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       0.2777 0.0336 20000 0.2116 0.3434
#> b        0.4580 0.0306 20000 0.3984 0.5183
#> a        0.4582 0.0318 20000 0.3951 0.5199
#> Y~~Y     0.9309 0.0411 20000 0.8503 1.0120
#> M~~M     0.9912 0.0444 20000 0.9049 1.0781
```

```
#> X~~X     1.0093 0.0000 20000 1.0093 1.0093
#> indirect 0.2099 0.0201 20000 0.1718 0.2502
#> direct   0.2777 0.0336 20000 0.2116 0.3434
#> total    0.4875 0.0335 20000 0.4223 0.5536
```

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        1.136768e-03 -4.375662e-04  5.896044e-06 -3.606490e-06  7.829920e-06
#> b        -4.375662e-04  9.438352e-04 -1.309317e-05 -1.194636e-06 -2.232049e-05
#> a         5.896044e-06 -1.309317e-05  9.745362e-04  5.070569e-06  1.500531e-05
#> Y~~Y     -3.606490e-06 -1.194636e-06  5.070569e-06  1.727433e-03 -1.474145e-05
#> M~~M      7.829920e-06 -2.232049e-05  1.500531e-05 -1.474145e-05  1.971296e-03
#> X~~X      1.682183e-05 -4.602366e-06  2.824467e-06  6.703624e-07 -8.235774e-06
#> indirect -1.976156e-04  4.264400e-04  4.402349e-04  1.326856e-06 -3.254169e-06
#> direct    1.136768e-03 -4.375662e-04  5.896044e-06 -3.606490e-06  7.829920e-06
#> total     9.391528e-04 -1.112614e-05  4.461309e-04 -2.279633e-06  4.575751e-06
#>                    X~~X      indirect        direct         total
#> cp        1.682183e-05 -1.976156e-04  1.136768e-03  9.391528e-04
```

```
#> b        -4.602366e-06  4.264400e-04 -4.375662e-04 -1.112614e-05
#> a         2.824467e-06  4.402349e-04  5.896044e-06  4.461309e-04
#> Y~~Y      6.703624e-07  1.326856e-06 -3.606490e-06 -2.279633e-06
#> M~~M     -8.235774e-06 -3.254169e-06  7.829920e-06  4.575751e-06
#> X~~X      2.046420e-03 -9.193125e-07  1.682183e-05  1.590252e-05
#> indirect -9.193125e-07  3.979031e-04 -1.976156e-04  2.002875e-04
#> direct    1.682183e-05 -1.976156e-04  1.136768e-03  9.391528e-04
#> total     1.590252e-05  2.002875e-04  9.391528e-04  1.139440e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2376 0.0283 20000 0.1459 0.1646 0.1815 0.2924 0.3097 0.3293
#> b        0.4279 0.0265 20000 0.3412 0.3596 0.3748 0.4790 0.4949 0.5119
#> a        0.4197 0.0259 20000 0.3304 0.3512 0.3680 0.4700 0.4859 0.5070
#> Y~~Y     0.6752 0.0243 20000 0.5938 0.6103 0.6261 0.7216 0.7356 0.7505
#> M~~M     0.8238 0.0218 20000 0.7430 0.7639 0.7791 0.8646 0.8767 0.8908
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1796 0.0159 20000 0.1311 0.1393 0.1490 0.2110 0.2226 0.2332
#> direct   0.2376 0.0283 20000 0.1459 0.1646 0.1815 0.2924 0.3097 0.3293
#> total    0.4171 0.0263 20000 0.3292 0.3460 0.3644 0.4675 0.4830 0.4984
```

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