Student performance prediction using bayesian logistic regression

Contents

1 Introduction . . . 2
2 Data . . . 2
3 Description of the models . . . 4
    Bayesian logistic regression . . . 4
4 Stan model implementation . . . 5
5 Convergence diagnostics . . . 8
6 Posterior predictive checks . . . 11
7 Model Comparison . . . 13
8 Prediction performance . . . 16
9 Sensitivity analysis with respect to prior choices . . . 17
10 Discussion of issues . . . 19
11 Conclusion . . . 19
12 Self reflection . . . 20

1 Introduction

Education is a key issue when we talk about a country's development. The learning process needs continuous effort from both students and teachers. Thus, student performance in a course is a matter that concerns teachers. For example, it is shown that even students with the same intelligence level and in the same school could have a clear difference in the result. It is also very clear that the performance of the students doesn't only rely on their understanding of the subject, and that it is affected by other factors. It might also be different for the same courses in different schools.

In this project, we aim to predict how an individual student performs in the course while taking into consideration variables such as class participation, free time, father's and mother's education level, and school supplementary. This will be modeled using a Bayesian logistic regression. The order of the project is as follows: in section 2 the data is presented and discussed, and in section 3, we introduced the used models in the project and we defined suitable priors for scaled variables. In section 4, Stan codes for the two models are represented, while in section 5 and 6, the convergence of the models is discussed and posterior predictive check for the prediction problem is illustrated. In section 7, we compare the difference between the two models in context of the results, convergence and the reliability of estimates. In section 8, the prediction accuracy of the two models is shown. In section 9, we assessed the prior sensitivity for the two models. In the final part of the project, suggestions for improvement and discussion are presented.

2 Data

The data used in the project contains information on Portuguese secondary school students, collected during 2005 and 2006. The original dataset was collected and analysed in a 2008 study conducted by Paulo Cortez and Alice Silva, "Using data mining to predict secondary school student performance" [1]. In the study Cortez and Silva trained various machine learning models in order to predict student performance based on features such as study habits, family conditions and more.

This project will utilize a subset of the original data, with a total of 10 covariates and a single target variable. The first variable (school) will act as a grouping variable in the hierarchical model.

variable	type
school	binary: 1 - "Gabriel Pereira" or 2 - "Mousinho da Silveira"
sex	binary: "F" - female or "M" - male
mother's education level	numeric: 1 - 5
father's education level	numeric: 1 - 5
travel time	numeric: 1 - 5
study time	numeric: 1 - 5
failures	numeric: 1 - 3 if class failures between this, 4 for 4 or greater
extra educational support	binary: "yes" or "no"
health state	numeric: 1 - 5
number of absences	numeric: 0 - 93
final math grade	numeric: 0 - 20

In this project, Bayesian logistic regression models will be used to predict whether the final math grade will be below or above 10.

```r
#import needed libraries
library(cmdstanr)
library(posterior)
library(loo)
library(tidyr)
library(dplyr)
library(ggplot2)
library(gridExtra)
library(bayesplot)
#set_cmdstan_path('/coursedata/cmdstan')
options(mc.cores=4)
#set.seed(1234)
```

Data preprocessing Reading the original dataset:

```
student <- read.csv('student-mat.csv',sep = ';')
```

Mapping the grouping variable (school) to 1's and 2's, and the binary categorical variables and the target variable to 0's and 1's:

```
student$school <- with(student, case_when(school == 'GP' ~ 1,
                                           TRUE ~ 2))
student$sex <- with(student, case_when(sex == 'F' ~ 0,
                                           TRUE ~ 1))
student$schoolsup <- with(student, case_when(schoolsup == 'yes' ~ 1,
                                           TRUE ~ 0))
student$G3 <- with(student, case_when(G3 < 10 ~ 0,
                                           TRUE ~ 1))
```

Extracting the variables used in this analysis:

```
index<-c(1,2,7,8,13,14,15,16,29,30,33)
student_new<- student[,index]
```

Shuffling the data:

```
student_new <- student_new[sample(nrow(student_new)),]
```

Scaling each variable (except for the grouping and the target variables) by substracting the variable mean μ_d and dividing by standard deviation σ_d:

```
for (i in 2:10) {
      student_new[i] <- scale(student_new[i])
}
```

Dividing the data into training and testing:

```
index<-1:20
train_data<-student_new[-index,]
test_data<-student_new[index,]
```

Data after preprocessing:

```
head(train_data)
```

```
##     school        sex       Medu       Fedu traveltime   studytime   failures
## 163      1  1.0533202 -1.5979820 -0.4792490  0.7912473 -1.23378606  0.8953431
## 240      1  1.0533202 -0.6845191 -0.4792490 -0.6424347 -0.04223229  0.8953431
## 236      1  1.0533202  0.2289439 -0.4792490  0.7912473  1.14932149 -0.4493737
## 178      1  1.0533202  0.2289439  0.4396993 -0.6424347 -0.04223229 -0.4493737
## 39       1 -0.9469754  0.2289439  1.3586476 -0.6424347  1.14932149 -0.4493737
## 199      1 -0.9469754  1.1424068  1.3586476  0.7912473 -1.23378606  0.8953431
##      schoolsup     health   absences G3
## 163 -0.3845523  1.0397512 -0.7133316  0
## 240 -0.3845523 -1.1180512 -0.7133316  0
## 236 -0.3845523 -1.1180512  0.5361849  1
## 178 -0.3845523  0.3204837 -0.2135250  0
## 39   2.5938431  1.0397512 -0.4634283  1
## 199 -0.3845523 -1.1180512  2.2855080  1
```

3 Description of the models

Bayesian logistic regression

The likelihood function for a binary outcome variable for a single observation y_i follows the binomial distribution

$$p(y_i|\pi) = \pi(x_i)^{y_i}(1-\pi(x_i))^{1-y_i}$$

where $\pi(x_i)$ is the probability of outcome y_i with some predictor vector x_i. In the logistic regression model, this is written as

$$\pi(x) = \frac{e^{\alpha+\beta_1 x_1+\ldots+\beta x_d}}{1+e^{\alpha+\beta_1 x_1+\ldots+\beta x_d}},$$

Where α (intercept) and $\beta_1, \ldots\beta_d$ are the unkown model parametes.

Over the whole dataset with N independent observations, the likelihood is

$$\prod_{i=1}^{N} p(y_i|\pi(x_i)).$$

Thus, the joint posterior distribution for all parameters β is proportional to the product of the independent priors and the N likelihood contributions $p(y_i|\pi(x_i)$.

In this project, two different bayesian logistic regression models are used and compared: 1. a pooled model, and 2. a hierarchical model.

1. Pooled model: In the pooled model, all of the data is treated as identically and independently distributed, and shared regression parameters α β are used. Each parameter gets a similar weakly informative prior

$$\alpha \sim N(\mu, \sigma)$$
$$\beta_d \sim N(\mu, \sigma)$$

with fixed mean and standard deviation $\mu = 0, \sigma = 10$.

These weakly informative normal priors are suitable for the model because (prior to seeing the data), each parameter β_d can be either positive or negative, and are unlikely to be far from zero - normal prior with zero mean accounts for this prior knowledge.

