Package 'semmcci'

December 23, 2022

Title Monte Carlo Confidence Intervals in Structural Equation Modeling

Version 1.0.4.9000

Description 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. 'semmcci' 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. Preacher and Selig (2012) <doi:10.1080/19312458.2012.679848>.

URL https://github.com/jeksterslab/semmcci, https://jeksterslab.github.io/semmcci/

BugReports https://github.com/jeksterslab/semmcci/issues

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 3.0.0)

Imports stats, methods, lavaan

Suggests knitr, rmarkdown, testthat, MASS

RoxygenNote 7.2.3

NeedsCompilation no

Author Ivan Jacob Agaloos Pesigan [aut, cre, cph] (<https://orcid.org/0000-0003-4818-8420>), Shu Fai Cheung [ctb] (<https://orcid.org/0000-0002-9871-9448>)

Maintainer Ivan Jacob Agaloos Pesigan <r.jeksterslab@gmail.com>

R topics documented:

`coef.semmcci` *Parameter Estimates*

Description

Parameter Estimates

Usage

```
## S3 method for class 'semmcci'
coef(object, ...)
```

Arguments

`object`	Object of class `semmcci`.
`...`	additional arguments.

Value

Returns a vector of parameter estimates.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
library(semmcci)
library(lavaan)

# Generate Data ------------------------------------------------------------
n <- 1000
a <- 0.50
b <- 0.50
cp <- 0.25
s2_em <- 1 - a^2
s2_ey <- 1 - cp^2 - a^2 * b^2 - b^2 * s2_em - 2 * cp * a * b
em <- rnorm(n = n, mean = 0, sd = sqrt(s2_em))
ey <- rnorm(n = n, mean = 0, sd = sqrt(s2_ey))
X <- rnorm(n = n)
M <- a * X + em
Y <- cp * X + b * M + ey
df <- data.frame(X, M, Y)

# Fit Model in lavaan ------------------------------------------------------
model <- "
  Y ~ cp * X + b * M
  M ~ a * X
  indirect := a * b
  direct := cp
  total := cp + (a * b)
"
fit <- sem(data = df, model = model)

# Monte Carlo --------------------------------------------------------------
unstd <- MC(
  fit,
  R = 100L, # use a large value e.g., 20000L for actual research
  alpha = c(0.001, 0.01, 0.05)
)
coef(unstd)
```

`coef.semmccistd` *Standardized Parameter Estimates*

Description

Standardized Parameter Estimates

Usage

```
## S3 method for class 'semmccistd'
coef(object, ...)
```

Arguments

object	Object of class semmccistd.
...	additional arguments.

Value

Returns a vector of standardized parameter estimates.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
library(semmcci)
library(lavaan)

# Generate Data ----------------------------------------------------------
n <- 1000
a <- 0.50
b <- 0.50
cp <- 0.25
s2_em <- 1 - a^2
s2_ey <- 1 - cp^2 - a^2 * b^2 - b^2 * s2_em - 2 * cp * a * b
em <- rnorm(n = n, mean = 0, sd = sqrt(s2_em))
ey <- rnorm(n = n, mean = 0, sd = sqrt(s2_ey))
X <- rnorm(n = n)
M <- a * X + em
Y <- cp * X + b * M + ey
df <- data.frame(X, M, Y)

# Fit Model in lavaan ----------------------------------------------------
model <- "
  Y ~ cp * X + b * M
  M ~ a * X
  indirect := a * b
  direct := cp
  total := cp + (a * b)
"
fit <- sem(data = df, model = model, fixed.x = FALSE)

# Monte Carlo ------------------------------------------------------------
unstd <- MC(
  fit,
  R = 100L, # use a large value e.g., 20000L for actual research
  alpha = c(0.001, 0.01, 0.05)
)

# Standardized Monte Carlo -----------------------------------------------
std <- MCStd(unstd)
coef(std)
```

