

# 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:

```
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.

## 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 = c(0.001, 0.01, 0.05))

#> Monte Carlo Confidence Intervals
#>           est      se      R 0.05%  0.5%  2.5% 97.5% 99.5% 99.95%
#> cp      0.2392 0.0346 20000 0.1287 0.1512 0.1717 0.3070 0.3294 0.3524
#> b      0.4916 0.0311 20000 0.3871 0.4102 0.4304 0.5519 0.5711 0.5921
#> a      0.4496 0.0322 20000 0.3451 0.3671 0.3871 0.5134 0.5316 0.5526
#> Y~~Y    0.9300 0.0415 20000 0.7882 0.8221 0.8485 1.0113 1.0373 1.0620
#> M~~M    0.9534 0.0426 20000 0.8121 0.8437 0.8697 1.0369 1.0638 1.0938
#> indirect 0.2210 0.0212 20000 0.1560 0.1694 0.1810 0.2638 0.2784 0.2947
#> direct  0.2392 0.0346 20000 0.1287 0.1512 0.1717 0.3070 0.3294 0.3524

```

```
#> total      0.4603 0.0353 20000 0.3438 0.3696 0.3911 0.5303 0.5510 0.5734
```

## 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 = c(0.001, 0.01, 0.05))
```

```
MCStd(unstd)
```

```
#> Standardized Monte Carlo Confidence Intervals
```

#>	est	se	R	0.05%	0.5%	2.5%	97.5%	99.5%	99.95%
#> cp	0.1970	0.0285	20000	0.1010	0.1213	0.1402	0.2525	0.2711	0.2906
#> b	0.4507	0.0262	20000	0.3624	0.3812	0.3980	0.5007	0.5158	0.5327
#> a	0.4037	0.0265	20000	0.3158	0.3345	0.3509	0.4543	0.4696	0.4917
#> Y~~Y	0.6863	0.0242	20000	0.6045	0.6232	0.6380	0.7323	0.7462	0.7628
#> M~~M	0.8370	0.0213	20000	0.7582	0.7795	0.7936	0.8769	0.8881	0.9003
#> X~~X	1.0000	0.0000	20000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
#> indirect	0.1820	0.0163	20000	0.1294	0.1411	0.1504	0.2142	0.2253	0.2392
#> direct	0.1970	0.0285	20000	0.1010	0.1213	0.1402	0.2525	0.2711	0.2906
#> total	0.3789	0.0271	20000	0.2886	0.3057	0.3242	0.4311	0.4472	0.4668

## 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](https://doi.org/10.1207/s15327906mbr3901_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. (2022). *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>