Package 'SEMsensitivity'

August 21, 2025

Title SEM Sensitivity Analysis

Version 0.1.0

Maintainer Biying Zhou <zby.zhou@mail.utoronto.ca>

Description This package performs sensitivity analysis for Structural Equation Modeling (SEM). It determines which sample points need to be removed for the sign of a specific path in the SEM model to change, thus assessing the robustness of the model.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports lavaan,
dplyr,
R.utils,
semfindr,
stats,
methods,

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Contents

SEMsensitivity *Run Specified Test for SEM Path Sign or Fit Measure Change*

Description

This function runs a specified test by method name or method index to determine either (1) when a specific path in a Structural Equation Modeling (SEM) model changes sign (positive or negative) by iteratively removing data points, or (2) whether a fit measure (e.g., CFI, TLI) exceeds or falls below a specified threshold (e.g., 0.9). The function outputs relevant results for the specified test.

Usage

```
SEMsensitivity(
  df,
  model,
  var_one = NULL,
  var_two = NULL,
  PAR = NULL,
  threshold = 10,
  fit,
  estimates = NULL,
  conc = NULL,
  int = NULL,
  par_value = NULL,
  max_final,
  N,
  signFactor = NULL,
  equalCons = 0,
  calcMeth = NULL,
  ratio = NULL,
  adaptA = NULL,
  alpha = NULL,
  maxTime = NULL,
  pruneNum = NULL,
  measureTest = "cfi",
  fitThreshold = 0.9,
  highGood = TRUE,
  method = 2,
  ...
)
```

Arguments

df	A data frame containing the dataset.
model	A specified SEM model.
var_one	The first variable of interest.
var_two	The second variable of interest.
PAR	The path of interest.
threshold	The threshold for percentage of data dropped.

fit	The fit measure to assess the model.
estimates	The estimates from the SEM model.
conc	The convergence criterion for the SEM model.
int	The interval for checking the parameter.
par_value	The original value of the parameter of interest.
max_final	The maximum number of influential data points to consider.
N	The total number of data points.
signFactor	A factor indicating the direction of parameter change (positive or negative).
equalCons	The equality constraint used in the SEM model (default is 0).
calcMeth	The method used for approximation (default is 'Hessian').
ratio	The ratio used to determine the number of points to check using the exact method (default is 2) in combined method.
adaptA	Logical indicating whether to use adaptive pruning (default is FALSE).
alpha	Manual tuning parameter (default is 0.25).
maxTime	Maximum time allowed for the search in seconds (default is 300).
pruneNum	The number of branches to explore from each node (default is 3).
measureTest	The fit measurement name to be tested (e.g., "cfi", "tli", "rmsea"). Default is "cfi".
fitThreshold	The threshold of the fit measurement. For example, for CFI (measureTest = "cfi"), the threshold could be 0.9. Default is 0.9.
highGood	A boolean indicating whether higher values of the fit measure are better. For instance, for CFI, this should be set to TRUE. Default is TRUE.
method	The method name or index specifying which test to run. Default is "Naive Method with Approximate Influence".
...	Other arguments.

1: "Naive Method with Exact Influence"

2: "Naive Method with Approximate Influence"

3: "Specified Approximation Method"

4: "Simple Depth Method"

5: "Combined Method"

7: "Negamax Search Algorithm Function"

8: "Use Depth Method to Try to Switch Sign of Parameter"

9: "Use Depth Method to Try to Switch Sign of Parameter with Negamax"

11: "Simulated Annealing Method"

12: "Particle Swarm Optimization"

13: "Brute Search with Cut Method"

10: "Use depth method to drop fit measure below threshold"

61: "Finding case deletions required to change fit metric using exact influences"

62: "Finding case deletions required to change fit metric using approximate method"

Value

A list containing the results of the specified test.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)

# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

var_one <- 'dem65' # first term
var_two <- 'ind60' # second term
PAR <- c("dem65~ind60") # full relation
threshold <- 10

# Fit SEM model
fit <- sem(model, data = df)
summary(fit)

# Get Estimates of Parameters from SEM
estimates <- parameterEstimates(fit)

# Determine The Value of The Parameter of Interest
conc <- data.frame(lhs = estimates$lhs, rhs = estimates$rhs, est = estimates$est)
int <- conc %>% filter(lhs == var_one & rhs == var_two)
par_value <- int$est # this is the value of the parameter of interest

# Compute max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100) # perform rounding if necessary
N <- nrow(df) # store number of observations in df for convenience

# Determine whether parameter is negative or positive in order
# to assess which direction to perturb it
signFactor <- ifelse(par_value >= 0, TRUE, FALSE)

# Run specified test by method name
result_by_name <- SEMsensitivity(df, model, var_one, var_two, PAR, threshold, fit,
estimates, conc, int, par_value, max_final, N, signFactor,
```

```
"Naive Method with Exact Influence")
summary(result_by_name)

# Run specified test by method index
result_by_index <- SEMsensitivity(df, model, var_one, var_two, PAR, threshold, fit,
estimates, conc, int, par_value, max_final, N, signFactor, 1)
summary(result_by_index)

## End(Not run)
```

Test1 *Naive Method with Exact Influence*

Description

Remove a fixed percentage of samples (determined by the exact influence) at a time and refit the model to observe the change in the path of interest. Use method 1 - Naive Method with Exact Influence.

Usage

```
Test1(
  df,
  model,
  var_one,
  var_two,
  PAR,
  threshold,
  fit,
  estimates,
  conc,
  int,
  par_value,
  max_final,
  N,
  signFactor,
  ...
)
```

Arguments

df	A data frame containing the dataset.
model	A specified SEM model.
var_one	The first variable of interest.
var_two	The second variable of interest.
PAR	The path of interest.
threshold	The threshold for percentage of data dropped.
fit	The SEM object.
estimates	The estimates from the SEM model.

conc	A data frame containing the parameter of interest.
int	The value of the path of interest.
par_value	The original value of the parameter of interest.
max_final	The maximum number of influential data points to consider.
N	The total number of data points.
signFactor	A factor indicating the direction of parameter change (positive or negative).
...	Other arguments.

Value

A list of class TestResult containing:

value	The value of the parameter after dropping the influential points.
points	The indices of the most influential data points.
methodname	The name of the method used.
testindex	The index of the test performed.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)

# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

var_one <- 'dem65' # first term
var_two <- 'ind60' # second term
PAR <- c("dem65~ind60") # full relation
threshold <- 10

# Fit SEM model
fit <- lavaan::sem(model, data = df)
```

```
summary(fit)

# Get Estimates of Parameters from SEM
estimates <- parameterEstimates(fit)

# Determine The Value of The Parameter of Interest
conc <- data.frame(lhs = estimates$lhs, rhs = estimates$rhs, est = estimates$est)
int <- conc %>% filter(lhs == var_one & rhs == var_two)
par_value <- int$est # this is the value of the parameter of interest

# Compute max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100) # perform rounding if necessary
N <- nrow(df) # store number of observations in df for convenience

# Determine whether parameter is negative or positive in order
# to assess which direction to perturb it
signFactor <- ifelse(par_value >= 0, TRUE, FALSE)

Test1_result <- Test1(df, model, var_one, var_two, PAR, threshold, fit, estimates,
 conc, int, par_value, max_final, N, signFactor)
summary(Test1_result)

## End(Not run)
```

`Test10` *Use depth method to drop fit measure below threshold*

Description

This function uses the depth method to iteratively remove the most influential data points based on influence scores to reduce the fit measure below a specified threshold. The method is based on approximating the influence of each data point and is similar to TEST 8 but focused on fit metrics like CFI, RMSEA, etc.

Usage

```
Test10(
  df,
  model,
  fit,
  max_final,
  N,
  measureTest = "cfi",
  fitThreshold = 0.9,
  highGood = TRUE,
  ...
)
```

Arguments

`df`	A data frame containing the dataset.
`model`	A specified SEM model.

fit	The SEM object after fitting the model.
max_final	The maximum number of influential data points to consider for removal.
N	The total number of data points.
measureTest	The fit measurement name to be tested (e.g., "cfi", "tli", "rmsea"). Default is "cfi".
fitThreshold	The threshold of the fit measurement. For example, for CFI (measureTest = "cfi"), the threshold could be 0.9. Default is 0.9.
highGood	A boolean indicating whether higher values of the fit measure are better. For instance, for CFI, this should be set to TRUE. Default is TRUE.
...	Other arguments.

Value

A list of class `TestResult10` containing:

methodname	The name of the method used.
testindex	The index of the test performed.
original_fit_value	The original value of the fit measurement before data points were removed.
final_fit_value	The fit value after the influential data points were removed.
num_drops	The number of data points dropped to achieve the desired fit value reduction.
depthdiffFit	The difference between the original fit value and the threshold.
depthDropScore	The cumulative drop in the fit value after removing points.
final_drops	The indices of the most influential data points dropped.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)

# Import data
df <- PoliticalDemocracy

# Build SEM model
model <- '
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
'

fit <- lavaan::sem(model, data = df)
max_final <- ceiling(10 * nrow(df) / 100)  # dropping 10% of the data points
N <- nrow(df)

Test10_result <- Test10(df, model, fit, max_final, N,
measureTest = "cfi", fitThreshold = 0.9, highGood = TRUE)
```

```
summary(Test10_result)

## End(Not run)
```

`Test11` *Use Simulated Annealing Method to Try to Switch Sign of Parameter*

Description

This function uses the simulated annealing method to iteratively remove data points in order to switch the sign of a specific path in a Structural Equation Modeling (SEM) model.

Usage

