

Heart Failure Prediction using Bayesian Approach

Teemu Sormunen, Abdullah Günay, Nicola Brazzale

11/18/2020

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 3 |
| 1.1 | The problem | 3 |
| 1.2 | The motivation | 3 |
| 1.3 | Modeling idea | 3 |
| 2 | Dataset | 4 |
| 2.1 | Term explanation | 4 |
| 2.2 | Dataset introduction | 4 |
| 3 | Packages | 6 |
| 4 | