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       0.2646 0.0341 20000 0.1982 0.3320
#> b        0.4902 0.0307 20000 0.4299 0.5507
#> a        0.4749 0.0319 20000 0.4122 0.5377
#> Y~~Y     0.9684 0.0433 20000 0.8838 1.0531
#> M~~M     1.0211 0.0460 20000 0.9312 1.1120
```

```
#> X~~X     1.0257 0.0000 20000 1.0257 1.0257
#> indirect 0.2328 0.0216 20000 0.1928 0.2766
#> direct   0.2646 0.0341 20000 0.1982 0.3320
#> total    0.4975 0.0346 20000 0.4304 0.5656
```

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        1.162572e-03 -4.541802e-04  3.032236e-06  6.622871e-07  1.309437e-06
#> b        -4.541802e-04  9.634692e-04 -1.135966e-05 -7.927341e-06  6.568289e-06
#> a         3.032236e-06 -1.135966e-05  9.727951e-04 -2.075129e-05 -1.497071e-05
#> Y~~Y      6.622871e-07 -7.927341e-06 -2.075129e-05  1.865077e-03 -2.689177e-06
#> M~~M      1.309437e-06  6.568289e-06 -1.497071e-05 -2.689177e-06  2.046887e-03
#> X~~X      2.229382e-05 -5.486232e-06 -1.206341e-06 -1.546843e-05 -1.887899e-05
#> indirect -2.141804e-04  4.522494e-04  4.716192e-04 -1.359472e-05 -4.139786e-06
#> direct    1.162572e-03 -4.541802e-04  3.032236e-06  6.622871e-07  1.309437e-06
#> total     9.483912e-04 -1.930838e-06  4.746514e-04 -1.293243e-05 -2.830349e-06
#>                    X~~X      indirect        direct         total
#> cp        2.229382e-05 -2.141804e-04  1.162572e-03  9.483912e-04
```

```
#> b         -5.486232e-06  4.522494e-04 -4.541802e-04 -1.930838e-06
#> a         -1.206341e-06  4.716192e-04  3.032236e-06  4.746514e-04
#> Y~~Y      -1.546843e-05 -1.359472e-05  6.622871e-07 -1.293243e-05
#> M~~M      -1.887899e-05 -4.139786e-06  1.309437e-06 -2.830349e-06
#> X~~X       2.048471e-03 -3.090316e-06  2.229382e-05  1.920351e-05
#> indirect -3.090316e-06  4.471103e-04 -2.141804e-04  2.329299e-04
#> direct    2.229382e-05 -2.141804e-04  1.162572e-03  9.483912e-04
#> total     1.920351e-05  2.329299e-04  9.483912e-04  1.181321e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>              est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp        0.2212 0.0281 20000 0.1308 0.1478 0.1663 0.2762 0.2921 0.3110
#> b         0.4529 0.0263 20000 0.3606 0.3833 0.4006 0.5036 0.5196 0.5356
#> a         0.4298 0.0256 20000 0.3451 0.3638 0.3792 0.4796 0.4939 0.5076
#> Y~~Y      0.6598 0.0244 20000 0.5765 0.5950 0.6101 0.7066 0.7211 0.7401
#> M~~M      0.8153 0.0220 20000 0.7424 0.7561 0.7700 0.8562 0.8676 0.8809
#> X~~X      1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1947 0.0163 20000 0.1428 0.1543 0.1633 0.2272 0.2374 0.2523
#> direct    0.2212 0.0281 20000 0.1308 0.1478 0.1663 0.2762 0.2921 0.3110
#> total     0.4159 0.0262 20000 0.3303 0.3466 0.3628 0.4665 0.4816 0.5010
```

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