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 = c(0.001, 0.01, 0.05))

#> Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2321 0.0339 20000 0.1126 0.1457 0.1655 0.2989 0.3196 0.3408
#> b        0.5103 0.0301 20000 0.4188 0.4350 0.4517 0.5696 0.5878 0.6119
#> a        0.4237 0.0330 20000 0.3144 0.3390 0.3587 0.4879 0.5105 0.5333
#> Y~~Y     0.9333 0.0418 20000 0.7894 0.8274 0.8505 1.0152 1.0419 1.0702
#> M~~M     1.0167 0.0455 20000 0.8626 0.8988 0.9272 1.1051 1.1349 1.1610
```

```
#> X~~X     0.9401 0.0000 20000 0.9401 0.9401 0.9401 0.9401 0.9401 0.9401
#> indirect 0.2163 0.0212 20000 0.1521 0.1644 0.1763 0.2594 0.2741 0.2924
#> direct   0.2321 0.0339 20000 0.1126 0.1457 0.1655 0.2989 0.3196 0.3408
#> total    0.4484 0.0356 20000 0.3341 0.3575 0.3789 0.5190 0.5407 0.5652
```

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 = c(0.001, 0.01, 0.05))
vcov(unstd)

#>                      cp             b             a          Y~~Y          M~~M
#> cp        1.143324e-03 -3.813178e-04 -6.365104e-06  1.789535e-07 4.580055e-06
#> b        -3.813178e-04  9.121698e-04 -3.338436e-06  3.392521e-07 9.706387e-07
#> a        -6.365104e-06 -3.338436e-06  1.072746e-03 -4.740374e-06 3.043147e-06
#> Y~~Y      1.789535e-07  3.392521e-07 -4.740374e-06  1.734546e-03 1.571921e-05
#> M~~M      4.580055e-06  9.706387e-07  3.043147e-06  1.571921e-05 2.065415e-03
#> X~~X     -1.525884e-06  2.766045e-06 -5.391959e-06  2.911062e-05 6.286989e-06
#> indirect -1.647930e-04  3.847711e-04  5.461518e-04 -2.383720e-06 1.660159e-06
#> direct    1.143324e-03 -3.813178e-04 -6.365104e-06  1.789535e-07 4.580055e-06
#> total     9.785313e-04  3.453334e-06  5.397867e-04 -2.204766e-06 6.240214e-06
#>                    X~~X      indirect        direct         total
#> cp       -1.525884e-06 -1.647930e-04  1.143324e-03  9.785313e-04
```

```
#> b         2.766045e-06  3.847711e-04 -3.813178e-04  3.453334e-06
#> a        -5.391959e-06  5.461518e-04 -6.365104e-06  5.397867e-04
#> Y~~Y      2.911062e-05 -2.383720e-06  1.789535e-07 -2.204766e-06
#> M~~M      6.286989e-06  1.660159e-06  4.580055e-06  6.240214e-06
#> X~~X      1.782912e-03 -1.716801e-06 -1.525884e-06 -3.242685e-06
#> indirect -1.716801e-06  4.427759e-04 -1.647930e-04  2.779829e-04
#> direct   -1.525884e-06 -1.647930e-04  1.143324e-03  9.785313e-04
#> total    -3.242685e-06  2.779829e-04  9.785313e-04  1.256514e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.1911 0.0275 20000 0.1024 0.1203 0.1377 0.2454 0.2613 0.2780
#> b        0.4718 0.0252 20000 0.3888 0.4062 0.4224 0.5205 0.5345 0.5542
#> a        0.3774 0.0270 20000 0.2902 0.3071 0.3244 0.4300 0.4459 0.4648
#> Y~~Y     0.6729 0.0242 20000 0.5909 0.6092 0.6239 0.7190 0.7336 0.7453
#> M~~M     0.8576 0.0204 20000 0.7840 0.8012 0.8151 0.8948 0.9057 0.9158
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1780 0.0161 20000 0.1284 0.1388 0.1472 0.2103 0.2208 0.2316
#> direct   0.1911 0.0275 20000 0.1024 0.1203 0.1377 0.2454 0.2613 0.2780
#> total    0.3691 0.0271 20000 0.2810 0.2985 0.3154 0.4210 0.4372 0.4555
```

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