```
Test11(
  df,
  model,
  var_one,
  var_two,
  PAR,
  threshold,
  fit,
  estimates,
  conc,
  int,
  par_value,
  max_final,
  N,
  signFactor,
  ...
)
```

Arguments

`df`	A data frame containing the dataset.
`model`	A specified SEM model.
`var_one`	The first variable of interest.
`var_two`	The second variable of interest.
`PAR`	The path of interest.
`threshold`	The threshold for the percentage of data dropped.
`fit`	The SEM object.
`estimates`	The estimates from the SEM model.
`conc`	A data frame containing the parameter of interest.
`int`	The value of the path of interest.
`par_value`	The original value of the parameter of interest.
`max_final`	The maximum number of influential data points to consider.
`N`	The total number of data points.
`signFactor`	A factor indicating the direction of parameter change (positive or negative).
`...`	Other arguments.

Value

A list of class `TestResult11` containing:

`annealingDrops`	The indices of the most influential data points selected by the simulated annealing method.
`initialValue`	The original value of the parameter.
`finalValue`	The final value of the parameter after applying the simulated annealing method.
`methodname`	The name of the method used.
`testindex`	The index of the test performed.
`PAR`	The path of interest that was evaluated.
`threshold`	The threshold used for the percentage of dropped points.
`N`	The total number of data points.
`max_final`	The maximum number of points allowed to be dropped.
`par_value`	The original value of the parameter.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)

# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

var_one <- 'dem65' # first term
var_two <- 'ind60' # second term
PAR <- c("dem65~ind60") # full relation
threshold <- 10

# Fit SEM model
fit <- lavaan::sem(model, data = df)
summary(fit)
```

```
# Get Estimates of Parameters from SEM
estimates <- parameterEstimates(fit)

# Determine the value of the parameter of interest
conc <- data.frame(lhs = estimates$lhs, rhs = estimates$rhs, est = estimates$est)
int <- conc %>% filter(lhs == var_one & rhs == var_two)
par_value <- int$est # this is the value of the parameter of interest

# Compute the max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100) # perform rounding if necessary
N <- nrow(df) # store the number of observations in df for convenience

# Determine whether the parameter is negative or positive in order
# to assess which direction to perturb it
signFactor <- ifelse(par_value >= 0, TRUE, FALSE)

Test11_result = Test11(df, model, var_one, var_two, PAR, threshold, fit, estimates,
conc, int, par_value, max_final, N, signFactor)
summary(Test11_result)

## End(Not run)
```

`Test12` *Use Particle Swarm Optimization (PSO) to Try to Switch Sign of Parameter*

Description

This function uses the Particle Swarm Optimization (PSO) method to iteratively remove data points in order to switch the sign of a specific path in a Structural Equation Modeling (SEM) model.

Usage

```
Test12(
  df,
  model,
  var_one,
  var_two,
  PAR,
  threshold,
  fit,
  estimates,
  conc,
  int,
  par_value,
  max_final,
  N,
  signFactor,
  ...
)
```

Arguments

df	A data frame containing the dataset.
model	A specified SEM model.
var_one	The first variable of interest.
var_two	The second variable of interest.
PAR	The path of interest.
threshold	The threshold for the percentage of data dropped.
fit	The SEM object.
estimates	The estimates from the SEM model.
conc	A data frame containing the parameter of interest.
int	The value of the path of interest.
par_value	The original value of the parameter of interest.
max_final	The maximum number of influential data points to consider.
N	The total number of data points.
signFactor	A factor indicating the direction of parameter change (positive or negative).
...	Other arguments.

Value

A list of class `TestResult12` containing:

psoDrops	The indices of the most influential data points selected by the PSO method.
initialValue	The original value of the parameter.
finalValue	The final value of the parameter after applying the PSO method.
methodname	The name of the method used.
testindex	The index of the test performed.
PAR	The path of interest that was evaluated.
threshold	The threshold used for the percentage of dropped points.
N	The total number of data points.
max_final	The maximum number of points allowed to be dropped.
par_value	The original value of the parameter.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)

# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
```

```
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

var_one <- 'dem65' # first term
var_two <- 'ind60' # second term
PAR <- c("dem65~ind60") # full relation
threshold <- 10

# Fit SEM model
fit <- lavaan::sem(model, data = df)
summary(fit)

# Get Estimates of Parameters from SEM
estimates <- parameterEstimates(fit)

# Determine the value of the parameter of interest
conc <- data.frame(lhs = estimates$lhs, rhs = estimates$rhs, est = estimates$est)
int <- conc %>% filter(lhs == var_one & rhs == var_two)
par_value <- int$est # this is the value of the parameter of interest

# Compute the max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100) # perform rounding if necessary
N <- nrow(df) # store the number of observations in df for convenience

# Determine whether the parameter is negative or positive in order
# to assess which direction to perturb it
signFactor <- ifelse(par_value >= 0, TRUE, FALSE)

Test12_result = Test12(df, model, var_one, var_two, PAR, threshold, fit, estimates,
conc, int, par_value, max_final, N, signFactor)
summary(Test12_result)

## End(Not run)
```

`Test13` *Use Brute Search with Cut Method to Try to Switch Sign of Parameter*

Description

This function uses a brute-force search method with pruning (cutting) to iteratively search for combinations of data points that switch the sign of a specific path in a Structural Equation Modeling (SEM) model.

Usage

