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       0.2790 0.0365 20000 0.1588 0.1847 0.2081 0.3503 0.3725 0.3993
#> b        0.4833 0.0322 20000 0.3775 0.3999 0.4201 0.5461 0.5664 0.5875
#> a        0.5042 0.0319 20000 0.4042 0.4215 0.4412 0.5670 0.5860 0.6063
#> Y~~Y     1.0480 0.0466 20000 0.9022 0.9277 0.9565 1.1397 1.1672 1.2002
#> M~~M     1.0095 0.0452 20000 0.8614 0.8923 0.9209 1.0980 1.1258 1.1590
```

```
#> X~~X     0.9831 0.0000 20000 0.9831 0.9831 0.9831 0.9831 0.9831 0.9831
#> indirect 0.2437 0.0222 20000 0.1724 0.1896 0.2012 0.2883 0.3041 0.3232
#> direct   0.2790 0.0365 20000 0.1588 0.1847 0.2081 0.3503 0.3725 0.3993
#> total    0.5227 0.0361 20000 0.3964 0.4299 0.4527 0.5943 0.6155 0.6366
```

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        1.350026e-03 -5.260672e-04 -6.628412e-06 -1.586572e-05 -7.171063e-06
#> b        -5.260672e-04  1.018883e-03  1.810003e-06  5.722308e-06 -7.968539e-06
#> a        -6.628412e-06  1.810003e-06  1.002549e-03 -1.714054e-05  1.155884e-05
#> Y~~Y     -1.586572e-05  5.722308e-06 -1.714054e-05  2.213346e-03  2.907773e-05
#> M~~M     -7.171063e-06 -7.968539e-06  1.155884e-05  2.907773e-05  2.056484e-03
#> X~~X     -1.145099e-05  7.010750e-06 -1.081699e-05 -1.490131e-06 -3.424184e-05
#> indirect -2.687236e-04  5.146629e-04  4.846145e-04 -5.442931e-06  1.566823e-06
#> direct    1.350026e-03 -5.260672e-04 -6.628412e-06 -1.586572e-05 -7.171063e-06
#> total     1.081303e-03 -1.140430e-05  4.779861e-04 -2.130865e-05 -5.604241e-06
#>                    X~~X      indirect        direct         total
#> cp       -1.145099e-05 -2.687236e-04  1.350026e-03  1.081303e-03
```

```
#> b         7.010750e-06  5.146629e-04 -5.260672e-04 -1.140430e-05
#> a        -1.081699e-05  4.846145e-04 -6.628412e-06  4.779861e-04
#> Y~~Y     -1.490131e-06 -5.442931e-06 -1.586572e-05 -2.130865e-05
#> M~~M     -3.424184e-05  1.566823e-06 -7.171063e-06 -5.604241e-06
#> X~~X      1.937358e-03 -1.307122e-06 -1.145099e-05 -1.275811e-05
#> indirect -1.307122e-06  4.943621e-04 -2.687236e-04  2.256385e-04
#> direct   -1.145099e-05 -2.687236e-04  1.350026e-03  1.081303e-03
#> total    -1.275811e-05  2.256385e-04  1.081303e-03  1.306941e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2220 0.0288 20000 0.1252 0.1471 0.1653 0.2777 0.2958 0.3152
#> b        0.4353 0.0267 20000 0.3457 0.3652 0.3826 0.4868 0.5036 0.5227
#> a        0.4454 0.0251 20000 0.3604 0.3801 0.3955 0.4938 0.5098 0.5265
#> Y~~Y     0.6751 0.0241 20000 0.5973 0.6117 0.6273 0.7219 0.7354 0.7488
#> M~~M     0.8016 0.0223 20000 0.7228 0.7401 0.7561 0.8435 0.8555 0.8701
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1939 0.0165 20000 0.1417 0.1530 0.1620 0.2266 0.2378 0.2497
#> direct   0.2220 0.0288 20000 0.1252 0.1471 0.1653 0.2777 0.2958 0.3152
#> total    0.4159 0.0262 20000 0.3292 0.3463 0.3638 0.4661 0.4813 0.5022
```

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