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 = 0.05)

#> Monte Carlo Confidence Intervals
#>             est     se     R   2.5%  97.5%
#> cp       0.2674 0.0362 20000 0.1958 0.3387
#> b        0.4862 0.0309 20000 0.4248 0.5465
#> a        0.5437 0.0328 20000 0.4796 0.6073
#> Y~~Y     0.9835 0.0436 20000 0.8979 1.0684
#> M~~M     1.0121 0.0451 20000 0.9253 1.1010
```

```
#> X~~X     0.9469 0.0000 20000 0.9469 0.9469
#> indirect 0.2643 0.0232 20000 0.2202 0.3110
#> direct   0.2674 0.0362 20000 0.1958 0.3387
#> total    0.5318 0.0358 20000 0.4613 0.6020
```

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        1.323669e-03 -5.177524e-04 -9.489955e-06  1.995146e-05 -1.187401e-05
#> b        -5.177524e-04  9.653011e-04  9.207130e-07 -9.725114e-06 -1.267670e-05
#> a        -9.489955e-06  9.207130e-07  1.065205e-03  5.278372e-06 -1.010667e-05
#> Y~~Y      1.995146e-05 -9.725114e-06  5.278372e-06  1.943872e-03 -6.445783e-07
#> M~~M     -1.187401e-05 -1.267670e-05 -1.010667e-05 -6.445783e-07  2.073663e-03
#> X~~X     -2.863144e-05  1.126716e-05 -1.210011e-06 -1.058040e-05 -7.374167e-06
#> indirect -2.853608e-04  5.245633e-04  5.186076e-04 -2.673343e-06 -1.154335e-05
#> direct    1.323669e-03 -5.177524e-04 -9.489955e-06  1.995146e-05 -1.187401e-05
#> total     1.038308e-03  6.810937e-06  5.091176e-04  1.727812e-05 -2.341737e-05
#>                   X~~X      indirect        direct         total
#> cp       -2.863144e-05 -2.853608e-04  1.323669e-03  1.038308e-03
```

```
#> b         1.126716e-05  5.245633e-04 -5.177524e-04  6.810937e-06
#> a        -1.210011e-06  5.186076e-04 -9.489955e-06  5.091176e-04
#> Y~~Y     -1.058040e-05 -2.673343e-06  1.995146e-05  1.727812e-05
#> M~~M     -7.374167e-06 -1.154335e-05 -1.187401e-05 -2.341737e-05
#> X~~X      1.793310e-03  6.163557e-06 -2.863144e-05 -2.246788e-05
#> indirect  6.163557e-06  5.380916e-04 -2.853608e-04  2.527309e-04
#> direct   -2.863144e-05 -2.853608e-04  1.323669e-03  1.038308e-03
#> total    -2.246788e-05  2.527309e-04  1.038308e-03  1.291039e-03
```

```
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2132 0.0285 20000 0.1203 0.1381 0.1556 0.2684 0.2854 0.3055
#> b        0.4527 0.0267 20000 0.3627 0.3825 0.4000 0.5044 0.5206 0.5389
#> a        0.4654 0.0248 20000 0.3787 0.3986 0.4149 0.5120 0.5263 0.5395
#> Y~~Y     0.6598 0.0242 20000 0.5790 0.5965 0.6119 0.7063 0.7227 0.7363
#> M~~M     0.7834 0.0230 20000 0.7090 0.7230 0.7378 0.8279 0.8411 0.8566
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2107 0.0171 20000 0.1554 0.1678 0.1776 0.2447 0.2557 0.2677
#> direct   0.2132 0.0285 20000 0.1203 0.1381 0.1556 0.2684 0.2854 0.3055
#> total    0.4238 0.0258 20000 0.3377 0.3549 0.3710 0.4727 0.4871 0.5042
```

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