2. Hierarchical model: In the hierarchical model, the data is separated into L groups, each of which get their own vector of parameters $\beta_l in \mathbb{R}^D$, where D is the number of predictor variables in the data. Then for each group l, each individual parameter $\beta_{l,d}$ is given a prior

$$\beta_{l,d} \sim N(\mu_d, \sigma_d)$$

,

where both μ_d and σ_d are given hyper priors

$$\mu_d \sim N(0, 10)$$

$$\sigma_d \sim Inv - \chi^2(10), \sigma_d > 0,$$

which are common for all of the groups.

The intercept α_l will be drawn for each group separately, with prior distribution

$$\alpha_l \sim N(0, 10).$$

Similar to the pooled model, this selection of priors reasonably reflect the possible values of the true distribution while still allowing relatively large variation.

4 Stan model implementation

For all the Stan models, we used 1000 samples: half of these samples is for warm up and 4 chains.

```
pm <- cmdstan_model(stan_file = "project_pooled.stan")
pm$print()
```

1. Pooled model Stan implementation:

```
## data {
##   int<lower=1> D; // number of variables
##   int<lower=0> N_train; // number of datapoints in the training set
##   int<lower=0> N_test; // number of datapoints in the testing set
##   array[N_train] int<lower=0, upper=1> y; // outcome
##   array[N_train] row_vector[D] x_train; // covariate matrix for the training set
##   array[N_test] row_vector[D] x_pred; // covariate vector for the testing set
## }
## parameters {
##   vector[D] beta; // regression parameters
##   real alpha; // intercept
## }
## model {
##   //priors
##   alpha ~ normal(0,10);
##   for (d in 1:D) {
##     beta[d] ~ normal(0, 10);
##   }
##   //likelihood
##   for (n in 1:N_train) {
##     y[n] ~ bernoulli(inv_logit(alpha + x_train[n] * beta));
##   }
## }
## generated quantities {
##   //log likelihood values for the ELPD-LOO calculation
##   vector[N_train] log_lik;
##   for (n in 1:N_train)
##     //log_lik[n]=inv_logit(alpha + x_train[n] * beta);
##     log_lik[n]=bernoulli_logit_lpmf(y[n] | alpha + x_train[n] * beta);
##
##   // posterior predictions for the training data
##   array[N_train] int y_rep;
##   for (n in 1:N_train){
##     y_rep[n] = bernoulli_logit_rng(alpha + x_train[n, ] * beta);
##   }
##   // posterior predictions for unseen data
##   array[N_test] int y_pred;
##   for (n in 1:N_test){
##     y_pred[n] = bernoulli_logit_rng(alpha + x_pred[n, ] * beta);
##   }
## }
```

```
hm <- cmdstan_model(stan_file = "project_hierarchical.stan")
hm$print()
```

2. Hierarchical model Stan implementation:

```
## data {
##   int<lower=1> D; // number of variables
##   int<lower=0> N_train; // number of datapoints in the training set
##   int<lower=0> N_test; // number of datapoints in the testing set
##   int<lower=1> L; // number of groups
##   array[N_train] int<lower=0, upper=1> y; // outcome
##   array[N_train] int<lower=1, upper=L> ll; // group column for train data
##   array[N_train] row_vector[D] x_train; // covariate matrix for the training set
##   array[N_test] row_vector[D] x_pred; // covariate matrix for the testing set
##   array[N_test] int<lower=1, upper=L> ll_pred; // group column for test data
## }
## parameters {
##   // hyperparameters:
##   array[D] real mu;
##   array[D] real<lower=0> sigma;
##   // regression parameters
##   array[L] vector[D] beta;
##   // intercepts
##   array[L] real alpha;
## }
## model {
##   for (d in 1:D) {
##     mu[d] ~ normal(0, 10); // hyperprior (same for both groups)
##     sigma[d] ~ inv_chi_square(10); // hyperprior (same for both groups)
##     for (l in 1:L) {
##       alpha[l] ~ normal(0,10); //prior, intercept
##       beta[l, d] ~ normal(mu[d], sigma[d]); //prior, beta
##     }
##   }
##   //likelihood
##   for (n in 1:N_train) {
##     y[n] ~ bernoulli(inv_logit(alpha[ll[n]] + x_train[n] * beta[ll[n]]));
##   }
## }
## generated quantities {
##   //log likelihood values for ELDP-LOO calculation
##   vector[N_train] log_lik;
##   for (n in 1:N_train)
##     //log_lik[n]=inv_logit(alpha[ll[n]] + x_train[n] * beta[ll[n]]);
##     log_lik[n]=bernoulli_logit_lpmf(y[n]|alpha[ll[n]] + x_train[n] * beta[ll[n]]);
##
##   // posterior predictions for the training data
##   array[N_train]int y_rep;
##   for (n in 1:N_train){
##     y_rep[n] = bernoulli_logit_rng(alpha[ll[n]] + x_train[n, ] * beta[ll][n]);
##   }
##   // posterior predictions for unseen data
##   array[N_test] int y_pred;
##   for (n in 1:N_test){
##     y_pred[n] = bernoulli_logit_rng(alpha[ll[n]] + x_pred[n, ] * beta[ll_pred][n]);
##   }
## }
```

```
#fit and simulate the draws using the Stan models
stan_data <- list(
                  D = ncol(student_new)-2,
                  N_train = nrow(train_data),
                  N_test = nrow(test_data),
                  y = train_data$G3,
                  x_train = train_data[-c(1,11)],
                  x_pred= test_data[-c(1,11)]
)
pooled_model <- pm$sample(data = stan_data, refresh=0, show_messages = FALSE, chains = 4, iter_warmup =
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.9 seconds.
## Chain 2 finished in 0.8 seconds.
## Chain 3 finished in 0.9 seconds.
## Chain 4 finished in 0.8 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.8 seconds.
## Total execution time: 1.1 seconds.
```

```
draws_pooled<- as_draws_df(pooled_model$draws())
```

```
stan_data <- list(
                  D = ncol(student_new)-2,
                  N_train = nrow(train_data),
                  N_test = nrow(test_data),
                  L = 2,
                  y = train_data$G3,
                  ll = train_data$school,
                  x_train = train_data[-c(1,11)],
                  x_pred = test_data[-c(1,11)],
                  ll_pred = test_data$school
)
hierarchical_model <- hm$sample(data = stan_data, refresh=0, show_messages = FALSE, chains = 4, iter_wa
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 3 finished in 8.2 seconds.
## Chain 1 finished in 8.3 seconds.
## Chain 4 finished in 8.3 seconds.
## Chain 2 finished in 8.5 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 8.3 seconds.
## Total execution time: 8.7 seconds.
```

```
draws_hierarchical <- as_draws_df(hierarchical_model$draws())
```

5 Convergence diagnostics

$\widehat{R}$ values:

The potential scale reduction factor $\widehat{R}$, defined as the square root of the ratio between estimated marginal posterior variance of all chains and the intra-chain variance. If the MCMC chains converge as the number of iterations increase, the ratio

$$\frac{\widehat{var^+(\psi|y)}}{W}$$

approaches 1. Thus, $\widehat{R}$ estimates the factor by how much the scale of the current chain distribution ψ could be reduced with more iterations. [2]

Effective sample sizes and the ESS / real sample size ratio:

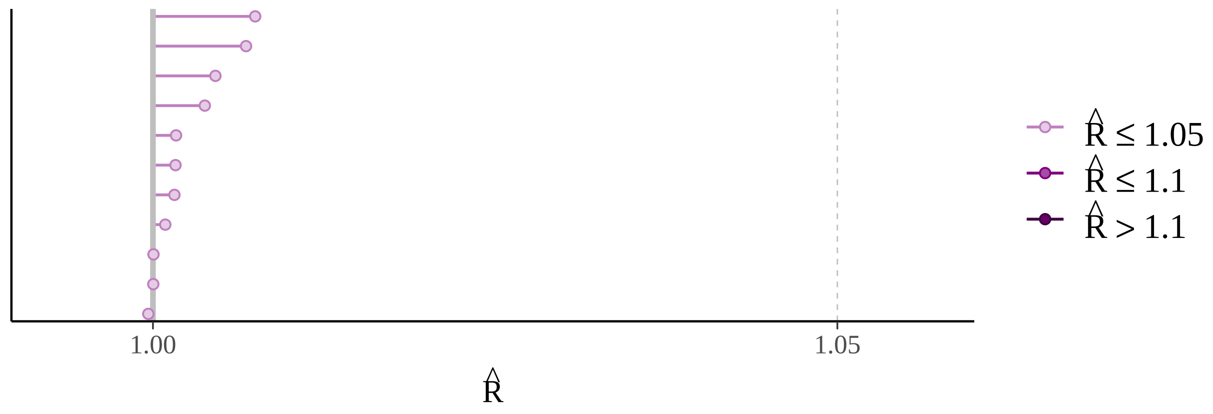
The effective sample size (ESS) is a measure by how much autocorrelation in MCMC samples increases uncertainty relative to an independent sample.

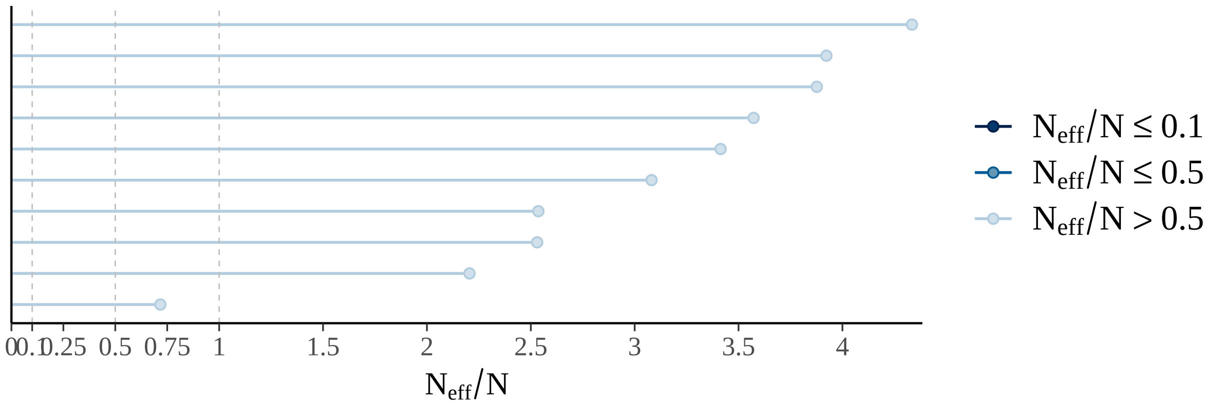
Below is a visualization of the ratios ESS / N, where N is the number of MCMC samples. Generally, a ratio below 0.1 indicates that the posterior predictions may not be reliable.

CmdstanR also has an inbuilt function diagnose(), which in addition to the above metrics will analyze divergent transitions, transitions hitting maximum tree depth, and E-BFMI values.

```
#rhat values for variables excluding ypred and log_lik
color_scheme_set("purple")
rhatp <- mcmc_rhat(pooled_model$summary()$rhat[1:11])

N = 1000 # total MCMC sample size
color_scheme_set("blue")
neffp <- mcmc_neff(pooled_model$summary()$ess_bulk[1:10] / N)

bayesplot_grid(rhatp, neffp, grid_args = list(nrow=2))
```

Pooled model:





```
pooled_model$cmdstan_diagnose()
```

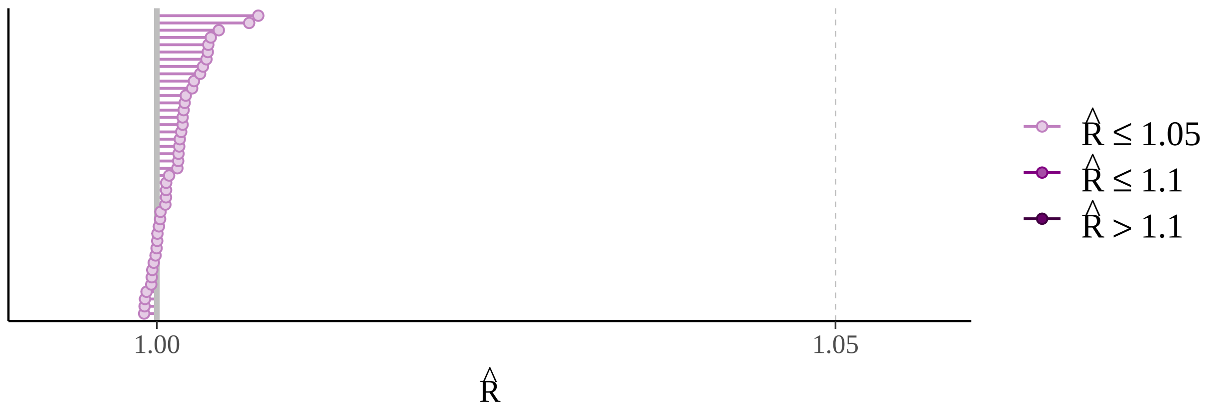
```
## Processing csv files: C:/Users/Juho/AppData/Local/Temp/RtmpyCKUfK/project_pooled-202212141843-1-35255
##
## Checking sampler transitions treedepth.
## Treedepth satisfactory for all transitions.
##
## Checking sampler transitions for divergences.
## No divergent transitions found.
##
## Checking E-BFMI - sampler transitions HMC potential energy.
## E-BFMI satisfactory.
##
## Effective sample size satisfactory.
##
## Split R-hat values satisfactory all parameters.
##
## Processing complete, no problems detected.
```

