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       0.2321 0.0353 20000 0.1629 0.3019
#> b        0.5076 0.0327 20000 0.4427 0.5714
#> a        0.4982 0.0305 20000 0.4388 0.5585
#> Y~~Y     1.0038 0.0449 20000 0.9149 1.0904
#> M~~M     0.9581 0.0429 20000 0.8737 1.0413
```

```
#> X~~X     1.0256 0.0000 20000 1.0256 1.0256
#> indirect 0.2529 0.0224 20000 0.2099 0.2973
#> direct   0.2321 0.0353 20000 0.1629 0.3019
#> total    0.4850 0.0348 20000 0.4174 0.5539
```

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        1.246635e-03 -5.262934e-04  2.165942e-06  1.326064e-05  6.302160e-06
#> b        -5.262934e-04  1.043108e-03 -4.707951e-06  3.262687e-06 -7.562341e-07
#> a         2.165942e-06 -4.707951e-06  9.187481e-04 -1.066025e-05  3.741640e-07
#> Y~~Y      1.326064e-05  3.262687e-06 -1.066025e-05  2.030658e-03  9.479320e-06
#> M~~M      6.302160e-06 -7.562341e-07  3.741640e-07  9.479320e-06  1.831551e-03
#> X~~X     -9.676818e-06  1.603139e-05 -1.467461e-06 -2.250509e-05  1.796609e-06
#> indirect -2.612536e-04  5.179862e-04  4.636879e-04 -4.372389e-06 -2.156581e-07
#> direct    1.246635e-03 -5.262934e-04  2.165942e-06  1.326064e-05  6.302160e-06
#> total     9.853810e-04 -8.307273e-06  4.658538e-04  8.888254e-06  6.086502e-06
#>                    X~~X      indirect        direct         total
#> cp       -9.676818e-06 -2.612536e-04  1.246635e-03  9.853810e-04
```

```
#> b          1.603139e-05  5.179862e-04 -5.262934e-04 -8.307273e-06
#> a         -1.467461e-06  4.636879e-04  2.165942e-06  4.658538e-04
#> Y~~Y      -2.250509e-05 -4.372389e-06  1.326064e-05  8.888254e-06
#> M~~M       1.796609e-06 -2.156581e-07  6.302160e-06  6.086502e-06
#> X~~X       2.068142e-03  7.381620e-06 -9.676818e-06 -2.295198e-06
#> indirect   7.381620e-06  4.945662e-04 -2.612536e-04  2.333125e-04
#> direct    -9.676818e-06 -2.612536e-04  1.246635e-03  9.853810e-04
#> total     -2.295198e-06  2.333125e-04  9.853810e-04  1.218694e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>              est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp        0.1924 0.0289 20000 0.0958 0.1183 0.1358 0.2490 0.2672 0.2870
#> b         0.4577 0.0267 20000 0.3695 0.3885 0.4037 0.5087 0.5257 0.5463
#> a         0.4582 0.0248 20000 0.3769 0.3944 0.4091 0.5056 0.5196 0.5362
#> Y~~Y      0.6728 0.0242 20000 0.5945 0.6095 0.6239 0.7190 0.7351 0.7508
#> M~~M      0.7901 0.0227 20000 0.7125 0.7300 0.7444 0.8327 0.8444 0.8580
#> X~~X      1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2097 0.0171 20000 0.1574 0.1677 0.1770 0.2438 0.2549 0.2695
#> direct    0.1924 0.0289 20000 0.0958 0.1183 0.1358 0.2490 0.2672 0.2870
#> total     0.4021 0.0265 20000 0.3138 0.3324 0.3491 0.4532 0.4692 0.4848
```

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