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       0.2776 0.0350 20000 0.2102 0.3473
#> b        0.4518 0.0328 20000 0.3878 0.5175
#> a        0.4666 0.0305 20000 0.4056 0.5256
#> Y~~Y     1.0103 0.0453 20000 0.9221 1.0989
#> M~~M     0.9322 0.0419 20000 0.8499 1.0153
```

```
#> X~~X     1.0059 0.0000 20000 1.0059 1.0059
#> indirect 0.2108 0.0205 20000 0.1720 0.2524
#> direct   0.2776 0.0350 20000 0.2102 0.3473
#> total    0.4884 0.0346 20000 0.4211 0.5562
```

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        1.241740e-03 -5.044582e-04  3.224169e-06 -4.845097e-07 -4.995636e-06
#> b        -5.044582e-04  1.100884e-03  3.933688e-06 -4.671758e-06  1.304705e-05
#> a         3.224169e-06  3.933688e-06  9.190526e-04 -2.741582e-06  3.236884e-06
#> Y~~Y     -4.845097e-07 -4.671758e-06 -2.741582e-06  2.064290e-03  1.512788e-05
#> M~~M     -4.995636e-06  1.304705e-05  3.236884e-06  1.512788e-05  1.736320e-03
#> X~~X      9.411219e-06 -2.679261e-06 -3.777447e-06  1.594913e-05 -6.069559e-06
#> indirect -2.344077e-04  5.159730e-04  4.175757e-04 -4.165204e-06  7.283963e-06
#> direct    1.241740e-03 -5.044582e-04  3.224169e-06 -4.845097e-07 -4.995636e-06
#> total     1.007333e-03  1.151478e-05  4.207999e-04 -4.649713e-06  2.288327e-06
#>                    X~~X      indirect        direct         total
#> cp        9.411219e-06 -2.344077e-04  1.241740e-03  1.007333e-03
```

```
#> b         -2.679261e-06  5.159730e-04 -5.044582e-04  1.151478e-05
#> a         -3.777447e-06  4.175757e-04  3.224169e-06  4.207999e-04
#> Y~~Y       1.594913e-05 -4.165204e-06 -4.845097e-07 -4.649713e-06
#> M~~M      -6.069559e-06  7.283963e-06 -4.995636e-06  2.288327e-06
#> X~~X       1.988894e-03 -2.811817e-06  9.411219e-06  6.599402e-06
#> indirect -2.811817e-06  4.308903e-04 -2.344077e-04  1.964826e-04
#> direct     9.411219e-06 -2.344077e-04  1.241740e-03  1.007333e-03
#> total      6.599402e-06  1.964826e-04  1.007333e-03  1.203815e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>              est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp        0.2319 0.0289 20000 0.1347 0.1561 0.1752 0.2880 0.3060 0.3249
#> b         0.4039 0.0278 20000 0.3122 0.3321 0.3489 0.4576 0.4748 0.4945
#> a         0.4361 0.0255 20000 0.3473 0.3704 0.3861 0.4856 0.5014 0.5214
#> Y~~Y      0.7013 0.0244 20000 0.6195 0.6363 0.6513 0.7481 0.7616 0.7738
#> M~~M      0.8098 0.0222 20000 0.7282 0.7486 0.7642 0.8509 0.8628 0.8794
#> X~~X      1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1762 0.0162 20000 0.1276 0.1371 0.1452 0.2086 0.2198 0.2340
#> direct   0.2319 0.0289 20000 0.1347 0.1561 0.1752 0.2880 0.3060 0.3249
#> total    0.4081 0.0264 20000 0.3203 0.3397 0.3556 0.4584 0.4741 0.4918
```

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