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       0.2216 0.0363 20000 0.1065 0.1275 0.1500 0.2921 0.3142 0.3386
#> b        0.4646 0.0304 20000 0.3666 0.3868 0.4054 0.5242 0.5427 0.5646
#> a        0.5041 0.0341 20000 0.3903 0.4161 0.4374 0.5717 0.5917 0.6173
#> Y~~Y     0.9958 0.0446 20000 0.8509 0.8795 0.9068 1.0818 1.1107 1.1372
#> M~~M     1.0633 0.0482 20000 0.9060 0.9394 0.9695 1.1591 1.1877 1.2174
#> indirect 0.2342 0.0219 20000 0.1693 0.1816 0.1928 0.2783 0.2939 0.3145
#> direct   0.2216 0.0363 20000 0.1065 0.1275 0.1500 0.2921 0.3142 0.3386
```

```
#> total    0.4558 0.0363 20000 0.3375 0.3580 0.3847 0.5273 0.5497 0.5760
```

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        1.315850e-03 -4.636186e-04 -1.072275e-05 -4.116232e-06 -2.046418e-05
#> b        -4.636186e-04  9.437244e-04  1.143637e-05  1.679943e-05 -2.650908e-06
#> a        -1.072275e-05  1.143637e-05  1.139880e-03 -4.781269e-06  1.064716e-05
#> Y~~Y     -4.116232e-06  1.679943e-05 -4.781269e-06  2.000069e-03 -7.983082e-06
#> M~~M     -2.046418e-05 -2.650908e-06  1.064716e-05 -7.983082e-06  2.275534e-03
#> X~~X     -1.099100e-05  2.380082e-06  1.282772e-05 -1.840637e-05  4.685796e-06
#> indirect -2.388469e-04  4.816551e-04  5.350540e-04  6.573539e-06  3.893553e-06
#> direct    1.315850e-03 -4.636186e-04 -1.072275e-05 -4.116232e-06 -2.046418e-05
#> total     1.077004e-03  1.803645e-05  5.243313e-04  2.457307e-06 -1.657063e-05
#>                    X~~X      indirect        direct         total
#> cp       -1.099100e-05 -2.388469e-04  1.315850e-03  1.077004e-03
#> b         2.380082e-06  4.816551e-04 -4.636186e-04  1.803645e-05
#> a         1.282772e-05  5.350540e-04 -1.072275e-05  5.243313e-04
#> Y~~Y     -1.840637e-05  6.573539e-06 -4.116232e-06  2.457307e-06
```

```
#> M~~M      4.685796e-06  3.893553e-06 -2.046418e-05 -1.657063e-05
#> X~~X      1.740597e-03  6.716136e-06 -1.099100e-05 -4.274869e-06
#> indirect  6.716136e-06  4.926443e-04 -2.388469e-04  2.537974e-04
#> direct   -1.099100e-05 -2.388469e-04  1.315850e-03  1.077004e-03
#> total    -4.274869e-06  2.537974e-04  1.077004e-03  1.330801e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>              est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp        0.1798 0.0291 20000 0.0866 0.1038 0.1222 0.2366 0.2538 0.2790
#> b         0.4448 0.0269 20000 0.3538 0.3748 0.3910 0.4963 0.5119 0.5274
#> a         0.4273 0.0259 20000 0.3399 0.3577 0.3756 0.4772 0.4925 0.5092
#> Y~~Y      0.7015 0.0244 20000 0.6203 0.6364 0.6528 0.7482 0.7626 0.7790
#> M~~M      0.8174 0.0221 20000 0.7407 0.7574 0.7723 0.8590 0.8720 0.8845
#> X~~X      1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1901 0.0167 20000 0.1388 0.1484 0.1577 0.2231 0.2343 0.2469
#> direct    0.1798 0.0291 20000 0.0866 0.1038 0.1222 0.2366 0.2538 0.2790
#> total     0.3698 0.0274 20000 0.2774 0.2974 0.3144 0.4222 0.4384 0.4560
```

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