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         0.2224 0.0347 20000 0.1544 0.2902
#> b          0.4928 0.0309 20000 0.4324 0.5537
#> a          0.4914 0.0319 20000 0.4280 0.5543
#> Y~~Y       0.9996 0.0447 20000 0.9125 1.0872
#> M~~M       1.0319 0.0463 20000 0.9414 1.1239
```

```
#> X~~X     1.0200 0.0000 20000 1.0200 1.0200
#> indirect 0.2421 0.0218 20000 0.2010 0.2858
#> direct   0.2224 0.0347 20000 0.1544 0.2902
#> total    0.4646 0.0347 20000 0.3958 0.5332
```

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.

```r
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        1.201065e-03 -4.764097e-04 -1.340611e-05  1.089074e-05 -1.772085e-06
#> b        -4.764097e-04  9.800866e-04  8.693837e-06  1.071566e-05 -9.242808e-07
#> a        -1.340611e-05  8.693837e-06  1.007072e-03 -7.981692e-06  8.284826e-06
#> Y~~Y      1.089074e-05  1.071566e-05 -7.981692e-06  2.017004e-03 -2.459042e-06
#> M~~M     -1.772085e-06 -9.242808e-07  8.284826e-06 -2.459042e-06  2.113629e-03
#> X~~X     -7.207768e-07  1.735376e-05 -6.847990e-06  1.108161e-05  6.597167e-06
#> indirect -2.406844e-04  4.859878e-04  5.008314e-04  1.234077e-06  3.974960e-06
#> direct    1.201065e-03 -4.764097e-04 -1.340611e-05  1.089074e-05 -1.772085e-06
#> total     9.603810e-04  9.578146e-06  4.874253e-04  1.212482e-05  2.202875e-06
#>                   X~~X      indirect        direct        total
#> cp       -7.207768e-07 -2.406844e-04  1.201065e-03 9.603810e-04
```

```
#> b         1.735376e-05  4.859878e-04 -4.764097e-04 9.578146e-06
#> a        -6.847990e-06  5.008314e-04 -1.340611e-05 4.874253e-04
#> Y~~Y      1.108161e-05  1.234077e-06  1.089074e-05 1.212482e-05
#> M~~M      6.597167e-06  3.974960e-06 -1.772085e-06 2.202875e-06
#> X~~X      2.069825e-03  5.232339e-06 -7.207768e-07 4.511563e-06
#> indirect  5.232339e-06  4.867980e-04 -2.406844e-04 2.461136e-04
#> direct   -7.207768e-07 -2.406844e-04  1.201065e-03 9.603810e-04
#> total     4.511563e-06  2.461136e-04  9.603810e-04 1.206495e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.1853 0.0286 20000 0.0909 0.1092 0.1281 0.2408 0.2566 0.2762
#> b        0.4595 0.0266 20000 0.3699 0.3895 0.4063 0.5106 0.5262 0.5441
#> a        0.4389 0.0254 20000 0.3503 0.3708 0.3877 0.4873 0.5040 0.5172
#> Y~~Y     0.6799 0.0243 20000 0.6023 0.6177 0.6316 0.7266 0.7410 0.7547
#> M~~M     0.8073 0.0222 20000 0.7325 0.7459 0.7625 0.8497 0.8625 0.8773
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2017 0.0170 20000 0.1502 0.1595 0.1686 0.2354 0.2458 0.2583
#> direct   0.1853 0.0286 20000 0.0909 0.1092 0.1281 0.2408 0.2566 0.2762
#> total    0.3869 0.0266 20000 0.2977 0.3153 0.3333 0.4373 0.4532 0.4706
```

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