confint.semmcci *Monte Carlo Confidence Intervals for the Parameter Estimates*

Description

Monte Carlo Confidence Intervals for the Parameter Estimates

Usage

```
## S3 method for class 'semmcci'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

object	Object of class semmcci.
parm	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
level	the confidence level required.
...	additional arguments.

Value

Returns a matrix of confidence intervals.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
library(semmcci)
library(lavaan)

# Generate Data ------------------------------------------------------------
n <- 1000
a <- 0.50
b <- 0.50
cp <- 0.25
s2_em <- 1 - a^2
s2_ey <- 1 - cp^2 - a^2 * b^2 - b^2 * s2_em - 2 * cp * a * b
em <- rnorm(n = n, mean = 0, sd = sqrt(s2_em))
ey <- rnorm(n = n, mean = 0, sd = sqrt(s2_ey))
X <- rnorm(n = n)
M <- a * X + em
Y <- cp * X + b * M + ey
df <- data.frame(X, M, Y)
```

```
# Fit Model in lavaan -----------------------------------------------------
model <- "
  Y ~ cp * X + b * M
  M ~ a * X
  indirect := a * b
  direct := cp
  total := cp + (a * b)
"
fit <- sem(data = df, model = model)

# Monte Carlo -------------------------------------------------------------
unstd <- MC(
  fit,
  R = 100L, # use a large value e.g., 20000L for actual research
  alpha = c(0.001, 0.01, 0.05)
)
confint(unstd)
```

`confint.semmccistd` *Monte Carlo Confidence Intervals for the Standardized Parameter Estimates*

Description

Monte Carlo Confidence Intervals for the Standardized Parameter Estimates

Usage

```
## S3 method for class 'semmccistd'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

`object`	Object of class `semmccistd`.
`parm`	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
`level`	the confidence level required.
`...`	additional arguments.

Value

Returns a matrix of confidence intervals.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
library(semmcci)
library(lavaan)

# Generate Data ------------------------------------------------------------
n <- 1000
a <- 0.50
b <- 0.50
cp <- 0.25
s2_em <- 1 - a^2
s2_ey <- 1 - cp^2 - a^2 * b^2 - b^2 * s2_em - 2 * cp * a * b
em <- rnorm(n = n, mean = 0, sd = sqrt(s2_em))
ey <- rnorm(n = n, mean = 0, sd = sqrt(s2_ey))
X <- rnorm(n = n)
M <- a * X + em
Y <- cp * X + b * M + ey
df <- data.frame(X, M, Y)

# Fit Model in lavaan ------------------------------------------------------
model <- "
  Y ~ cp * X + b * M
  M ~ a * X
  indirect := a * b
  direct := cp
  total := cp + (a * b)
"
fit <- sem(data = df, model = model, fixed.x = FALSE)

# Monte Carlo --------------------------------------------------------------
unstd <- MC(
  fit,
  R = 100L, # use a large value e.g., 20000L for actual research
  alpha = c(0.001, 0.01, 0.05)
)

# Standardized Monte Carlo -------------------------------------------------
std <- MCStd(unstd)
confint(std)
```

MC *Monte Carlo Confidence Intervals*

Description

Calculates Monte Carlo confidence intervals for free and defined parameters.

Usage

```
MC(
```

```
  object,
  R = 20000L,
  alpha = c(0.001, 0.01, 0.05),
  decomposition = "eigen",
  pd = TRUE,
  tol = 1e-06
)
```

Arguments

object	object of class lavaan.
R	Positive integer. Number of Monte Carlo replications.
alpha	Numeric vector. Significance level α. Default value is alpha = c(0.001, 0.01, 0.05).
decomposition	Character string. Matrix decomposition of the sampling variance-covariance matrix for the data generation. If decomposition = "chol", use Cholesky decomposition. If decomposition = "eigen", use eigenvalue decomposition. If decomposition = "svd", use singular value decomposition.
pd	Logical. If pd = TRUE, check if the sampling variance-covariance matrix is positive definite using tol.
tol	Numeric. Tolerance used for pd.

Details

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 free and defined parameters are generated using the simulated sampling distribution. Parameters can be defined using the := operator in the lavaan model syntax.

