empirical.R

mingya

2023-12-22

### :::::: Real Data Reduced Sample Size :::::: ###  
### ====== HLM HS ====== ###  
library(loo)

## This is loo version 2.6.0

## - Online documentation and vignettes at mc-stan.org/loo

## - As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'cores' argument or set options(mc.cores = NUM\_CORES) for an entire session.

library(rstanarm)

## Loading required package: Rcpp

## This is rstanarm version 2.21.4

## - See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!

## - Default priors may change, so it's safest to specify priors, even if equivalent to the defaults.

## - For execution on a local, multicore CPU with excess RAM we recommend calling

## options(mc.cores = parallel::detectCores())

library(rstan)

## Loading required package: StanHeaders

## Loading required package: ggplot2

## rstan (Version 2.21.8, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calling  
## options(mc.cores = parallel::detectCores()).  
## To avoid recompilation of unchanged Stan programs, we recommend calling  
## rstan\_options(auto\_write = TRUE)

library(LaplacesDemon)

##   
## Attaching package: 'LaplacesDemon'

## The following objects are masked from 'package:rstanarm':  
##   
## invlogit, logit

library(kableExtra)  
library(bayesplot)

## This is bayesplot version 1.10.0

## - Online documentation and vignettes at mc-stan.org/bayesplot

## - bayesplot theme set to bayesplot::theme\_default()

## \* Does \_not\_ affect other ggplot2 plots

## \* See ?bayesplot\_theme\_set for details on theme setting

options(mc.cores = parallel::detectCores())  
  
set.seed(53706)  
# data  
df0 <- read.csv("pisa2018.BayesBook.csv")  
  
df <- df0 %>%  
 dplyr::select(SchoolID, CNTSTUID, Female, ESCS, METASUM, PERFEED, HOMEPOS,   
 ADAPTIVITY, TEACHINT, ICTRES, ATTLNACT, COMPETE, JOYREAD,  
 WORKMAST, GFOFAIL, SWBP, MASTGOAL, BELONG, SCREADCOMP,   
 PISADIFF, Public, PV1READ, SCREADDIFF)  
  
  
# subset  
sch <- table(df$SchoolID)  
dt0 <- subset(df, SchoolID %in% names(sch[sch > 10]))  
  
  
# check  
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.2 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ lubridate 1.9.2 ✔ tibble 3.2.1  
## ✔ purrr 1.0.1 ✔ tidyr 1.3.0

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ lubridate::dst() masks LaplacesDemon::dst()  
## ✖ tidyr::extract() masks rstan::extract()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::group\_rows() masks kableExtra::group\_rows()  
## ✖ lubridate::interval() masks LaplacesDemon::interval()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ purrr::partial() masks LaplacesDemon::partial()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

dt0 %>%  
 group\_by(SchoolID)%>%  
 summarise(n=n()) # 84 in total

## # A tibble: 157 × 2  
## SchoolID n  
## <int> <int>  
## 1 84000001 28  
## 2 84000002 36  
## 3 84000004 35  
## 4 84000005 28  
## 5 84000008 27  
## 6 84000009 30  
## 7 84000010 30  
## 8 84000011 35  
## 9 84000012 22  
## 10 84000013 24  
## # ℹ 147 more rows

#===============================#  
### ::: For reduced sample::: ###  
#===============================#  
# randomly select 15 students in each group  
dt <- dt0 %>% group\_by(SchoolID) %>% slice\_sample(n = 10)  
  
df <- dt[401:600, ]  
  