```
Test13(
  df,
  model,
  var_one,
  var_two,
  PAR,
  threshold,
  fit,
  estimates,
  conc,
  int,
  par_value,
  max_final,
  N,
  signFactor,
  ...
)
```

Arguments

df	A data frame containing the dataset.
model	A specified SEM model.
var_one	The first variable of interest.
var_two	The second variable of interest.
PAR	The path of interest.
threshold	The threshold for the percentage of data dropped.
fit	The SEM object.
estimates	The estimates from the SEM model.
conc	A data frame containing the parameter of interest.
int	The value of the path of interest.
par_value	The original value of the parameter of interest.
max_final	The maximum number of influential data points to consider.
N	The total number of data points.
signFactor	A factor indicating the direction of parameter change (positive or negative).
...	Other arguments.

Value

A list of class `TestResult13` containing:

bruteSearchDrops	The indices of the most influential data points selected by the brute search method.
initialValue	The original value of the parameter.
finalValue	The final value of the parameter after applying the brute search method.
methodname	The name of the method used.

testindex	The index of the test performed.
PAR	The path of interest that was evaluated.
threshold	The threshold used for the percentage of dropped points.
N	The total number of data points.
max_final	The maximum number of points allowed to be dropped.
par_value	The original value of the parameter.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)

# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

var_one <- 'dem65' # first term
var_two <- 'ind60' # second term
PAR <- c("dem65~ind60") # full relation
threshold <- 10

# Fit SEM model
fit <- lavaan::sem(model, data = df)
summary(fit)

# Get Estimates of Parameters from SEM
estimates <- parameterEstimates(fit)

# Determine the value of the parameter of interest
conc <- data.frame(lhs = estimates$lhs, rhs = estimates$rhs, est = estimates$est)
int <- conc %>% filter(lhs == var_one & rhs == var_two)
par_value <- int$est # this is the value of the parameter of interest

# Compute the max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100) # perform rounding if necessary
```

```
N <- nrow(df) # store the number of observations in df for convenience

# Determine whether the parameter is negative or positive in order
# to assess which direction to perturb it
signFactor <- ifelse(par_value >= 0, TRUE, FALSE)

Test13_result = Test13(df, model, var_one, var_two, PAR, threshold, fit, estimates,
conc, int, par_value, max_final, N, signFactor)
summary(Test13_result)

## End(Not run)
```

`Test2` *Naive Method with Approximate Influence*

Description

Remove a fixed percentage of samples (determined by the approximate influence) at a time and refit the model to observe the change in the parameter of interest. Use method 2 - Naive Method with Approximate Influence.

Usage

```
Test2(
  df,
  model,
  var_one,
  var_two,
  PAR,
  threshold,
  fit,
  estimates,
  conc,
  int,
  par_value,
  max_final,
  N,
  signFactor,
  equalCons = 0,
  calcMeth = "Hessian",
  ...
)
```

Arguments

`df`	A data frame containing the dataset.
`model`	A specified SEM model.
`var_one`	The first variable of interest.
`var_two`	The second variable of interest.
`PAR`	The path of interest.
`threshold`	The threshold for percentage of data dropped.

fit	The SEM object.
estimates	The estimates from the SEM model.
conc	A data frame containing the parameter of interest.
int	The value of the path of interest.
par_value	The original value of the parameter of interest.
max_final	The maximum number of influential data points to consider.
N	The total number of data points.
signFactor	A factor indicating the direction of parameter change (positive or negative).
equalCons	The equality constraint used in the SEM model (default is 0).
calcMeth	The method used for approximation (default is 'Hessian').
...	Other arguments.

Value

A list of class `TestResult` containing:

value	The value of the parameter after dropping the influential points.
points	The indices of the most influential data points.
methodname	The name of the method used.
testindex	The index of the test performed.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)

# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

var_one <- 'dem65' # first term
var_two <- 'ind60' # second term
```

```
PAR <- c("dem65~ind60") # full relation
threshold <- 10

# Fit SEM model
fit <- lavaan::sem(model, data = df)
summary(fit)

# Get Estimates of Parameters from SEM
estimates <- parameterEstimates(fit)

# Determine The Value of The Parameter of Interest
conc <- data.frame(lhs = estimates$lhs, rhs = estimates$rhs, est = estimates$est)
int <- conc %>% filter(lhs == var_one & rhs == var_two)
par_value <- int$est # this is the value of the parameter of interest

# Compute max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100) # perform rounding if necessary
N <- nrow(df) # store number of observations in df for convenience

# Determine whether parameter is negative or positive in order
# to assess which direction to perturb it
signFactor <- ifelse(par_value >= 0, TRUE, FALSE)

Test2_result <- Test2(df, model, var_one, var_two, PAR, threshold, fit, estimates,
conc, int, par_value, max_final, N, signFactor, equalCons, calcMeth)
summary(Test2_result)

## End(Not run)
```

`Test3` *Specified Approximation Method*

Description

This function determines when a specific path in a Structural Equation Modeling (SEM) model changes sign by iteratively removing most influential data points (determined by naive method) and outputs relevant results. Use method 3 - Specified Approximation Method.

Usage

```
Test3(
  df,
  model,
  var_one,
  var_two,
  PAR,
  threshold,
  fit,
  estimates,
  conc,
  int,
  par_value,
  max_final,
```

