BDA - Project
Active heart rate - Bayesian analysis

Anonymous

Contents

1. Introduction

Everyone has a unique heart rate which can tell a lot about health and life choices of an individual. It's a combination of different affecting measures, most of which one can have an influence on. Resting heart rate can be as useful thing to measure as body temperature: to see if there is any aberration from normal. Active heart rate during exercise might tell even more but takes more effort to measure. According to Harvard (2015), vigorous exercise is the most effective way to lower one's resting heart rate and improve one's health. It also increases the maximum heart rate and aerobic capacity, which is also associated with a lower risk of heart attack and death.

In this project, we try to reach similar conclusions by exploring Pulse Rates and Exercise Data from the collection of common R-datasets. We use the active heart rate as a measure of overall fitness of the individual: the lower the active heart rate, the better the physical shape. The logic behind this is that people who exercise more have a much higher threshold of getting tired and out of breath. We will try to model the effects of lifestyle habits such as smoking, BMI, etc. on one's active heart rate by using separate and hierarchical models, as well as present and interpret results from a Bayesian point of view.

Firstly, we preprocess the data and separate it in three groups based on the weekly hours of exercise. Then we present our chosen models: separate and hierarchical with their respective priors. We perform convergence diagnostics, posterior predictive checks, and sensitivity analysis. We also compare the two models with LOO-CV. The Stan codes are given in the Appendix.

2. Data: Pulse rates and exercise

The dataset is called "Pulse rates and exercise" and it is publicly accessible at Arel-Bundock (n.d.). To the best of our knowledge, this dataset has not been used in another case study.

It contains 232 observations with 7 variables:

- *Active* – heart rate after exercise (beats per minute)

- *Rest* – resting heart rate (beats per minute)
- *Smoke* – binary variable indicating if the person is a smoker or not (1=smoker, 0=non-smoker)
- *Sex* – binary variable indicating the sex of the person (1=female, 0=male)
- *Exercise* – typical hours of exercise per week (1, 2, or 3)
- *Hgt* – height (inches)
- *Wgt* – weight (pounds)

The data was collected by recording the resting heart rate of a group of students and then recording the 'active' heart rate after walking up and down a set of stairs. The students provided additional information about height, weight, exercise, and smoking habits via a survey.

2.1. Data Preprocessing

```r
data = read.csv('Pulse.csv')
```

Firstly, we noticed that for some students the active heart rate was lower than the resting heart rate. This did not seem logical, so we decided to remove these observations from the dataset.

```
data_clean <- subset(data, Active>Rest)
```

Secondly, we decided to combine the height and weight of the students in one variable – body mass index (BMI) with unit of (kg/m^2). It is calculated with the following formula (CDC (2019)):

$$\text{BMI} = \frac{\text{weight}}{\text{height}^2} \cdot 703$$

According to NHLBI (n.d.), BMI is a good measure for one's risk of certain diseases, and thus overall health for the general population. Our reasoning for including this variable is that we expect people with a lower BMI, and thus less fat, to be in a better physical condition (=lower active heart rate) compared to those with a higher BMI, and thus more fat.

After performing these steps, the dataset contains 227 observations with 6 variables:

```
head(data_clean)
```

```
##   Active Rest Smoke Sex Exercise      BMI
## 1     97   78     0   1        1 21.07760
## 2     82   68     1   0        3 32.28061
## 3     88   62     0   0        3 23.73167
## 4    106   74     0   0        3 23.05363
## 5     78   63     0   1        3 19.57563
## 6    109   65     0   0        3 24.13514
```

Active heart rate (Active in the table shown above) is a variable that we observe and want to predict. Active is used as the likelihood y while training the model and y_rep to get the prediction value. The other variables (Rest, Smoke, Sex, Exercise, and BMI) are used as the input X, which denotes the variables (covariates) in the model.

Furthermore, we divide this dataset into three subsets for each level of exercise (1, 2, or 3 hours per week), which we then use for feeding into the models as the different groups: group 1 with one hour of exercise per week, group 2 with two hours of exercise per week, and group 3 with three hours of exercise per week. We separate the data based on this condition because we expect to see differences between people who exercise more (and should be in better shape) and people who exercise less (and should be in worse shape).

```
data1 <- data_clean %>% filter(Exercise == 1)
data2 <- data_clean %>% filter(Exercise == 2)
data3 <- data_clean %>% filter(Exercise == 3)
```

3. Models

For this project, we decided to build three separate models for each group, and compare them to a hierarchical model.

The separate model treats each group of students individually. Here we make the assumption that prior distributions on the parameters of each group are independent, and we fit separate models allowing us to infer about one group independently of the inferences of the remaining groups. The decision about using a separate model is based on the reasoning that the assumption of independence is not always applicable though, as we would expect some relation between the groups.

Hence, we also build a hierarchical model which allows us to examine each group, as well as the population. This approach acknowledges the similarities and differences between the groups.

4. Priors

The priors that we use are weakly informative. The reason for this is that we do not want to influence the posterior distributions too much, but we want to set an appropriate scale, without considering our prior knowledge in this area, or at least limiting its impact.

4.1. Separate model

For our models, we are representing the mean of the distributions as a linear combination of the covariates, so the resulting distributions of the active heart rate can be represented in the following way:

$$y \sim \text{Normal}(\alpha + x \cdot \beta, \sigma)$$

α can be thought of as the intercept when all the other covariates are 0. Due to this fact, we are choosing Normal(110, 30) as prior distribution for alpha. With this distribution, most of the data is expected to fall within the range [20, 200], which is a relatively reasonable range for a heart rate. For β, we do not really know how each variable impacts the active heart rate, so we choose the prior Normal(0, 1), as the differences between the resting and active heart rate are not extremely big, so we do not expect the other variables to get very high coefficients.

In order to account for some uncertainty in the prior of the mean, we choose the prior Normal(0, 10) for the standard deviation.

The chosen priors are the following:

$$\begin{aligned}
\alpha &\sim \text{Normal}(110, 30)\\
\beta &\sim \text{Normal}(0, 1)\\
\sigma &\sim \text{Normal}(0, 10)
\end{aligned}$$

4.2. Hierarchical model

For the priors for the hierarchical models, we tried many different priors before deciding the final ones due to problems with convergence.

The final chosen priors based on the convergence analysis are the following:

$$\begin{aligned}
\mu_{0_\alpha} &\sim \text{Normal}(110, 30)\\
\sigma_{0_\alpha} &\sim \text{Normal}(0, 100)\\
\mu_{0_\beta} &\sim \text{Normal}(0, 2)\\
\sigma_{0_\beta} &\sim \text{Inv-}\chi^2(1)\\
\alpha &\sim \text{Normal}(\mu_{0_\alpha}, \sigma_{0_\alpha})\\
\beta &\sim \text{Normal}(\mu_{0_\beta}, \sigma_{0_\beta})\\
\sigma &\sim \text{Normal}(0, 10)
\end{aligned}$$

5. Fitting the Stan models

The Stan code for the separate and hierarchical models is given in the Appendix.

