0.8619
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1962 0.0168 20000 0.1443 0.1542 0.1640 0.2298 0.2403 0.2543
#> direct   0.2148 0.0289 20000 0.1158 0.1393 0.1577 0.2711 0.2889 0.3080
#> total    0.4110 0.0262 20000 0.3197 0.3412 0.3581 0.4613 0.4775 0.4953
```

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