```
  N,
  signFactor,
  equalCons = 0,
  calcMeth = "Hessian",
  ...
)
```

Arguments

df	A data frame containing the dataset.
model	A specified SEM model.
var_one	The first variable of interest.
var_two	The second variable of interest.
PAR	The path of interest.
threshold	The threshold for percentage of data dropped.
fit	The SEM object.
estimates	The estimates from the SEM model.
conc	A data frame containing the parameter of interest.
int	The value of the path of interest.
par_value	The original value of the parameter of interest.
max_final	The maximum number of influential data points to consider.
N	The total number of data points.
signFactor	A factor indicating the direction of parameter change (positive or negative).
equalCons	The equality constraint used in the SEM model (default is 0).
calcMeth	The method used for approximation (default is 'Hessian').
...	Other arguments.

Value

A list of class `TestResult3` containing:

methodname	The name of the method used.
testindex	The index of the test performed.
max_drops	The maximum number of data points dropped.
est_diff	The expected change in parameter value.
act_diff	The actual change in parameter value.
final_par_value	The parameter value after dropping the influential points.
initial_par_value	The original parameter value.
sign_switch_possible	Logical indicating whether it was possible to change the sign of the parameter.
dropped_points	The indices of the most influential data points.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)

# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

var_one <- 'dem65' # first term
var_two <- 'ind60' # second term
PAR <- c("dem65~ind60") # full relation
threshold <- 10

# Fit SEM model
fit <- lavaan::sem(model, data = df)
summary(fit)

# Get Estimates of Parameters from SEM
estimates <- parameterEstimates(fit)

# Determine The Value of The Parameter of Interest
conc <- data.frame(lhs = estimates$lhs, rhs = estimates$rhs, est = estimates$est)
int <- conc %>% filter(lhs == var_one & rhs == var_two)
par_value <- int$est # this is the value of the parameter of interest

# Compute max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100) # perform rounding if necessary
N <- nrow(df) # store number of observations in df for convenience

# Determine whether parameter is negative or positive in order
# to assess which direction to perturb it
signFactor <- ifelse(par_value >= 0, TRUE, FALSE)

Test3_result = Test3(df, model, var_one, var_two, PAR, threshold, fit, estimates,
conc, int, par_value, max_final, N, signFactor,equalCons = 0, calcMeth)
summary(Test3_result)
```

```
## End(Not run)
```

`Test4` *Simple depth method*

Description

This function determines a specific path in a Structural Equation Modeling (SEM) model value changing by removing samples iteratively, in which influences are determined by naive method and outputs relevant results. Use method 4 - Simple depth method.

Usage

```
Test4(
  df,
  model,
  var_one,
  var_two,
  PAR,
  threshold,
  fit,
  estimates,
  conc,
  int,
  par_value,
  max_final,
  N,
  signFactor,
  equalCons = 0,
  calcMeth = "Hessian",
  ...
)
```

Arguments

`df`	A data frame containing the dataset.
`model`	A specified SEM model.
`var_one`	The first variable of interest.
`var_two`	The second variable of interest.
`PAR`	The path of interest.
`threshold`	The threshold for percentage of data dropped.
`fit`	The SEM object.
`estimates`	The estimates from the SEM model.
`conc`	A data frame containing the parameter of interest.
`int`	The value of the path of interest.
`par_value`	The original value of the parameter of interest.
`max_final`	The maximum number of influential data points to consider.

N	The total number of data points.
signFactor	A factor indicating the direction of parameter change (positive or negative).
equalCons	The equality constraint used in the SEM model.
calcMeth	The method used for approximation (default is 'Hessian').
...	Other arguments.

Value

A list of class TestResult containing:

value	The value of the parameter after dropping the influential points.
points	The indices of the most influential data points.
methodname	The name of the method used.
testindex	The index of the test performed.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)

# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

var_one <- 'dem65' # first term
var_two <- 'ind60' # second term
PAR <- c("dem65~ind60") # full relation
threshold <- 10

# Fit SEM model
fit <- lavaan::sem(model, data = df)
summary(fit)

# Get Estimates of Parameters from SEM
```

```
estimates <- parameterEstimates(fit)

# Determine The Value of The Parameter of Interest
conc <- data.frame(lhs = estimates$lhs, rhs = estimates$rhs, est = estimates$est)
int <- conc %>% filter(lhs == var_one & rhs == var_two)
par_value <- int$est # this is the value of the parameter of interest

# Compute max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100) # perform rounding if necessary
N <- nrow(df) # store number of observations in df for convenience

# Determine whether parameter is negative or positive in order
# to assess which direction to perturb it
signFactor <- ifelse(par_value >= 0, TRUE, FALSE)

Test4_result = Test4(df, model, var_one, var_two, PAR, threshold, fit, estimates,
conc, int, par_value, max_final, N, signFactor,equalCons,calcMeth)
summary(Test4_result)

## End(Not run)
```

`Test5` *Combined Method*

Description

This function determines a specific path in a Structural Equation Modeling (SEM) model value changing by removing samples iteratively, in which influences are determined by both naive method and approximate method and outputs relevant results. Use method 5 - Combined Method.

Usage

