semmcci: Monte Carlo Confidence Intervals

Ivan Jacob Agaloos Pesigan

Installation

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

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

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

```r
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.

```r
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`.

```r
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.

```r
MC(fit, R = 20000L, alpha = 0.05)

#> Monte Carlo Confidence Intervals
#>             est     se     R   2.5%  97.5%
#> cp       0.2774 0.0362 20000 0.2070 0.3489
#> b        0.5535 0.0332 20000 0.4883 0.6179
#> a        0.5497 0.0302 20000 0.4912 0.6092
#> Y~~Y     1.0039 0.0449 20000 0.9157 1.0923
#> M~~M     0.9203 0.0414 20000 0.8386 1.0008
```

```
#> X~~X     1.0251 0.0000 20000 1.0251 1.0251
#> indirect 0.3042 0.0248 20000 0.2575 0.3543
#> direct   0.2774 0.0362 20000 0.2070 0.3489
#> total    0.5817 0.0355 20000 0.5127 0.6517
```

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 = 0.05)
vcov(unstd)
```

```
#>                     cp             b             a          Y~~Y          M~~M
#> cp        1.303373e-03 -6.118500e-04 4.817339e-06 -2.328676e-05  7.961675e-06
#> b        -6.118500e-04  1.108336e-03 9.039060e-06  2.461479e-07 -2.333999e-05
#> a         4.817339e-06  9.039060e-06 9.049849e-04  1.358197e-05  2.644763e-07
#> Y~~Y     -2.328676e-05  2.461479e-07 1.358197e-05  2.001253e-03  9.319276e-06
#> M~~M      7.961675e-06 -2.333999e-05 2.644763e-07  9.319276e-06  1.705314e-03
#> X~~X     -5.713580e-06  5.145664e-06 7.918838e-06  2.988791e-06  5.883026e-06
#> indirect -3.334333e-04  6.135267e-04 5.056937e-04  7.459974e-06 -1.248220e-05
#> direct    1.303373e-03 -6.118500e-04 4.817339e-06 -2.328676e-05  7.961675e-06
#> total     9.699395e-04  1.676705e-06 5.105110e-04 -1.582678e-05 -4.520527e-06
#>                   X~~X      indirect        direct         total
#> cp       -5.713580e-06 -3.334333e-04  1.303373e-03  9.699395e-04
```

```
#> b         5.145664e-06  6.135267e-04 -6.118500e-04  1.676705e-06
#> a         7.918838e-06  5.056937e-04  4.817339e-06  5.105110e-04
#> Y~~Y      2.988791e-06  7.459974e-06 -2.328676e-05 -1.582678e-05
#> M~~M      5.883026e-06 -1.248220e-05  7.961675e-06 -4.520527e-06
#> X~~X      2.099218e-03  6.988235e-06 -5.713580e-06  1.274655e-06
#> indirect  6.988235e-06  6.176712e-04 -3.334333e-04  2.842379e-04
#> direct   -5.713580e-06 -3.334333e-04  1.303373e-03  9.699395e-04
#> total     1.274655e-06  2.842379e-04  9.699395e-04  1.254177e-03
```

```
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2198 0.0283 20000 0.1251 0.1468 0.1644 0.2755 0.2931 0.3098
#> b        0.4804 0.0263 20000 0.3957 0.4112 0.4280 0.5314 0.5468 0.5666
#> a        0.5018 0.0238 20000 0.4205 0.4384 0.4532 0.5467 0.5591 0.5761
#> Y~~Y     0.6149 0.0242 20000 0.5360 0.5518 0.5670 0.6624 0.6762 0.6941
#> M~~M     0.7482 0.0238 20000 0.6682 0.6875 0.7011 0.7946 0.8078 0.8232
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2411 0.0179 20000 0.1820 0.1956 0.2056 0.2763 0.2876 0.3005
#> direct   0.2198 0.0283 20000 0.1251 0.1468 0.1644 0.2755 0.2931 0.3098
#> total    0.4609 0.0249 20000 0.3770 0.3946 0.4108 0.5088 0.5231 0.5394
```

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*, *56*(3), 1678–1696. 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. (2024). *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