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.

```
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 = 0.05)

#> Monte Carlo Confidence Intervals
#>             est     se     R   2.5%  97.5%
#> cp       0.2875 0.0361 20000 0.2171 0.3578
#> b        0.4840 0.0325 20000 0.4202 0.5475
#> a        0.4819 0.0314 20000 0.4204 0.5439
#> Y~~Y     1.0315 0.0457 20000 0.9420 1.1209
#> M~~M     0.9774 0.0434 20000 0.8919 1.0618
```

```
#> X~~X     0.9890 0.0000 20000 0.9890 0.9890
#> indirect 0.2333 0.0218 20000 0.1920 0.2777
#> direct   0.2875 0.0361 20000 0.2171 0.3578
#> total    0.5207 0.0357 20000 0.4505 0.5904
```

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        1.283360e-03 -5.047282e-04  7.738051e-06 -8.197431e-06 -8.998168e-06
#> b        -5.047282e-04  1.057005e-03 -1.069494e-06  4.222401e-06  5.455511e-06
#> a         7.738051e-06 -1.069494e-06  1.008059e-03 -1.183966e-05  6.931326e-06
#> Y~~Y     -8.197431e-06  4.222401e-06 -1.183966e-05  2.097277e-03  7.225417e-06
#> M~~M     -8.998168e-06  5.455511e-06  6.931326e-06  7.225417e-06  1.931057e-03
#> X~~X     -1.187301e-06 -1.284066e-05 -6.777046e-06  3.079056e-06 -1.781998e-05
#> indirect -2.390837e-04  5.085649e-04  4.875482e-04 -4.317011e-06  5.756591e-06
#> direct    1.283360e-03 -5.047282e-04  7.738051e-06 -8.197431e-06 -8.998168e-06
#> total     1.044276e-03  3.836664e-06  4.952862e-04 -1.251444e-05 -3.241577e-06
#>                    X~~X      indirect        direct         total
#> cp       -1.187301e-06 -2.390837e-04  1.283360e-03  1.044276e-03
```

```
#> b         -1.284066e-05  5.085649e-04 -5.047282e-04  3.836664e-06
#> a         -6.777046e-06  4.875482e-04  7.738051e-06  4.952862e-04
#> Y~~Y       3.079056e-06 -4.317011e-06 -8.197431e-06 -1.251444e-05
#> M~~M      -1.781998e-05  5.756591e-06 -8.998168e-06 -3.241577e-06
#> X~~X       1.964050e-03 -9.633742e-06 -1.187301e-06 -1.082104e-05
#> indirect -9.633742e-06  4.820432e-04 -2.390837e-04  2.429595e-04
#> direct   -1.187301e-06 -2.390837e-04  1.283360e-03  1.044276e-03
#> total    -1.082104e-05  2.429595e-04  1.044276e-03  1.287235e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>              est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp        0.2312 0.0283 20000 0.1397 0.1581 0.1758 0.2868 0.3041 0.3260
#> b         0.4301 0.0267 20000 0.3367 0.3611 0.3772 0.4819 0.4962 0.5158
#> a         0.4362 0.0258 20000 0.3475 0.3682 0.3846 0.4858 0.5011 0.5173
#> Y~~Y      0.6748 0.0243 20000 0.5936 0.6106 0.6259 0.7207 0.7351 0.7468
#> M~~M      0.8097 0.0225 20000 0.7324 0.7489 0.7640 0.8521 0.8645 0.8792
#> X~~X      1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1876 0.0163 20000 0.1370 0.1467 0.1561 0.2202 0.2301 0.2438
#> direct   0.2312 0.0283 20000 0.1397 0.1581 0.1758 0.2868 0.3041 0.3260
#> total    0.4188 0.0262 20000 0.3274 0.3499 0.3670 0.4695 0.4845 0.5027
```

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/