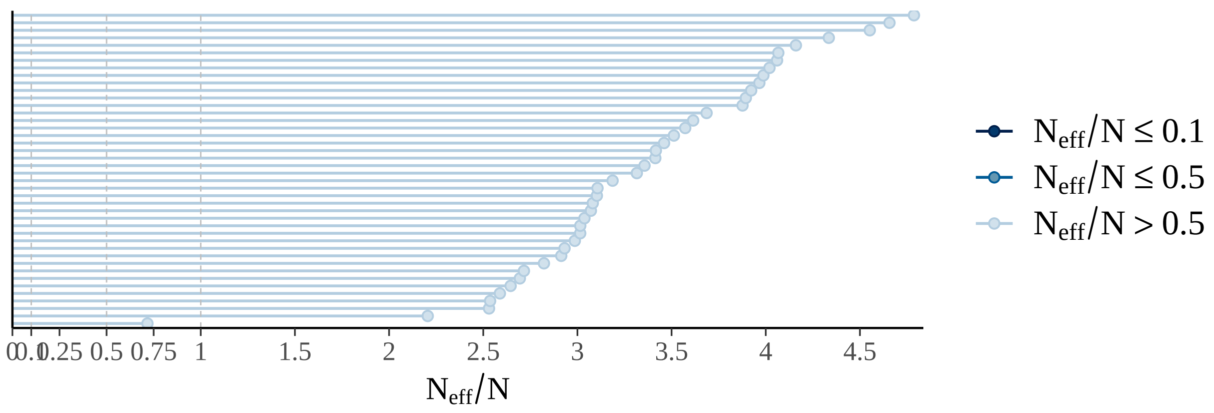
All $\widehat{R}$ values are close to 1, and the effective sample sizes are large enough for all of the estimated parameters. In addition, the cdstan_diagnose() results for tree depth and E-BFMI indicates proper convergence with no divergent transitions. The chains seem to have converged without problems.

```
#rhat values for variables excluding ypred and log_lik
color_scheme_set("purple")
rhath <-mcmc_rhat(pooled_model$summary()$rhat[1:42])
```

```
N = 1000 # total MCMC sample size
color_scheme_set("blue")
neffh <-mcmc_neff(pooled_model$summary()$ess_bulk[1:42] / N)

bayesplot_grid(rhath,neffh, grid_args = list(nrow=2))
```

Hierarchical model:





```
hierarchical_model$cmdstan_diagnose()
```

```
## Processing csv files: C:/Users/Juho/AppData/Local/Temp/RtmpyCKUfK/project_hierarchical-202212141843-
##
## Checking sampler transitions treedepth.
## Treedepth satisfactory for all transitions.
##
## Checking sampler transitions for divergences.
## No divergent transitions found.
##
## Checking E-BFMI - sampler transitions HMC potential energy.
## E-BFMI satisfactory.
##
## Effective sample size satisfactory.
##
## Split R-hat values satisfactory all parameters.
##
## Processing complete, no problems detected.
```

All $\widehat{R}$ values are close to 1, and the effective sample sizes are large enough for all of the estimated parameters. In addition, the cdstan_diagnose() results for tree depth and E-BFMI indicates proper convergence with no divergent transitions. The chains seem to have converged without problems.

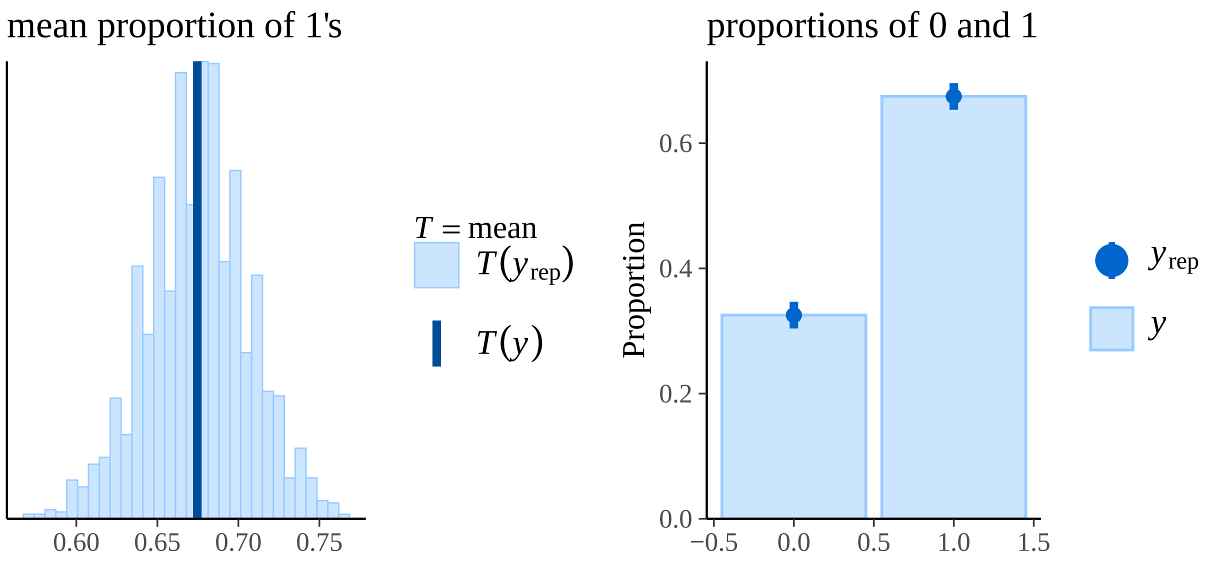
6 Posterior predictive checks

The posterior predictive check for the models will be done by calculating the predicted proportion of students that pass the course, and compare that against the one of the observed data.

