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 (2024).

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       0.2258 0.0349 20000 0.1577 0.2946
#> b        0.5437 0.0306 20000 0.4838 0.6044
#> a        0.5257 0.0323 20000 0.4623 0.5894
#> Y~~Y     1.0205 0.0454 20000 0.9314 1.1100
#> M~~M     1.0994 0.0490 20000 1.0036 1.1944
```

```
#> X~~X     1.0483 0.0000 20000 1.0483 1.0483
#> indirect 0.2858 0.0240 20000 0.2402 0.3342
#> direct   0.2258 0.0349 20000 0.1577 0.2946
#> total    0.5116 0.0356 20000 0.4420 0.5817
```

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        1.221749e-03 -4.902496e-04 -1.214125e-05 -3.284633e-07 -2.023017e-05
#> b        -4.902496e-04  9.224239e-04  1.195221e-05 -8.635838e-06 -1.755196e-06
#> a        -1.214125e-05  1.195221e-05  1.032803e-03  1.306815e-05 -2.055988e-05
#> Y~~Y     -3.284633e-07 -8.635838e-06  1.306815e-05  2.080198e-03 -2.717524e-05
#> M~~M     -2.023017e-05 -1.755196e-06 -2.055988e-05 -2.717524e-05  2.400729e-03
#> X~~X      1.220898e-05  5.572485e-07  2.287230e-06 -2.036124e-06  8.422999e-06
#> indirect -2.642937e-04  4.912222e-04  5.679065e-04  2.659857e-06 -1.168587e-05
#> direct    1.221749e-03 -4.902496e-04 -1.214125e-05 -3.284633e-07 -2.023017e-05
#> total     9.574551e-04  9.725343e-07  5.557652e-04  2.331394e-06 -3.191604e-05
#>                    X~~X      indirect        direct         total
#> cp        1.220898e-05 -2.642937e-04  1.221749e-03  9.574551e-04
```

```
#> b         5.572485e-07  4.912222e-04 -4.902496e-04  9.725343e-07
#> a         2.287230e-06  5.679065e-04 -1.214125e-05  5.557652e-04
#> Y~~Y     -2.036124e-06  2.659857e-06 -3.284633e-07  2.331394e-06
#> M~~M      8.422999e-06 -1.168587e-05 -2.023017e-05 -3.191604e-05
#> X~~X      2.208937e-03  1.209539e-06  1.220898e-05  1.341852e-05
#> indirect  1.209539e-06  5.679071e-04 -2.642937e-04  3.036134e-04
#> direct    1.220898e-05 -2.642937e-04  1.221749e-03  9.574551e-04
#> total     1.341852e-05  3.036134e-04  9.574551e-04  1.261069e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.1816 0.0279 20000 0.0884 0.1100 0.1273 0.2360 0.2536 0.2721
#> b        0.5035 0.0254 20000 0.4248 0.4369 0.4524 0.5522 0.5677 0.5850
#> a        0.4567 0.0250 20000 0.3773 0.3914 0.4063 0.5043 0.5194 0.5374
#> Y~~Y     0.6300 0.0241 20000 0.5495 0.5673 0.5820 0.6768 0.6903 0.7060
#> M~~M     0.7915 0.0228 20000 0.7112 0.7303 0.7457 0.8349 0.8468 0.8576
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2299 0.0175 20000 0.1742 0.1854 0.1960 0.2641 0.2756 0.2879
#> direct   0.1816 0.0279 20000 0.0884 0.1100 0.1273 0.2360 0.2536 0.2721
#> total    0.4116 0.0261 20000 0.3282 0.3428 0.3594 0.4617 0.4757 0.4935
```

References

Pesigan, I. J. A., & Cheung, S. F. (2024). 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

R Core Team. (2025). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/