The Journal of Open Source Software

SEMsensitivity: An R Package for Sensitivity Analysis in Structural Equation Modeling

Biying Zhou [1] **and Feng Ji** [1¶]

1 Department of Applied Psychology & Human Development, University of Toronto, Toronto, Canada ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 14 October 2025
Published: unpublished

Summary

The package, SEMsensitivity, addresses the need for robust methods to validate the stability and reliability of SEMs, particularly in light of the replication crisis in psychology and social sciences. Traditional sensitivity analyses often fall short in assessing the impact of individual data points on complex models. SEMsensitivity overcomes these limitations by providing tools to identify influential data points and evaluate their effect on SEM paths. The package integrates with popular R packages like lavaan, dplyr, and semfindr, offering a user-friendly interface and efficient computation. It employs various methods, including Taylor series approximations and the Negamax search algorithm, to iteratively remove data points and assess their influence on model parameters. It is publicly available on GitHub and can be easily installed and utilized by researchers to ensure the robustness and reliability of their SEM models.

Statement of Need

In statistical analysis, ensuring the robustness of models is significant, especially in the context of Structural Equation Modeling (SEM). The replication crisis in psychology and the social sciences highlights the critical need for methodologies that can validate the stability and reliability of statistical findings. Traditional sensitivity analyses often fall short in assessing the impact of individual data points on the overall model conclusions, particularly in complex, multi-dimensional datasets where influential observations may not be readily apparent.

The R package SEMsensitivity addresses this gap by providing a suite of tools specifically designed for performing sensitivity analysis on SEMs. This package allows researchers to determine which specific sample points need to be removed for a particular path in the SEM model to change sign, thus assessing the robustness of the model. By iteratively removing data points and observing changes in model parameters, SEMsensitivity offers a detailed view of how individual observations influence the overall model, facilitating a deeper understanding of model stability.

The importance of this package lies in its ability to streamline and simplify the process of influence analysis for SEMs. Traditionally, such analyses are computationally intensive and challenging to perform, often requiring bespoke solutions that are not readily accessible to all researchers. SEMsensitivity integrates with popular R packages such as lavaan, dplyr, and semfindr, providing a user-friendly interface and leveraging existing computational tools to deliver efficient and accurate results.

By automating the detection of influential data points and providing clear, interpretable outputs, SEMsensitivity empowers researchers to make informed decisions about data dropping and model robustness. This capability is particularly crucial in fields where data integrity and

the reproducibility of results are under intense scrutiny. The package not only enhances the methodological rigor of SEM analyses but also contributes to the broader effort of improving transparency and reliability in scientific research.

In summary, `SEMsensitivity` is a vital tool for researchers conducting SEMs, offering a robust framework for sensitivity analysis that is both accessible and computationally efficient. Its development addresses a critical need in the current research landscape, providing a practical solution to ensure the robustness and reliability of statistical models.

Method

The primary goal of `SEMsensitivity` is to provide robust methods for sensitivity analysis in SEM. This package includes multiple methods to determine the influence of individual data points on SEM paths and fit indices. Each method iteratively removes data points to identify those that significantly affect the model's parameters. Among these methods, one evaluates when a path coefficient changes its direction, such as from positive to negative. Additionally, another method is designed to monitor the model's fit indices, such as when the Comparative Fit Index (CFI) drops below a threshold like 0.9, indicating a decline in model adequacy. These methods provide a comprehensive assessment of how data points influence both model parameters and fit, making the package suitable for various scenarios and levels of computational complexity.

Case Deletions and Approximations in Path Parameters

The core functionality of `SEMsensitivity` revolves around case deletions and approximations, allowing researchers to assess the robustness of their SEM models by systematically removing data points and observing the effects. Following the ideas presented in (Broderick et al., 2020), we employ Taylor series approximations to efficiently compute influence scores. Thus, we implemented various approaches to drop samples, listed in the table below. Our methods can determine the value or sign change of a specific path by dropping some of the samples.

Method	Description	Path Change Type
Naive Method with Exact Influence	Remove a fixed percentage of samples (determined by the exact influence) at a time and refit the model to observe the change in the parameter of interest.	Value Change
Naive Method with Approximate Influence	Remove a fixed percentage of samples (determined by the approximate influence, see (Broderick et al., 2020)) at a time and refit the model to observe the change in the parameter of interest.	Value Change
Specified Approximation Method	Determines when a specific path in a SEM model changes sign by iteratively removing data points.	Sign Change

Method	Description	Path Change Type
Simple Depth Method	Determines a specific path in a SEM model value changing by removing samples iteratively, in which influences are determined by naive method and outputs relevant results.	Value Change
Combined Method	Determines a specific path in a SEM model value changing by removing samples iteratively, in which influences are determined by both naive method and approximate method.	Value Change
Negamax Search Algorithm	Utilizes a Negamax search algorithm to iteratively drop data points and update the SEM.	Value Change
Use Depth Method to Try to Switch Sign of Parameter	Uses a depth method to iteratively remove data points in order to switch the sign of a specific path in a SEM.	Sign Change
Use Depth Method to Try to Switch Sign of Parameter	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 SEM.	Sign Change
Simulated Annealing Method	Uses the simulated annealing method to identify a SEM model path value changing by removing samples iteratively.	Value Change
Particle Swarm Optimization (PSO)	Uses the PSO method to identify a SEM model path value changing by removing samples iteratively.	Value Change
Brute Search with Cut Method	To identify a SEM model path value changing by removing samples iteratively.	Value Change