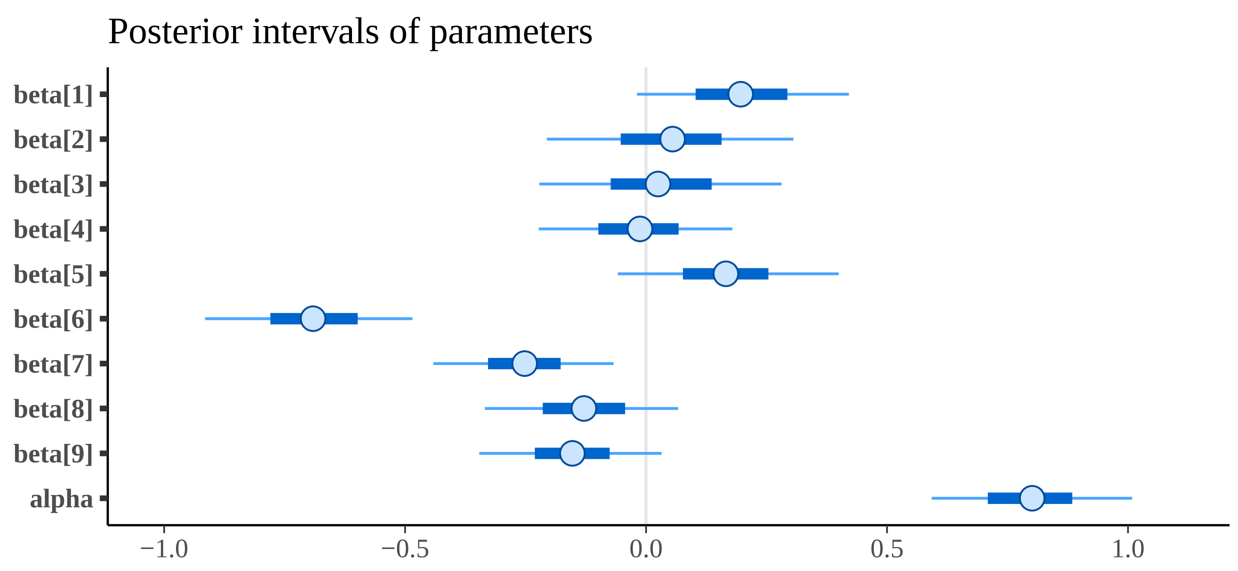
Posterior predictive check (Pooled):

```
color_scheme_set("brightblue")
ppc1 <-ppc_stat(y = train_data$G3, yrep = as.matrix(draws_pooled[,387:761]), stat = mean)

ppc2 <-ppc_bars(
  y =train_data$G3,
  yrep = as.matrix(draws_pooled[,387:761]),
  freq=FALSE,
  prob = 0.5,
  fatten = 1,
  size = 1.5
)

bayesplot_grid(ppc1,ppc2, grid_args = list(ncol=2), titles = c("mean proportion of 1's","proportions of
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



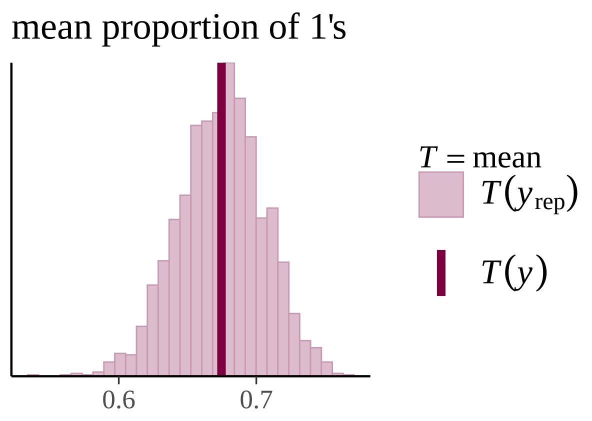
```
mcmc_intervals(draws_pooled, pars = c("beta[1]", "beta[2]","beta[3]","beta[4]","beta[5]","beta[6]","bet
```

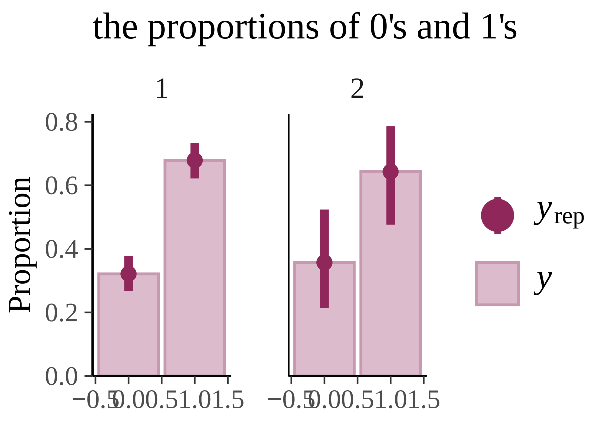


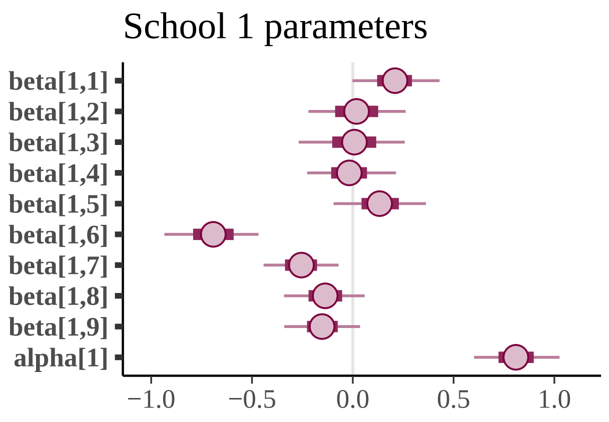
Posterior predictive check (Hierarchical):

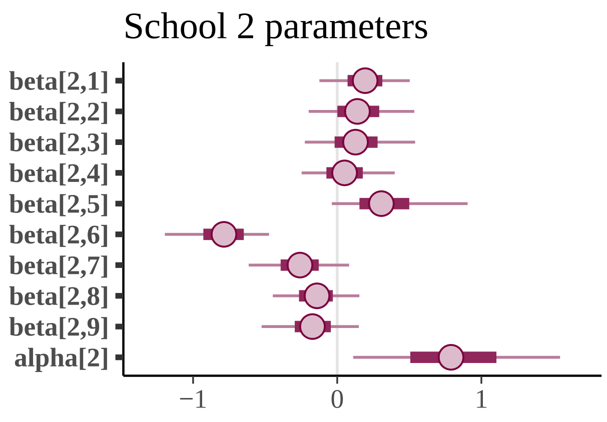
```
color_scheme_set("pink")
ppch1 <- ppc_stat(y = train_data$G3, yrep = as.matrix(draws_hierarchical[,415:789]), stat = mean)

ppch2 <-ppc_bars_grouped(
  y =train_data$G3,
  yrep = as.matrix(draws_hierarchical[,415:789]),
  group = train_data$school,
  freq=FALSE,
  prob = 0.9,
  fatten = 1,
  size = 1.5
)

ppch3 <- mcmc_intervals(draws_hierarchical, pars = c("beta[1,1]", "beta[1,2]","beta[1,3]","beta[1,4]","

ppch4 <- mcmc_intervals(draws_hierarchical, pars = c("beta[2,1]", "beta[2,2]","beta[2,3]","beta[2,4]","

bayesplot_grid(ppch1, ppch2, ppch3, ppch4 ,titles = c("mean proportion of 1's", "the proportions of 0's
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```









True proportion of passing students in school 1 and 2, respectively:

```r
sum(student_new$G3[student_new$school==1])/sum(student_new$school==1)
```

```
## [1] 0.6762178
```

```r
sum(student_new$G3[student_new$school==2])/sum(student_new$school==2)
```

```
## [1] 0.6304348
```

For both of the models, the mean predicted value matches that of the data.

The parameter interval plots describe the model parameter uncertainty related to their respective covariates: the wider the interval, the more uncertainty related to that parameter/covariate. It can be seen from the plots that the hierarchical model displays more uncertainty overall for students from the school 2, since the number of datapoints from the school is lower. The pooled model, on the other hand, shows combined and averaged uncertainty for the parameters as the group is not taken into account.

The magnitude of the parameters also reflect the importance of their respective covariates: beta[6], for example, corresponds to the number of past failures of a student and is valued highly in both of the models, whereas the parameters for travel time or parent's education level have lower impact.

