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.

```
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 = c(0.001, 0.01, 0.05))

#> Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2097 0.0350 20000 0.0907 0.1180 0.1409 0.2785 0.2993 0.3282
#> b        0.5145 0.0301 20000 0.4170 0.4368 0.4557 0.5737 0.5935 0.6142
#> a        0.5359 0.0325 20000 0.4317 0.4518 0.4725 0.5996 0.6183 0.6395
#> Y~~Y     0.9489 0.0423 20000 0.8067 0.8389 0.8658 1.0311 1.0573 1.0862
#> M~~M     1.0608 0.0473 20000 0.9031 0.9387 0.9676 1.1531 1.1828 1.2187
```

```
#> X~~X     0.9892 0.0000 20000 0.9892 0.9892 0.9892 0.9892 0.9892 0.9892
#> indirect 0.2757 0.0232 20000 0.2048 0.2196 0.2318 0.3226 0.3383 0.3558
#> direct   0.2097 0.0350 20000 0.0907 0.1180 0.1409 0.2785 0.2993 0.3282
#> total    0.4854 0.0354 20000 0.3725 0.3928 0.4148 0.5551 0.5782 0.5980
```

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 = c(0.001, 0.01, 0.05))
vcov(unstd)
```

```
#>                      cp             b             a          Y~~Y          M~~M
#> cp        1.201487e-03 -4.854989e-04  3.698507e-06 -1.034124e-06 -6.932198e-06
#> b        -4.854989e-04  9.070377e-04 -9.009780e-06  5.823466e-06  1.929803e-05
#> a         3.698507e-06 -9.009780e-06  1.080586e-03  1.308560e-05 -6.935492e-07
#> Y~~Y     -1.034124e-06  5.823466e-06  1.308560e-05  1.800983e-03  4.959985e-06
#> M~~M     -6.932198e-06  1.929803e-05 -6.935492e-07  4.959985e-06  2.252829e-03
#> X~~X      2.277153e-06  6.436120e-06  6.858243e-06  6.863582e-06  2.395241e-05
#> indirect -2.580805e-04  4.814375e-04  5.507825e-04  1.024658e-05  1.018783e-05
#> direct    1.201487e-03 -4.854989e-04  3.698507e-06 -1.034124e-06 -6.932198e-06
#> total     9.434068e-04 -4.061474e-06  5.544810e-04  9.212454e-06  3.255635e-06
#>                   X~~X      indirect        direct         total
#> cp        2.277153e-06 -2.580805e-04  1.201487e-03  9.434068e-04
```

```
#> b        6.436120e-06  4.814375e-04 -4.854989e-04 -4.061474e-06
#> a        6.858243e-06  5.507825e-04  3.698507e-06  5.544810e-04
#> Y~~Y     6.863582e-06  1.024658e-05 -1.034124e-06  9.212454e-06
#> M~~M     2.395241e-05  1.018783e-05 -6.932198e-06  3.255635e-06
#> X~~X     1.963878e-03  7.627357e-06  2.277153e-06  9.904510e-06
#> indirect 7.627357e-06  5.421654e-04 -2.580805e-04  2.840849e-04
#> direct   2.277153e-06 -2.580805e-04  1.201487e-03  9.434068e-04
#> total    9.904510e-06  2.840849e-04  9.434068e-04  1.227492e-03
```

```
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.1724 0.0283 20000 0.0804 0.0985 0.1165 0.2279 0.2439 0.2644
#> b        0.4933 0.0260 20000 0.4057 0.4262 0.4406 0.5431 0.5571 0.5744
#> a        0.4596 0.0250 20000 0.3752 0.3931 0.4098 0.5079 0.5231 0.5392
#> Y~~Y     0.6487 0.0242 20000 0.5716 0.5858 0.6003 0.6949 0.7088 0.7240
#> M~~M     0.7888 0.0230 20000 0.7093 0.7264 0.7421 0.8321 0.8455 0.8592
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2267 0.0175 20000 0.1726 0.1835 0.1925 0.2617 0.2718 0.2848
#> direct   0.1724 0.0283 20000 0.0804 0.0985 0.1165 0.2279 0.2439 0.2644
#> total    0.3992 0.0264 20000 0.3112 0.3298 0.3466 0.4502 0.4645 0.4795
```

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