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:

```
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.

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 = c(0.001, 0.01, 0.05))

#> Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2630 0.0361 20000 0.1393 0.1688 0.1925 0.3327 0.3561 0.3791
#> b        0.4398 0.0313 20000 0.3370 0.3584 0.3785 0.5015 0.5222 0.5456
#> a        0.4783 0.0333 20000 0.3676 0.3920 0.4131 0.5426 0.5640 0.5849
#> Y~~Y     0.9723 0.0436 20000 0.8302 0.8617 0.8868 1.0578 1.0844 1.1145
#> M~~M     0.9927 0.0449 20000 0.8482 0.8790 0.9044 1.0803 1.1096 1.1363
#> indirect 0.2104 0.0210 20000 0.1463 0.1583 0.1708 0.2528 0.2676 0.2841
#> direct   0.2630 0.0361 20000 0.1393 0.1688 0.1925 0.3327 0.3561 0.3791
```

```
#> total    0.4734 0.0361 20000 0.3559 0.3809 0.4027 0.5434 0.5651 0.5901
```

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 = c(0.001, 0.01, 0.05))
vcov(unstd)

#>                      cp             b            a          Y~~Y          M~~M
#> cp         1.298947e-03 -4.616340e-04 6.566082e-06 -1.341828e-05 -1.446039e-05
#> b         -4.616340e-04  9.641954e-04 6.162039e-06 -1.808978e-06  1.174851e-05
#> a          6.566082e-06  6.162039e-06 1.104344e-03  4.055176e-06  2.165742e-07
#> Y~~Y      -1.341828e-05 -1.808978e-06 4.055176e-06  1.939241e-03  7.935158e-06
#> M~~M      -1.446039e-05  1.174851e-05 2.165742e-07  7.935158e-06  1.986625e-03
#> X~~X       7.510104e-07 -6.561426e-06 2.111693e-05  1.827008e-05 -1.560982e-06
#> indirect  -2.178342e-04  4.641543e-04 4.889620e-04  2.428007e-07  4.754201e-06
#> direct     1.298947e-03 -4.616340e-04 6.566082e-06 -1.341828e-05 -1.446039e-05
#> total      1.081113e-03  2.520281e-06 4.955281e-04 -1.317548e-05 -9.706194e-06
#>                    X~~X      indirect        direct         total
#> cp         7.510104e-07 -2.178342e-04  1.298947e-03  1.081113e-03
#> b         -6.561426e-06  4.641543e-04 -4.616340e-04  2.520281e-06
#> a          2.111693e-05  4.889620e-04  6.566082e-06  4.955281e-04
#> Y~~Y       1.827008e-05  2.428007e-07 -1.341828e-05 -1.317548e-05
```

```
#> M~~M     -1.560982e-06  4.754201e-06 -1.446039e-05 -9.706194e-06
#> X~~X      1.628534e-03  5.971948e-06  7.510104e-07  6.722958e-06
#> indirect  5.971948e-06  4.383771e-04 -2.178342e-04  2.205429e-04
#> direct    7.510104e-07 -2.178342e-04  1.298947e-03  1.081113e-03
#> total     6.722958e-06  2.205429e-04  1.081113e-03  1.301656e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>              est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp        0.2137 0.0288 20000 0.1167 0.1380 0.1570 0.2695 0.2877 0.3070
#> b         0.4120 0.0272 20000 0.3203 0.3406 0.3583 0.4647 0.4807 0.4989
#> a         0.4149 0.0263 20000 0.3252 0.3458 0.3619 0.4650 0.4805 0.4971
#> Y~~Y      0.7116 0.0242 20000 0.6300 0.6476 0.6631 0.7580 0.7716 0.7845
#> M~~M      0.8279 0.0218 20000 0.7529 0.7691 0.7838 0.8691 0.8804 0.8942
#> X~~X      1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1709 0.0159 20000 0.1230 0.1324 0.1404 0.2028 0.2145 0.2261
#> direct    0.2137 0.0288 20000 0.1167 0.1380 0.1570 0.2695 0.2877 0.3070
#> total     0.3846 0.0271 20000 0.2945 0.3121 0.3301 0.4365 0.4525 0.4716
```

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

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. (2022). *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