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       0.2250 0.0352 20000 0.1564 0.2943
#> b        0.4701 0.0326 20000 0.4058 0.5340
#> a        0.4997 0.0302 20000 0.4406 0.5587
#> Y~~Y     1.0070 0.0452 20000 0.9183 1.0958
#> M~~M     0.9391 0.0422 20000 0.8574 1.0223
```

```
#> X~~X     1.0267 0.0000 20000 1.0267 1.0267
#> indirect 0.2349 0.0216 20000 0.1940 0.2785
#> direct   0.2250 0.0352 20000 0.1564 0.2943
#> total    0.4599 0.0345 20000 0.3934 0.5282
```

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        1.242906e-03 -5.335424e-04  5.099753e-06  1.322885e-05  4.421641e-07
#> b        -5.335424e-04  1.064732e-03 -7.650311e-06  2.764767e-07  1.134356e-05
#> a         5.099753e-06 -7.650311e-06  9.111222e-04 -2.272599e-05  6.375408e-06
#> Y~~Y      1.322885e-05  2.764767e-07 -2.272599e-05  2.060413e-03  2.464559e-05
#> M~~M      4.421641e-07  1.134356e-05  6.375408e-06  2.464559e-05  1.754257e-03
#> X~~X      9.092943e-06  6.010804e-06  5.896421e-06 -1.486709e-05 -2.235525e-05
#> indirect -2.644616e-04  5.293942e-04  4.238108e-04 -1.061396e-05  9.072968e-06
#> direct    1.242906e-03 -5.335424e-04  5.099753e-06  1.322885e-05  4.421641e-07
#> total     9.784446e-04 -4.148136e-06  4.289105e-04  2.614891e-06  9.515132e-06
#>                   X~~X      indirect        direct         total
#> cp        9.092943e-06 -2.644616e-04  1.242906e-03  9.784446e-04
```

```
#> b         6.010804e-06  5.293942e-04 -5.335424e-04 -4.148136e-06
#> a         5.896421e-06  4.238108e-04  5.099753e-06  4.289105e-04
#> Y~~Y     -1.486709e-05 -1.061396e-05  1.322885e-05  2.614891e-06
#> M~~M     -2.235525e-05  9.072968e-06  4.421641e-07  9.515132e-06
#> X~~X      2.111328e-03  5.503861e-06  9.092943e-06  1.459680e-05
#> indirect  5.503861e-06  4.648772e-04 -2.644616e-04  2.004156e-04
#> direct    9.092943e-06 -2.644616e-04  1.242906e-03  9.784446e-04
#> total     1.459680e-05  2.004156e-04  9.784446e-04  1.178860e-03
```

```
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.1905 0.0295 20000 0.0964 0.1146 0.1327 0.2473 0.2665 0.2851
#> b        0.4295 0.0277 20000 0.3393 0.3564 0.3743 0.4822 0.4991 0.5183
#> a        0.4631 0.0248 20000 0.3801 0.3980 0.4139 0.5114 0.5265 0.5431
#> Y~~Y     0.7034 0.0242 20000 0.6172 0.6390 0.6552 0.7497 0.7638 0.7790
#> M~~M     0.7855 0.0230 20000 0.7050 0.7228 0.7385 0.8287 0.8416 0.8555
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1989 0.0170 20000 0.1465 0.1571 0.1662 0.2331 0.2442 0.2566
#> direct   0.1905 0.0295 20000 0.0964 0.1146 0.1327 0.2473 0.2665 0.2851
#> total    0.3895 0.0268 20000 0.3022 0.3198 0.3358 0.4415 0.4566 0.4752
```

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