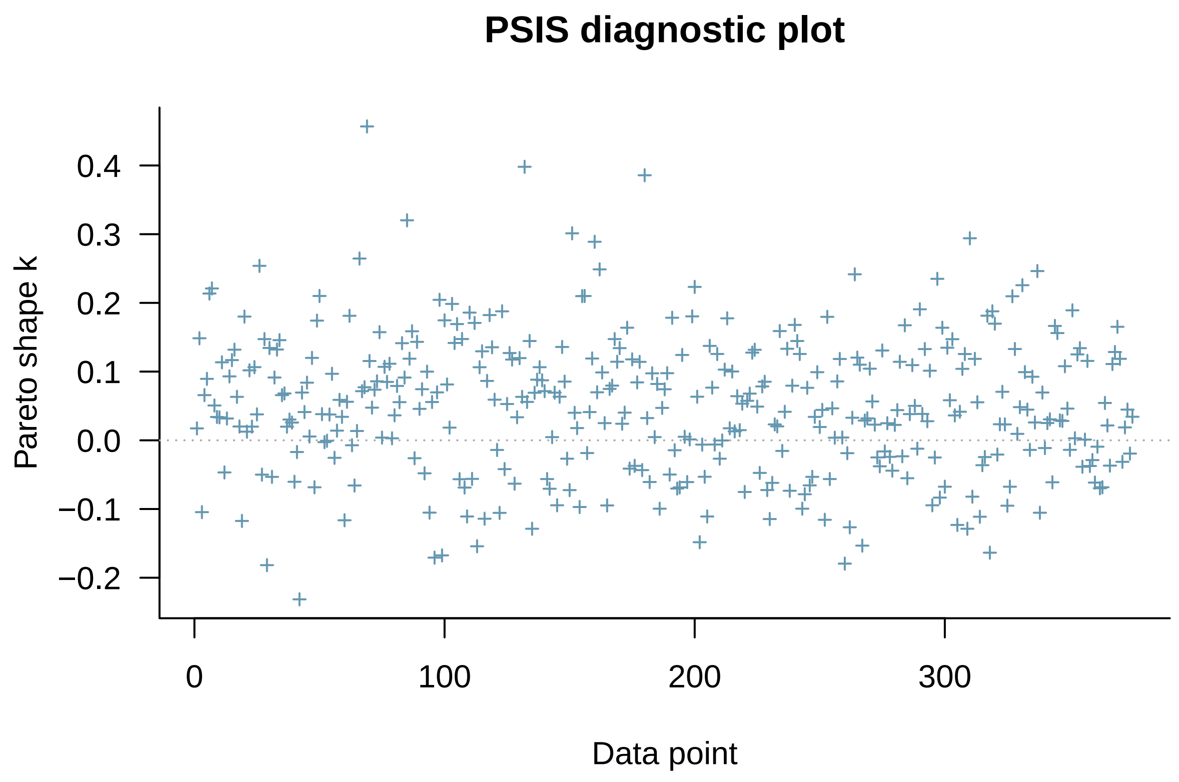
7 Model Comparison

Performing model comparison using PSIS-LOO cross validation:

```r
#values for the pooled model
loo_p <- loo(pooled_model$draws("log_lik"), r_eff = relative_eff(pooled_model$draws("log_lik")))
loo_p
```

```
##
## Computed from 2000 by 375 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -222.3  9.7
## p_loo        11.1  0.9
## looic       444.6 19.5
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
plot(loo_p)
```

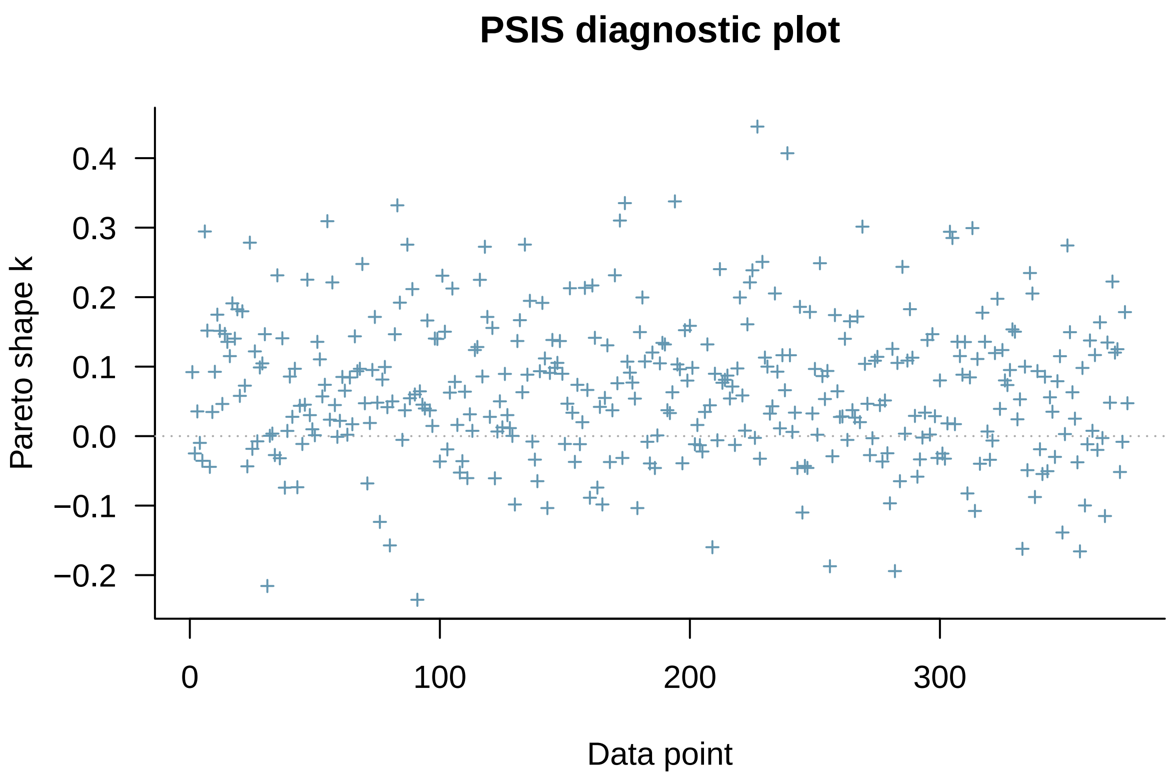


```
loo_h <- loo(hierarchical_model$draws("log_lik"), r_eff = relative_eff(hierarchical_model$draws("log_li
loo_h
```

```
##
## Computed from 2000 by 375 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -222.0  9.7
## p_loo        13.4  1.0
## looic       444.0 19.4
## ------
## Monte Carlo SE of elpd_loo is 0.1.
```

```
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
plot(loo_h)
```



```
loo_compare(list("pooled"=loo_p, "hierarchical"=loo_h))
```

```
##              elpd_diff se_diff
## hierarchical  0.0       0.0
## pooled       -0.3       1.0
```

As per above, the pareto k values are good for both of the models, and the ELPD-LOO estimates are accurate. The effective number of parameters is also reasonable for both of the models, indicating proper model behaviour.

The calculated ELPD-LOO value is slightly higher for the pooled model than the hierarchical model, though the difference is withing the Monte Carlo standard error: the models perform roughly equally well.

As also seen in posterior predictive check results, both of the models produce similar results. This is mostly due to how the hierarchical model is structured: the different sets of parameters for each group are estimated with the data, and similar groups are reflected as smaller differences between the group parameters. The key difference between the models is in the uncertainty between predictions for different groups: The predictions for the group with less datapoints will have more uncertainty in the hierarchical model, whereas the pooled model does not account for the group.

8 Prediction performance

10-folds Cross Validation for pooled model: We did K-folds Cross Validation with K=10 to assess the accuracy of the pooled model.

```
accuracy_pooled=0
index<-1:39
for (k in 1:10){
  index<-((k-1)*39+1):(k*39)
  train_data<-student_new[-index,]
  test_data<-student_new[index,]
  stan_data <- list(
                  D = ncol(student_new)-2,
                  N_train = nrow(train_data),
                  N_test = nrow(test_data),
                  y = train_data$G3,
                  x_train = train_data[-c(1,11)],
                  x_pred= test_data[-c(1,11)]
)
pooled_model <- pm$sample(data = stan_data, refresh=0, show_messages = FALSE, chains = 4, iter_warmup =
draws_pooled<- as_draws_df(pooled_model$draws())

accuracy_pooled[k]=mean(round(apply(draws_hierarchical[,752:790], 2, mean)) == test_data$G3)
}
accuracy_pooled
```