  
# model fitting  
bsm <- list()  
loo\_bs <- list()  
  
bsm[[1]] <- stan\_lmer(  
 PV1READ ~ Female + ESCS + HOMEPOS + ICTRES + (1 + ICTRES|SchoolID), data = df,   
 prior\_intercept = student\_t(3, 470, 100),  
 iter = 5000, chains = 4,  
 adapt\_delta=.999,thin=10)  
  
bsm[[2]] <- stan\_lmer(  
 PV1READ ~ JOYREAD + PISADIFF + SCREADCOMP + SCREADDIFF + (1|SchoolID),  
 data = df, prior\_intercept = student\_t(3, 470, 100),iter = 5000, chains = 4,  
 adapt\_delta=.999,thin=10)  
  
bsm[[3]] <- stan\_lmer(  
 PV1READ ~ METASUM + GFOFAIL + MASTGOAL + SWBP + WORKMAST + ADAPTIVITY + COMPETE + (1|SchoolID),  
 data = df, prior\_intercept = student\_t(3, 470, 100),iter = 5000, chains = 4,  
 adapt\_delta=.999,thin=10)  
  
bsm[[4]] <- stan\_lmer(  
 PV1READ ~ PERFEED + TEACHINT + BELONG + (1 + TEACHINT|SchoolID),  
 data = df, prior\_intercept = student\_t(3, 470, 100),iter = 5000, chains = 4,  
 adapt\_delta=.999,thin=10)  
  
  
# loo and weights  
loo\_bs[[1]] <- loo(log\_lik(bsm[[1]]))

## Warning: Relative effective sample sizes ('r\_eff' argument) not specified.  
## For models fit with MCMC, the reported PSIS effective sample sizes and   
## MCSE estimates will be over-optimistic.

## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for details.

loo\_bs[[2]] <- loo(log\_lik(bsm[[2]]))

## Warning: Relative effective sample sizes ('r\_eff' argument) not specified.  
## For models fit with MCMC, the reported PSIS effective sample sizes and   
## MCSE estimates will be over-optimistic.  
  
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for details.

loo\_bs[[3]] <- loo(log\_lik(bsm[[3]]))

## Warning: Relative effective sample sizes ('r\_eff' argument) not specified.  
## For models fit with MCMC, the reported PSIS effective sample sizes and   
## MCSE estimates will be over-optimistic.

## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.

loo\_bs[[4]] <- loo(log\_lik(bsm[[4]]))

## Warning: Relative effective sample sizes ('r\_eff' argument) not specified.  
## For models fit with MCMC, the reported PSIS effective sample sizes and   
## MCSE estimates will be over-optimistic.  
  
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.

w\_bs <- loo\_model\_weights(loo\_bs, method = "stacking")  
w\_pbma <- loo\_model\_weights(loo\_bs, method = "pseudobma", BB=FALSE)  
w\_pbmabb <- loo\_model\_weights(loo\_bs, method = "pseudobma")  
  
# Obtain the LPD  
lpd\_point <- as.matrix(cbind(loo\_bs[[1]]$pointwise[, "elpd\_loo"],  
 loo\_bs[[2]]$pointwise[, "elpd\_loo"],  
 loo\_bs[[3]]$pointwise[, "elpd\_loo"],  
 loo\_bs[[4]]$pointwise[, "elpd\_loo"]))  
  
  
# kld  
# bs  
n\_draws <- nrow(as.matrix(bsm[[1]]));print(n\_draws)

## [1] 1000

ypred\_bs <- matrix(NA, nrow = n\_draws, ncol = nobs(bsm[[1]]))  
for (d in 1:n\_draws) {  
 k <- sample(1:length(w\_bs), size = 1, prob = w\_bs)  
 ypred\_bs[d, ] <- posterior\_predict(bsm[[k]], draws = 1)  
}  
  
  
y\_bs <- colMeans(ypred\_bs)  
d1 <- density(y\_bs, kernel = c("gaussian"))$y  
d0 <- density(df$PV1READ, kernel = c("gaussian"))$y  
kld1 <- KLD(d1, d0)$sum.KLD.py.px  
  
# pbma  
  
ypred\_bma <- matrix(NA, nrow = n\_draws, ncol = nobs(bsm[[1]]))  
for (d in 1:n\_draws) {  
 k <- sample(1:length(w\_pbma), size = 1, prob = w\_pbma)  
 ypred\_bma[d, ] <- posterior\_predict(bsm[[k]], draws = 1)  
}  
  
  
y\_bma <- colMeans(ypred\_bma)  
d2 <- density(y\_bma, kernel = c("gaussian"))$y  
kld2 <- KLD(d2, d0)$sum.KLD.py.px  
  
