Computing complexity for the Bayes Factor in inequality constrained hypotheses

Mariëlle Zondervan-Zwijnenburg[1]*, Rens van de Schoot[1,2], Alan R. Johnson[3]

[1]*Department of Methods and Statistics, Utrecht Univeristy, The Netherlands*

[2]*Optentia Research Program, North-West University, South Africa*

[2]*EM LYON Business School, France*

The Bayes Factor (BF) is a measure of support used for model selection in the Bayesian context (Kass & Raftery, 1995). The BF can also be used for the evaluation of so called informative, or inequality constrained, hypotheses (Hoijtink, 2012). For more information and publications on informative hypotheses, visit: `www.informativehypotheses.wordpress.com`. Using the BF, the support in the data for an informative hypothesis (H_i) versus the unconstrained alternative (H_u) can be calculated using the fit of the hypothesis (f_i) and its complexity (c_i). As was shown by Klugkist, Laudy, and Hoijtink (2005):

$$BF_{\text{Hi1 vs. Hu}} = \frac{f_i}{c_i} \tag{1}$$

Fit here, is a measure of agreement between data and hypothesis: the posterior probability of the hypothesis given the data, which can only be calculated after the analysis. Complexity 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, Hoijtink, & de Leeuw, 2012) can be used to estimate the BF. The computation of the BF for Structural Equation Models is described by Van de Schoot, Hoijtink, Hallquist, and Boelen (2012). As these authors show, fit can be calculated by means of MplusAutomation (Hallquist, 2013), and complexity can be calculated manually (see also Van de Schoot et al., 2012). With few parameters and few constraints, this procedure is easy. When the number of parameters and constraints increases, however, it is preferable to automate the process. Therefore, an annotated script to perform the computations in R is introduced in this paper. We will consider two examples in which the goal is to obtain the complexity for the hypothesis of interest. 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: $(\beta_1 < \beta_2)$ & $(\beta_3 < \beta_4)$.

*Correspondence to: Mariëlle Zondervan-Zwijnenburg, Department of Methods and Statistics, Utrecht University, PO Box 80140, 3508TC, Utrecht, The Netherlands. E-mail: `m.a.j.z@zondervan.org`

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. Copy and run the (annotated) syntax for the complete function as given below. Running the syntax saves the calculations that R needs under the name complexity.

```
install.packages("combinat")

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)}
```

2. Next, enter the name of the function (i.e. complexity), the number of parameters, and the constraints as follows: `complexity(4,1,2,3,4)`. Here, the first `4` stands for the four parameters involved in the analylsis. 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)`.

3. After running `complexity(4,1,2,3,4)`, the following output is obtained:

```
$'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 its 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 ($= c_i$).

The second example comes from Johnson (2013) 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 after entering this code, the following output is obtained:

```
$`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.

In conclusion; although it is possible to calculate the BF 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 safer to use the complexity function in R to obtain the complexity for the Bayes Factor.

References

Hallquist, M. (2013, August). *Mplusautomation: Automating Mplus model estimation and interpretation Package MplusAutomation.* Retrieved from `http://cran.r-project.org/web/packages/MplusAutomation/MplusAutomation.pdf`

Hoijtink, H. (2012). *Informative hypotheses: Theory and practice for behavioral and social scientists.* CRC Press.

Johnson, A. R. (2013). *A social influence interpretation of members' interpersonal processes over time and team level outcomes using informative hypotheses and bayes factors.* Unpublished doctoral dissertation, EM LYON Business School, Écully, France.

Kass, R. E., & Raftery, A. E. (1995). Bayes factors. *Journal of the american statistical association*, *90*(430), 773–795. Retrieved from `http://amstat.tandfonline.com/doi/full/10.1080/01621459.1995.10476572`

Klugkist, I., Laudy, O., & Hoijtink, H. (2005). Inequality constrained analysis of variance: a bayesian approach. *Psychological Methods*, *10*(4), 477. Retrieved from `http://psycnet.apa.org/journals/met/10/4/477/`

Mulder, J., Hoijtink, H., & de Leeuw, C. (2012). BIEMS: A fortran 90 program for calculating bayes factors for inequality and equality constrained models. *Journal of Statistical Software*, *46*(2), 1–39.

Van de Schoot, R., Hoijtink, H., Hallquist, M. N., & Boelen, P. A. (2012). Bayesian evaluation of inequality-constrained hypotheses in sem models using m plus. *Structural Equation Modeling: A Multidisciplinary Journal*, *19*(4), 593–609. Retrieved from `http://www.tandfonline.com/doi/abs/10.1080/10705511.2012.713267`