How to evaluate theory-based hypotheses in a SEM model using the GORICA

Rebecca M. Kuiper

17 September 2024

Contents

Example 1: Confirmatory Factor Analysis 1

Example 2: Multiple Group Regression 4

This is a tutorial for using GORICA for Structural Equation Models. The GORICA is an information criterion that can be used to evaluate theory-driven hypotheses.

Below, you will find two examples for the use of the `goric` function in the `restriktor` R package. The first example covers Confirmatory Factor Analysis, while the second example covers Multigroup Regression.
More details can be found in:
Kuiper, R. (2021). AIC-type Theory-Based Model Selection for Structural Equation Models. *Structural Equation Modeling*. https://doi.org/10.1080/10705511.2020.1836967
Here, I will use the `restriktor` package; in this article, the gorica package is used in the main text.
R scripts can be found on 'https://github.com/rebeccakuiper/Tutorials/tree/main/GORICA%20in%20SEM'.

Note: For (more) information regarding interpreting the GORIC(A) output, see 'Guidelines_output_GORIC' (https://github.com/rebeccakuiper/Tutorials).

Example 1: Confirmatory Factor Analysis

First, install and call the `lavaan` library to conduct CFA and the `restriktor` library to load the `goric` function. If needed, it is possible to view the description of the function with the `?` operator or the `help` command.

```r
# To install restriktor in R:
#if (!require("restriktor")) install.packages("restriktor")

# To install restriktor from github:
# if (!require("devtools")) install.packages("devtools")
# library(devtools)
# install_github("LeonardV/restriktor")
library(restriktor)

# print docs in the help-tab to view arguments and explanations for the function
#?goric

# To install lavaan in R:
# if (!require("lavaan")) install.packages("lavaan")
library(lavaan)
```

```r
# Additionally, load the gorica library to obtain the data set 'sesamesim'
# if (!require("gorica")) install.packages("gorica")
library(gorica)
```

The second step involves specifying the confirmatory factor model. Because the `goric` function cannot use the default labeling, make sure to give your own labels to estimates by including them in the `lavaan` model.

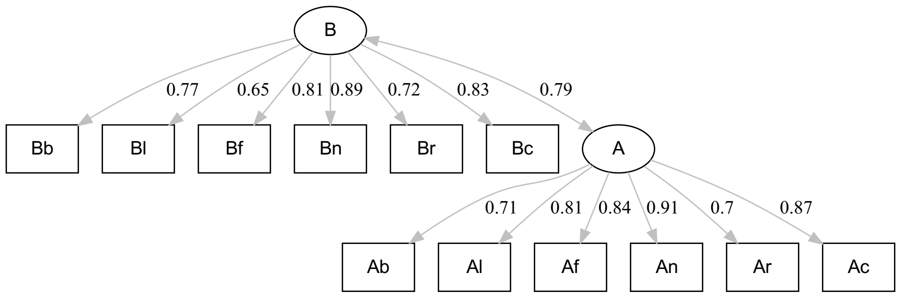
```r
model1 <- '
    A =~ A1*Ab + A2*Al + A3*Af + A4*An + A5*Ar + A6*Ac
    B =~ B1*Bb + B2*Bl + B3*Bf + B4*Bn + B5*Br + B6*Bc
'
```

Next, we can fit the confirmatory factor model using the `sem` function from `lavaan`.

```r
fit1 <- sem(model1, data = sesamesim, std.lv = TRUE)
```

We can also plot said model using `lavaanPlot`.

```r
# if (!require("lavaanPlot")) install.packages("lavaanPlot")
library(lavaanPlot)
lavaanPlot(model = fit1, node_options = list(shape = "box", fontname = "Helvetica"),
           edge_options = list(color = "grey"), coefs = T, stand = T, covs = T)
```



To call the `goric` function, we need to formulate the hypothesis of interest. The GORICA selects the best hypothesis out of a given set. Here, since we only have a single hypothesis, we will compare it to its complement.

In this case, the hypothesis of interest consists of 12 order restrictions. Note that when specifying the hypothesis we repeat the labeling using when specifying the lavaan model object. Also note that restrictions are connected by the character `;` (or `,` or `&`).

```r
H1.1 <- "
A1 > .6; A2 > .6; A3 > .6; A4 > .6; A5 > .6; A6 > .6;
B1 > .6; B2 > .6; B3 > .6; B4 > .6; B5 > .6; B6 > .6
```

```
"
```

After specifying the hypothesis of interest, we can call the `goric` function. Here, we use GORICA (i.e.,`type = "gorica"`) by default, because of entering a lavaan object. Additionally, because we need standardized estimates to obtain a meaningful comparison, we specify `standardized = TRUE`. Finally, because we are interested in one theory-based hypothesis, we compare it (by default) to its complement (i.e., by default: `comparison = "complement"`).

```
# Calculate GORICA values and weights for H1.1 and its complement.

set.seed(100)

#results1 <- goric(fit1,
#                  hypotheses = list(H1.1 = H1.1), comparison = "complement",
#                  type = "gorica", standardized = TRUE)
# or, because of the defaults:
results1 <- goric(fit1,
                  hypotheses = list(H1.1 = H1.1),
                  standardized = TRUE)

#summary(results1)
results1
```

```
restriktor (0.5-90): generalized order-restricted information criterion approximation:

Results:
        model  loglik  penalty   gorica  loglik.weights  penalty.weights  gorica.weights
1        H1.1  33.581    8.139  -50.884           0.669            0.977           0.988
2  complement  32.879   11.886  -41.985           0.331            0.023           0.012
---
The order-restricted hypothesis 'H1.1' has 85.56 times more support than its complement.
```

From the comparison of hypotheses, it seems that the hypothesis H1.1 is better supported by the data than its complement.

