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       0.2622 0.0362 20000 0.1921 0.3343
#> b        0.4757 0.0321 20000 0.4127 0.5382
#> a        0.4722 0.0324 20000 0.4083 0.5356
#> Y~~Y     1.0306 0.0460 20000 0.9403 1.1208
#> M~~M     1.0014 0.0449 20000 0.9139 1.0886
```

```
#> X~~X     0.9425 0.0000 20000 0.9425 0.9425
#> indirect 0.2246 0.0216 20000 0.1837 0.2675
#> direct   0.2622 0.0362 20000 0.1921 0.3343
#> total    0.4868 0.0364 20000 0.4159 0.5579
```

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        1.308196e-03 -4.834553e-04  4.492790e-06  6.934977e-06 -6.131892e-06
#> b        -4.834553e-04  1.030661e-03 -2.943056e-07 -1.796954e-08  3.481664e-06
#> a         4.492790e-06 -2.943056e-07  1.071252e-03 -2.757813e-06 -2.482743e-06
#> Y~~Y      6.934977e-06 -1.796954e-08 -2.757813e-06  2.129822e-03  3.485683e-06
#> M~~M     -6.131892e-06  3.481664e-06 -2.482743e-06  3.485683e-06  1.991172e-03
#> X~~X      7.494678e-06 -1.050541e-05 -1.445090e-05  3.165216e-05  1.211394e-05
#> indirect -2.253706e-04  4.861553e-04  5.090337e-04 -1.166948e-06  5.777921e-07
#> direct    1.308196e-03 -4.834553e-04  4.492790e-06  6.934977e-06 -6.131892e-06
#> total     1.082825e-03  2.699956e-06  5.135265e-04  5.768029e-06 -5.554100e-06
#>                    X~~X      indirect        direct         total
#> cp        7.494678e-06 -2.253706e-04  1.308196e-03  1.082825e-03
```

```
#> b         -1.050541e-05  4.861553e-04 -4.834553e-04  2.699956e-06
#> a         -1.445090e-05  5.090337e-04  4.492790e-06  5.135265e-04
#> Y~~Y       3.165216e-05 -1.166948e-06  6.934977e-06  5.768029e-06
#> M~~M       1.211394e-05  5.777921e-07 -6.131892e-06 -5.554100e-06
#> X~~X       1.776455e-03 -1.113070e-05  7.494678e-06 -3.636024e-06
#> indirect -1.113070e-05  4.724321e-04 -2.253706e-04  2.470616e-04
#> direct     7.494678e-06 -2.253706e-04  1.308196e-03  1.082825e-03
#> total     -3.636024e-06  2.470616e-04  1.082825e-03  1.329887e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>              est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp        0.2092 0.0284 20000 0.1124 0.1352 0.1532 0.2646 0.2830 0.3065
#> b         0.4303 0.0268 20000 0.3400 0.3612 0.3771 0.4823 0.4973 0.5148
#> a         0.4164 0.0261 20000 0.3293 0.3484 0.3647 0.4672 0.4827 0.5026
#> Y~~Y      0.6961 0.0241 20000 0.6162 0.6330 0.6472 0.7415 0.7562 0.7771
#> M~~M      0.8266 0.0218 20000 0.7474 0.7670 0.7817 0.8670 0.8786 0.8916
#> X~~X      1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1792 0.0161 20000 0.1279 0.1387 0.1481 0.2113 0.2215 0.2348
#> direct   0.2092 0.0284 20000 0.1124 0.1352 0.1532 0.2646 0.2830 0.3065
#> total    0.3884 0.0267 20000 0.2969 0.3171 0.3343 0.4395 0.4543 0.4711
```

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