5.1. Separate model

The data used for the separate model is defined in the following way:

```r
separate_data <- list(J=3,
                      N1=length(data1$Active),
                      N2=length(data2$Active),
                      N3=length(data3$Active),
                      K=length(data_clean)-1,
                      x1=data1[2:length(data1)],
                      x2=data2[2:length(data2)],
                      x3=data3[2:length(data3)],
                      y1=data1[['Active']],
                      y2=data2[['Active']],
                      y3=data3[['Active']],
                      prior=1)
```

The separate model was fitted with the default Stan settings:

```r
separate_fit <- stan(file = "separate1.stan",
                     data = separate_data,
                     seed = SEED)

sep_samples <- extract(separate_fit)
```

5.2. Hierarchical model

The data used for the hierarchical model is defined in the following way:

```r
hierarchical_data <- list(J=3,
                      N1=length(data1$Active),
                      N2=length(data2$Active),
                      N3=length(data3$Active),
                      K=length(data_clean)-1,
                      x1=data1[2:length(data1)],
                      x2=data2[2:length(data2)],
                      x3=data3[2:length(data3)],
                      y1=data1[['Active']],
                      y2=data2[['Active']],
                      y3=data3[['Active']],
                      prior=1)  # 1 for default, 2 for sigma0 change, 3 for sigma0 change,
                                # 4 for mu0 change, 5 for mu0 change
```

The hierarchical model was fitted with modified settings. The maximum tree depth and adapt_delta were increased, as well as the number of iterations.

```r
hierarchical_fit <- stan(file = "hier_N2.stan",
                     data = hierarchical_data,
                     seed = SEED,
                     control = list(adapt_delta = 0.99, max_treedepth = 15),
                     warmup = 2000,
                     iter = 4000)
```

```
hier_samples <- extract(hierarchical_fit)
```

6. Convergence diagnostics

For convergence diagnostics, we use $\widehat{R}$, effective sample size (ESS), and divergent transitions.

```
sep_summary <- as.data.frame(summary(separate_fit)$summary[, c('Rhat', 'n_eff')])
sep_r_hat <- sep_summary %>% filter(Rhat > 1.05)
sep_r_hat <- sep_r_hat$Rhat

sep_n_eff <- sep_summary %>% filter(n_eff < 400)
sep_n_eff <- sep_n_eff$n_eff
```

```
hier_summary <- as.data.frame(summary(hierarchical_fit)$summary[, c('Rhat', 'n_eff')])
hier_r_hat <- hier_summary %>% filter(Rhat > 1.05)
hier_r_hat <- hier_r_hat$Rhat

hier_n_eff <- hier_summary %>% filter(n_eff < 400)
hier_n_eff <- hier_n_eff$n_eff
```

6.1. $\widehat{R}$

$\widehat{R}$ is the potential scale reduction factor. The function for calculating $\widehat{R}$ compares the between- and within-chain estimates for model parameters and other univariate quantities of interest. If chains have not mixed well, i.e. they don't converge, the $\widehat{R}$ value is larger than 1. However, as per Stan documentation, the acceptable $\widehat{R}$ values for posterior draws are less than 1.05, so we use that condition in our analysis.

Number of $\widehat{R}$ values higher than 1.05 for the separate and hierarchical model is zero:

```
length(sep_r_hat)
```

```
## [1] 0
```

```
length(hier_r_hat)
```

```
## [1] 0
```

For both models all $\widehat{R}$ values are less than 1.05, which indicates the chains have converged and mixed well.

6.2. Effective sample size

N_{eff} is a crude measure of effective sample size. An ESS > 100 per chain is considered good, as per the Stan output. We use four chains, which means the effective sample size values should be over 400.

Number of n_{eff} values lower than 400 for the separate and hierarchical model is zero:

```r
length(sep_n_eff)
```

```
## [1] 0
```

```r
length(hier_n_eff)
```

```
## [1] 0
```

For both models all of the n_{eff} values are larger than the required minimum of 400, which indicates that the simulated sequences achieve the required stability.

6.3. Divergent transitions

Divergent transitions happen when the simulated Hamiltonian trajectory departs from the true trajectory. This will be measured by the departure of the Hamiltonian value from its initial value. The simulation cannot be trusted if the number of divergent transitions is too high.

```r
sep_sampler_params <- get_sampler_params(separate_fit, inc_warmup = FALSE)
sep_divergent_iterations_by_chain <- sapply(sep_sampler_params,
                                            function(x) sum(x[, "divergent__"]))
sum(sep_divergent_iterations_by_chain)
```

```
## [1] 0
```

There are no divergent transitions for the separate model, which further confirms that the chains have converged.

```r
hier_sampler_params <- get_sampler_params(hierarchical_fit, inc_warmup = FALSE)
hier_divergent_iterations_by_chain <- sapply(hier_sampler_params,
                                             function(x) sum(x[, "divergent__"]))
sum(hier_divergent_iterations_by_chain)
```

```
## [1] 6
```

For the hierarchical model there are 6 divergent transitions, which means the model has not fully converged.

6.4. Conclusion about convergence

Considering all metrics' results ($\widehat{R}, n_{eff}$, divergent transitions) are satisfactory for the separate model, we can conclude that this model has converged. There are a few divergent transitions for the hierarchical model, which indicates that there is a risk of biased Markov chain Monte Carlo (MCMC) estimates.

6.5. Improving convergence

The separate model converged on the first try without the need to modify the fit parameters for the model, the priors, etc.

However, the hierarchical model did not converge and modifications were needed. We decided to change the hyper-priors depending on the number of divergences, $\widehat{R}$ value, and ESS. Stan allows us to check these parameters automatically and provide the warnings for each run.

We tried to change μ_{0_α} and σ_{0_α} to be consistent with the mean of data for all groups, and μ_{0_β} and σ_{0_β} to be consistent with the weight of each parameters in data for all groups.

The steps we took are listed below.

