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        0.2849 0.0348 20000 0.2168 0.3532
#> b         0.4771 0.0324 20000 0.4140 0.5407
#> a         0.4456 0.0317 20000 0.3822 0.5068
#> Y~~Y      1.0183 0.0457 20000 0.9273 1.1077
#> M~~M      0.9873 0.0444 20000 0.9004 1.0739
```

```
#> X~~X     0.9957 0.0000 20000 0.9957 0.9957
#> indirect 0.2126 0.0210 20000 0.1729 0.2547
#> direct   0.2849 0.0348 20000 0.2168 0.3532
#> total    0.4975 0.0352 20000 0.4286 0.5659
```

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         1.248401e-03 -4.676146e-04  9.392423e-06 -1.889257e-05 -3.559211e-06
#> b         -4.676146e-04  1.026838e-03 -1.234636e-06  5.473286e-06 -3.586824e-06
#> a          9.392423e-06 -1.234636e-06  9.806538e-04 -6.555363e-07 -8.616884e-06
#> Y~~Y      -1.889257e-05  5.473286e-06 -6.555363e-07  2.106829e-03 -8.126829e-06
#> M~~M      -3.559211e-06 -3.586824e-06 -8.616884e-06 -8.126829e-06  1.961568e-03
#> X~~X       6.895899e-06  1.304820e-05 -1.752680e-06  5.575149e-06  4.281129e-07
#> indirect -2.039078e-04  4.566610e-04  4.683887e-04  1.918496e-06 -5.020972e-06
#> direct     1.248401e-03 -4.676146e-04  9.392423e-06 -1.889257e-05 -3.559211e-06
#> total      1.044493e-03 -1.095362e-05  4.777811e-04 -1.697407e-05 -8.580183e-06
#>                    X~~X      indirect        direct         total
#> cp         6.895899e-06 -2.039078e-04  1.248401e-03  1.044493e-03
```

```
#> b         1.304820e-05  4.566610e-04 -4.676146e-04 -1.095362e-05
#> a        -1.752680e-06  4.683887e-04  9.392423e-06  4.777811e-04
#> Y~~Y      5.575149e-06  1.918496e-06 -1.889257e-05 -1.697407e-05
#> M~~M      4.281129e-07 -5.020972e-06 -3.559211e-06 -8.580183e-06
#> X~~X      1.949282e-03  4.526837e-06  6.895899e-06  1.142274e-05
#> indirect  4.526837e-06  4.283482e-04 -2.039078e-04  2.244404e-04
#> direct    6.895899e-06 -2.039078e-04  1.248401e-03  1.044493e-03
#> total     1.142274e-05  2.244404e-04  1.044493e-03  1.268933e-03
```

```r
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2329 0.0284 20000 0.1335 0.1583 0.1772 0.2883 0.3054 0.3261
#> b        0.4255 0.0265 20000 0.3374 0.3576 0.3733 0.4766 0.4919 0.5061
#> a        0.4085 0.0262 20000 0.3227 0.3390 0.3558 0.4591 0.4748 0.4914
#> Y~~Y     0.6837 0.0244 20000 0.6056 0.6207 0.6348 0.7304 0.7434 0.7580
#> M~~M     0.8331 0.0214 20000 0.7585 0.7746 0.7893 0.8734 0.8851 0.8959
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1738 0.0158 20000 0.1251 0.1352 0.1439 0.2052 0.2155 0.2283
#> direct   0.2329 0.0284 20000 0.1335 0.1583 0.1772 0.2883 0.3054 0.3261
#> total    0.4068 0.0267 20000 0.3127 0.3359 0.3530 0.4575 0.4743 0.4886
```

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