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       0.2900 0.0357 20000 0.2200 0.3601
#> b        0.4917 0.0324 20000 0.4275 0.5550
#> a        0.5001 0.0312 20000 0.4392 0.5608
#> Y~~Y     0.9987 0.0447 20000 0.9101 1.0865
#> M~~M     0.9525 0.0430 20000 0.8690 1.0364
```

```
#> X~~X     0.9737 0.0000 20000 0.9737 0.9737
#> indirect 0.2459 0.0224 20000 0.2035 0.2904
#> direct   0.2900 0.0357 20000 0.2200 0.3601
#> total    0.5358 0.0356 20000 0.4658 0.6054
```

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        1.273002e-03 -5.136552e-04  6.265059e-06 -5.205106e-06 -1.656462e-05
#> b        -5.136552e-04  1.051997e-03 -1.278561e-05  1.487780e-05  8.202789e-06
#> a         6.265059e-06 -1.278561e-05  9.901778e-04 -6.962774e-06  7.500225e-06
#> Y~~Y     -5.205106e-06  1.487780e-05 -6.962774e-06  2.000656e-03  2.381919e-05
#> M~~M     -1.656462e-05  8.202789e-06  7.500225e-06  2.381919e-05  1.813228e-03
#> X~~X     -1.841598e-05  8.795574e-06 -3.566459e-06  2.458500e-05 -1.906289e-05
#> indirect -2.535180e-04  5.195288e-04  4.800477e-04  4.757221e-06  8.008702e-06
#> direct    1.273002e-03 -5.136552e-04  6.265059e-06 -5.205106e-06 -1.656462e-05
#> total     1.019484e-03  5.873510e-06  4.863127e-04 -4.478854e-07 -8.555914e-06
#>                   X~~X      indirect        direct         total
#> cp       -1.841598e-05 -2.535180e-04  1.273002e-03  1.019484e-03
```

```
#> b         8.795574e-06  5.195288e-04 -5.136552e-04  5.873510e-06
#> a        -3.566459e-06  4.800477e-04  6.265059e-06  4.863127e-04
#> Y~~Y      2.458500e-05  4.757221e-06 -5.205106e-06 -4.478854e-07
#> M~~M     -1.906289e-05  8.008702e-06 -1.656462e-05 -8.555914e-06
#> X~~X      1.866320e-03  2.987359e-06 -1.841598e-05 -1.542862e-05
#> indirect  2.987359e-06  4.965298e-04 -2.535180e-04  2.430118e-04
#> direct   -1.841598e-05 -2.535180e-04  1.273002e-03  1.019484e-03
#> total    -1.542862e-05  2.430118e-04  1.019484e-03  1.262496e-03
```

```
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2330 0.0281 20000 0.1392 0.1600 0.1777 0.2880 0.3066 0.3244
#> b        0.4378 0.0266 20000 0.3475 0.3667 0.3847 0.4880 0.5043 0.5218
#> a        0.4512 0.0253 20000 0.3653 0.3834 0.4004 0.4994 0.5135 0.5309
#> Y~~Y     0.6620 0.0242 20000 0.5784 0.5980 0.6131 0.7082 0.7221 0.7411
#> M~~M     0.7964 0.0227 20000 0.7181 0.7363 0.7506 0.8396 0.8530 0.8665
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1975 0.0165 20000 0.1442 0.1557 0.1655 0.2303 0.2401 0.2532
#> direct   0.2330 0.0281 20000 0.1392 0.1600 0.1777 0.2880 0.3066 0.3244
#> total    0.4305 0.0256 20000 0.3427 0.3626 0.3793 0.4800 0.4951 0.5103
```

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