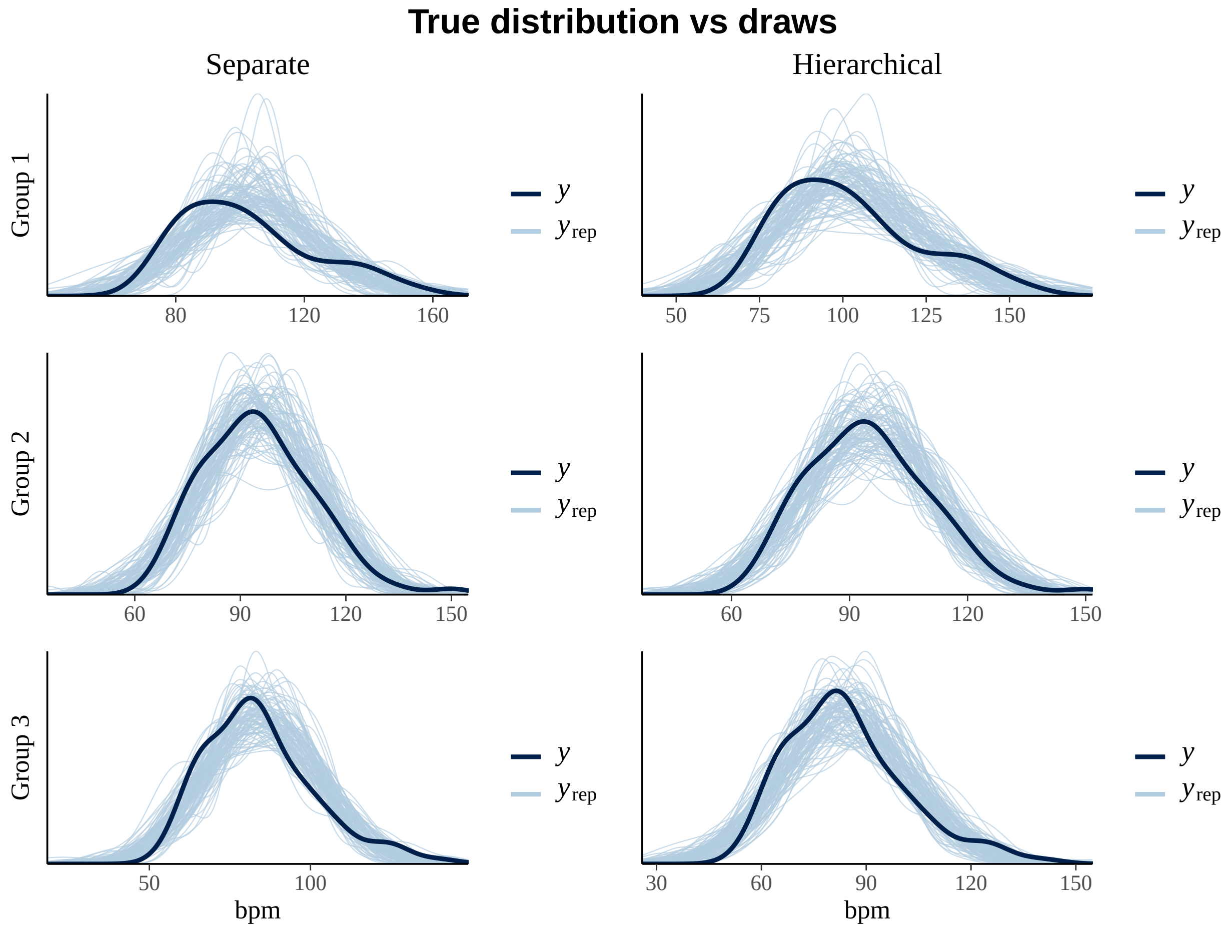
1. Change the hyper-prior until the number of divergences is less than 50.
2. Increase number of iterations from 2000 to 4000. Change the warmup from 1000 to 2000. Increasing iterations allows each chain to be converge and prevents bulk ESS and tail ESS from being too low.
3. Increase the target acceptance rate (adapt_delta) to 0.99.
4. Increase the maximum tree depth from 10 (default value) to 15.

After doing this, we managed to achieve satisfactory convergence for the hierarchical model and reduce the number of divergent transitions.

7. Posterior predictive checks

For checking the posterior predictive performance, we plotted the density estimates from several draws against the true distributions.

```r
sep_y1_samp <- sample(nrow(sep_samples$y1_rep), 100)
sep_y2_samp <- sample(nrow(sep_samples$y2_rep), 100)
sep_y3_samp <- sample(nrow(sep_samples$y3_rep), 100)

hier_y1_samp <- sample(nrow(hier_samples$y1_rep), 100)
hier_y2_samp <- sample(nrow(hier_samples$y2_rep), 100)
hier_y3_samp <- sample(nrow(hier_samples$y3_rep), 100)
```



For group 1, we found that the predicted mean of active heart rate is shifted from the observed mean (~103 bpm). The true distribution also has an 'irregular' shape, which makes it more difficult for the models to follow it well as it does not match the normal distribution exactly.

In group 2, the distributions of observed and predicted model are similar.

In group 3, the distributions of observed and predicted model are almost similar. However, each predicted draw plot does not have the obvious peak around the true mean value (~85 bpm), as the true distribution is again a bit different than the standard normal distribution.

Overall, according to all three groups result, we can conclude that our predictive posteriors are quite reasonable, but they can be improved, especially for group 1.

8. Model comparison with LOO-CV

LOO-CV i.e. leave-one-out cross-validation is a method to validate Bayesian models. From our posterior draws, we can compute efficient approximate LOO-CV by using Pareto smoothed importance sampling (PSIS), which is a new method for stabilizing importance ratios.

K-values are used to assess the reliability of PSIS-based estimates. The ideal target value for k would be below 0.5, but values up to 0.7 have been observed to give good results. After that, the convergence rates get impractical. Below 0.5 the distribution of raw importance ratios has finite variance and the central limit theorem holds, which is great. After that the convergence of the estimate is slower with increasing k. When the values of k are between 0.5 and 1.0, the variance of raw importance ratios is infinite, but the mean still exists. However, if k is between 0.5 and 0.7, we can observe practically useful convergence rates and Monte

Carlo error estimates with PSIS. If k is more than 0.7, we cannot observe the practical and reliable result from both of them. After 1.0, neither the variance nor the mean of the raw importance ratios exist and the convergence ratio is close to 0.

```
separate_log_lik <- loo::extract_log_lik(separate_fit, merge_chains = FALSE)
separate_r_eff <- loo::relative_eff(exp(separate_log_lik))
separate_loo <- loo::loo(separate_log_lik, r_eff = separate_r_eff)
print(separate_loo)
```

```
##
## Computed from 4000 by 227 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -938.1 15.3
## p_loo         9.8  2.1
## looic      1876.2 30.6
## ------
## 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.
```