```
Test5(
  df,
  model,
  var_one,
  var_two,
  PAR,
  threshold,
  fit,
  estimates,
  conc,
  int,
  par_value,
  max_final,
  N,
  signFactor,
  equalCons = 0,
  calcMeth = "Hessian",
  ratio = 2,
  ...
)
```

Arguments

df	A data frame containing the dataset.
model	A specified SEM model.
var_one	The first variable of interest.
var_two	The second variable of interest.
PAR	The path of interest.
threshold	The threshold for percentage of data dropped.
fit	The SEM object.
estimates	The estimates from the SEM model.
conc	A data frame containing the parameter of interest.
int	The value of the path of interest.
par_value	The original value of the parameter of interest.
max_final	The maximum number of influential data points to consider.
N	The total number of data points.
signFactor	A factor indicating the direction of parameter change (positive or negative).
equalCons	The equality constraint used in the SEM model.
calcMeth	The method used for approximation (default is 'Hessian').
ratio	The ratio used to determine the number of points to check using the exact method (default is 2).
...	Other arguments.

Value

A list of class `TestResult` containing:

value	The value of the parameter after dropping the influential points.
points	The indices of the most influential data points.
methodname	The name of the method used.
testindex	The index of the test performed.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)

# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
```

```
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

var_one <- 'dem65' # first term
var_two <- 'ind60' # second term
PAR <- c("dem65~ind60") # full relation
threshold <- 10

# Fit SEM model
fit <- lavaan::sem(model, data = df)
summary(fit)

# Get Estimates of Parameters from SEM
estimates <- parameterEstimates(fit)

# Determine The Value of The Parameter of Interest
conc <- data.frame(lhs = estimates$lhs, rhs = estimates$rhs, est = estimates$est)
int <- conc %>% filter(lhs == var_one & rhs == var_two)
par_value <- int$est # this is the value of the parameter of interest

# Compute max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100) # perform rounding if necessary
N <- nrow(df) # store number of observations in df for convenience

# Determine whether parameter is negative or positive in order
# to assess which direction to perturb it
signFactor <- ifelse(par_value >= 0, TRUE, FALSE)

Test5_result = Test5(df, model, var_one, var_two, PAR, threshold, fit, estimates,
conc, int, par_value, max_final, N, signFactor, equalCons,calcMeth)
summary(Test5_result)

## End(Not run)
```

`Test61` *Finding case deletions required to change fit metric using exact influences*

Description

Remove a fixed percentage of samples (determined by the exact influence) at a time and refit the model to observe the change in the path of interest. Use method 1 - Naive Method with Exact Influence.

Usage

```
Test61(
```

```
  df,
  model,
  threshold,
  fit,
  max_final,
  N,
  measureTest = "cfi",
  fitThreshold = 0.9,
  highGood = T,
  ...
)
```

Arguments

df	A data frame containing the dataset.
model	A specified SEM model.
threshold	The threshold for percentage of data dropped.
fit	The SEM object.
max_final	The maximum number of influential data points to consider.
N	The total number of data points.
measureTest	The fit measurement name. Can be "cfi", "chisq", "tli", "rmsea".
fitThreshold	The threshold of the fit measurement to be a "good" model. For example, for CFI (measureTest = "cfi), this threshold can be 0.9.
highGood	A boolean argument stating if the fit measurement is higher the better. For CFI, this argument is TRUE.
...	Other arguments.

Value

A list of class `TestResult61` containing:

methodname	The name of the method used.
testindex	The index of the test performed.
original_fit_value	The original value of the fit measurement.
final_fit_value	The fit value after dropping the influential points.
num_drops	The number of data points dropped.
threshold_crossed	Logical indicating whether the threshold was crossed.
final_drops	The indices of the most influential data points dropped.
measureTest	The name of the fit measurement used.
fitThreshold	The threshold of the fit measurement used.
highGood	Logical indicating if a higher value is better for the fit measurement.
exact_threshold_tally	The tally value required to cross the threshold.
model_exact_threshold_final	The final fit value after dropping points sufficient to cross the threshold.
max_final	The maximum number of influential data points to consider.
N	The total number of data points.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)

# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

threshold <- 10

# Fit SEM model
fit <- lavaan::sem(model, data = df)
summary(fit)

# Compute max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100)
N <- nrow(df)

Test61_result <- Test61(df, model, threshold, fit, max_final, N,
 measureTest = "cfi", fitThreshold = 0.9, highGood = T)
summary(Test61_result)

## End(Not run)
```

`Test62` *Finding case deletions required to change fit metric using approximate method*

Description

Remove a fixed percentage of samples (determined by the approximate influence) at a time and refit the model to observe the change in the fit metric of interest. Use method 2 - Approximate Method.

Usage

