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.

```r
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`.

```r
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.

```r
MC(fit, R = 20000L, alpha = 0.05)

#> Monte Carlo Confidence Intervals
#>               est     se     R   2.5%  97.5%
#> cp         0.2746 0.0390 20000 0.1979 0.3502
#> b          0.4910 0.0348 20000 0.4228 0.5600
#> a          0.5259 0.0312 20000 0.4652 0.5870
#> Y~~Y       1.1282 0.0507 20000 1.0282 1.2279
#> M~~M       0.9413 0.0420 20000 0.8583 1.0236
```

```
#> X~~X     0.9617 0.0000 20000 0.9617 0.9617
#> indirect 0.2582 0.0240 20000 0.2132 0.3070
#> direct   0.2746 0.0390 20000 0.1979 0.3502
#> total    0.5328 0.0376 20000 0.4588 0.6067
```

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        1.495411e-03 -6.152539e-04 -9.897186e-06 -5.088251e-06 -2.441421e-05
#> b        -6.152539e-04  1.175255e-03 -9.228023e-07 -2.970535e-06 -1.035878e-05
#> a        -9.897186e-06 -9.228023e-07  9.851359e-04  3.776605e-06  1.749047e-05
#> Y~~Y     -5.088251e-06 -2.970535e-06  3.776605e-06  2.533545e-03  9.319945e-07
#> M~~M     -2.441421e-05 -1.035878e-05  1.749047e-05  9.319945e-07  1.778481e-03
#> X~~X      2.355619e-05 -1.511108e-05  1.141487e-05  1.339382e-05  6.209458e-06
#> indirect -3.286821e-04  6.178976e-04  4.832467e-04  1.982476e-07  3.990484e-06
#> direct    1.495411e-03 -6.152539e-04 -9.897186e-06 -5.088251e-06 -2.441421e-05
#> total     1.166729e-03  2.643789e-06  4.733495e-04 -4.890003e-06 -2.042372e-05
#>                   X~~X      indirect        direct         total
#> cp        2.355619e-05 -3.286821e-04  1.495411e-03  1.166729e-03
```

```
#> b        -1.511108e-05  6.178976e-04 -6.152539e-04  2.643789e-06
#> a         1.141487e-05  4.832467e-04 -9.897186e-06  4.733495e-04
#> Y~~Y      1.339382e-05  1.982476e-07 -5.088251e-06 -4.890003e-06
#> M~~M      6.209458e-06  3.990484e-06 -2.441421e-05 -2.042372e-05
#> X~~X      1.860573e-03 -2.480529e-06  2.355619e-05  2.107566e-05
#> indirect -2.480529e-06  5.635301e-04 -3.286821e-04  2.348480e-04
#> direct    2.355619e-05 -3.286821e-04  1.495411e-03  1.166729e-03
#> total     2.107566e-05  2.348480e-04  1.166729e-03  1.401577e-03
```

```
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2110 0.0293 20000 0.1134 0.1360 0.1527 0.2676 0.2864 0.3068
#> b        0.4228 0.0275 20000 0.3297 0.3502 0.3684 0.4766 0.4928 0.5104
#> a        0.4694 0.0247 20000 0.3842 0.4041 0.4199 0.5165 0.5326 0.5483
#> Y~~Y     0.6929 0.0241 20000 0.6125 0.6288 0.6441 0.7379 0.7523 0.7684
#> M~~M     0.7797 0.0231 20000 0.6994 0.7164 0.7332 0.8237 0.8367 0.8524
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.1985 0.0170 20000 0.1431 0.1559 0.1656 0.2319 0.2436 0.2554
#> direct   0.2110 0.0293 20000 0.1134 0.1360 0.1527 0.2676 0.2864 0.3068
#> total    0.4095 0.0264 20000 0.3208 0.3387 0.3565 0.4599 0.4749 0.4922
```

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