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 (2023).

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        0.1870 0.0343 20000 0.1198 0.2547
#> b         0.5128 0.0297 20000 0.4539 0.5705
#> a         0.5363 0.0327 20000 0.4728 0.6014
#> Y~~Y      0.9690 0.0433 20000 0.8841 1.0538
#> M~~M      1.1080 0.0495 20000 1.0110 1.2047
```

```
#> X~~X     1.0393 0.0000 20000 1.0393 1.0393
#> indirect 0.2750 0.0232 20000 0.2311 0.3221
#> direct   0.1870 0.0343 20000 0.1198 0.2547
#> total    0.4620 0.0347 20000 0.3951 0.5307
```

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        1.181717e-03 -4.705091e-04  8.261433e-06 -1.344359e-05 -2.595224e-06
#> b        -4.705091e-04  8.759326e-04 -1.308366e-05 -1.966912e-06 -1.312343e-05
#> a         8.261433e-06 -1.308366e-05  1.049035e-03 -4.884435e-06 -1.708792e-06
#> Y~~Y     -1.344359e-05 -1.966912e-06 -4.884435e-06  1.860756e-03 -1.483980e-06
#> M~~M     -2.595224e-06 -1.312343e-05 -1.708792e-06 -1.483980e-06  2.426911e-03
#> X~~X     -8.138886e-06  6.557347e-06  2.922167e-06 -9.395330e-06 -7.011709e-06
#> indirect -2.484595e-04  4.635080e-04  5.310531e-04 -3.542631e-06 -8.075768e-06
#> direct    1.181717e-03 -4.705091e-04  8.261433e-06 -1.344359e-05 -2.595224e-06
#> total     9.332576e-04 -7.001085e-06  5.393146e-04 -1.698622e-05 -1.067099e-05
#>                    X~~X      indirect        direct         total
#> cp       -8.138886e-06 -2.484595e-04  1.181717e-03  9.332576e-04
```

```
#> b         6.557347e-06  4.635080e-04 -4.705091e-04 -7.001085e-06
#> a         2.922167e-06  5.310531e-04  8.261433e-06  5.393146e-04
#> Y~~Y     -9.395330e-06 -3.542631e-06 -1.344359e-05 -1.698622e-05
#> M~~M     -7.011709e-06 -8.075768e-06 -2.595224e-06 -1.067099e-05
#> X~~X      2.105082e-03  5.202160e-06 -8.138886e-06 -2.936726e-06
#> indirect  5.202160e-06  5.221204e-04 -2.484595e-04  2.736609e-04
#> direct   -8.138886e-06 -2.484595e-04  1.181717e-03  9.332576e-04
#> total    -2.936726e-06  2.736609e-04  9.332576e-04  1.206919e-03
```

```
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.1566 0.0286 20000 0.0650 0.0830 0.1014 0.2118 0.2286 0.2500
#> b        0.4996 0.0258 20000 0.4138 0.4317 0.4476 0.5489 0.5644 0.5818
#> a        0.4609 0.0247 20000 0.3766 0.3964 0.4118 0.5086 0.5225 0.5411
#> Y~~Y     0.6537 0.0242 20000 0.5715 0.5901 0.6055 0.6995 0.7147 0.7314
#> M~~M     0.7876 0.0228 20000 0.7072 0.7269 0.7414 0.8304 0.8428 0.8582
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2303 0.0176 20000 0.1747 0.1867 0.1964 0.2655 0.2763 0.2898
#> direct   0.1566 0.0286 20000 0.0650 0.0830 0.1014 0.2118 0.2286 0.2500
#> total    0.3869 0.0269 20000 0.2980 0.3174 0.3337 0.4382 0.4541 0.4704
```

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

Pesigan, I. J. A., & Cheung, S. F. (2023). Monte Carlo confidence intervals for the indirect effect with missing data. *Behavior Research Methods*. https://doi.org/10.3758/s13428-023-02114-4

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. (2023). *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