Value

Returns an object of class semmcci which is a list with the following elements:

R Number of Monte Carlo replications.

alpha Significance level α specified.

lavaan lavaan object.

decomposition Matrix decomposition used to generate multivariate normal random variates.

thetahat Parameter estimates $\hat{\theta}$.

thetahatstar Sampling distribution of parameter estimates $\hat{\theta}^*$.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
library(semmcci)
library(lavaan)

# Generate Data ------------------------------------------------------------
n <- 1000
a <- 0.50
b <- 0.50
cp <- 0.25
s2_em <- 1 - a^2
s2_ey <- 1 - cp^2 - a^2 * b^2 - b^2 * s2_em - 2 * cp * a * b
em <- rnorm(n = n, mean = 0, sd = sqrt(s2_em))
ey <- rnorm(n = n, mean = 0, sd = sqrt(s2_ey))
X <- rnorm(n = n)
M <- a * X + em
Y <- cp * X + b * M + ey
df <- data.frame(X, M, Y)

# Fit Model in lavaan ------------------------------------------------------
model <- "
  Y ~ cp * X + b * M
  M ~ a * X
  indirect := a * b
  direct := cp
  total := cp + (a * b)
"
fit <- sem(data = df, model = model)

# Monte Carlo --------------------------------------------------------------
MC(
  fit,
  R = 100L, # use a large value e.g., 20000L for actual research
  alpha = c(0.001, 0.01, 0.05)
)
```

MCStd *Standardized Monte Carlo Confidence Intervals*

Description

Calculates standardized Monte Carlo confidence intervals for free and defined parameters.

Usage

```
MCStd(object, alpha = c(0.001, 0.01, 0.05))
```

Arguments

object	object of class semmcci. Output of the MC() function.

alpha Numeric vector. Significance level α. Default value is alpha = c(0.001, 0.01, 0.05).

Details

The empirical sampling distribution of parameter estimates from the argument object is standardized, that is, each randomly generated vector of parameters is standardized. Defined parameters are computed from the standardized component parameters. Confidence intervals are generated using the standardized empirical sampling distribution.

Value

Returns an object of class semmccistd which is a list with the following elements:

R Number of Monte Carlo replications.

alpha Significance level α specified.

lavaan lavaan object.

decomposition Matrix decomposition used to generate multivariate normal random variates.

thetahat Parameter estimates $\hat{\theta}$.

thetahatstar Sampling distribution of parameter estimates $\hat{\theta}^*$.

ci Confidence intervals.

thetahat_std Standardized parameter estimates $\hat{\theta}_{\text{std}}$.

thetahatstar_std Standardized sampling distribution of parameter estimates $\hat{\theta}^*_{\text{std}}$.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
library(semmcci)
library(lavaan)

# Generate Data ------------------------------------------------------------
n <- 1000
a <- 0.50
b <- 0.50
cp <- 0.25
s2_em <- 1 - a^2
s2_ey <- 1 - cp^2 - a^2 * b^2 - b^2 * s2_em - 2 * cp * a * b
em <- rnorm(n = n, mean = 0, sd = sqrt(s2_em))
ey <- rnorm(n = n, mean = 0, sd = sqrt(s2_ey))
X <- rnorm(n = n)
M <- a * X + em
Y <- cp * X + b * M + ey
df <- data.frame(X, M, Y)

# Fit Model in lavaan ------------------------------------------------------
model <- "
```

```
  Y ~ cp * X + b * M
  M ~ a * X
  indirect := a * b
  direct := cp
  total := cp + (a * b)
"
fit <- sem(data = df, model = model, fixed.x = FALSE)

# Monte Carlo ----------------------------------------------------------
output <- MC(
  fit,
  R = 100L, # use a large value e.g., 20000L for actual research
  alpha = c(0.001, 0.01, 0.05)
)

# Standardized Monte Carlo ---------------------------------------------
MCStd(output)
```

`print.semmcci` *Print Method for Object of Class* `semmcci`

Description

Print Method for Object of Class `semmcci`

Usage

```
## S3 method for class 'semmcci'
print(x, digits = 4, ...)
```

Arguments

`x`	an object of class `semmcci`.
`digits`	Integer indicating the number of decimal places to display.
`...`	further arguments.

