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       0.1729 0.0344 20000 0.1047 0.2405
#> b        0.5689 0.0321 20000 0.5062 0.6320
#> a        0.4747 0.0302 20000 0.4146 0.5332
#> Y~~Y     0.9744 0.0436 20000 0.8884 1.0602
#> M~~M     0.9398 0.0416 20000 0.8582 1.0221
```

```
#> X~~X     1.0231 0.0000 20000 1.0231 1.0231
#> indirect 0.2700 0.0229 20000 0.2266 0.3170
#> direct   0.1729 0.0344 20000 0.1047 0.2405
#> total    0.4429 0.0355 20000 0.3725 0.5123
```

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        1.180530e-03 -4.936396e-04 -7.508230e-06 -9.221576e-06  2.298480e-06
#> b        -4.936396e-04  1.051544e-03  7.053770e-06  7.299499e-06  1.155124e-05
#> a        -7.508230e-06  7.053770e-06  9.280787e-04  2.059317e-06  1.007318e-05
#> Y~~Y     -9.221576e-06  7.299499e-06  2.059317e-06  1.875264e-03  1.452651e-05
#> M~~M      2.298480e-06  1.155124e-05  1.007318e-05  1.452651e-05  1.778793e-03
#> X~~X     -1.709801e-06  1.469474e-05 -1.278219e-06  5.983750e-06 -8.441276e-06
#> indirect -2.382347e-04  5.029975e-04  5.316741e-04  4.823751e-06  1.102335e-05
#> direct    1.180530e-03 -4.936396e-04 -7.508230e-06 -9.221576e-06  2.298480e-06
#> total     9.422951e-04  9.357890e-06  5.241658e-04 -4.397825e-06  1.332183e-05
#>                   X~~X      indirect        direct         total
#> cp       -1.709801e-06 -2.382347e-04  1.180530e-03  9.422951e-04
```

```
#> b          1.469474e-05  5.029975e-04 -4.936396e-04  9.357890e-06
#> a         -1.278219e-06  5.316741e-04 -7.508230e-06  5.241658e-04
#> Y~~Y       5.983750e-06  4.823751e-06 -9.221576e-06 -4.397825e-06
#> M~~M      -8.441276e-06  1.102335e-05  2.298480e-06  1.332183e-05
#> X~~X       2.112052e-03  6.555401e-06 -1.709801e-06  4.845600e-06
#> indirect   6.555401e-06  5.423177e-04 -2.382347e-04  3.040830e-04
#> direct    -1.709801e-06 -2.382347e-04  1.180530e-03  9.422951e-04
#> total      4.845600e-06  3.040830e-04  9.422951e-04  1.246378e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>              est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp        0.1438 0.0284 20000 0.0510 0.0710 0.0873 0.1994 0.2178 0.2361
#> b         0.5060 0.0257 20000 0.4105 0.4373 0.4543 0.5558 0.5726 0.5906
#> a         0.4438 0.0255 20000 0.3607 0.3772 0.3931 0.4926 0.5092 0.5275
#> Y~~Y      0.6587 0.0244 20000 0.5741 0.5948 0.6095 0.7051 0.7207 0.7413
#> M~~M      0.8030 0.0226 20000 0.7217 0.7408 0.7573 0.8455 0.8577 0.8699
#> X~~X      1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect  0.2246 0.0178 20000 0.1689 0.1801 0.1900 0.2600 0.2717 0.2864
#> direct    0.1438 0.0284 20000 0.0510 0.0710 0.0873 0.1994 0.2178 0.2361
#> total     0.3683 0.0273 20000 0.2779 0.2964 0.3139 0.4205 0.4370 0.4536
```

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