Supplementary Material

Part 2

The aim of this section is to demonstrate how to easily apply our proposed sequential method for model selection based on confidence intervals (CIs) for LOO or WAIC within the R environment. R and the necessary R packages should be installed, and we recommend using RStudio for a smoother experience. The input dataset is data.dat. The complete R code can be found in single dataset code.R.

This material includes the following steps:

* Prepare the environment and read the input data
* Fit each model to the dataset
* Calculate LOO and WAIC for each model and rank the models based on them
* Calculate the confidence intervals for ΔLOO and ΔWAIC and check whether they include zero
* Compare all the competing models with using a sequential procedure

Step 1: Prepare the environment and read the input data

To prepare the environment for the subsequent analyses, five packages need to be loaded. The R packages tidyverse, MASS, and dplyr are commonly used for data manipulation and applied statistics These packages are dependencies for brms. The brms package is used for fitting Bayesian regression models and calculating LOO and WAIC for MELSMs. The cmdstanr package provides a lightweight interface to Stan, enabling Bayesian modeling in R. The set\_cmdstan\_path() function is used to specify the location of your cmdstanr installation.

rm(list=ls())

library(tidyverse)

library(MASS)

library(dplyr)

library(brms)

library(cmdstanr)

options(scipen = 10)

set\_cmdstan\_path("C:/Users/YueLiu/Documents/.cmdstan/cmdstan-2.32.2")

We use the function read.table() to read the input data data.dat. The dataset contains four columns: *I* (level–2 indicator), *J* (level–1 indicator), *Y* (outcome variable), and *X* (outcome variable).



Step 2: Fit each model to the dataset

M1 to M4 are fitted to the dataset, and the estimated parameters are saved in separate output files: output1.1.0, output2.1.0, output1.2.0, and output2.2.0respectively.

#Fit M1 to the dataset

get\_prior(bf(Y~1+X+(1|c|I),sigma~1),data)

prior1.1.0 <- prior(normal(0,1000000),class=b)

make\_stancode(bf(Y~1+X+(1|c|I),sigma~1),data,prior = prior1.1.0)

model1.1.0 <- brm(bf(Y~1+X+(1|c|I),sigma~1),data,

inits=0,cores=4,sample\_prior=T,iter=4000,

prior = prior1.1.0,seed = 1234,

backend="cmdstanr")

#Fit M2 to the dataset

get\_prior(bf(Y~1+X+(1+X|c|I),sigma~1),data)

prior2.1.0 <- prior(normal(0,1000000),class=b)

make\_stancode(bf(Y~1+X+(1+X|c|I),sigma~1),data,prior = prior2.1.0)

model2.1.0 <- brm(bf(Y~1+X+(1+X|c|I),sigma~1),data,

inits=0,cores=4,sample\_prior=T,iter=4000,

prior = prior2.1.0,seed = 1234,

backend="cmdstanr")

#Fit M3 to the dataset

get\_prior(bf(Y~1+X+(1|c|I),sigma~1+(1|c|I)),data)

prior1.2.0 <- prior(normal(0,1000000),class=b)

make\_stancode(bf(Y~1+X+(1|c|I),sigma~1+(1|c|I)),data,prior = prior1.2.0)

model1.2.0 <- brm(bf(Y~1+X+(1|c|I),sigma~1+(1|c|I)),data,

inits=0,cores=4,sample\_prior=T,iter=4000,

prior = prior1.2.0,seed = 1234,

backend="cmdstanr")

#Fit M4 to the dataset

get\_prior(bf(Y~1+X+(1+X|c|I),sigma~1+(1|c|I)),data)

prior2.2.0 <- prior(normal(0,1000000),class=b)

make\_stancode(bf(Y~1+X+(1+X|c|I),sigma~1+(1|c|I)),data,prior = prior2.2.0)

model2.2.0 <- brm(bf(Y~1+X+(1+X|c|I),sigma~1+(1|c|I)),data,

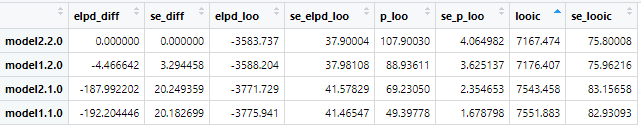
inits=0,cores=4,sample\_prior=T,iter=4000,

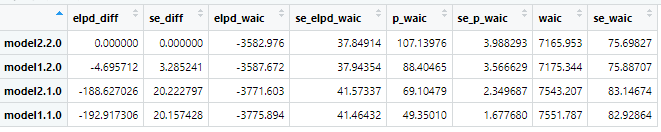
prior = prior2.2.0,seed = 1234,

backend="cmdstanr")

Step 3: Calculate LOO and WAIC for each model and rank the models based on them

The fitted models are evaluated using loo() andwaic() functions. The resulting LOO and WAIC values are saved as loo1.1.0, loo2.1.0, loo1.2.0, and loo2.2.0, and waic1.1.0, waic2.1.0, waic1.2.0, and waic2.2.0 for M1 to M4, respectively. Then, fitted models are ranked based on ELPD values using loo\_compare()andwaic\_compare(). The function outputs a matrix where rows represent models, and columns include the ELPD differences (elpd\_diff) and their SEs (se\_diff) relative to the one with the largest ELPD (, the model in the first row). The models are sorted in descending order of fit.





A matrix output is created to store the model selection results. Before that, each model is labeled with a number in ascending order of complexity (e.g., 110 for the simplest model, 220 for the most complex). The columns waic1 and loo1 in the matrix output represent the selected model based on the point methods (i.e., ). Other columns contain details such as the model label, the difference in fit index between and other models, and its SE. For example, loo2, loo2\_diff, and loo2\_SEdiff represent the second-best model based on LOO (i.e., ), the ΔLOO between and , and its SE, respectively.



Step 4: Calculate the confidence intervals for ΔLOO and ΔWAIC and check whether they include zero

The 90% CI, 70% CI, and 50% CI for ΔLOO and ΔWAIC are calculated. Then, we check whether these CIs include zero.

For example, we use the following syntax to calculate the lower level (CI$loo2\_90L) and upper level (CI$loo2\_90U) bounds of the 90 %CI for ΔLOO between and .

CI$loo2\_90L <- output$loo2\_diff-1.64\*output$loo2\_SEdiff

CI$loo2\_90U <- output$loo2\_diff+1.64\*output$loo2\_SEdiff

Next, we check whether the 90% CI includes zero:

CI$loo2\_90 <- CI$loo2\_90L<=0 & CI$loo2\_90U>=0

If the CI includes zero, the result will be “TRUE”, otherwise, it will be “FALSE”. A part of the results is shown below.



Step 5: Compare all the competing models with using a sequential procedure

The *compare* vector is created to store the labels of models that are not significantly worse than .

# output$loo1 saved model label of in the sequential procedure.

compare <- c(output$loo1)

{if(is.na(CI$ loo2\_90)){compare <- c(compare,NA)}

# output$loo2 saved the label of in the sequential procedure. If the 90% CI of ΔLOO between and includes zero, then add the model label in the vector *compare.*

else if (CI$ loo2\_90==T) {compare <- c(compare,output$loo2)}}

{if(is.na(CI$loo3\_90)){compare <- c(compare,NA)}

Finally, the function min(compare) selects the simplest model among the models that are not significantly worse than. The parameter estimates for the selected model can be extracted from the corresponding output files (output1.1.0, output2.1.0, output1.2.0, and output2.2.0) for further analysis.