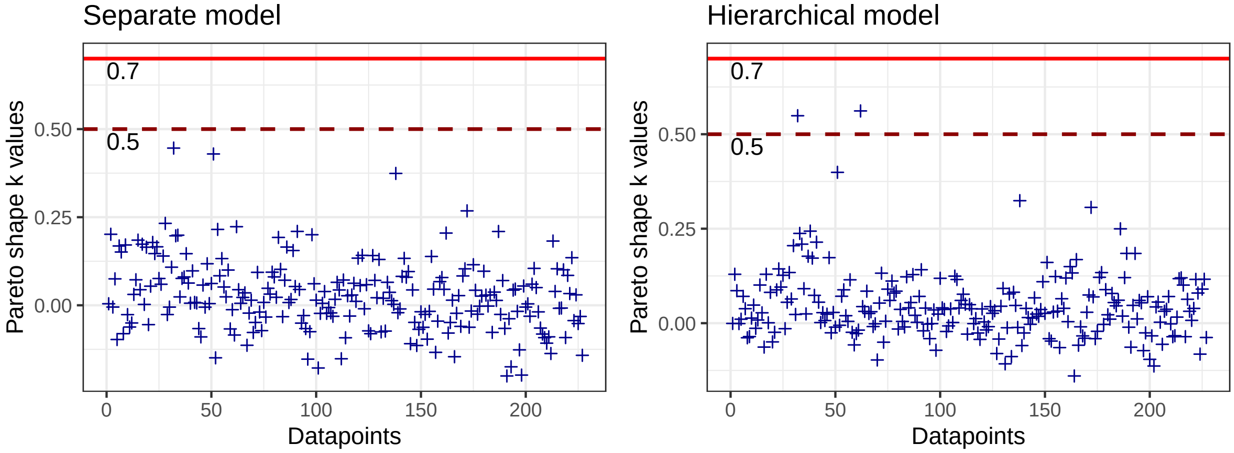
```
hier_log_lik <- loo::extract_log_lik(hierarchical_fit, merge_chains = FALSE)
hier_r_eff <- loo::relative_eff(exp(hier_log_lik))
hier_loo <- loo::loo(hier_log_lik, r_eff = hier_r_eff)
print(hier_loo)
```

```
##
## Computed from 8000 by 227 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -932.2 16.0
## p_loo         8.1  1.5
## looic      1864.4 32.1
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     225   99.1%   2183
##  (0.5, 0.7]   (ok)         2    0.9%   899
##    (0.7, 1]   (bad)        0    0.0%   <NA>
##    (1, Inf)   (very bad)   0    0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo_compare(separate_loo, hier_loo)
```

```
##        elpd_diff se_diff
## model2  0.0       0.0
## model1 -5.9       3.6
```

According to the LOO-CV comparison table, the elpd_diff represents the difference of elpd relative to the largest elpd that our model can provide. In our case, the hierarchical model (model 2) has the largest elpd value. When considering the elpd_diff and se_diff (the standard error of the difference), we can conclude that the hierarchical model performs better than the separate one.


Separate model

Hierarchical model

All k values for the separate model are under 0.5, while the hierarchical model has a few over 0.5. However, all k values for both models are below 0.7. This indicates that both of them can be considered as reliable.

9. Predictive performance assessment

The task of this project is not a classification task; we are trying to explore and predict heart rate, which is a regression task. For this purpose, LOO-CV is an adequate measure of performance.

10. Sensitivity analysis

For checking whether the models are sensitive to the prior, we perform a comparison of the results obtained with the default priors (used in the previous sections) and several different priors. For the comparison we used the k values, as well as a visual representation of the posterior distributions.

For the separate model, we are checking the results when using a wider and a more narrow priors. The used priors are the following:

- Wide prior

$$\begin{aligned}
\alpha &\sim \text{Normal}(0, 100) \\
\beta &\sim \text{Normal}(0, 2) \\
\sigma &\sim \text{Normal}(0, 30)
\end{aligned}$$

```r
separate_data_wide <- list(J=3,
                           N1=length(data1$Active),
                           N2=length(data2$Active),
                           N3=length(data3$Active),
                           K=length(data_clean)-1,
                           x1=data1[2:length(data1)],
                           x2=data2[2:length(data2)],
                           x3=data3[2:length(data3)],
                           y1=data1[['Active']],
                           y2=data2[['Active']],
                           y3=data3[['Active']],
                           prior=2) # 1 for default, 2 for wide, 3 for narrow


separate_fit_wide <- stan(file = "separate1.stan",
                          data = separate_data_wide,
                          seed = SEED)

samples_wide <- extract(separate_fit_wide)
```

- Narrow prior

$$\begin{aligned}
\alpha &\sim \text{Normal}(110, 5) \\
\beta &\sim \text{Normal}(0, 0.5) \\
\sigma &\sim \text{Normal}(0, 5)
\end{aligned}$$

```r
separate_data_narrow <- list(J=3,
                             N1=length(data1$Active),
                             N2=length(data2$Active),
                             N3=length(data3$Active),
                             K=length(data_clean)-1,
                             x1=data1[2:length(data1)],
                             x2=data2[2:length(data2)],
                             x3=data3[2:length(data3)],
                             y1=data1[['Active']],
                             y2=data2[['Active']],
                             y3=data3[['Active']],
                             prior=3) # 1 for default, 2 for wide, 3 for narrow


separate_fit_narrow <- stan(file = "separate1.stan",
                            data = separate_data_narrow,
                            seed = SEED)

samples_narrow <- extract(separate_fit_narrow)
```

For the hierarchical model first we tried different hyper-priors of α and β (σ_{0_α} and σ_{0_β}) in hier2 and hier3, respectively.

- Hier2 hyper-prior:

$$
\begin{aligned}
\mu_{0_\alpha} &\sim \text{Normal}(110, 30) \\
\sigma_{0_\alpha} &\sim \text{Normal}(0, 10) \\
\mu_{0_\beta} &\sim \text{Normal}(0, 2) \\
\sigma_{0_\beta} &\sim \text{Inv-}\chi^2(0.05) \\
\alpha &\sim \text{Normal}(\mu_{0_\alpha}, \sigma_{0_\alpha}) \\
\beta &\sim \text{Normal}(\mu_{0_\beta}, \sigma_{0_\beta}) \\
\sigma &\sim \text{Normal}(0, 10)
\end{aligned}
$$

```
hier_data2 <- list(J=3,
                    N1=length(data1$Active),
                    N2=length(data2$Active),
                    N3=length(data3$Active),
                    K=length(data_clean)-1,
                    x1=data1[2:length(data1)],
                    x2=data2[2:length(data2)],
                    x3=data3[2:length(data3)],
                    y1=data1[['Active']],
                    y2=data2[['Active']],
                    y3=data3[['Active']],
                    prior=2) # 1 for default, 2 for sigma0 change, 3 for sigma0 change,
                            # 4 for mu0 change, 5 for mu0 change


hier_fit2 <- stan(file = "hier_N2.stan",
                  data = hier_data2,
                  seed = SEED,
                  control = list(adapt_delta = 0.99, max_treedepth = 15),
                  warmup = 2000,
                  iter = 4000)

samples_hier2 <- extract(hier_fit2)
```

