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       0.1979 0.0351 20000 0.1297 0.2674
#> b        0.5299 0.0305 20000 0.4690 0.5893
#> a        0.5587 0.0317 20000 0.4970 0.6209
#> Y~~Y     0.9751 0.0435 20000 0.8913 1.0614
#> M~~M     1.0345 0.0461 20000 0.9439 1.1247
```

```
#> X~~X     1.0267 0.0000 20000 1.0267 1.0267
#> indirect 0.2961 0.0239 20000 0.2504 0.3442
#> direct   0.1979 0.0351 20000 0.1297 0.2674
#> total    0.4940 0.0350 20000 0.4253 0.5633
```

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        1.247388e-03 -5.432114e-04 -1.162381e-05  1.806066e-05 -8.312134e-06
#> b        -5.432114e-04  9.532573e-04  8.786011e-07 -1.256510e-05 -8.766127e-06
#> a        -1.162381e-05  8.786011e-07  1.013588e-03  9.304416e-06 -1.140726e-06
#> Y~~Y      1.806066e-05 -1.256510e-05  9.304416e-06  1.931026e-03 -2.218847e-05
#> M~~M     -8.312134e-06 -8.766127e-06 -1.140726e-06 -2.218847e-05  2.135951e-03
#> X~~X      7.151784e-06 -1.005933e-05 -2.470264e-05  3.965685e-06  8.244087e-06
#> indirect -3.094292e-04  5.327509e-04  5.376804e-04 -2.023025e-06 -6.078824e-06
#> direct    1.247388e-03 -5.432114e-04 -1.162381e-05  1.806066e-05 -8.312134e-06
#> total     9.379587e-04 -1.046045e-05  5.260566e-04  1.603764e-05 -1.439096e-05
#>                   X~~X      indirect        direct         total
#> cp        7.151784e-06 -3.094292e-04  1.247388e-03  9.379587e-04
```

```
#> b        -1.005933e-05  5.327509e-04 -5.432114e-04 -1.046045e-05
#> a        -2.470264e-05  5.376804e-04 -1.162381e-05  5.260566e-04
#> Y~~Y      3.965685e-06 -2.023025e-06  1.806066e-05  1.603764e-05
#> M~~M      8.244087e-06 -6.078824e-06 -8.312134e-06 -1.439096e-05
#> X~~X      2.129462e-03 -1.862737e-05  7.151784e-06 -1.147558e-05
#> indirect -1.862737e-05  5.834060e-04 -3.094292e-04  2.739768e-04
#> direct    7.151784e-06 -3.094292e-04  1.247388e-03  9.379587e-04
#> total    -1.147558e-05  2.739768e-04  9.379587e-04  1.211935e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.1629 0.0289 20000 0.0627 0.0884 0.1066 0.2194 0.2365 0.2608
#> b        0.5009 0.0264 20000 0.4113 0.4316 0.4487 0.5520 0.5679 0.5865
#> a        0.4864 0.0241 20000 0.4014 0.4222 0.4378 0.5322 0.5464 0.5607
#> Y~~Y     0.6432 0.0241 20000 0.5630 0.5798 0.5945 0.6900 0.7053 0.7180
#> M~~M     0.7635 0.0234 20000 0.6856 0.7015 0.7168 0.8083 0.8218 0.8389
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2436 0.0181 20000 0.1883 0.1981 0.2083 0.2796 0.2907 0.3078
#> direct   0.1629 0.0289 20000 0.0627 0.0884 0.1066 0.2194 0.2365 0.2608
#> total    0.4065 0.0262 20000 0.3161 0.3371 0.3545 0.4567 0.4734 0.4904
```

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