semmcci: Monte Carlo Confidence Intervals

Ivan Jacob Agaloos Pesigan

Installation

You can install the CRAN release of `semmcci` with:

```r
install.packages("semmcci")
```

You can install the development version of `semmcci` from GitHub with:

```r
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       0.2643 0.03424 20000 0.1509 0.1780 0.1969 0.3307 0.3519 0.3739
#> b        0.4456 0.03076 20000 0.3456 0.3662 0.3855 0.5060 0.5248 0.5434
#> a        0.4478 0.03192 20000 0.3450 0.3670 0.3858 0.5102 0.5310 0.5532
#> Y~~Y     0.9583 0.04289 20000 0.8198 0.8479 0.8733 1.0419 1.0670 1.0956
#> M~~M     1.0131 0.04554 20000 0.8685 0.8968 0.9237 1.1021 1.1311 1.1600
#> indirect 0.1995 0.01987 20000 0.1398 0.1515 0.1620 0.2406 0.2541 0.2680
#> direct   0.2643 0.03424 20000 0.1509 0.1780 0.1969 0.3307 0.3519 0.3739
```

```
#> total    0.4639 0.03427 20000 0.3545 0.3767 0.3962 0.5311 0.5510 0.5673
```

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       0.2250 0.0284 20000 0.1301 0.1506 0.1687 0.2806 0.2986 0.3167
#> b        0.4186 0.0269 20000 0.3297 0.3482 0.3653 0.4698 0.4851 0.5031
#> a        0.4056 0.0262 20000 0.3186 0.3359 0.3537 0.4560 0.4708 0.4876
#> Y~~Y     0.6977 0.0244 20000 0.6185 0.6345 0.6487 0.7444 0.7581 0.7785
#> M~~M     0.8355 0.0212 20000 0.7622 0.7784 0.7920 0.8749 0.8871 0.8985
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1698 0.0158 20000 0.1190 0.1308 0.1394 0.2011 0.2107 0.2212
#> direct   0.2250 0.0284 20000 0.1301 0.1506 0.1687 0.2806 0.2986 0.3167
#> total    0.3948 0.0267 20000 0.3052 0.3233 0.3407 0.4457 0.4611 0.4803
```

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