# pbmabb  
ypred\_bmabb <- matrix(NA, nrow = n\_draws, ncol = nobs(bsm[[1]]))  
for (d in 1:n\_draws) {  
 k <- sample(1:length(w\_pbmabb), size = 1, prob = w\_pbmabb)  
 ypred\_bmabb[d, ] <- posterior\_predict(bsm[[k]], draws = 1)  
}  
  
  
y\_bmabb <- colMeans(ypred\_bmabb)  
d3 <- density(y\_bmabb, kernel = c("gaussian"))$y  
kld3 <- KLD(d3, d0)$sum.KLD.py.px  
  
  
### ::: For BHS ::: ###  
# Build the model  
d\_discrete = 1  
X = df[, c("ESCS","HOMEPOS","ICTRES",  
 "JOYREAD","PISADIFF","SCREADCOMP","SCREADDIFF",  
 "METASUM","GFOFAIL","MASTGOAL","SWBP","WORKMAST","ADAPTIVITY","COMPETE",  
 "PERFEED","TEACHINT","BELONG")]   
  
stan\_bhs <- list(X = X, N = nrow(X), d = ncol(X), d\_discrete = d\_discrete,  
 lpd\_point = lpd\_point, K = ncol(lpd\_point), tau\_mu = 1,  
 tau\_sigma = 1, tau\_discrete = .5, tau\_con = 1)  
  
fit\_bhs<- stan("bhs\_stan.stan", data = stan\_bhs, chains = 4, iter = 5000)  
  
# weights  
wts\_bhs <- rstan::extract(fit\_bhs, pars = 'w')$w  
w\_bhs\_r <- apply(wts\_bhs, c(2,3), mean)  
w\_bhs\_m <- as.matrix(apply(wts\_bhs, 3, mean))  
  
# Obtain the KLD  
ypred\_bhs\_r <- matrix(NA, nrow = n\_draws, ncol = nobs(bsm[[1]]))  
for (d in 1:n\_draws) {  
 k <- sample(1:4, size = 1, prob = w\_bhs\_m)  
 ypred\_bhs\_r[d, ] <- posterior\_predict(bsm[[k]], draws = 1)  
}  
  
y\_bhs\_r <- colMeans(ypred\_bhs\_r)  
  
# lpd\_bhs <- lpd\_point\*w\_bhs\_r  
  
# KLD  
d4 <- density(y\_bhs\_r, kernel = c("gaussian"))$y  
kld4 <- KLD(d4, d0)$sum.KLD.py.px  
  
  
# summarize the weights and lpd  
wr <- data.frame(as.matrix(w\_bs), as.matrix(w\_pbma), as.matrix(w\_pbmabb), w\_bhs\_m)  
colnames(wr) <- c("bs","pbma", "pbmabb", "bhs")  
  
klds <- rbind(kld1, kld2, kld3, kld4)  
  
  
  
  
#===============================#  
 ### ::: For full sample ::: ###  
#===============================#  
  
  
df <- df0 %>%  
 dplyr::select(SchoolID, CNTSTUID, Female, ESCS, METASUM, PERFEED, HOMEPOS,   
 ADAPTIVITY, TEACHINT, ICTRES, ATTLNACT, COMPETE, JOYREAD,  
 WORKMAST, GFOFAIL, SWBP, MASTGOAL, BELONG, SCREADCOMP,   
 PISADIFF, Public, PV1READ, SCREADDIFF)  
  
