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       0.3083 0.0358 20000 0.2379 0.3790
#> b        0.4787 0.0309 20000 0.4188 0.5392
#> a        0.5376 0.0326 20000 0.4748 0.6023
#> Y~~Y     0.9857 0.0443 20000 0.8983 1.0720
#> M~~M     1.0192 0.0454 20000 0.9296 1.1087
```

```
#> X~~X     0.9753 0.0000 20000 0.9753 0.9753
#> indirect 0.2574 0.0228 20000 0.2141 0.3039
#> direct   0.3083 0.0358 20000 0.2379 0.3790
#> total    0.5657 0.0352 20000 0.4976 0.6353
```

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        1.272891e-03 -5.068481e-04  1.034055e-05 -8.480185e-06 -8.648039e-07
#> b        -5.068481e-04  9.520114e-04  3.993674e-06  1.467242e-05  1.700575e-05
#> a         1.034055e-05  3.993674e-06  1.037166e-03  8.991439e-06 -2.189206e-06
#> Y~~Y     -8.480185e-06  1.467242e-05  8.991439e-06  1.945073e-03 -6.134803e-06
#> M~~M     -8.648039e-07  1.700575e-05 -2.189206e-06 -6.134803e-06  2.086706e-03
#> X~~X      3.060753e-06 -1.188971e-05 -2.201021e-06  1.053228e-05 -1.453283e-05
#> indirect -2.674402e-04  5.130541e-04  4.984255e-04  1.191840e-05  8.368105e-06
#> direct    1.272891e-03 -5.068481e-04  1.034055e-05 -8.480185e-06 -8.648039e-07
#> total     1.005451e-03  6.206023e-06  5.087660e-04  3.438214e-06  7.503302e-06
#>                    X~~X      indirect        direct         total
#> cp        3.060753e-06 -2.674402e-04  1.272891e-03  1.005451e-03
```

```
#> b         -1.188971e-05  5.130541e-04 -5.068481e-04  6.206023e-06
#> a         -2.201021e-06  4.984255e-04  1.034055e-05  5.087660e-04
#> Y~~Y       1.053228e-05  1.191840e-05 -8.480185e-06  3.438214e-06
#> M~~M      -1.453283e-05  8.368105e-06 -8.648039e-07  7.503302e-06
#> X~~X       1.895551e-03 -7.180774e-06  3.060753e-06 -4.120022e-06
#> indirect -7.180774e-06  5.149466e-04 -2.674402e-04  2.475064e-04
#> direct    3.060753e-06 -2.674402e-04  1.272891e-03  1.005451e-03
#> total    -4.120022e-06  2.475064e-04  1.005451e-03  1.252957e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>              est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp        0.2461 0.0279 20000 0.1490 0.1734 0.1913 0.3008 0.3171 0.3377
#> b         0.4412 0.0264 20000 0.3493 0.3709 0.3880 0.4910 0.5087 0.5274
#> a         0.4655 0.0247 20000 0.3804 0.3995 0.4157 0.5131 0.5265 0.5479
#> Y~~Y      0.6437 0.0242 20000 0.5630 0.5817 0.5961 0.6907 0.7043 0.7188
#> M~~M      0.7833 0.0230 20000 0.6998 0.7228 0.7367 0.8272 0.8404 0.8553
#> X~~X      1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2054 0.0166 20000 0.1538 0.1636 0.1728 0.2382 0.2488 0.2619
#> direct   0.2461 0.0279 20000 0.1490 0.1734 0.1913 0.3008 0.3171 0.3377
#> total    0.4515 0.0251 20000 0.3660 0.3849 0.4001 0.4990 0.5135 0.5276
```

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