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.

```r
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`.

```r
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.

```r
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       0.2260 0.0347 20000 0.1133 0.1357 0.1574 0.2936 0.3148 0.3449
#> b        0.4988 0.0325 20000 0.3906 0.4151 0.4356 0.5623 0.5821 0.6042
#> a        0.4709 0.0304 20000 0.3735 0.3932 0.4116 0.5307 0.5497 0.5713
#> Y~~Y     1.0124 0.0454 20000 0.8705 0.8965 0.9237 1.1014 1.1314 1.1653
#> M~~M     0.9589 0.0434 20000 0.8168 0.8477 0.8745 1.0440 1.0725 1.1079
#> indirect 0.2349 0.0214 20000 0.1723 0.1821 0.1946 0.2782 0.2925 0.3089
#> direct   0.2260 0.0347 20000 0.1133 0.1357 0.1574 0.2936 0.3148 0.3449
```

```
#> total    0.4609 0.0347 20000 0.3488 0.3712 0.3940 0.5294 0.5510 0.5688
```

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       0.1901 0.0290 20000 0.0939 0.1141 0.1328 0.2461 0.2630 0.2847
#> b        0.4484 0.0270 20000 0.3606 0.3771 0.3949 0.5008 0.5167 0.5321
#> a        0.4406 0.0255 20000 0.3579 0.3746 0.3894 0.4895 0.5054 0.5202
#> Y~~Y     0.6877 0.0242 20000 0.6043 0.6236 0.6394 0.7343 0.7483 0.7619
#> M~~M     0.8059 0.0225 20000 0.7294 0.7446 0.7604 0.8484 0.8597 0.8719
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1976 0.0168 20000 0.1441 0.1559 0.1653 0.2308 0.2422 0.2551
#> direct   0.1901 0.0290 20000 0.0939 0.1141 0.1328 0.2461 0.2630 0.2847
#> total    0.3877 0.0269 20000 0.2961 0.3154 0.3334 0.4380 0.4545 0.4716
```

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