semmcci: Monte Carlo Confidence Intervals

Ivan Jacob Agaloos Pesigan

Installation

You can install the CRAN release of `semmcci` with:

```r
install.packages("semmcci")
```

You can install the development version of `semmcci` from GitHub with:

```r
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         0.2746 0.0354 20000 0.2057 0.3444
#> b          0.4536 0.0297 20000 0.3948 0.5114
#> a          0.5583 0.0331 20000 0.4928 0.6231
#> Y~~Y       0.9674 0.0434 20000 0.8826 1.0520
#> M~~M       1.0934 0.0489 20000 0.9977 1.1889
```

```
#> X~~X     0.9827 0.0000 20000 0.9827 0.9827
#> indirect 0.2532 0.0225 20000 0.2098 0.2987
#> direct   0.2746 0.0354 20000 0.2057 0.3444
#> total    0.5278 0.0344 20000 0.4617 0.5957
```

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        1.273657e-03 -4.937050e-04  6.428523e-06  2.535270e-06 -1.456908e-06
#> b        -4.937050e-04  8.808833e-04 -2.027978e-07 -8.264881e-06  9.667984e-06
#> a         6.428523e-06 -2.027978e-07  1.106024e-03  9.971128e-06  1.920802e-05
#> Y~~Y      2.535270e-06 -8.264881e-06  9.971128e-06  1.865884e-03 -1.508133e-05
#> M~~M     -1.456908e-06  9.667984e-06  1.920802e-05 -1.508133e-05  2.384944e-03
#> X~~X     -7.130463e-07 -9.855083e-07  2.045900e-06  2.648879e-06  1.905907e-06
#> indirect -2.731830e-04  4.917119e-04  5.016149e-04  1.172963e-07  1.446485e-05
#> direct    1.273657e-03 -4.937050e-04  6.428523e-06  2.535270e-06 -1.456908e-06
#> total     1.000474e-03 -1.993098e-06  5.080435e-04  2.652566e-06  1.300794e-05
#>                   X~~X      indirect        direct         total
#> cp       -7.130463e-07 -2.731830e-04  1.273657e-03  1.000474e-03
```

```
#> b        -9.855083e-07  4.917119e-04 -4.937050e-04 -1.993098e-06
#> a         2.045900e-06  5.016149e-04  6.428523e-06  5.080435e-04
#> Y~~Y      2.648879e-06  1.172963e-07  2.535270e-06  2.652566e-06
#> M~~M      1.905907e-06  1.446485e-05 -1.456908e-06  1.300794e-05
#> X~~X      1.901554e-03  5.418577e-07 -7.130463e-07 -1.711886e-07
#> indirect  5.418577e-07  5.030301e-04 -2.731830e-04  2.298471e-04
#> direct   -7.130463e-07 -2.731830e-04  1.273657e-03  1.000474e-03
#> total    -1.711886e-07  2.298471e-04  1.000474e-03  1.230321e-03
```

```r
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2248 0.0287 20000 0.1314 0.1505 0.1680 0.2814 0.2981 0.3172
#> b        0.4432 0.0269 20000 0.3546 0.3715 0.3897 0.4951 0.5106 0.5304
#> a        0.4678 0.0245 20000 0.3846 0.4024 0.4187 0.5151 0.5290 0.5484
#> Y~~Y     0.6598 0.0244 20000 0.5798 0.5984 0.6120 0.7072 0.7218 0.7378
#> M~~M     0.7812 0.0229 20000 0.6992 0.7201 0.7346 0.8247 0.8381 0.8521
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2073 0.0169 20000 0.1557 0.1654 0.1748 0.2410 0.2523 0.2648
#> direct   0.2248 0.0287 20000 0.1314 0.1505 0.1680 0.2814 0.2981 0.3172
#> total    0.4321 0.0258 20000 0.3472 0.3645 0.3805 0.4814 0.4957 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