The default way of calculating the penalty scores in `goric` is slower when the number of parameters is large. Here, there are 12 parameters; so, we may want to use: `mix_weights = "boot"`. This means that the function will use bootstrapping in the calculation of the so-called level probabilities (LPs) needed in the penalties.

The results of course do not change, but the computation time may decrease.

```
# if (!require("parallel")) install.packages("parallel")
#library(parallel)
#nrCPUcores <- detectCores(all.tests = FALSE, logical = TRUE)
# Increasing the number of cores does not always help, so for now:
#nrCPUcores <- 1
```

We repeat the same analysis with bootstrapping:

```
set.seed(100)

results1_b <- goric(fit1,
                    hypotheses = list(H1.1 = H1.1),
                    standardized = TRUE,
                    mix_weights = "boot")

#summary(results1_b)
```

```
results1_b
```

```
restriktor (0.5-90): generalized order-restricted information criterion approximation:

Results:
        model  loglik  penalty   gorica  loglik.weights  penalty.weights  gorica.weights
1        H1.1  33.581    8.152  -50.857           0.669            0.977           0.988
2  complement  32.879   11.884  -41.990           0.331            0.023           0.012
---
The order-restricted hypothesis 'H1.1' has 84.26 times more support than its complement.
```

As you can see from the results, we arrive at the same GORICA weights as earlier.

Example 2: Multiple Group Regression

As for Example 1, ensure that the required libraries are loaded in the R workspace.

```r
# To install restriktor in R:
#if (!require("restriktor")) install.packages("restriktor")

# To install restriktor from github:
# if (!require("devtools")) install.packages("devtools")
# library(devtools)
# install_github("LeonardV/restriktor")
library(restriktor)

# To install lavaan in R:
# if (!require("lavaan")) install.packages("lavaan")
library(lavaan)

# Additionally, load the gorica library to obtain the data set 'sesamesim'
# if (!require("gorica")) install.packages("gorica")
library(gorica)
```

Before specifying the regression model ensure that the variable *sex* in the *sesamesim* data set is a factor with the right labels.

```r
sesamesim$sex <- factor(sesamesim$sex, labels = c("boy", "girl"))
```

Then specify the model being tested, which in this case is a multiple group regression model. Again, because the `goric` function cannot use the default labeling, make sure to include your own labels in the model.

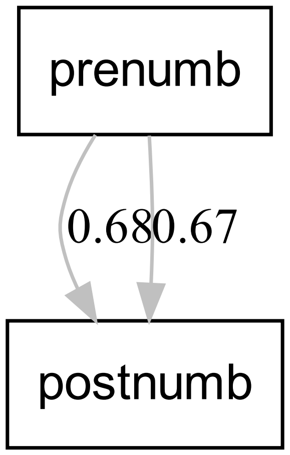
```r
model2 <- '
    postnumb ~ c(Pre_b, Pre_g)*prenumb
'
```

Fit the model using the `sem` function from `lavaan` and, if desired, use the `lavaanPlot` function to obtain a graph of the model.

```r
fit2 <- sem(model2, data = sesamesim, std.lv = TRUE, group = "sex")

#if (!require("lavaanPlot")) install.packages("lavaanPlot") # install this package first (once)
library(lavaanPlot)

lavaanPlot(model = fit2, node_options = list(shape = "box", fontname = "Helvetica"),
           edge_options = list(color = "grey"), coefs = T, stand = T, covs = T)
```



Proceed to formulate the hypothesis of interest using the same labels as in the model definition:

```r
H2.1 <- "Pre_b < Pre_g"
# versus its complement
```

Call the `goric` function to evaluate the hypothesis versus its complement using the GORICA. Note that, because we need standardized estimates for a meaningful comparison, we use `standardized = TRUE`.

```r
set.seed(100)

results2 <- goric(fit2, hypotheses = list(H2.1 = H2.1),
                  standardized = TRUE)

#summary(results2)
results2
```

```
restriktor (0.5-90): generalized order-restricted information criterion approximation:

Results:
        model  loglik  penalty  gorica  loglik.weights  penalty.weights  gorica.weights
1        H2.1   4.420    1.500  -5.841           0.498            0.500           0.498
2  complement   4.428    1.500  -5.856           0.502            0.500           0.502
---
The order-restricted hypothesis 'H2.1' has 0.99 times more support than its complement.
```

The output table shows that the hypothesis of interest and its complement are equally likely, since both have a weight of approximately .50. In other words, both are equally supported.

Since the hypotheses do not overlap and are equally complex (i.e., they have the same penalty value), this implies that their boundary is the preferred hypothesis; that is, H0: Pre_b = Pre_g.

Thus, there is support for the boundary of the hypothesis of interest and its complement, indicating that the relationship between postnumb and prenumb is equally high for girls and boys.