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 (2024).

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.

```r
library(semmcci)
library(lavaan)
```

Data

```r
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       0.2574 0.0350 20000 0.1880 0.3269
#> b        0.5258 0.0331 20000 0.4604 0.5903
#> a        0.5243 0.0292 20000 0.4670 0.5810
#> Y~~Y     1.0145 0.0451 20000 0.9279 1.1038
#> M~~M     0.9099 0.0405 20000 0.8299 0.9889
#> X~~X     1.0816 0.0000 20000 1.0816 1.0816
#> indirect 0.2757 0.0232 20000 0.2316 0.3226
#> direct   0.2574 0.0350 20000 0.1880 0.3269
#> total    0.5331 0.0341 20000 0.4660 0.5992
```

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        1.246630e-03 -5.964375e-04 -2.308608e-06 -5.275647e-06  7.826043e-06
#> b        -5.964375e-04  1.125408e-03  9.374162e-06  1.802717e-05 -7.799464e-07
#> a        -2.308608e-06  9.374162e-06  8.482760e-04 -7.009591e-06 -1.472575e-07
#> Y~~Y     -5.275647e-06  1.802717e-05 -7.009591e-06  2.067194e-03  1.103505e-05
#> M~~M      7.826043e-06 -7.799464e-07 -1.472575e-07  1.103505e-05  1.608832e-03
#> X~~X     -1.519385e-05  1.936833e-05 -9.091972e-07  4.228669e-06  5.540622e-06
#> indirect -3.141821e-04  5.953213e-04  4.511887e-04  5.669676e-06 -4.224236e-07
#> direct    1.246630e-03 -5.964375e-04 -2.308608e-06 -5.275647e-06  7.826043e-06
#> total     9.324484e-04 -1.116280e-06  4.488801e-04  3.940292e-07  7.403619e-06
#>                   X~~X      indirect        direct         total
#> cp       -1.519385e-05 -3.141821e-04  1.246630e-03  9.324484e-04
#> b          1.936833e-05  5.953213e-04 -5.964375e-04 -1.116280e-06
#> a         -9.091972e-07  4.511887e-04 -2.308608e-06  4.488801e-04
#> Y~~Y       4.228669e-06  5.669676e-06 -5.275647e-06  3.940292e-07
#> M~~M       5.540622e-06 -4.224236e-07  7.826043e-06  7.403619e-06
#> X~~X       2.320328e-03  9.070718e-06 -1.519385e-05 -6.123131e-06
#> indirect   9.070718e-06  5.506341e-04 -3.141821e-04  2.364520e-04
#> direct    -1.519385e-05 -3.141821e-04  1.246630e-03  9.324484e-04
#> total     -6.123131e-06  2.364520e-04  9.324484e-04  1.168900e-03
```

```
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2134 0.0289 20000 0.1202 0.1406 0.1573 0.2697 0.2872 0.3041
#> b        0.4606 0.0270 20000 0.3717 0.3908 0.4070 0.5124 0.5289 0.5474
#> a        0.4963 0.0239 20000 0.4166 0.4330 0.4492 0.5422 0.5562 0.5721
#> Y~~Y     0.6447 0.0242 20000 0.5667 0.5813 0.5965 0.6913 0.7042 0.7187
#> M~~M     0.7537 0.0236 20000 0.6727 0.6907 0.7061 0.7982 0.8125 0.8265
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2286 0.0178 20000 0.1727 0.1834 0.1942 0.2642 0.2756 0.2895
#> direct   0.2134 0.0289 20000 0.1202 0.1406 0.1573 0.2697 0.2872 0.3041
#> total    0.4420 0.0254 20000 0.3585 0.3770 0.3915 0.4910 0.5052 0.5199
```

References

Pesigan, I. J. A., & Cheung, S. F. (2024). 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

R Core Team. (2025). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/