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:

```
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.

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       0.2190 0.0353 20000 0.1033 0.1280 0.1497 0.2883 0.3095 0.3330
#> b        0.4968 0.0320 20000 0.3942 0.4160 0.4344 0.5596 0.5793 0.5981
#> a        0.5173 0.0307 20000 0.4155 0.4383 0.4569 0.5775 0.5975 0.6217
#> Y~~Y     1.0067 0.0446 20000 0.8549 0.8917 0.9192 1.0944 1.1188 1.1514
#> M~~M     0.9826 0.0436 20000 0.8446 0.8703 0.8977 1.0688 1.0948 1.1243
#> indirect 0.2570 0.0226 20000 0.1906 0.2030 0.2143 0.3027 0.3173 0.3376
#> direct   0.2190 0.0353 20000 0.1033 0.1280 0.1497 0.2883 0.3095 0.3330
```

```
#> total    0.4760 0.0349 20000 0.3638 0.3863 0.4073 0.5442 0.5649 0.5872
```

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))
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.1839 0.0296 20000 0.0836 0.1075 0.1253 0.2412 0.2587 0.2790
#> b        0.4580 0.0273 20000 0.3672 0.3867 0.4040 0.5116 0.5279 0.5457
#> a        0.4713 0.0244 20000 0.3901 0.4062 0.4224 0.5180 0.5325 0.5481
#> Y~~Y     0.6771 0.0243 20000 0.5914 0.6128 0.6285 0.7231 0.7375 0.7562
#> M~~M     0.7779 0.0229 20000 0.6996 0.7164 0.7317 0.8216 0.8350 0.8479
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2159 0.0175 20000 0.1608 0.1723 0.1818 0.2512 0.2634 0.2767
#> direct   0.1839 0.0296 20000 0.0836 0.1075 0.1253 0.2412 0.2587 0.2790
#> total    0.3998 0.0265 20000 0.3095 0.3300 0.3465 0.4504 0.4671 0.4841
```

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

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. (2022). *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