10-folds Cross Validation for hierarchical model: We did K-folds Cross Validation with K=10 to assess the accuracy of the hierarchical model.

```
accuracy_hierarchical =0
index<-1:39
for (k in 1:10){
  index<-((k-1)*39+1):(k*39)
  train_data<-student_new[-index,]
  test_data<-student_new[index,]
  stan_data <- list(
                  D = ncol(student_new)-2,
                  N_train = nrow(train_data),
                  N_test = nrow(test_data),
                  L = 2,
                  y = train_data$G3,
                  ll = train_data$school,
                  x_train = train_data[-c(1,11)],
                  x_pred = test_data[-c(1,11)],
                  ll_pred = test_data$school
)
hierarchical_model <- hm$sample(data = stan_data, refresh=0, show_messages = FALSE, chains = 4, iter_wa
draws_hierarchical <- as_draws_df(hierarchical_model$draws())

accuracy_hierarchical [k]=mean(round(apply(draws_hierarchical[,752:790], 2, mean)) == test_data$G3)
}
```

```
accuracy_pooled
```

```
##  [1] 0.5641026 0.5641026 0.5641026 0.7692308 0.6666667 0.5384615 0.5641026
##  [8] 0.5384615 0.7692308 0.6666667
```

```
mean(accuracy_pooled)
```

```
## [1] 0.6205128
```

```
accuracy_hierarchical
```

```
##  [1] 0.6410256 0.6923077 0.5641026 0.7179487 0.8205128 0.7179487 0.7435897
##  [8] 0.6153846 0.6153846 0.7692308
```

```
mean(accuracy_hierarchical)
```

```
## [1] 0.6897436
```

```
sd(accuracy_pooled)
```

```
## [1] 0.09109684
```

```
sd(accuracy_hierarchical)
```

```
## [1] 0.07967536
```

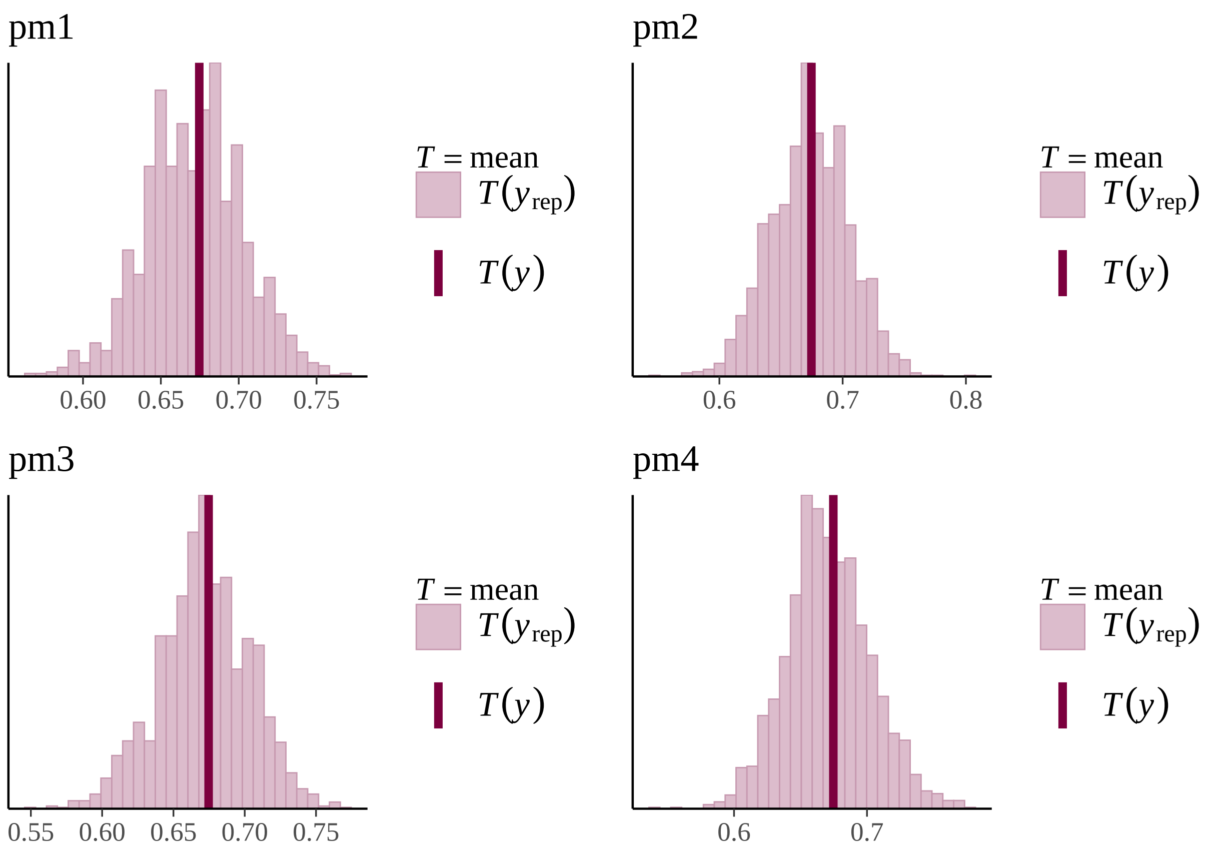
Despite seeing similar performance in the previous model performance estimates like ELPD-loo values and posterior predictive checks, it can be seen that the average value of the validation accuracy for the hierarchical model is better than the pooled one. The average accuracy for the hierarchical model is 71% and for the pooled is 61%.

9 Sensitivity analysis with respect to prior choices

Testing the model performance with different priors: The sensitivity is estimated by varying the prior distributions and their parameter selections, and calculating the posterior mean for the proportion of passing students against the true proportion in the data.

```

p1<-ppc_stat(y = train_data$G3, yrep = as.matrix(draws_pooled_prior1[,387:761]), stat = mean)

p2<-ppc_stat(y = train_data$G3, yrep = as.matrix(draws_pooled_prior2[,387:761]), stat = mean)

p3<-ppc_stat(y = train_data$G3, yrep = as.matrix(draws_pooled_prior3[,387:761]), stat = mean)

p4<-ppc_stat(y = train_data$G3, yrep = as.matrix(draws_pooled_prior4[,387:761]), stat = mean)

bayesplot_grid(p1,p2,p3,p4, grid_args = list(nrow=2,ncol=2), titles = c("pm1", "pm2", "pm3", "pm4"))
```