- Hier3 hyper-prior:

$$
\begin{aligned}
\mu_{0_\alpha} &\sim \text{Normal}(110, 30)\\
\sigma_{0_\alpha} &\sim \text{Normal}(0, 1000)\\
\mu_{0_\beta} &\sim \text{Normal}(0, 2)\\
\sigma_{0_\beta} &\sim \text{Inv-}\chi^2(10)\\
\alpha &\sim \text{Normal}(\mu_{0_\alpha}, \sigma_{0_\alpha})\\
\beta &\sim \text{Normal}(\mu_{0_\beta}, \sigma_{0_\beta})\\
\sigma &\sim \text{Normal}(0, 10)
\end{aligned}
$$

```r
hier_data3 <- list(J=3,
                   N1=length(data1$Active),
                   N2=length(data2$Active),
                   N3=length(data3$Active),
                   K=length(data_clean)-1,
                   x1=data1[2:length(data1)],
                   x2=data2[2:length(data2)],
                   x3=data3[2:length(data3)],
                   y1=data1[['Active']],
                   y2=data2[['Active']],
                   y3=data3[['Active']],
                   prior=3) # 1 for default, 2 for sigma0 change, 3 for sigma0 change,
                            # 4 for mu0 change, 5 for mu0 change


hier_fit3 <- stan(file = "hier_N2.stan",
                   data = hier_data3,
                   seed = SEED,
                   control = list(adapt_delta = 0.99, max_treedepth = 15),
                   warmup = 2000,
                   iter = 4000)

samples_hier3 <- extract(hier_fit3)
```

Then, we tried different hyper-priors of α and β (μ_{0_α} and μ_{0_β}) in hier4 and hier5.

- Hier4 hyper-prior:

$$
\begin{aligned}
\mu_{0_\alpha} &\sim \text{Normal}(0, 100)\\
\sigma_{0_\alpha} &\sim \text{Normal}(0, 100)\\
\mu_{0_\beta} &\sim \text{Normal}(0, 20)\\
\sigma_{0_\beta} &\sim \text{Inv-}\chi^2(1)\\
\alpha &\sim \text{Normal}(\mu_{0_\alpha}, \sigma_{0_\alpha})\\
\beta &\sim \text{Normal}(\mu_{0_\beta}, \sigma_{0_\beta})\\
\sigma &\sim \text{Normal}(0, 10)
\end{aligned}
$$

```r
hier_data4 <- list(J=3,
                  N1=length(data1$Active),
                  N2=length(data2$Active),
                  N3=length(data3$Active),
                  K=length(data_clean)-1,
                  x1=data1[2:length(data1)],
                  x2=data2[2:length(data2)],
                  x3=data3[2:length(data3)],
                  y1=data1[['Active']],
                  y2=data2[['Active']],
                  y3=data3[['Active']],
                  prior=4) # 1 for default, 2 for sigma0 change, 3 for sigma0 change,
                           # 4 for mu0 change, 5 for mu0 change


hier_fit4 <- stan(file = "hier_N2.stan",
                  data = hier_data4,
                  seed = SEED,
                  control = list(adapt_delta = 0.99, max_treedepth = 15),
                  warmup = 2000,
                  iter = 4000)

samples_hier4 <- extract(hier_fit4)
```

- Hier5 hyper-prior:

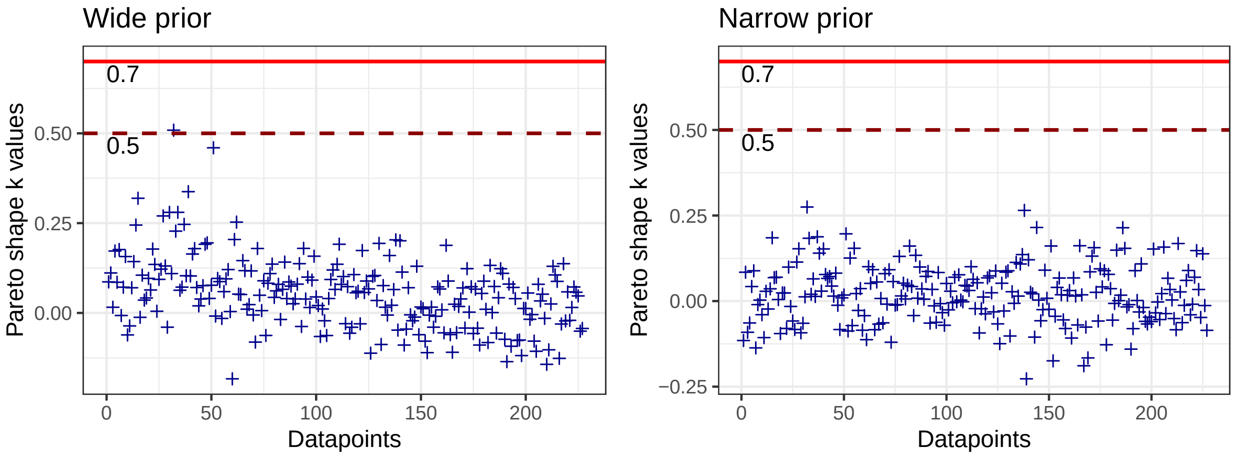
$$
\begin{aligned}
\mu_{0_\alpha} &\sim \text{Normal}(200, 500)\\
\sigma_{0_\alpha} &\sim \text{Normal}(0, 100)\\
\mu_{0_\beta} &\sim \text{Normal}(20, 50)\\
\sigma_{0_\beta} &\sim \text{Inv-}\chi^2(1)\\
\alpha &\sim \text{Normal}(\mu_{0_\alpha}, \sigma_{0_\alpha})\\
\beta &\sim \text{Normal}(\mu_{0_\beta}, \sigma_{0_\beta})\\
\sigma &\sim \text{Normal}(0, 10)
\end{aligned}
$$

```r
hier_data5 <- list(J=3,
                   N1=length(data1$Active),
                   N2=length(data2$Active),
                   N3=length(data3$Active),
                   K=length(data_clean)-1,
                   x1=data1[2:length(data1)],
                   x2=data2[2:length(data2)],
                   x3=data3[2:length(data3)],
                   y1=data1[['Active']],
                   y2=data2[['Active']],
                   y3=data3[['Active']],
                   prior=5) # 1 for default, 2 for sigma0 change, 3 for sigma0 change,
                            # 4 for mu0 change, 5 for mu0 change

hier_fit5 <- stan(file = "hier_N2.stan",
                  data = hier_data5,
                  seed = SEED,
                  control = list(adapt_delta = 0.99, max_treedepth = 15),
                  warmup = 2000,
                  iter = 4000)

samples_hier5 <- extract(hier_fit5)
```

10.1. LOO-CV

10.1.1. Separate model

```
wide_log_lik <- loo::extract_log_lik(separate_fit_wide, merge_chains = FALSE)
wide_r_eff <- loo::relative_eff(exp(wide_log_lik))
wide_loo <- loo::loo(wide_log_lik, r_eff = wide_r_eff)

narrow_log_lik <- loo::extract_log_lik(separate_fit_narrow, merge_chains = FALSE)
narrow_r_eff <- loo::relative_eff(exp(narrow_log_lik))
narrow_loo <- loo::loo(narrow_log_lik, r_eff = narrow_r_eff)
```