# model fitting  
bsm <- list()  
loo\_bs <- list()  
  
bsm[[1]] <- stan\_lmer(  
 PV1READ ~ Female + ESCS + HOMEPOS + ICTRES + (1 + ICTRES|SchoolID), data = dt,   
 prior\_intercept = student\_t(3, 470, 100),  
 iter = 5000, chains = 4,  
 adapt\_delta=.999,thin=10)  
  
bsm[[2]] <- stan\_lmer(  
 PV1READ ~ JOYREAD + PISADIFF + SCREADCOMP + SCREADDIFF + (1|SchoolID),  
 data = dt, prior\_intercept = student\_t(3, 470, 100),iter = 5000, chains = 4,  
 adapt\_delta=.999,thin=10)  
  
bsm[[3]] <- stan\_lmer(  
 PV1READ ~ METASUM + GFOFAIL + MASTGOAL + SWBP + WORKMAST + ADAPTIVITY + COMPETE + (1|SchoolID),  
 data = dt, prior\_intercept = student\_t(3, 470, 100),iter = 5000, chains = 4,  
 adapt\_delta=.999,thin=10)  
  
bsm[[4]] <- stan\_lmer(  
 PV1READ ~ PERFEED + TEACHINT + BELONG + (1 + TEACHINT|SchoolID),  
 data = dt, prior\_intercept = student\_t(3, 470, 100),iter = 5000, chains = 4,  
 adapt\_delta=.999,thin=10)  
  
  
# loo and weights  
  
loo\_bs[[1]] <- loo(log\_lik(bsm[[1]]))

## Warning: Relative effective sample sizes ('r\_eff' argument) not specified.  
## For models fit with MCMC, the reported PSIS effective sample sizes and   
## MCSE estimates will be over-optimistic.

## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for details.

loo\_bs[[2]] <- loo(log\_lik(bsm[[2]]))

## Warning: Relative effective sample sizes ('r\_eff' argument) not specified.  
## For models fit with MCMC, the reported PSIS effective sample sizes and   
## MCSE estimates will be over-optimistic.  
  
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for details.

loo\_bs[[3]] <- loo(log\_lik(bsm[[3]]))

## Warning: Relative effective sample sizes ('r\_eff' argument) not specified.  
## For models fit with MCMC, the reported PSIS effective sample sizes and   
## MCSE estimates will be over-optimistic.  
  
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for details.

loo\_bs[[4]] <- loo(log\_lik(bsm[[4]]))

## Warning: Relative effective sample sizes ('r\_eff' argument) not specified.  
## For models fit with MCMC, the reported PSIS effective sample sizes and   
## MCSE estimates will be over-optimistic.  
  
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for details.

w\_bs <- loo\_model\_weights(loo\_bs, method = "stacking")  
w\_pbma <- loo\_model\_weights(loo\_bs, method = "pseudobma", BB=FALSE)  
w\_pbmabb <- loo\_model\_weights(loo\_bs, method = "pseudobma")  
  
  
# Obtain the LPD  
lpd\_point <- as.matrix(cbind(loo\_bs[[1]]$pointwise[, "elpd\_loo"],  
 loo\_bs[[2]]$pointwise[, "elpd\_loo"],  
 loo\_bs[[3]]$pointwise[, "elpd\_loo"],  
 loo\_bs[[4]]$pointwise[, "elpd\_loo"]))  
  
  
# kld  
# bs  
n\_draws <- nrow(as.matrix(bsm[[1]]));print(n\_draws)

## [1] 1000

ypred\_bs <- matrix(NA, nrow = n\_draws, ncol = nobs(bsm[[1]]))  
for (d in 1:n\_draws) {  
 k <- sample(1:length(w\_bs), size = 1, prob = w\_bs)  
 ypred\_bs[d, ] <- posterior\_predict(bsm[[k]], draws = 1)  
}  
  
  
y\_bs <- colMeans(ypred\_bs)  
d1 <- density(y\_bs, kernel = c("gaussian"))$y  
d0 <- density(df$PV1READ, kernel = c("gaussian"))$y  
kld1 <- KLD(d1, d0)$sum.KLD.py.px  
  
# pbma  
  
ypred\_bma <- matrix(NA, nrow = n\_draws, ncol = nobs(bsm[[1]]))  
for (d in 1:n\_draws) {  
 k <- sample(1:length(w\_pbma), size = 1, prob = w\_pbma)  
 ypred\_bma[d, ] <- posterior\_predict(bsm[[k]], draws = 1)  
}  
  
  
y\_bma <- colMeans(ypred\_bma)  
d2 <- density(y\_bma, kernel = c("gaussian"))$y  
kld2 <- KLD(d2, d0)$sum.KLD.py.px  
  
