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       0.2517 0.0342 20000 0.1850 0.3185
#> b        0.4892 0.0315 20000 0.4270 0.5506
#> a        0.5161 0.0303 20000 0.4571 0.5755
#> Y~~Y     0.9609 0.0431 20000 0.8762 1.0454
#> M~~M     0.9780 0.0435 20000 0.8930 1.0641
```

```
#> X~~X     1.0678 0.0000 20000 1.0678 1.0678
#> indirect 0.2525 0.0222 20000 0.2115 0.2973
#> direct   0.2517 0.0342 20000 0.1850 0.3185
#> total    0.5042 0.0336 20000 0.4392 0.5713
```

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        1.166869e-03 -5.138947e-04  1.535497e-05 -8.148513e-06 -6.908199e-07
#> b        -5.138947e-04  9.795365e-04 -6.658198e-06  6.228099e-06 -3.504815e-06
#> a         1.535497e-05 -6.658198e-06  9.176202e-04 -8.947865e-06  5.916583e-06
#> Y~~Y     -8.148513e-06  6.228099e-06 -8.947865e-06  1.864451e-03  1.928943e-05
#> M~~M     -6.908199e-07 -3.504815e-06  5.916583e-06  1.928943e-05  1.912183e-03
#> X~~X     -1.210856e-05  8.435540e-06  1.088003e-05 -2.508988e-07 -2.035580e-05
#> indirect -2.576034e-04  5.018469e-04  4.455333e-04 -1.049641e-06  7.982461e-07
#> direct    1.166869e-03 -5.138947e-04  1.535497e-05 -8.148513e-06 -6.908199e-07
#> total     9.092660e-04 -1.204780e-05  4.608883e-04 -9.198154e-06  1.074262e-07
#>                   X~~X      indirect        direct         total
#> cp       -1.210856e-05 -2.576034e-04  1.166869e-03  9.092660e-04
```

```
#> b          8.435540e-06  5.018469e-04 -5.138947e-04 -1.204780e-05
#> a          1.088003e-05  4.455333e-04  1.535497e-05  4.608883e-04
#> Y~~Y      -2.508988e-07 -1.049641e-06 -8.148513e-06 -9.198154e-06
#> M~~M      -2.035580e-05  7.982461e-07 -6.908199e-07  1.074262e-07
#> X~~X       2.311125e-03  9.635475e-06 -1.210856e-05 -2.473083e-06
#> indirect  9.635475e-06  4.776476e-04 -2.576034e-04  2.200441e-04
#> direct   -1.210856e-05 -2.576034e-04  1.166869e-03  9.092660e-04
#> total    -2.473083e-06  2.200441e-04  9.092660e-04  1.129310e-03
```

```
MCStd(unstd)

#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2148 0.0287 20000 0.1194 0.1412 0.1583 0.2708 0.2885 0.3042
#> b        0.4539 0.0267 20000 0.3643 0.3860 0.4011 0.5062 0.5225 0.5372
#> a        0.4747 0.0246 20000 0.3894 0.4102 0.4257 0.5224 0.5364 0.5587
#> Y~~Y     0.6553 0.0243 20000 0.5741 0.5905 0.6062 0.7017 0.7144 0.7331
#> M~~M     0.7747 0.0233 20000 0.6879 0.7123 0.7271 0.8188 0.8317 0.8484
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2154 0.0172 20000 0.1615 0.1723 0.1826 0.2495 0.2607 0.2740
#> direct   0.2148 0.0287 20000 0.1194 0.1412 0.1583 0.2708 0.2885 0.3042
#> total    0.4302 0.0259 20000 0.3448 0.3622 0.3784 0.4802 0.4954 0.5117
```

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