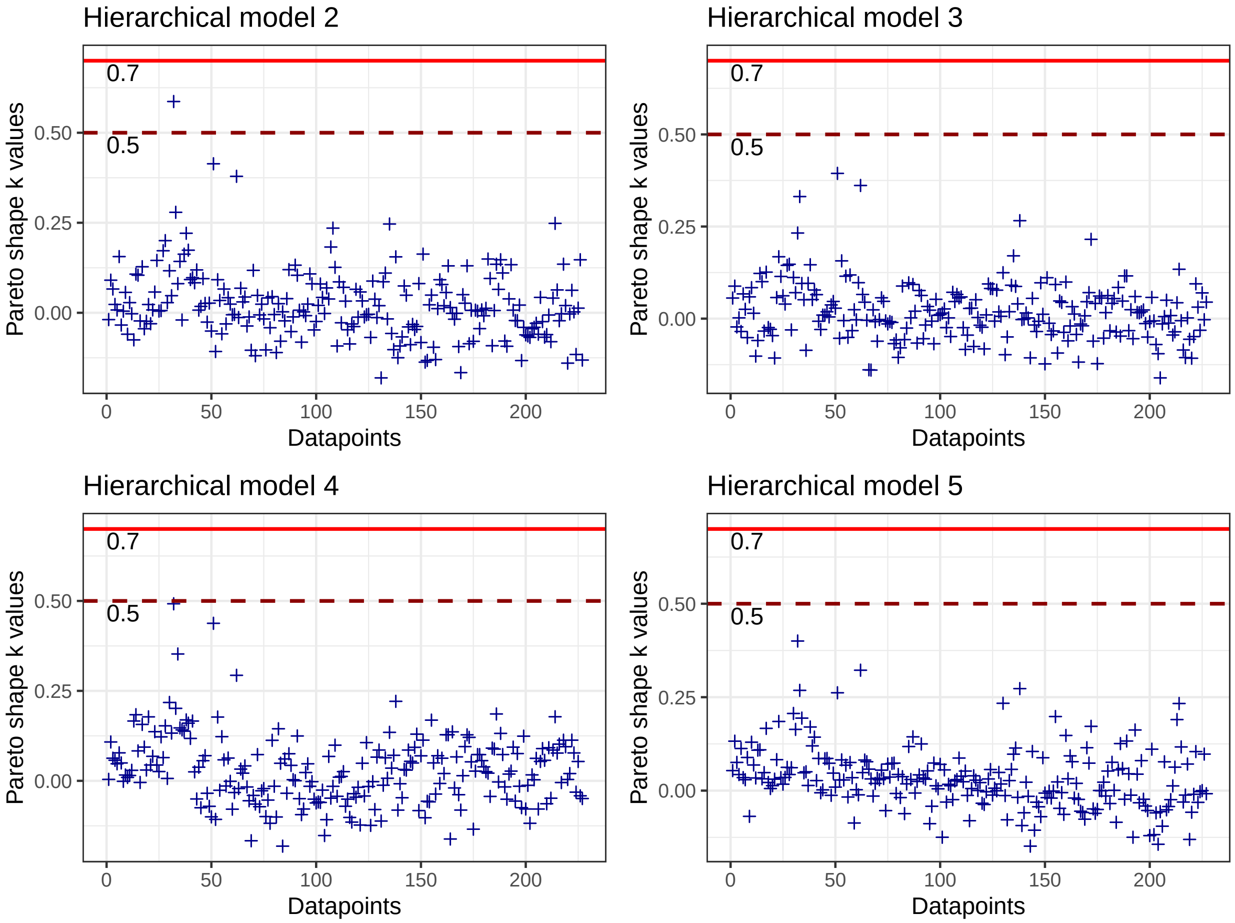


For both the wide and the narrow prior, all k values are below 0.7, which indicates the models are reliable.

10.1.2. Hierarchical model

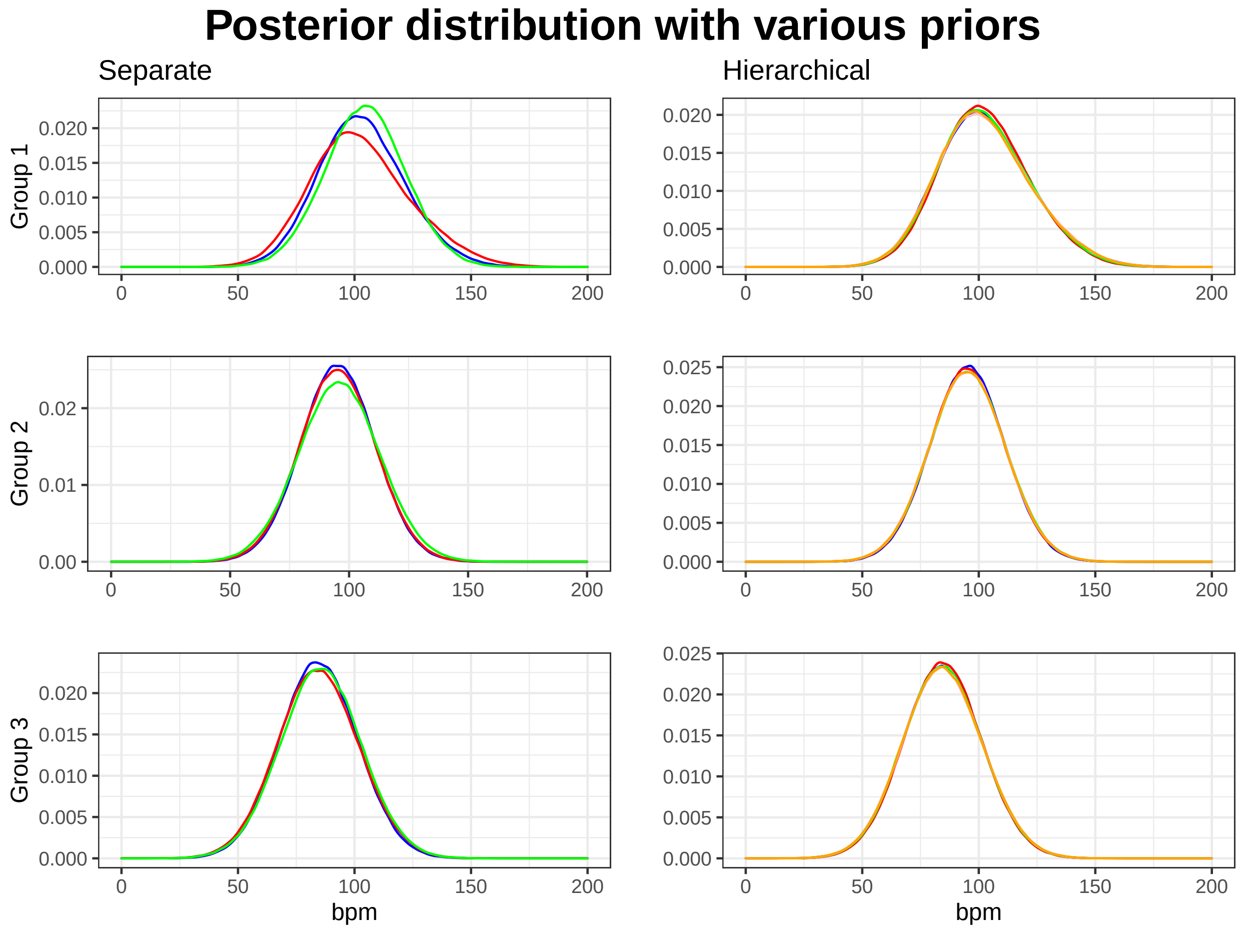
```
hierarchical2_log_lik <- loo::extract_log_lik(hier_fit2, merge_chains = FALSE)
hierarchical2_r_eff <- loo::relative_eff(exp(hierarchical2_log_lik))
hierarchical2_loo <- loo::loo(hierarchical2_log_lik, r_eff = hierarchical2_r_eff)


hierarchical3_log_lik <- loo::extract_log_lik(hier_fit3, merge_chains = FALSE)
hierarchical3_r_eff <- loo::relative_eff(exp(hierarchical3_log_lik))
hierarchical3_loo <- loo::loo(hierarchical3_log_lik, r_eff = hierarchical3_r_eff)
```

```r
hierarchical4_log_lik <- loo::extract_log_lik(hier_fit4, merge_chains = FALSE)
hierarchical4_r_eff <- loo::relative_eff(exp(hierarchical4_log_lik))
hierarchical4_loo <- loo::loo(hierarchical4_log_lik, r_eff = hierarchical4_r_eff)


hierarchical5_log_lik <- loo::extract_log_lik(hier_fit5, merge_chains = FALSE)
hierarchical5_r_eff <- loo::relative_eff(exp(hierarchical5_log_lik))
hierarchical5_loo <- loo::loo(hierarchical5_log_lik, r_eff = hierarchical5_r_eff)
```