Case Deletions in Fit Indices

We use the `semfindr` package to identify data points that exert the most influence on model fit indices. By calculating the impact of individual cases, the package helps detect points that cause significant changes to indices like the CFI or RMSEA. Once the most influential points

are identified, they can be deleted in bulk or individually. Bulk deletion allows for the removal of a specified number of cases at once, enabling quick evaluation of whether the model fit becomes inadequate (e.g., CFI dropping below 0.9). Alternatively, cases can be deleted one by one to monitor the incremental impact on fit and determine when the model fails to meet desired thresholds.

Availability

The R package `SEMsensitivity` is publicly available on Github. It could be installed and run by using the following commands:

```
devtools::install_github("Feng-Ji-Lab/SEMsensitivity")
library(SEMsensitivity)
```

Example

We take the dataset `PoliticalDemocracy` provided in `lavaan` as an example. First, define a SEM and fit it with lavaan. Then, indicate the path of interest and the two variables of interest. Then, run the `SEMsensitivity` function and see the results. The possible method names are shown below:

- Naive Method with Exact Influence
- Naive Method with Approximate Influence
- Specified Approximation Method
- Simple Depth Method
- Combined Method
- Negamax Search Algorithm Function
- Use Depth Method to Try to Switch Sign of Parameter
- Use Depth Method to Try to Switch Sign of Parameter with Negamax

Below is a sample code.

First, install all needed packages and import dataset.

```
library(lavaan)
library(dplyr)
library(semfindr)
library(R.utils)
library(SEMsensitivity)

df <- PoliticalDemocracy
```

State the model (same as the model in `lavaan` package).

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

State the parameters and path of interest. Use the `lavaan` package to calculate the initial path parameters and extract them.

```
var_one <- 'dem65'
var_two <- 'ind60'
PAR <- c("dem65~ind60")
threshold <- 10

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

estimates <- parameterEstimates(fit)

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

max_final <- ceiling(threshold * nrow(df) / 100)
N <- nrow(df)

signFactor <- ifelse(par_value >= 0, TRUE, FALSE)
```

Arguments include:

- df: Dataset used by lavaan.
- model: A lavaan model syntax string.
- var_one, var_two: LHS and RHS variable names of the target path (e.g., "dem65", "ind60"). Required for path-sign/value methods.
- PAR: Full path label like "dem65~ind60". Required for path methods.
- threshold: Maximum percentage of cases allowed to drop (e.g., 10 means up to ceiling(0.10*N) cases).
- fit: Fitted baseline model.
- signFactor: TRUE if the path is currently $>= 0$ (we try to decrease it), FALSE if < 0 (we try to increase it).
- method: Which method to run (see next section). You may pass either a name or its index.

Call the function `SEMsensitivity` to do sensitivity analysis. The method can be specified by either method name or index. The two ways should desplay same outputs. This runs method 1 (exact influence) to see which cases flip the sign of `dem65 ~ ind60`.

```
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)

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)
```

After running this, we can see the result as shown below. It shows the path of interest, original value, changed parameter value and the dropped samples indices list.

- Method: which algorithm was used.

- Initial path value: estimate before dropping cases.
- Final path value: estimate after dropping selected cases.
- Deleted case indices: rows of df that were identified as most influential.

The results show that dropping these 10 cases will change the path value from 0.57 to 0.076.

```
Summary of SEM sensitivity analysis result:
Method Name: Naive exact method
Path: dem65~ind60
Original Value: 0.572336
Original Number of Samples: 75
Drop Points Percentage: 10
Dropped Number of Samples: 8
New Parameter Value: 0.076319
drop points list:
[1] 30 38  2 47 36 14 53 26
```

We also provide an example of fit indices changing from a good-fit model to an inadequate model.

```
fit_result <- SEMsensitivity(df, model, fit = fit, max_final = max_final, N = N, measure
summary(fit_result)
```

The result is shown below:

```
Method Name:  Use depth method to try to drop fit measure below threshold
Original Fit Value:  0.9954
Final Fit Value After Dropping Points:  0.8823
Number of Data Points Dropped:  19
Difference Between Original Fit and Threshold:  0.0954
Cumulative Drop in Fit Value:  0.0963
Dropped Points:  74, 41, 12, 69, 63, 45, 66, 36, 18, 56, 53, 2, 59, 53, 4, 12, 53, 16, 5
```

This output reports the fit index before and after deletions, the number of cases removed, and the indices of those cases. In this example, the CFI drops from 0.9954 (good fit) to 0.8823 (below the 0.90 threshold), indicating that removing these 19 observations is enough to make the model appear inadequate.

References

Broderick, T., Giordano, R., & Meager, R. (2020). An automatic finite-sample robustness metric: When can dropping a little data make a big difference? *arXiv Preprint arXiv:2011.14999*.