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       0.1921 0.0349 20000 0.1238 0.2598
#> b        0.5080 0.0323 20000 0.4441 0.5706
#> a        0.5170 0.0303 20000 0.4576 0.5770
#> Y~~Y     0.9859 0.0446 20000 0.8997 1.0733
#> M~~M     0.9538 0.0429 20000 0.8686 1.0370
```

```
#> X~~X     1.0494 0.0000 20000 1.0494 1.0494
#> indirect 0.2626 0.0227 20000 0.2194 0.3080
#> direct   0.1921 0.0349 20000 0.1238 0.2598
#> total    0.4548 0.0342 20000 0.3872 0.5216
```

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        1.217474e-03 -5.286936e-04 -6.759262e-06 -4.576467e-06  2.534699e-05
#> b        -5.286936e-04  1.024945e-03  1.377169e-06  2.161332e-05 -1.185860e-05
#> a        -6.759262e-06  1.377169e-06  8.963937e-04 -6.753065e-06 -1.803727e-05
#> Y~~Y     -4.576467e-06  2.161332e-05 -6.753065e-06  1.969210e-03 -9.299605e-07
#> M~~M      2.534699e-05 -1.185860e-05 -1.803727e-05 -9.299605e-07  1.805217e-03
#> X~~X      8.623741e-07 -4.164452e-06 -3.692446e-06  8.269935e-06  1.475164e-05
#> indirect -2.766576e-04  5.299114e-04  4.561216e-04  7.188698e-06 -1.531321e-05
#> direct    1.217474e-03 -5.286936e-04 -6.759262e-06 -4.576467e-06  2.534699e-05
#> total     9.408163e-04  1.217797e-06  4.493624e-04  2.612231e-06  1.003378e-05
#>                    X~~X      indirect        direct         total
#> cp        8.623741e-07 -2.766576e-04  1.217474e-03  9.408163e-04
```

```
#> b         -4.164452e-06  5.299114e-04 -5.286936e-04  1.217797e-06
#> a         -3.692446e-06  4.561216e-04 -6.759262e-06  4.493624e-04
#> Y~~Y       8.269935e-06  7.188698e-06 -4.576467e-06  2.612231e-06
#> M~~M       1.475164e-05 -1.531321e-05  2.534699e-05  1.003378e-05
#> X~~X       2.194621e-03 -4.782314e-06  8.623741e-07 -3.919940e-06
#> indirect -4.782314e-06  5.062794e-04 -2.766576e-04  2.296218e-04
#> direct     8.623741e-07 -2.766576e-04  1.217474e-03  9.408163e-04
#> total     -3.919940e-06  2.296218e-04  9.408163e-04  1.170438e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>              est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp        0.1635 0.0294 20000 0.0630 0.0871 0.1064 0.2206 0.2396 0.2628
#> b         0.4688 0.0269 20000 0.3765 0.3980 0.4146 0.5197 0.5384 0.5554
#> a         0.4767 0.0244 20000 0.3919 0.4115 0.4277 0.5229 0.5370 0.5536
#> Y~~Y      0.6804 0.0241 20000 0.5979 0.6177 0.6326 0.7268 0.7404 0.7582
#> M~~M      0.7727 0.0232 20000 0.6935 0.7117 0.7266 0.8171 0.8307 0.8464
#> X~~X      1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2235 0.0177 20000 0.1680 0.1794 0.1890 0.2582 0.2705 0.2863
#> direct    0.1635 0.0294 20000 0.0630 0.0871 0.1064 0.2206 0.2396 0.2628
#> total     0.3870 0.0268 20000 0.2963 0.3154 0.3337 0.4382 0.4542 0.4752
```

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