Computing complexity for the Bayes Factor in inequality constrained hypotheses

M.A.J. Zondervan-Zwijnenburg, A. R. Johnson, R. Van de Schoot

Over a decade of research on the topic of informative hypotheses has resulted in a bunch of methodological, tutorial, and applied papers, books (e.g., Hoijtink, 2012), and several software packages (see informative-hypotheses.sites.uu.nl). Informative hypotheses with simple order constraints, such as H_i: $\mu_1 > \mu_2 > \mu_3$, $H_i : \mu_1 > (\mu_2, \mu_3)$ have proven their use: not only do these hypotheses represent researchers' expectations better, they also have more statistical power than classical hypotheses (Vanbrabant et al., 2015). The Bayes Factor (BF) is a Bayesian measure of support used for model selection (Kass and raftery, 1995). The BF can also be used for the evaluation of informative hypotheses. Using the BF, the support in the data for an informative hypothesis (H_i) versus the unconstrained alternative (H_u) can be calculated. As was shown by Klugkist et al. (2005):

$$BF_{H_i,H_u} = \frac{f_i}{c_i},$$

where f_i denotes fit for the hypothesis H_i, and c_i denotes complexity for the hypothesis H_i. f_i is a measure of agreement between the data and H_i, it is the posterior probability of the hypothesis given the data. f_i can only be calculated after the data is observed. c_i refers to the proportion of the parameter space in agreement with the hypothesis by chance: the a priori probability of the hypothesis. Complexity can be calculated before conducting the analysis.

In case of multivariate analyses, BIEMS (Mulder et al., 2012) can be used to estimate the BF. For structural equation models, BF-SEM has been developed (Gu et al., submitted). The computation of the BF for structural equation models calculated in Mplus (Muthén and Muthén, 1998-2012) is described by Van de Schoot et al. (2012). As these authors show, fit can be calculated by means of **MplusAutomation** (Hallquist, 2013), and complexity can be calculated manually. With few parameters and few constraints, this manual computation of c_i is easy. When the number of parameters and constraints increases, however, it is preferable to automate the process. Therefore, we published the CRAN package **complexity** (Zondervan-Zwijnenburg, 2017). The code for the complexity function is provided in Appendix A. We will consider two examples in which the goal is to obtain the complexity for the hypothesis of interest to demonstrate the **complexity** package.

Example 1

The first example includes four regression coefficients: $\beta_1, \beta_2, \beta_3$, and β_4. The expectation is that β_1 is smaller than β_2 and that β_3 is smaller than β_4, which can be expressed as, $H_i : (\beta_1 < \beta_2)$ & $(\beta_3 < \beta_4)$.

Manual computation of c_i

1. Obtain all possible ways in which the parameters can be ordered; for the first example there are $4! = 4 \times 3 \times 2 \times 1 = 24$ different ways of ordering the parameters.
2. Count the number of possible orderings that are in line with each of the informative hypotheses: in this example it is six.
3. Divide the value obtained in step 2 by the value obtained in step 1. In our example: $c_i = 6/24 = 0.25$.

Automated computation of c_i

1. Use the complexity function from the **complexity** package as follows: `complexity(4,1,2,3,4)`. Here, the first 4 stands for the four parameters involved in the analysis. Every following set of two numbers, represents a constraint in which the first parameter is constrained to be lower than the second parameter. Hence, `1,2` refers to ($\beta_1 < \beta_2$) and `3,4` refers to ($\beta_3 < \beta_4$). If the hypothesis would have been ($\beta_1 < \beta_2$) & ($\beta_2 > \beta_4$), the specification would be `complexity(4,1,2,4,2)`. Parameter 3 is then unrestricted. For $\beta_1 < \beta_2 < \beta_3 < \beta_4$ the correct specification is: `complexity(4,1,2,2,3,3,4)`. `complexity(4,1,2,2,3,3,4)` generates the following output:

```
## $`true permutations`
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    1    4    2    3
## [3,]    1    3    2    4
## [4,]    3    4    1    2
## [5,]    2    3    1    4
## [6,]    2    4    1    3
##
## $`total number of permutations`
## [1] 24
##
## $`number true`
## [1] 6
##
## $`complexity (proportion)`
## [1] 0.25
```

The output shows:

1. the list of permutations in accordance with the request, where the column represents parameter and the numbers represent it's position.
2. the total number of permutations.
3. the number of permutations in agreement with the hypothesis.
4. the percentage of permutations in agreement with the hypothesis ($= c_i$).

