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       0.2375 0.0335 20000 0.1712 0.3040
#> b        0.4436 0.0301 20000 0.3834 0.5025
#> a        0.4392 0.0324 20000 0.3757 0.5029
#> Y~~Y     0.9544 0.0428 20000 0.8705 1.0383
#> M~~M     1.0398 0.0468 20000 0.9484 1.1319
```

```
#> X~~X     0.9874 0.0000 20000 0.9874 0.9874
#> indirect 0.1948 0.0195 20000 0.1581 0.2346
#> direct   0.2375 0.0335 20000 0.1712 0.3040
#> total    0.4323 0.0340 20000 0.3656 0.4990
```

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        1.136209e-03 -4.025716e-04  1.065149e-05  8.755369e-08  5.134447e-06
#> b        -4.025716e-04  9.182522e-04 -3.254555e-06  7.809414e-06  1.085602e-06
#> a         1.065149e-05 -3.254555e-06  1.056501e-03 -1.089049e-05 -2.854462e-07
#> Y~~Y      8.755369e-08  7.809414e-06 -1.089049e-05  1.825587e-03  1.161806e-05
#> M~~M      5.134447e-06  1.085602e-06 -2.854462e-07  1.161806e-05  2.171324e-03
#> X~~X      4.862660e-06  4.812005e-06  4.267997e-06  2.437995e-05  3.124866e-06
#> indirect -1.720248e-04  4.016971e-04  4.669076e-04 -1.734010e-06  2.825545e-07
#> direct    1.136209e-03 -4.025716e-04  1.065149e-05  8.755369e-08  5.134447e-06
#> total     9.641838e-04 -8.744242e-07  4.775591e-04 -1.646456e-06  5.417001e-06
#>                   X~~X      indirect        direct         total
#> cp       4.862660e-06 -1.720248e-04  1.136209e-03  9.641838e-04
```

```
#> b        4.812005e-06  4.016971e-04 -4.025716e-04 -8.744242e-07
#> a        4.267997e-06  4.669076e-04  1.065149e-05  4.775591e-04
#> Y~~Y     2.437995e-05 -1.734010e-06  8.755369e-08 -1.646456e-06
#> M~~M     3.124866e-06  2.825545e-07  5.134447e-06  5.417001e-06
#> X~~X     1.897192e-03  3.857698e-06  4.862660e-06  8.720358e-06
#> indirect 3.857698e-06  3.843095e-04 -1.720248e-04  2.122847e-04
#> direct   4.862660e-06 -1.720248e-04  1.136209e-03  9.641838e-04
#> total    8.720358e-06  2.122847e-04  9.641838e-04  1.176469e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2036 0.0285 20000 0.1093 0.1301 0.1469 0.2587 0.2775 0.2968
#> b        0.4245 0.0268 20000 0.3326 0.3553 0.3705 0.4754 0.4903 0.5120
#> a        0.3934 0.0268 20000 0.3024 0.3226 0.3403 0.4455 0.4609 0.4775
#> Y~~Y     0.7104 0.0241 20000 0.6296 0.6469 0.6618 0.7568 0.7700 0.7868
#> M~~M     0.8452 0.0211 20000 0.7720 0.7876 0.8015 0.8842 0.8959 0.9086
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1670 0.0157 20000 0.1174 0.1277 0.1367 0.1985 0.2089 0.2216
#> direct   0.2036 0.0285 20000 0.1093 0.1301 0.1469 0.2587 0.2775 0.2968
#> total    0.3706 0.0273 20000 0.2785 0.2976 0.3157 0.4228 0.4402 0.4601
```

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*, *56*(3), 1678–1696. 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. (2024). *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