For all the different priors, all k values are below 0.7, which indicates the models are reliable.

10.2. Posterior distributions



From the plots for the separate model we can see that for groups 2 and 3 there is not much difference in the posterior even with very different priors. Thus, they are not sensitive to the prior distribution. Group 1 seems to be the one most sensitive to the prior distribution, even though the differences in the posterior are not very large.

The hierarchical model is more robust and we can see almost no change in the resulting posteriors, which means that they are not sensitive to the proposed hyper-priors.

11. Discussion of issues and potential improvements

The biggest issue that we have is the convergence of the hierarchical model. Even with our best efforts, there are still some divergent iterations. This might be due to the “weird” shape of some of the underlying distributions, but potentially also due to the way we are fitting the Stan model. Improvements for future work would be to look deeper into this issue and try to improve the convergence.

Furthermore, the analysis showed that while our models give reasonable results, they do not capture the underlying distributions perfectly, as can be seen in the posterior predictive check, so there is room for improvement. Potential tweaks could be using different priors that might work better for our data, using non-linear models to capture non-linear dependencies between the variables, etc.

Also, for the separate model we noticed sensitivity to the prior distribution for one of the groups, although it was not major. In any case, it is an issue that should be looked into in more detail.

12. Conclusion

After conducting all the analyses, the conclusion is that the hierarchical model performs better, with the acknowledgement that the chains do not fully converge. It has a higher elpd value and it is less sensitive to the priors. Therefore, the hierarchical model will be the one used for the following analysis.

If we take the means of the parameters given in the hierarchical model, we obtain the following equations for each group:

- Group 1 (one hour of exercise per week):

$$y\_rep1 \sim -9.59 + 1.17 \cdot \text{rest} + 0.98 \cdot \text{smoke} + 0.99 \cdot \text{sex} + 0.95 \cdot \text{exercise} + 0.9 \cdot \text{BMI}$$

- Group 2 (two hours of exercise per week):

$$y\_rep2 \sim 15 + 0.85 \cdot \text{rest} + 0.88 \cdot \text{smoke} + 1.03 \cdot \text{sex} + 0.93 \cdot \text{exercise} + 0.75 \cdot \text{BMI}$$

- Group 3 (three hours of exercise per week):

$$y\_rep3 \sim -5.85 + 1.08 \cdot \text{rest} + 0.95 \cdot \text{smoke} + 0.96 \cdot \text{sex} + 0.93 \cdot \text{exercise} + 0.79 \cdot \text{BMI}$$

Taking a look at the coefficients, we can draw some conclusions about the groups.

People who exercise only one hour per week (group 1) tend to have a higher active hear rate. It might be because the coefficient values of 'Rest' and 'BMI' are significantly higher than those of group 2 and 3. The 'Rest' and 'BMI' values are also high compared to other variables with an average of 'Rest' being 68, and average of 'BMI' 24, while other variables are 0, 1, 2, or 3. Although the value of α in y_rep1 is more negative, the products of 'BMI' and 'Rest' have larger impact, which leads to the higher active rate of group 1.

Furthermore, we can notice that the active heart rate is expected to increase much more in group 1 with the increase of BMI than in groups 2 and 3. We can interpret this in the way that if people who exercise more gain weight, their overall physical fitness level suffers less than people who are in worse shape.

Lastly, we can notice that if one is a smoker, the active heart rate is higher than for non-smokers, however the effect in people who exercise only one hour per week is worse than in people who exercise more.

13. Self-reflection

This project gave us an idea what it would be like to do Bayesian data analysis with all the tools that we have learned so far during the course. While working on the project, we found several gaps in our knowledge, so we had to go back and revisit the lectures and course materials. However, we learned a lot and got the experience of working with a bit more complex dataset than the ones used in the assignment. The dataset was unique and not used before in Bayesian analysis, so it was not as easy to find the right way of doing things and compare the results with someone else. In practice, this meant that it took longer than usual to get our models and priors right. After that, the project slowly started to unfold. The evaluation part was interesting and felt easier to do. The only difficult part was to make sure that our answers were correct or at least close to it. After all, the results don't necessarily need to be perfect, since we used only two models, but rather directional so that we could draw conclusions from them.