```
Test62(
  df,
  model,
  threshold,
  fit,
  max_final,
  N,
  equalCons = 0,
  measureTest = "cfi",
  fitThreshold = 0.9,
  highGood = T,
  ...
)
```

Arguments

df	A data frame containing the dataset.
model	A specified SEM model.
threshold	The threshold for percentage of data dropped.
fit	The SEM object.
max_final	The maximum number of influential data points to consider.
N	The total number of data points.
equalCons	Logical; whether equality constraints exist in the model. The approximate method can only be run if no equality constraints are present (default is 0).
measureTest	The fit measurement name. Can be "cfi", "chisq", "tli", "rmsea".
fitThreshold	The threshold of the fit measurement to be a "good" model. For example, for CFI (measureTest = "cfi"), this threshold can be 0.9.
highGood	A boolean argument stating if the fit measurement is higher the better. For CFI, this argument is TRUE.
...	Other arguments.

Value

A list of class `TestResult62` containing:

methodname	The name of the method used.
testindex	The index of the test performed.
original_fit_value	The original value of the fit measurement.
final_fit_value	The fit value after dropping the influential points using the approximate method.
num_drops	The number of data points dropped.
threshold_crossed	Logical indicating whether the threshold was crossed.
final_drops	The indices of the most influential data points dropped using the approximate method.
measureTest	The name of the fit measurement used.

fitThreshold	The threshold of the fit measurement used.
highGood	Logical indicating if a higher value is better for the fit measurement.
appx_threshold_tally	The tally value required to cross the threshold using the approximate method.
model_appx_threshold_final	The final fit value after dropping points sufficient to cross the threshold using the approximate method.
max_final	The maximum number of influential data points to consider.
N	The total number of data points.
equalCons	Logical indicating whether equality constraints exist in the model.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)

# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

threshold <- 10

# Fit SEM model
fit <- lavaan::sem(model, data = df)
summary(fit)

# Compute max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100)
N <- nrow(df)

Test62_result <- Test62(df, model, threshold, fit, max_final, N, equalCons = 0,
 measureTest = "cfi", fitThreshold = 0.9, highGood = T)
summary(Test62_result)

## End(Not run)
```

Test7 *Negamax Search Algorithm Function*

Description

Utilizes a Negamax search algorithm to iteratively drop data points and update the SEM, ultimately aiming to minimize the parameter value of interest. Use method 7 - Negamax Search Algorithm Function.

Usage

```
Test7(
  df,
  model,
  var_one,
  var_two,
  PAR,
  threshold,
  fit,
  estimates,
  conc,
  int,
  par_value,
  max_final,
  N,
  signFactor,
  equalCons,
  calcMeth = "Hessian",
  adaptA = F,
  alpha = 0.25,
  maxTime = 300,
  pruneNum = 3,
  ...
)
```

Arguments

df	A data frame containing the dataset.
model	A specified SEM model.
var_one	The first variable of interest.
var_two	The second variable of interest.
PAR	The path of interest.
threshold	The threshold for percentage of data dropped.
fit	The SEM object.
estimates	The estimates from the SEM model.
conc	A data frame containing the parameter of interest.
int	The value of the path of interest.
par_value	The original value of the parameter of interest.

max_final	The maximum number of influential data points to consider.
N	The total number of data points.
signFactor	A factor indicating the direction of parameter change (positive or negative).
equalCons	The equality constraint used in the SEM model.
calcMeth	The method used for approximation (default is 'Hessian').
adaptA	Logical indicating whether to use adaptive pruning (default is FALSE).
alpha	Manual tuning parameter (default is 0.25).
maxTime	Maximum time allowed for the search in seconds (default is 300).
pruneNum	The number of branches to explore from each node (default is 3).
...	Other arguments.

Value

A list of class `TestResult7` containing:

value	The final value of the parameter after dropping the influential points.
methodname	The name of the method used.
testindex	The index of the test performed.
deletedPoints	The indices of the most influential data points.
initialValue	The original value of the parameter.
finalValue	The final value of the parameter.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)

# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'
```

```
var_one <- 'dem65' # first term
var_two <- 'ind60' # second term
PAR <- c("dem65~ind60") # full relation
threshold <- 10

# Fit SEM model
fit <- lavaan::sem(model, data = df)
summary(fit)

# Get Estimates of Parameters from SEM
estimates <- parameterEstimates(fit)

# Determine The Value of The Parameter of Interest
conc <- data.frame(lhs = estimates$lhs, rhs = estimates$rhs, est = estimates$est)
int <- conc %>% filter(lhs == var_one & rhs == var_two)
par_value <- int$est # this is the value of the parameter of interest

# Compute max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100) # perform rounding if necessary
N <- nrow(df) # store number of observations in df for convenience

# Determine whether parameter is negative or positive in order
# to assess which direction to perturb it
signFactor <- ifelse(par_value >= 0, TRUE, FALSE)

  Test7_result = Test7(df, model, var_one, var_two, PAR, threshold, fit, estimates,
  conc, int, par_value, max_final, N, signFactor,equalCons,calcMeth = "Hessian",
  adaptA = F, alpha= 0.25, maxTime = 300, pruneNum = 3)
summary(Test7_result)

## End(Not run)
```

`Test8` *Use Depth Method to Try to Switch Sign of Parameter*

Description

This function uses a depth method to iteratively remove data points in order to switch the sign of a specific path in a Structural Equation Modeling (SEM) model.

Usage

```
Test8(
  df,
  model,
  var_one,
  var_two,
  PAR,
  threshold,
  fit,
  estimates,
  conc,
  int,
```