Value

Returns a matrix of estimates, standard errors, number of Monte Carlo replications, and confidence intervals.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
library(semmcci)
library(lavaan)

# Generate Data ------------------------------------------------------------
n <- 1000
a <- 0.50
b <- 0.50
cp <- 0.25
s2_em <- 1 - a^2
s2_ey <- 1 - cp^2 - a^2 * b^2 - b^2 * s2_em - 2 * cp * a * b
em <- rnorm(n = n, mean = 0, sd = sqrt(s2_em))
ey <- rnorm(n = n, mean = 0, sd = sqrt(s2_ey))
X <- rnorm(n = n)
M <- a * X + em
Y <- cp * X + b * M + ey
df <- data.frame(X, M, Y)

# Fit Model in lavaan ------------------------------------------------------
model <- "
  Y ~ cp * X + b * M
  M ~ a * X
  indirect := a * b
  direct := cp
  total := cp + (a * b)
"
fit <- sem(data = df, model = model)

# Monte Carlo --------------------------------------------------------------
unstd <- MC(
  fit,
  R = 100L, # use a large value e.g., 20000L for actual research
  alpha = c(0.001, 0.01, 0.05)
)
print(unstd)
```

`print.semmccistd` *Print Method for Object of Class* `semmccistd`

Description

Print Method for Object of Class `semmccistd`

Usage

```
## S3 method for class 'semmccistd'
print(x, digits = 4, ...)
```

Arguments

x	an object of class semmccistd.
digits	Integer indicating the number of decimal places to display.
...	further arguments.

Value

Returns a matrix of estimates, standard errors, number of Monte Carlo replications, and confidence intervals.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
library(semmcci)
library(lavaan)

# Generate Data ------------------------------------------------------------
n <- 1000
a <- 0.50
b <- 0.50
cp <- 0.25
s2_em <- 1 - a^2
s2_ey <- 1 - cp^2 - a^2 * b^2 - b^2 * s2_em - 2 * cp * a * b
em <- rnorm(n = n, mean = 0, sd = sqrt(s2_em))
ey <- rnorm(n = n, mean = 0, sd = sqrt(s2_ey))
X <- rnorm(n = n)
M <- a * X + em
Y <- cp * X + b * M + ey
df <- data.frame(X, M, Y)

# Fit Model in lavaan ------------------------------------------------------
model <- "
  Y ~ cp * X + b * M
  M ~ a * X
  indirect := a * b
  direct := cp
  total := cp + (a * b)
"
fit <- sem(data = df, model = model, fixed.x = FALSE)

# Monte Carlo --------------------------------------------------------------
unstd <- MC(
  fit,
  R = 100L, # use a large value e.g., 20000L for actual research
  alpha = c(0.001, 0.01, 0.05)
)

# Standardized Monte Carlo -------------------------------------------------
```

```
std <- MCStd(unstd)
print(std)
```

---

summary.semmcci *Summary Method for an Object of Class* semmcci

---

Description

Summary Method for an Object of Class semmcci

Usage

```
## S3 method for class 'semmcci'
summary(object, digits = 4, ...)
```

Arguments

object	Object of class semmcci.
digits	Digits to print.
...	additional arguments.

Value

Returns a matrix of estimates, standard errors, number of Monte Carlo replications, and confidence intervals.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
library(semmcci)
library(lavaan)

# Generate Data ------------------------------------------------------------
n <- 1000
a <- 0.50
b <- 0.50
cp <- 0.25
s2_em <- 1 - a^2
s2_ey <- 1 - cp^2 - a^2 * b^2 - b^2 * s2_em - 2 * cp * a * b
em <- rnorm(n = n, mean = 0, sd = sqrt(s2_em))
ey <- rnorm(n = n, mean = 0, sd = sqrt(s2_ey))
X <- rnorm(n = n)
M <- a * X + em
Y <- cp * X + b * M + ey
df <- data.frame(X, M, Y)
```

```
# Fit Model in lavaan -----------------------------------------------------
model <- "
  Y ~ cp * X + b * M
  M ~ a * X
  indirect := a * b
  direct := cp
  total := cp + (a * b)
"
fit <- sem(data = df, model = model)

# Monte Carlo -------------------------------------------------------------
unstd <- MC(
  fit,
  R = 100L, # use a large value e.g., 20000L for actual research
  alpha = c(0.001, 0.01, 0.05)
)
summary(unstd)
```

`summary.semmccistd` *Summary Method for an Object of Class* `semmccistd`

Description

Summary Method for an Object of Class `semmccistd`

Usage

