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 = 0.05)

#> Monte Carlo Confidence Intervals
#>             est     se     R   2.5%  97.5%
#> cp       0.2402 0.0345 20000 0.1727 0.3072
#> b        0.5053 0.0302 20000 0.4461 0.5650
#> a        0.5079 0.0327 20000 0.4434 0.5721
#> Y~~Y     0.9726 0.0435 20000 0.8870 1.0586
#> M~~M     1.0855 0.0484 20000 0.9913 1.1820
```

```
#> X~~X     1.0047 0.0000 20000 1.0047 1.0047
#> indirect 0.2567 0.0225 20000 0.2141 0.3021
#> direct   0.2402 0.0345 20000 0.1727 0.3072
#> total    0.4969 0.0349 20000 0.4282 0.5655
```

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        1.197046e-03 -4.533073e-04  7.960046e-07 -7.362958e-06  1.621333e-06
#> b        -4.533073e-04  8.989838e-04 -4.923184e-06 -5.651537e-06 -2.964170e-06
#> a         7.960046e-07 -4.923184e-06  1.075995e-03  9.674287e-06 -1.544503e-05
#> Y~~Y     -7.362958e-06 -5.651537e-06  9.674287e-06  1.888410e-03 -1.147526e-05
#> M~~M      1.621333e-06 -2.964170e-06 -1.544503e-05 -1.147526e-05  2.352962e-03
#> X~~X     -4.680534e-06 -7.884830e-07  2.108971e-05  6.365275e-06  1.807853e-05
#> indirect -2.295119e-04  4.540516e-04  5.416343e-04  1.843029e-06 -9.175996e-06
#> direct    1.197046e-03 -4.533073e-04  7.960046e-07 -7.362958e-06  1.621333e-06
#> total     9.675338e-04  7.442738e-07  5.424303e-04 -5.519929e-06 -7.554664e-06
#>                   X~~X      indirect        direct         total
#> cp       -4.680534e-06 -2.295119e-04  1.197046e-03  9.675338e-04
```

```
#> b        -7.884830e-07  4.540516e-04 -4.533073e-04  7.442738e-07
#> a         2.108971e-05  5.416343e-04  7.960046e-07  5.424303e-04
#> Y~~Y      6.365275e-06  1.843029e-06 -7.362958e-06 -5.519929e-06
#> M~~M      1.807853e-05 -9.175996e-06  1.621333e-06 -7.554664e-06
#> X~~X      2.009137e-03  1.024644e-05 -4.680534e-06  5.565909e-06
#> indirect  1.024644e-05  5.054562e-04 -2.295119e-04  2.759443e-04
#> direct   -4.680534e-06 -2.295119e-04  1.197046e-03  9.675338e-04
#> total     5.565909e-06  2.759443e-04  9.675338e-04  1.243478e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.1967 0.0280 20000 0.1057 0.1243 0.1413 0.2509 0.2675 0.2867
#> b        0.4788 0.0257 20000 0.3923 0.4118 0.4284 0.5285 0.5442 0.5605
#> a        0.4391 0.0256 20000 0.3509 0.3715 0.3880 0.4883 0.5047 0.5191
#> Y~~Y     0.6493 0.0243 20000 0.5621 0.5860 0.6012 0.6960 0.7097 0.7284
#> M~~M     0.8072 0.0225 20000 0.7305 0.7453 0.7615 0.8494 0.8620 0.8769
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2102 0.0170 20000 0.1562 0.1679 0.1781 0.2435 0.2551 0.2681
#> direct   0.1967 0.0280 20000 0.1057 0.1243 0.1413 0.2509 0.2675 0.2867
#> total    0.4070 0.0264 20000 0.3176 0.3351 0.3536 0.4571 0.4716 0.4881
```

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