1. Pooled model:

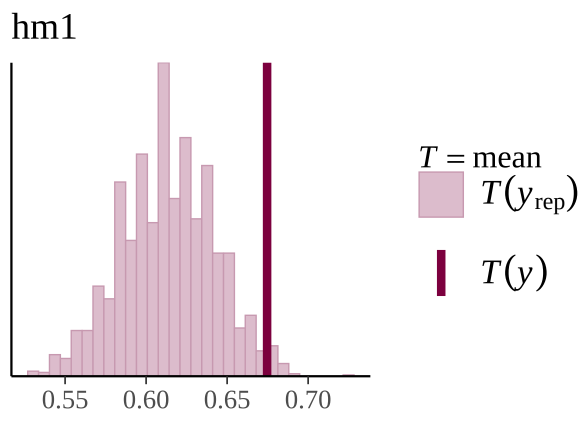


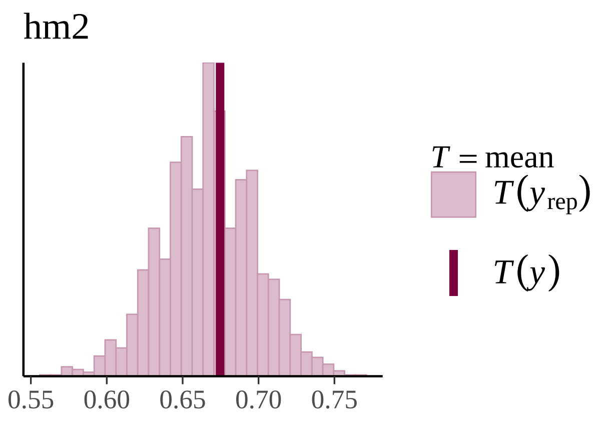
Model	α prior	β prior	$\mu(T_{yrep})$	T_{yrep} 2.5%	T_{yrep} 97.5%	divergent transitions	% -pareto k > 0.7
pm1	normal(0,1)	normal(0,1)	0.67	0.61	0.73	0	0
pm2	cauchy(0,10)	cauchy(0,10)	0.67	0.61	0.73	0	0
pm3	cauchy(0,1)	cauchy(0,1)	0.67	0.61	0.73	0	0
pm4	normal(0,1)	cauchy(0,1	0.67	0.61	0.73	0	0

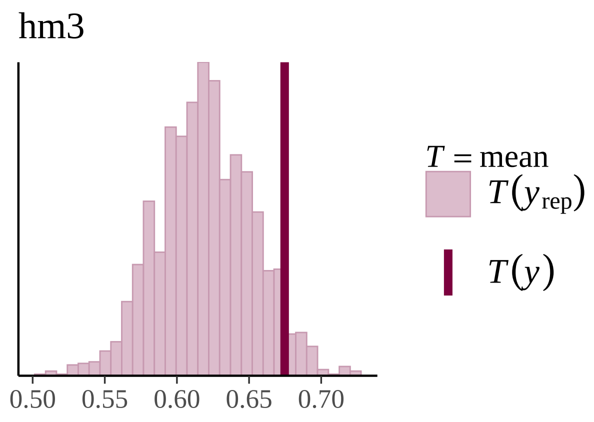
With different priors, the posterior mean stays close to the observation mean. The pooled model is therefore rather robust, and the changes in the prior distributions have little effect on the posterior.

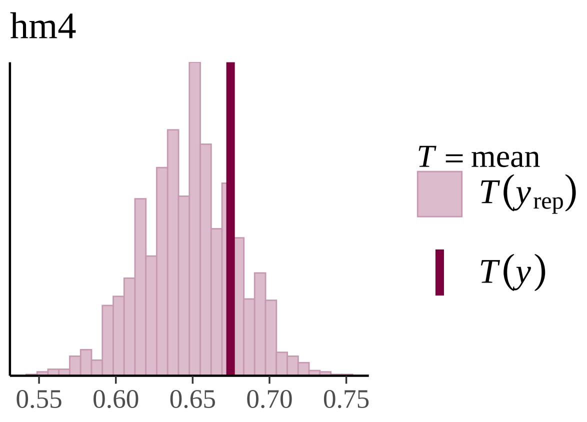
```
p1<-ppc_stat(y = train_data$G3, yrep = as.matrix(draws_hierarchical_prior1[,415:789]), stat = mean)

p2<-ppc_stat(y = train_data$G3, yrep = as.matrix(draws_hierarchical_prior2[,415:789]), stat = mean)

p3<-ppc_stat(y = train_data$G3, yrep = as.matrix(draws_hierarchical_prior3[,415:789]), stat = mean)

p4<-ppc_stat(y = train_data$G3, yrep = as.matrix(draws_hierarchical_prior4[,415:789]), stat = mean)

bayesplot_grid(p1,p2,p3,p4, grid_args = list(nrow=2, ncol=2), titles = c("hm1", "hm2", "hm3", "hm4"))
```

Hierarchical model:









Model	μ_d prior	σ_d prior	α_l prior	$\mu(T_{yrep})$	T_{yrep} 2.5%	T_{yrep} 97.5%	divergent transitions	% -pareto k >0.7
hm1	normal(0,0.5)	Inv-χ^2(10)	normal(0,0.5)	0.61	0.55	0.67	0	0
hm2	normal(0,2)	Inv-χ^2(10)	normal(0,2)	0.66	0.60	0.72	0	0
hm3	cauchy(0,0.5)	cauchy(0,0.5)	cauchy(0,0.5)	0.62	0.56	0.68	775	2.1
hm4	cauchy(0,1)	cauchy(0,1)	cauchy(0,1)	0.64	0.59	0.70	236	0

As seen in the graphs above, the posterior distribution of the point estimate varies noticeably between different prior choices. With more informative prior choices, especially for the intercept parameter α, the posterior distribution is subject to change: the hierarchical model is thus more sensitive to different priors than the pooled model.

10 Discussion of issues

The two models have good PSIS values, and both have good k-values, for the hierarchical one, one should think about the priors carefully as with priors having less variation around the mean, the divergence of chains is increasing. Despite this issue, it gives better accuracy for the prediction. Because the data has many variables, the model has many different parameters to estimate, which makes it more complicated to assess various model performance metrics.

To improve the model, one can think about variable selection criteria since the data has 30 variables and we only used 10 covariates. In the current settings, the prediction accuracy is good, but it might increase with another chosen set of variables.

11 Conclusion

In conclusion, the models have performed fairly in the context of convergence diagnostics, and posterior predictive checks. When predicting the unseen data, the hierarchical model has much better performance.

It has shown that the chosen set of variables are able to achieve fair accuracy which means that the performance of a student is not only depending on the student but rather to this set of variables such as parental education, number of failures, number of absences,.. They are really playing a crucial role in the performance of the students as they can have positive or negative influences on the performance.

In the hierarchical model, the uncertainty for the second school increases because it has low proportion of the data whereas in the pooled model where data is combined, the uncertainty is also combined.

12 Self reflection

The process of conducting and writing an extensive Bayesian data analysis report was both a demanding and rewarding task. While working on the project, the group had to reread and practically implement both the theory and analysis workflow taught during the course, in such a way which knit the individual topics into a coherent whole.

While the project taught many of the topics well and deepened our understanding of them, working on it would have been a lot more streamlined and structured had the course contents been studied more thoroughly before: this caused us to go through iterations that could have been skipped with better understanding of the subject.

As an example of the practical Bayesian data analysis, we learnt that not any data can be used for hierarchical models. The data must have levels from the beginning, one cannot use a covariate in the data to associate it with outcome and make it as levels for the hierarchical model. Moreover, we learnt a lot about plotting new things such as ppc, and mcmc_intervals, how to use hierarchical logistic regression and pooled one. How to do a posterior predictive check for a prediction problem. While doing the sensitivity analysis, we learnt how the prior has effect on the results and on the divergence.

[1] P. Cortez and A. Silva, "Using data mining to predict secondary school student performance. In a. Brito and j. Teixeira eds," *Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008)*, pp. 5–12, 2008, Available: http://www3.dsi.uminho.pt/pcortez/student.pdf

[2] A. Vehtari, A. Gelman, D. Simpson, B. Carpenter, and P.-C. Bürkner, "Rank-normalization, folding, and localization: An improved R^ for assessing convergence of MCMC (with discussion)," *Bayesian Analysis*, vol. 16, no. 2, Jun. 2021, doi: 10.1214/20-ba1221.