# 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 $\alpha$ 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.

```r
library(semmcci)
library(lavaan)
```

### Data

```r
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        0.2481 0.0353 20000 0.1797 0.3170
#> b         0.5154 0.0323 20000 0.4531 0.5790
#> a         0.5057 0.0307 20000 0.4457 0.5658
#> Y~~Y      1.0321 0.0464 20000 0.9414 1.1241
#> M~~M      0.9815 0.0436 20000 0.8953 1.0675
```

```
#> X~~X     1.0442 0.0000 20000 1.0442 1.0442
#> indirect 0.2607 0.0229 20000 0.2179 0.3073
#> direct   0.2481 0.0353 20000 0.1797 0.3170
#> total    0.5088 0.0351 20000 0.4397 0.5782
```

### 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        1.233296e-03 -5.224632e-04  1.159246e-05 -2.863771e-06  1.808729e-06
#> b        -5.224632e-04  1.049480e-03  3.503789e-06  3.367873e-06  5.341893e-06
#> a         1.159246e-05  3.503789e-06  9.268685e-04 -8.296009e-07  3.426777e-06
#> Y~~Y     -2.863771e-06  3.367873e-06 -8.296009e-07  2.106950e-03 -4.921518e-06
#> M~~M      1.808729e-06  5.341893e-06  3.426777e-06 -4.921518e-06  1.926186e-03
#> X~~X     -5.766921e-06 -1.136013e-05  4.037988e-06  7.934019e-06 -5.566672e-06
#> indirect -2.580171e-04  5.322663e-04  4.792198e-04  1.983531e-06  4.169332e-06
#> direct    1.233296e-03 -5.224632e-04  1.159246e-05 -2.863771e-06  1.808729e-06
#> total     9.752793e-04  9.803117e-06  4.908123e-04 -8.802400e-07  5.978060e-06
#>                   X~~X      indirect        direct         total
#> cp       -5.766921e-06 -2.580171e-04  1.233296e-03  9.752793e-04
```

```
#> b        -1.136013e-05  5.322663e-04 -5.224632e-04  9.803117e-06
#> a         4.037988e-06  4.792198e-04  1.159246e-05  4.908123e-04
#> Y~~Y      7.934019e-06  1.983531e-06 -2.863771e-06 -8.802400e-07
#> M~~M     -5.566672e-06  4.169332e-06  1.808729e-06  5.978060e-06
#> X~~X      2.195345e-03 -4.136809e-06 -5.766921e-06 -9.903730e-06
#> indirect -4.136809e-06  5.168524e-04 -2.580171e-04  2.588353e-04
#> direct   -5.766921e-06 -2.580171e-04  1.233296e-03  9.752793e-04
#> total    -9.903730e-06  2.588353e-04  9.752793e-04  1.234115e-03
```

```
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
#>             est     se     R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> cp       0.2028 0.0283 20000 0.1092 0.1296 0.1471 0.2574 0.2763 0.2970
#> b        0.4607 0.0265 20000 0.3705 0.3900 0.4074 0.5115 0.5273 0.5432
#> a        0.4625 0.0247 20000 0.3805 0.3974 0.4134 0.5099 0.5248 0.5435
#> Y~~Y     0.6603 0.0243 20000 0.5819 0.5972 0.6127 0.7068 0.7217 0.7388
#> M~~M     0.7861 0.0228 20000 0.7046 0.7246 0.7400 0.8291 0.8421 0.8552
#> X~~X     1.0000 0.0000 20000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
#> indirect 0.2131 0.0171 20000 0.1587 0.1702 0.1801 0.2470 0.2573 0.2684
#> direct   0.2028 0.0283 20000 0.1092 0.1296 0.1471 0.2574 0.2763 0.2970
#> total    0.4158 0.0260 20000 0.3243 0.3480 0.3635 0.4664 0.4804 0.4952
```

# 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