```
## S3 method for class 'semmccistd'
summary(object, digits = 4, ...)
```

Arguments

`object`	Object of class `semmccistd`.
`digits`	Digits to print.
`...`	additional arguments.

Value

Returns a matrix of estimates, standard errors, number of Monte Carlo replications, and confidence intervals.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
library(semmcci)
library(lavaan)

# Generate Data ------------------------------------------------------------
n <- 1000
a <- 0.50
b <- 0.50
cp <- 0.25
s2_em <- 1 - a^2
s2_ey <- 1 - cp^2 - a^2 * b^2 - b^2 * s2_em - 2 * cp * a * b
em <- rnorm(n = n, mean = 0, sd = sqrt(s2_em))
ey <- rnorm(n = n, mean = 0, sd = sqrt(s2_ey))
X <- rnorm(n = n)
M <- a * X + em
Y <- cp * X + b * M + ey
df <- data.frame(X, M, Y)

# Fit Model in lavaan ------------------------------------------------------
model <- "
  Y ~ cp * X + b * M
  M ~ a * X
  indirect := a * b
  direct := cp
  total := cp + (a * b)
"
fit <- sem(data = df, model = model, fixed.x = FALSE)

# Monte Carlo --------------------------------------------------------------
unstd <- MC(
  fit,
  R = 100L, # use a large value e.g., 20000L for actual research
  alpha = c(0.001, 0.01, 0.05)
)

# Standardized Monte Carlo -------------------------------------------------
std <- MCStd(unstd)
summary(std)
```

`vcov.semmcci` *Sampling Covariance Matrix of the Parameter Estimates*

Description

Sampling Covariance Matrix of the Parameter Estimates

Usage

```
## S3 method for class 'semmcci'
vcov(object, ...)
```

Arguments

`object`	Object of class `semmccistd`.
`...`	additional arguments.

Value

Returns a matrix of the variance-covariance matrix of parameter estimates.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
library(semmcci)
library(lavaan)

# Generate Data ------------------------------------------------------------
n <- 1000
a <- 0.50
b <- 0.50
cp <- 0.25
s2_em <- 1 - a^2
s2_ey <- 1 - cp^2 - a^2 * b^2 - b^2 * s2_em - 2 * cp * a * b
em <- rnorm(n = n, mean = 0, sd = sqrt(s2_em))
ey <- rnorm(n = n, mean = 0, sd = sqrt(s2_ey))
X <- rnorm(n = n)
M <- a * X + em
Y <- cp * X + b * M + ey
df <- data.frame(X, M, Y)

# Fit Model in lavaan ------------------------------------------------------
model <- "
  Y ~ cp * X + b * M
  M ~ a * X
  indirect := a * b
  direct := cp
  total := cp + (a * b)
"
fit <- sem(data = df, model = model)

# Monte Carlo --------------------------------------------------------------
unstd <- MC(
  fit,
  R = 100L, # use a large value e.g., 20000L for actual research
  alpha = c(0.001, 0.01, 0.05)
)
vcov(unstd)
```

`vcov.semmccistd` *Sampling Covariance Matrix of the Standardized Parameter Estimates*

Description

Sampling Covariance Matrix of the Standardized Parameter Estimates

Usage

```
## S3 method for class 'semmccistd'
vcov(object, ...)
```

Arguments

`object`	Object of class `semmccistd`.
`...`	additional arguments.

Value

Returns a matrix of the variance-covariance matrix of standardized parameter estimates.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
library(semmcci)
library(lavaan)

# Generate Data ------------------------------------------------------------
n <- 1000
a <- 0.50
b <- 0.50
cp <- 0.25
s2_em <- 1 - a^2
s2_ey <- 1 - cp^2 - a^2 * b^2 - b^2 * s2_em - 2 * cp * a * b
em <- rnorm(n = n, mean = 0, sd = sqrt(s2_em))
ey <- rnorm(n = n, mean = 0, sd = sqrt(s2_ey))
X <- rnorm(n = n)
M <- a * X + em
Y <- cp * X + b * M + ey
df <- data.frame(X, M, Y)

# Fit Model in lavaan ------------------------------------------------------
model <- "
  Y ~ cp * X + b * M
  M ~ a * X
  indirect := a * b
```

```
  direct := cp
  total := cp + (a * b)
"
fit <- sem(data = df, model = model, fixed.x = FALSE)

# Monte Carlo ------------------------------------------------------------
unstd <- MC(
  fit,
  R = 100L, # use a large value e.g., 20000L for actual research
  alpha = c(0.001, 0.01, 0.05)
)

# Standardized Monte Carlo -----------------------------------------------
std <- MCStd(unstd)
vcov(std)
```

Index