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       0.2318 0.03509 20000 0.1197 0.1420 0.1626 0.3006 0.3205 0.3446
#> b        0.4937 0.03160 20000 0.3893 0.4130 0.4316 0.5557 0.5743 0.5955
#> a        0.5005 0.03153 20000 0.3996 0.4199 0.4394 0.5629 0.5817 0.6009
#> Y~~Y     0.9702 0.04339 20000 0.8315 0.8601 0.8865 1.0545 1.0826 1.1163
#> M~~M     0.9648 0.04321 20000 0.8237 0.8547 0.8801 1.0492 1.0765 1.1113
#> indirect 0.2471 0.02210 20000 0.1811 0.1926 0.2047 0.2918 0.3064 0.3238
#> direct   0.2318 0.03509 20000 0.1197 0.1420 0.1626 0.3006 0.3205 0.3446
```

```
#> total    0.4789 0.03508 20000 0.3668 0.3892 0.4103 0.5475 0.5695 0.5935
```

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       0.1917 0.0290 20000 0.0957 0.1165 0.1347 0.2486 0.2657 0.2851
#> b        0.4542 0.0268 20000 0.3652 0.3852 0.4009 0.5059 0.5222 0.5410
#> a        0.4501 0.0254 20000 0.3610 0.3824 0.3996 0.4990 0.5143 0.5313
#> Y~~Y     0.6786 0.0244 20000 0.5971 0.6136 0.6290 0.7249 0.7390 0.7546
#> M~~M     0.7974 0.0228 20000 0.7178 0.7355 0.7510 0.8403 0.8538 0.8697
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2044 0.0171 20000 0.1498 0.1625 0.1715 0.2386 0.2488 0.2643
#> direct   0.1917 0.0290 20000 0.0957 0.1165 0.1347 0.2486 0.2657 0.2851
#> total    0.3961 0.0270 20000 0.3051 0.3263 0.3426 0.4479 0.4639 0.4837
```

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