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       0.2684 0.0351 20000 0.1996 0.3370
#> b        0.4688 0.0326 20000 0.4054 0.5328
#> a        0.4530 0.0313 20000 0.3910 0.5147
#> Y~~Y     1.0154 0.0458 20000 0.9261 1.1047
#> M~~M     0.9520 0.0426 20000 0.8672 1.0351
```

```
#> X~~X     0.9737 0.0000 20000 0.9737 0.9737
#> indirect 0.2124 0.0211 20000 0.1726 0.2553
#> direct   0.2684 0.0351 20000 0.1996 0.3370
#> total    0.4807 0.0352 20000 0.4114 0.5488
```

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         1.225313e-03 -4.765715e-04  1.709001e-06  9.877822e-07 -1.494146e-05
#> b         -4.765715e-04  1.055575e-03  3.381486e-06 -1.527644e-05 -1.667865e-06
#> a          1.709001e-06  3.381486e-06  9.763070e-04 -1.042272e-06  5.765571e-06
#> Y~~Y       9.877822e-07 -1.527644e-05 -1.042272e-06  2.067816e-03 -1.851375e-05
#> M~~M      -1.494146e-05 -1.667865e-06  5.765571e-06 -1.851375e-05  1.830704e-03
#> X~~X       1.393254e-05 -6.471605e-06  4.647784e-06 -1.354214e-05  1.068189e-05
#> indirect -2.147223e-04  4.791813e-04  4.591259e-04 -7.294856e-06  2.623207e-06
#> direct     1.225313e-03 -4.765715e-04  1.709001e-06  9.877822e-07 -1.494146e-05
#> total      1.010591e-03  2.609784e-06  4.608349e-04 -6.307074e-06 -1.231825e-05
#>                    X~~X      indirect        direct         total
#> cp         1.393254e-05 -2.147223e-04  1.225313e-03  1.010591e-03
```

```
#> b        -6.471605e-06  4.791813e-04 -4.765715e-04  2.609784e-06
#> a         4.647784e-06  4.591259e-04  1.709001e-06  4.608349e-04
#> Y~~Y     -1.354214e-05 -7.294856e-06  9.877822e-07 -6.307074e-06
#> M~~M      1.068189e-05  2.623207e-06 -1.494146e-05 -1.231825e-05
#> X~~X      1.866801e-03 -6.040455e-07  1.393254e-05  1.332849e-05
#> indirect -6.040455e-07  4.330291e-04 -2.147223e-04  2.183068e-04
#> direct    1.393254e-05 -2.147223e-04  1.225313e-03  1.010591e-03
#> total     1.332849e-05  2.183068e-04  1.010591e-03  1.228898e-03
```

```
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2200 0.0283 20000 0.1211 0.1453 0.1637 0.2752 0.2915 0.3083
#> b        0.4179 0.0270 20000 0.3259 0.3472 0.3642 0.4699 0.4859 0.5023
#> a        0.4165 0.0261 20000 0.3277 0.3485 0.3643 0.4661 0.4829 0.4978
#> Y~~Y     0.7004 0.0242 20000 0.6199 0.6371 0.6515 0.7467 0.7610 0.7782
#> M~~M     0.8265 0.0217 20000 0.7522 0.7668 0.7828 0.8673 0.8785 0.8926
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1740 0.0159 20000 0.1246 0.1345 0.1429 0.2055 0.2159 0.2293
#> direct   0.2200 0.0283 20000 0.1211 0.1453 0.1637 0.2752 0.2915 0.3083
#> total    0.3940 0.0265 20000 0.3052 0.3231 0.3408 0.4442 0.4606 0.4771
```

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*, *56*(3), 1678–1696. 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. (2024). *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