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       0.2229 0.0348 20000 0.1119 0.1361 0.1553 0.2911 0.3124 0.3319
#> b        0.5117 0.0322 20000 0.4065 0.4294 0.4491 0.5750 0.5943 0.6159
#> a        0.5199 0.0301 20000 0.4230 0.4437 0.4610 0.5790 0.5980 0.6183
#> Y~~Y     0.9831 0.0439 20000 0.8417 0.8702 0.8977 1.0683 1.0947 1.1246
#> M~~M     0.9531 0.0427 20000 0.8120 0.8437 0.8691 1.0376 1.0635 1.0954
#> indirect 0.2660 0.0228 20000 0.1961 0.2108 0.2231 0.3123 0.3286 0.3484
#> direct   0.2229 0.0348 20000 0.1119 0.1361 0.1553 0.2911 0.3124 0.3319
```

```
#> total    0.4889 0.0343 20000 0.3844 0.4038 0.4225 0.5568 0.5787 0.6041
```

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))
vcov(unstd)

#>                      cp             b             a          Y~~Y          M~~M
#> cp        1.215470e-03 -5.299003e-04 -6.647093e-06  5.729494e-06  2.393705e-06
#> b        -5.299003e-04  1.024082e-03 -1.255760e-06 -1.054616e-06 -1.577298e-06
#> a        -6.647093e-06 -1.255760e-06  9.002456e-04 -7.219866e-06 -1.098628e-06
#> Y~~Y      5.729494e-06 -1.054616e-06 -7.219866e-06  1.912975e-03  2.123468e-05
#> M~~M      2.393705e-06 -1.577298e-06 -1.098628e-06  2.123468e-05  1.839214e-03
#> X~~X      1.777453e-05 -3.107733e-06 -2.420914e-05  2.425611e-05  1.263475e-05
#> indirect -2.788855e-04  5.325357e-04  4.596715e-04 -4.180271e-06 -2.145417e-06
#> direct    1.215470e-03 -5.299003e-04 -6.647093e-06  5.729494e-06  2.393705e-06
#> total     9.365849e-04  2.635403e-06  4.530244e-04  1.549222e-06  2.482874e-07
#>                    X~~X      indirect        direct        total
#> cp        1.777453e-05 -2.788855e-04  1.215470e-03 9.365849e-04
#> b        -3.107733e-06  5.325357e-04 -5.299003e-04 2.635403e-06
#> a        -2.420914e-05  4.596715e-04 -6.647093e-06 4.530244e-04
#> Y~~Y      2.425611e-05 -4.180271e-06  5.729494e-06 1.549222e-06
```

```
#> M~~M      1.263475e-05 -2.145417e-06  2.393705e-06 2.482874e-07
#> X~~X      2.224741e-03 -1.376256e-05  1.777453e-05 4.011963e-06
#> indirect -1.376256e-05  5.132233e-04 -2.788855e-04 2.343379e-04
#> direct    1.777453e-05 -2.788855e-04  1.215470e-03 9.365849e-04
#> total     4.011963e-06  2.343379e-04  9.365849e-04 1.170923e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>              est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp        0.1878 0.0290 20000 0.0941 0.1122 0.1307 0.2443 0.2634 0.2824
#> b         0.4673 0.0267 20000 0.3755 0.3969 0.4138 0.5193 0.5348 0.5524
#> a         0.4797 0.0242 20000 0.3994 0.4149 0.4316 0.5258 0.5390 0.5540
#> Y~~Y      0.6622 0.0242 20000 0.5776 0.5980 0.6134 0.7081 0.7209 0.7371
#> M~~M      0.7699 0.0232 20000 0.6931 0.7094 0.7235 0.8137 0.8279 0.8404
#> X~~X      1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2241 0.0175 20000 0.1678 0.1801 0.1904 0.2587 0.2698 0.2868
#> direct   0.1878 0.0290 20000 0.0941 0.1122 0.1307 0.2443 0.2634 0.2824
#> total    0.4119 0.0262 20000 0.3228 0.3435 0.3599 0.4623 0.4790 0.4944
```

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