Example 2

The second example comes from Johnson et al. (2015) and extends on the first example with two extra parameters that we hypothesize to be ordered as follows: ($\beta_1 < \beta_2$) & ($\beta_2, \beta_3 > \beta_4$) & ($\beta_4, \beta_5 < \beta_6$). Step two of the manual method will show that there are 720 ways in which these parameters can be ordered. Writing down all possibilities and counting the number of possibilities in agreement with the hypothesis is time consuming and prone to errors. Hence, we switch to the automated procedure. To determine the correct input, we need to specify all the separate constraints in terms of parameters smaller than other parameters. In this case: ($\beta_1 < \beta_2$) & ($\beta_4 < \beta_2$) & ($\beta_4 < \beta_3$) & ($\beta_4 < \beta_6$) & ($\beta_5 < \beta_6$). The number of parameters involved is six. Hence, the required input is: `complexity(6, 1,2,4,2,4,3,4,6,5,6)`. Almost immediately, the **complexity** package generates the list of permutations in accordance with the request, and the following output:

```
## $`total number of permutations'
## [1] 720
##
## $`number true'
## [1] 66
##
## $`complexity (proportion)'
## [1] 0.09166667
```

The output shows that 66 out of 720 possibilities are in agreement with our hypothesis, which results in a complexity of .09. In addition, the output provides all permutations in agreement with the hypothesis for visual inspection of the results. The **complexity** package also includes a function that launches a **Shiny** (Chang et al., 2016) application with `runShiny()`. The output is the same as the output obtained with the `complexity` function.

Although it is possible to calculate the proportion of the parameter space in line with the hypothesis by chance (i.e., c_i) manually, the second example showed that especially when the number of parameters involved becomes larger or when the constraints are more complex, it is easier and more reliable to use the complexity function in R to obtain the complexity for the Bayes Factor.

To compute the Bayes Factor for Example 2, the remaining element to compute is f_i. To obtain f_i we need to calculate the proportion of iterations in the Gibbs sampler that is in agreement with $(\beta_1 < \beta_2)$ & $(\beta_2, \beta_3 > \beta_4)$ & $(\beta_4, \beta_5 < \beta_6)$. Using **MplusAutomation** we find that for [**?**] f_i is .353.

$$BF_{H_i,H_u} = \frac{.353}{.092} = 3.837.$$

Thus, taking the model complexity into account, H_i: $(\beta_1 < \beta_2)$ & $(\beta_2, \beta_3 > \beta_4)$ & $(\beta_4, \beta_5 < \beta_6)$ fits better than H_u: μ_1, μ_2, μ_3: the informative hypothesis receives more than three times the support.

Summary

The complexity of an informative hypothesis is required to calculate BF_{H_i,H_u}. The **complexity** package enables the user to calculate c_i quickly, even for more complicated hypotheses.

Funding

MZ is supported by the Consortium Individual Development (CID), which is funded through the Gravitation program of the Dutch Ministry of Education, Culture, and Science and the Netherlands Organization for Scientific Research (NWO grant number 024.001.003). RS is supported by a VIDI grant from the Netherlands Organization for Scientific Research (NWO grant number 452.14.006).