Appendix

A. Stan code

A.1. Separate model

```
data {
  int<lower=0> J; //number of groups
  int<lower=0> N1; //number of observations for group 1
  int<lower=0> N2; //number of observations for group 2
  int<lower=0> N3; //number of observations for group 3
  int<lower=1> K; //number of features

  matrix[N1, K] x1;
  matrix[N2, K] x2;
  matrix[N3, K] x3;

  vector[N1] y1;
  vector[N2] y2;
  vector[N3] y3;

  int prior;
}

parameters {
  real alpha1;
  vector[K] beta1;

  real alpha2;
  vector[K] beta2;

  real alpha3;
  vector[K] beta3;

  real<lower=0> sigma;
}

model {
  // default priors
  if (prior == 1) {
    alpha1 ~ normal(110, 30);
    beta1 ~ normal(0, 1);

    alpha2 ~ normal(110, 30);
    beta2 ~ normal(0, 1);

    alpha3 ~ normal(110, 30);
    beta3 ~ normal(0, 1);

    sigma ~ normal(0, 10);
  }

  // wide priors
  else if (prior == 2) {
    alpha1 ~ normal(0, 100);
    beta1 ~ normal(0, 2);

    alpha2 ~ normal(0, 100);
    beta2 ~ normal(0, 2);

    alpha3 ~ normal(0, 100);
    beta3 ~ normal(0, 2);

    sigma ~ normal(0, 30);
  }

  // narrow priors
  else if (prior == 3) {
    alpha1 ~ normal(110, 5);
    beta1 ~ normal(0, 0.5);

    alpha2 ~ normal(110, 5);
    beta2 ~ normal(0, 0.5);

    alpha3 ~ normal(110, 5);
    beta3 ~ normal(0, 0.5);

    sigma ~ normal(0, 5);
  }

  //likelihood
  y1 ~ normal(alpha1 + x1*beta1, sigma);
  y2 ~ normal(alpha2 + x2*beta2, sigma);
  y3 ~ normal(alpha3 + x3*beta3, sigma);
}

generated quantities {
  //posterior predictive distribution for posterior predictive check
  real y1_rep[N1] = normal_rng(alpha1 + x1*beta1, sigma);
  real y2_rep[N2] = normal_rng(alpha2 + x2*beta2, sigma);
  real y3_rep[N3] = normal_rng(alpha3 + x3*beta3, sigma);

  //log-likelihood
  vector[N1+N2+N3] log_lik;

  for (i in 1:N1) {
    log_lik[i] = normal_lpdf(y1[i] | x1[i] * beta1 + alpha1, sigma);
  }

  for (i in 1:N2) {
    log_lik[i+N1] = normal_lpdf(y2[i] | x2[i] * beta2 + alpha2, sigma);
  }

  for (i in 1:N3) {
    log_lik[i+N1+N2] = normal_lpdf(y3[i] | x3[i] * beta3 + alpha3, sigma);
  }
}
```

A.2. Hierarchical model

```
data {
  int<lower=0> J; //number of groups
  int<lower=0> N1; //number of observations for group 1
  int<lower=0> N2; //number of observations for group 2
  int<lower=0> N3; //number of observations for group 3
  int<lower=1> K; //number of features

  matrix[N1, K] x1;
  matrix[N2, K] x2;
  matrix[N3, K] x3;

  vector[N1] y1;
  vector[N2] y2;
  vector[N3] y3;

  int prior;
}


parameters {

  real mu0_alpha;
  real<lower=0> sigma0_alpha;
  real mu0_beta;
  real<lower=0> sigma0_beta;

  real alpha1;
  vector[K] beta1;

  real alpha2;
  vector[K] beta2;

  real alpha3;
  vector[K] beta3;

  real<lower=0> sigma;
}

model {
  // default priors
  if (prior == 1) {
    mu0_alpha ~ normal(110, 30);
    sigma0_alpha ~ normal(0, 100);
    mu0_beta ~ normal(0, 2);
    sigma0_beta ~ inv_chi_square(1);
  }
     // change sigma0
  else if (prior == 2) {
    mu0_alpha ~ normal(110, 30);
    sigma0_alpha ~ normal(0, 10);
    mu0_beta ~ normal(0, 2);
    sigma0_beta ~ inv_chi_square(0.05);
  }
  // change sigma0
  else if (prior == 3) {
    mu0_alpha ~ normal(110, 30);
    sigma0_alpha ~ normal(0, 1000);
    mu0_beta ~ normal(0, 2);
    sigma0_beta ~ inv_chi_square(10);
  }
  // change mu0
  else if (prior == 4) {
    mu0_alpha ~ normal(0, 100);
    sigma0_alpha ~ normal(0, 100);
    mu0_beta ~ normal(0, 20);
    sigma0_beta ~ inv_chi_square(1);
  }
  // change mu0
  else if (prior == 5) {
    mu0_alpha ~ normal(200, 500);
    sigma0_alpha ~ normal(0, 100);
    mu0_beta ~ normal(20, 50);
    sigma0_beta ~ inv_chi_square(1);
  }

  alpha1 ~ normal(mu0_alpha, sigma0_alpha);
  beta1 ~ normal(mu0_beta, sigma0_beta);

  alpha2 ~ normal(mu0_alpha, sigma0_alpha);
  beta2 ~ normal(mu0_beta, sigma0_beta);

  alpha3 ~ normal(mu0_alpha, sigma0_alpha);
  beta3 ~ normal(mu0_beta, sigma0_beta);

  sigma ~ normal(0, 10);

  //likelihood
  y1 ~ normal(alpha1 + x1*beta1, sigma);
  y2 ~ normal(alpha2 + x2*beta2, sigma);
  y3 ~ normal(alpha3 + x3*beta3, sigma);
}

generated quantities {
  //posterior predictive distribution for posterior predictive check
  real y1_rep[N1] = normal_rng(alpha1 + x1*beta1, sigma);
  real y2_rep[N2] = normal_rng(alpha2 + x2*beta2, sigma);
  real y3_rep[N3] = normal_rng(alpha3 + x3*beta3, sigma);

  //log-likelihood
  vector[N1+N2+N3] log_lik;

  for (i in 1:N1) {
    log_lik[i] = normal_lpdf(y1[i] | x1[i] * beta1 + alpha1, sigma);
  }

  for (i in 1:N2) {
    log_lik[i+N1] = normal_lpdf(y2[i] | x2[i] * beta2 + alpha2, sigma);
  }

  for (i in 1:N3) {
    log_lik[i+N1+N2] = normal_lpdf(y3[i] | x3[i] * beta3 + alpha3, sigma);
  }
}
```

B. Loaded packages

```
SEED = 42
library(aaltobda)
library(rstan)
library(dplyr)
library(ggplot2)
library(loo)
library(bayesplot)
library(gridExtra)
library(grid)
```

References

Arel-Bundock, Vincent. n.d. "R Datasets." Accessed December 3, 2021. https://vincentarelbundock.github.io/Rdatasets/datasets.html.

CDC. 2019. "Calculating BMI Using the English System." https://www.cdc.gov/nccdphp/dnpao/growthcharts/training/bmiage/page5_2.html.

Harvard. 2015. "What Your Heart Rate Is Telling You." *Harvard Health*. https://www.health.harvard.edu/heart-health/what-your-heart-rate-is-telling-you.

NHLBI. n.d. "Assessing Your Weight and Health Risk." Accessed December 3, 2021. https://www.nhlbi.nih.gov/health/educational/lose_wt/risk.htm.