```
  par_value,
  max_final,
  N,
  signFactor,
  equalCons,
  calcMeth = "Hessian",
  ...
)
```

Arguments

df	A data frame containing the dataset.
model	A specified SEM model.
var_one	The first variable of interest.
var_two	The second variable of interest.
PAR	The path of interest.
threshold	The threshold for percentage of data dropped.
fit	The SEM object.
estimates	The estimates from the SEM model.
conc	A data frame containing the parameter of interest.
int	The value of the path of interest.
par_value	The original value of the parameter of interest.
max_final	The maximum number of influential data points to consider.
N	The total number of data points.
signFactor	A factor indicating the direction of parameter change (positive or negative).
equalCons	The equality constraint used in the SEM model.
calcMeth	The method used for approximation (default is 'Hessian').
...	Other arguments.

Value

A list of class `TestResult8` containing:

deletedPoints	The indices of the most influential data points.
initialValue	The original value of the parameter.
finalValue	The final value of the parameter.
methodname	The name of the method used.
testindex	The index of the test performed.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)
```

```
# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

var_one <- 'dem65' # first term
var_two <- 'ind60' # second term
PAR <- c("dem65~ind60") # full relation
threshold <- 10

# Fit SEM model
fit <- lavaan::sem(model, data = df)
summary(fit)

# Get Estimates of Parameters from SEM
estimates <- parameterEstimates(fit)

# Determine The Value of The Parameter of Interest
conc <- data.frame(lhs = estimates$lhs, rhs = estimates$rhs, est = estimates$est)
int <- conc %>% filter(lhs == var_one & rhs == var_two)
par_value <- int$est # this is the value of the parameter of interest

# Compute max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100) # perform rounding if necessary
N <- nrow(df) # store number of observations in df for convenience

# Determine whether parameter is negative or positive in order
# to assess which direction to perturb it
signFactor <- ifelse(par_value >= 0, TRUE, FALSE)

  Test8_result = Test8(df, model, var_one, var_two, PAR, threshold, fit, estimates,
  conc, int, par_value, max_final, N, signFactor,equalCons,calcMeth = "Hessian")
summary(Test8_result)

## End(Not run)
```

Test9 *Use Depth Method to Try to Switch Sign of Parameter*

Description

This function uses a depth method combined with a Negamax search algorithm to iteratively remove data points in order to switch the sign of a specific path in a Structural Equation Modeling (SEM) model.

Usage

```
Test9(
  df,
  model,
  var_one,
  var_two,
  PAR,
  threshold,
  fit,
  estimates,
  conc,
  int,
  par_value,
  max_final,
  N,
  signFactor,
  equalCons,
  calcMeth = "Hessian",
  ...
)
```

Arguments

`df`	A data frame containing the dataset.
`model`	A specified SEM model.
`var_one`	The first variable of interest.
`var_two`	The second variable of interest.
`PAR`	The path of interest.
`threshold`	The threshold for percentage of data dropped.
`fit`	The SEM object.
`estimates`	The estimates from the SEM model.
`conc`	A data frame containing the parameter of interest.
`int`	The value of the path of interest.
`par_value`	The original value of the parameter of interest.
`max_final`	The maximum number of influential data points to consider.
`N`	The total number of data points.
`signFactor`	A factor indicating the direction of parameter change (positive or negative).
`equalCons`	The equality constraint used in the SEM model.
`calcMeth`	The method used for approximation (default is 'Hessian').
`...`	Other arguments.

Value

A list of class `TestResult9` containing:

`deletedPoints`	The indices of the most influential data points.
`initialValue`	The original value of the parameter.
`finalValue`	The final value of the parameter.
`methodname`	The name of the method used.
`testindex`	The index of the test performed.
`r_squared`	R-squared value for the approximation, if applicable.
`predictedReduction`	Predicted reduction in parameter, if applicable.
`message`	Summary message of the results.
`failureMessage`	Failure message if the sign switch was unsuccessful.

Examples

```
## Not run:
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(simsem)

# Import data
df <- PoliticalDemocracy

# Build Model
model <- '
  # measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

var_one <- 'dem65' # first term
var_two <- 'ind60' # second term
PAR <- c("dem65~ind60") # full relation
threshold <- 10

# Fit SEM model
fit <- lavaan::sem(model, data = df)
summary(fit)

# Get Estimates of Parameters from SEM
```

```
estimates <- parameterEstimates(fit)

# Determine The Value of The Parameter of Interest
conc <- data.frame(lhs = estimates$lhs, rhs = estimates$rhs, est = estimates$est)
int <- conc %>% filter(lhs == var_one & rhs == var_two)
par_value <- int$est # this is the value of the parameter of interest

# Compute max number of points to be dropped
max_final <- ceiling(threshold * nrow(df) / 100) # perform rounding if necessary
N <- nrow(df) # store number of observations in df for convenience

# Determine whether parameter is negative or positive in order
# to assess which direction to perturb it
signFactor <- ifelse(par_value >= 0, TRUE, FALSE)

Test9_result <- Test9(df, model, var_one, var_two, PAR, threshold, fit, estimates,
conc, int, par_value, max_final, N, signFactor, equalCons, calcMeth = "Hessian")
summary(Test9_result)

## End(Not run)
```

Index