About the authors

Mariëlle Zondervan-Zwijnenburg
Department of Methodology & Statistics
Utrecht University
The Netherlands

Alan R. Johnson
Nord University Business School
Bodø, Norway
and
RATIO Research Institute
Stockholm, Sweden

Rens van de Schoot
Department of Methodology & Statistics

Utrecht University
The Netherlands

References

- W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. shiny: Web Application Framework for R, 2016. URL https://CRAN.R-project.org/package=shiny. R package version 0.13.2.
- X. Gu, H. Hoijtink, J. Mulder, and Y. Rosseel. BF-SEM: A Fortran program for Bayesian testing of order constrained hypotheses in structural equation models. submitted.
- M. Hallquist and J. Wiley. Mplusautomation: Automating Mplus Model Estimation and Interpretation, 2016. URL https://CRAN.R-project.org/package=MplusAutomation.
- H. Hoijtink. Informative hypotheses: Theory and practice for behavioral and social scientists. CRC Press, 2012.
- A. R. Johnson, R. van de Schoot, F. Delmar, and W. D. Crano. Social influence interpretation of interpersonal processes and team performance over time using Bayesian model selection. Journal of Management, 41(2):574–606, 2015. doi: 10.1177/0149206314539351.
- R. E. Kass and A. E. Raftery. Bayes factors. Journal of the american statistical association, 90(430):773–795, 1995. URL http://amstat.tandfonline.com/doi/full/10.1080/01621459.1995.10476572.
- I. Klugkist, O. Laudy, and H. Hoijtink. Inequality constrained analysis of variance: a bayesian approach. Psychological Methods, 10(4):477, 2005. URL http://psycnet.apa.org/journals/met/10/4/477/.
- J. Mulder, H. Hoijtink, and C. de Leeuw. BIEMS: A fortran 90 program for calculating bayes factors for inequality and equality constrained models. Journal of Statistical Software, 46(2):1–39, 2012.
- L. K. Muthén and B. O. Muthén. Mplus user's guide. Muthén & Muthén, Los Angeles, CA, 7 edition, 1998-2012.
- R. Van de Schoot, H. Hoijtink, M. N. Hallquist, and P. A. Boelen. Bayesian evaluation of inequality-constrained hypotheses in SEM models using Mplus. Structural Equation Modeling: A Multidisciplinary Journal, 19(4):593–609, 2012. doi: 10.1080/10705511.2012.713267.
- L. Vanbrabant, R. Van de Schoot, and Y. Rosseel. Constrained statistical inference: sample-size tables for ANOVA and regression. Frontiers in Psychology, 5:1565, 2015. doi: 10.3389/fpsyg.2014.01565.
- M. Zondervan-Zwijnenburg. complexity: Calculate the Proportion of Permutations in Line with an Informative Hypothesis, 2017. URL https://CRAN.R-project.org/package=complexity. R package version 1.1.1.

Appendix A

```
complexity <- function(npar,...){

  require(combinat)
  values <- c(1:npar)                         #the parameters to permute
  perm <- permn(values)                       #all permutations in a list
  length <- length(perm)                      #number of permutations
  perm.matrix <- matrix(unlist(perm),length, byrow=TRUE)
                                              #permutations in rows of matrix
  dim <- dim(perm.matrix)                     #length and width of permutation matrix
  z <- matrix(unlist(list(...)),ncol=2,byrow=TRUE)
                                              #one set of restriction values in each row
  dimz <- dim(z)
  n <- dimz[1]                              #number of restrictions
  logical <- logical.last <- rep(0, nrow=dimz[1], ncol=1)
                                            #empty vectors with length perm.matrix

  for (i in 1:n){
    logical <- logical.last                 #fill logical with logical.last
    z1<- z[i,1]                             #get first restriction value
    z2<- z[i,2]                             #get second restriction value
    logical <-                              #restrictions, TRUEs and FALSES saved in logical
      perm.matrix[,z1]<perm.matrix[,z2]     #see if 1st restr. value column < 2nd restr.
    perm.matrix <- matrix(perm.matrix[logical],ncol=dim[2])
                                            #save TRUE selection in perm.matrix
    dimp <- dim(perm.matrix)                #dimensions of new matrix
    logical.last <- rep(0,nrow=dimp[1],ncol=1)
                                            #new empty vector with length new matrix
  }

  y <- sum(logical)                         #number of TRUEs in set
  true <- perm.matrix                       #matrix with TRUE permutations
  prop <- y/length                          #TRUE n / total n = complexity

  list("true permutations" = true,
       "total number of permutations"=length,
       "number true"=y, "complexity (proportion)"=prop)}
```