# pbmabb  
ypred\_bmabb <- matrix(NA, nrow = n\_draws, ncol = nobs(bsm[[1]]))  
for (d in 1:n\_draws) {  
 k <- sample(1:length(w\_pbmabb), size = 1, prob = w\_pbmabb)  
 ypred\_bmabb[d, ] <- posterior\_predict(bsm[[k]], draws = 1)  
}  
  
  
y\_bmabb <- colMeans(ypred\_bmabb)  
d3 <- density(y\_bmabb, kernel = c("gaussian"))$y  
kld3 <- KLD(d3, d0)$sum.KLD.py.px  
  
  
### ::: For BHS ::: ###  
# Build the model  
d\_discrete = 1  
X = dt[, c("ESCS","HOMEPOS","ICTRES",  
 "JOYREAD","PISADIFF","SCREADCOMP","SCREADDIFF",  
 "METASUM","GFOFAIL","MASTGOAL","SWBP","WORKMAST","ADAPTIVITY","COMPETE",  
 "PERFEED","TEACHINT","BELONG")]   
  
stan\_bhs <- list(X = X, N = nrow(X), d = ncol(X), d\_discrete = d\_discrete,  
 lpd\_point = lpd\_point, K = ncol(lpd\_point), tau\_mu = 1,  
 tau\_sigma = 1, tau\_discrete = .5, tau\_con = 1)  
  
fit\_bhs<- stan("bhs\_stan.stan", data = stan\_bhs, chains = 4, iter = 5000)  
  
# weights  
wts\_bhs <- rstan::extract(fit\_bhs, pars = 'w')$w  
w\_bhs\_r <- apply(wts\_bhs, c(2,3), mean)  
w\_bhs\_m <- as.matrix(apply(wts\_bhs, 3, mean))  
  
# Obtain the KLD  
ypred\_bhs\_r <- matrix(NA, nrow = n\_draws, ncol = nobs(bsm[[1]]))  
for (d in 1:n\_draws) {  
 k <- sample(1:4, size = 1, prob = w\_bhs\_m)  
 ypred\_bhs\_r[d, ] <- posterior\_predict(bsm[[k]], draws = 1)  
}  
  
y\_bhs\_r <- colMeans(ypred\_bhs\_r)  
  
# lpd\_bhs <- lpd\_point\*w\_bhs\_r  
  
# KLD  
d4 <- density(y\_bhs\_r, kernel = c("gaussian"))$y  
kld4 <- KLD(d4, d0)$sum.KLD.py.px  
  
  
# summarize the weights and lpd  
wr\_full <- data.frame(as.matrix(w\_bs), as.matrix(w\_pbma), as.matrix(w\_pbmabb), w\_bhs\_m)  
colnames(wr\_full) <- c("bs","pbma", "pbmabb", "bhs")  
  
klds\_full <- rbind(kld1, kld2, kld3, kld4)  
  
wr

## bs pbma pbmabb bhs  
## model1 2.096941e-02 2.849811e-10 1.103971e-02 0.1784932  
## model2 7.656283e-01 1.000000e+00 9.489290e-01 0.4520309  
## model3 2.134008e-01 8.866585e-09 3.994603e-02 0.2312057  
## model4 1.580064e-06 9.033222e-13 8.530829e-05 0.1382702

wr\_full

## bs pbma pbmabb bhs  
## model1 7.999249e-04 2.539611e-56 3.017902e-34 0.13460339  
## model2 5.961561e-01 1.000000e+00 9.374626e-01 0.47690623  
## model3 4.030435e-01 2.424552e-17 6.253741e-02 0.30975801  
## model4 4.463208e-07 1.758939e-80 1.537618e-61 0.07873236

klds

## [,1]  
## kld1 0.04180680  
## kld2 0.03743275  
## kld3 0.03739351  
## kld4 0.06758794

klds\_full

## [,1]  
## kld1 0.09983849  
## kld2 0.02687427  
## kld3 0.03587321  
## kld4 0.09911333