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       0.2959 0.0366 20000 0.2241 0.3671
#> b        0.4656 0.0325 20000 0.4023 0.5291
#> a        0.5265 0.0316 20000 0.4646 0.5881
#> Y~~Y     1.0603 0.0472 20000 0.9680 1.1528
#> M~~M     1.0002 0.0443 20000 0.9137 1.0878
```

```
#> X~~X     1.0071 0.0000 20000 1.0071 1.0071
#> indirect 0.2452 0.0225 20000 0.2023 0.2906
#> direct   0.2959 0.0366 20000 0.2241 0.3671
#> total    0.5410 0.0355 20000 0.4719 0.6108
```

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         1.355864e-03 -5.733885e-04  1.207901e-05 -2.208337e-07  2.542532e-05
#> b         -5.733885e-04  1.054961e-03 -2.220329e-07  1.518010e-05 -1.919745e-05
#> a          1.207901e-05 -2.220329e-07  9.864266e-04 -1.335642e-06 -8.774624e-06
#> Y~~Y      -2.208337e-07  1.518010e-05 -1.335642e-06  2.212003e-03 -1.579605e-06
#> M~~M       2.542532e-05 -1.919745e-05 -8.774624e-06 -1.579605e-06  2.013900e-03
#> X~~X       1.031335e-05 -1.147627e-05  2.824960e-06  7.511804e-07 -7.169286e-06
#> indirect -2.966750e-04  5.554871e-04  4.592158e-04  8.094866e-06 -1.423029e-05
#> direct     1.355864e-03 -5.733885e-04  1.207901e-05 -2.208337e-07  2.542532e-05
#> total      1.059189e-03 -1.790131e-05  4.712948e-04  7.874033e-06  1.119503e-05
#>                    X~~X      indirect        direct         total
#> cp         1.031335e-05 -2.966750e-04  1.355864e-03  1.059189e-03
```

```
#> b        -1.147627e-05  5.554871e-04 -5.733885e-04 -1.790131e-05
#> a         2.824960e-06  4.592158e-04  1.207901e-05  4.712948e-04
#> Y~~Y      7.511804e-07  8.094866e-06 -2.208337e-07  7.874033e-06
#> M~~M     -7.169286e-06 -1.423029e-05  2.542532e-05  1.119503e-05
#> X~~X      2.038923e-03 -5.142624e-06  1.031335e-05  5.170728e-06
#> indirect -5.142624e-06  5.074203e-04 -2.966750e-04  2.107453e-04
#> direct    1.031335e-05 -2.966750e-04  1.355864e-03  1.059189e-03
#> total     5.170728e-06  2.107453e-04  1.059189e-03  1.269934e-03
```

```
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2368 0.0290 20000 0.1449 0.1615 0.1798 0.2928 0.3124 0.3308
#> b        0.4201 0.0273 20000 0.3307 0.3478 0.3659 0.4729 0.4878 0.5049
#> a        0.4671 0.0247 20000 0.3833 0.4023 0.4178 0.5148 0.5300 0.5443
#> Y~~Y     0.6745 0.0240 20000 0.5933 0.6122 0.6265 0.7207 0.7351 0.7512
#> M~~M     0.7818 0.0231 20000 0.7037 0.7191 0.7350 0.8255 0.8381 0.8531
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1962 0.0167 20000 0.1447 0.1544 0.1641 0.2299 0.2401 0.2529
#> direct   0.2368 0.0290 20000 0.1449 0.1615 0.1798 0.2928 0.3124 0.3308
#> total    0.4330 0.0257 20000 0.3478 0.3646 0.3806 0.4823 0.4984 0.5139
```

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