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:

```
if (!require("remotes")) 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. A description of the package and code examples are presented in Pesigan and Cheung (2023).

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       0.2510 0.0357 20000 0.1347 0.1599 0.1808 0.3208 0.3433 0.3650
#> b        0.5247 0.0315 20000 0.4210 0.4441 0.4632 0.5856 0.6040 0.6286
#> a        0.5098 0.0317 20000 0.4100 0.4281 0.4475 0.5725 0.5908 0.6149
#> Y~~Y     1.0166 0.0456 20000 0.8675 0.8976 0.9263 1.1059 1.1315 1.1638
#> M~~M     1.0273 0.0455 20000 0.8651 0.9113 0.9380 1.1178 1.1455 1.1825
```

```
#> X~~X     1.0255 0.0000 20000 1.0255 1.0255 1.0255 1.0255 1.0255 1.0255
#> indirect 0.2675 0.0231 20000 0.2000 0.2111 0.2240 0.3146 0.3295 0.3492
#> direct   0.2510 0.0357 20000 0.1347 0.1599 0.1808 0.3208 0.3433 0.3650
#> total    0.5185 0.0359 20000 0.4015 0.4265 0.4484 0.5893 0.6113 0.6338
```

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))
vcov(unstd)

#>                      cp             b             a          Y~~Y          M~~M
#> cp        1.247990e-03 -5.080888e-04 -6.208667e-06 -1.403976e-06  9.653267e-07
#> b        -5.080888e-04  9.686711e-04  2.941175e-06 -1.232668e-05 -2.951775e-06
#> a        -6.208667e-06  2.941175e-06  9.885115e-04 -4.779755e-06 -2.435769e-06
#> Y~~Y     -1.403976e-06 -1.232668e-05 -4.779755e-06  2.024636e-03 -2.836394e-05
#> M~~M      9.653267e-07 -2.951775e-06 -2.435769e-06 -2.836394e-05  2.111537e-03
#> X~~X      9.388660e-06 -1.622702e-05 -2.013081e-05 -4.451909e-06  1.525728e-05
#> indirect -2.623957e-04  4.949801e-04  5.205527e-04 -8.347697e-06 -2.775001e-06
#> direct    1.247990e-03 -5.080888e-04 -6.208667e-06 -1.403976e-06  9.653267e-07
#> total     9.855944e-04 -1.310866e-05  5.143441e-04 -9.751673e-06 -1.809674e-06
#>                    X~~X      indirect        direct         total
#> cp        9.388660e-06 -2.623957e-04  1.247990e-03  9.855944e-04
```

```
#> b         -1.622702e-05  4.949801e-04 -5.080888e-04 -1.310866e-05
#> a         -2.013081e-05  5.205527e-04 -6.208667e-06  5.143441e-04
#> Y~~Y      -4.451909e-06 -8.347697e-06 -1.403976e-06 -9.751673e-06
#> M~~M       1.525728e-05 -2.775001e-06  9.653267e-07 -1.809674e-06
#> X~~X       2.079248e-03 -1.894010e-05  9.388660e-06 -9.551436e-06
#> indirect -1.894010e-05  5.264536e-04 -2.623957e-04  2.640579e-04
#> direct    9.388660e-06 -2.623957e-04  1.247990e-03  9.855944e-04
#> total    -9.551436e-06  2.640579e-04  9.855944e-04  1.249652e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2025 0.0282 20000 0.1130 0.1313 0.1470 0.2575 0.2747 0.2953
#> b        0.4755 0.0258 20000 0.3885 0.4082 0.4242 0.5247 0.5400 0.5619
#> a        0.4538 0.0249 20000 0.3683 0.3882 0.4048 0.5019 0.5157 0.5321
#> Y~~Y     0.6454 0.0240 20000 0.5671 0.5835 0.5978 0.6910 0.7066 0.7232
#> M~~M     0.7940 0.0225 20000 0.7168 0.7340 0.7481 0.8361 0.8493 0.8644
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2158 0.0170 20000 0.1618 0.1735 0.1830 0.2495 0.2603 0.2749
#> direct   0.2025 0.0282 20000 0.1130 0.1313 0.1470 0.2575 0.2747 0.2953
#> total    0.4184 0.0259 20000 0.3279 0.3511 0.3663 0.4682 0.4830 0.5011
```

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

Pesigan, I. J. A., & Cheung, S. F. (2023). Monte Carlo confidence intervals for the indirect effect with missing data. *Behavior Research Methods*. https://doi.org/10.3758/s13428-023-02114-4

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. (2023). *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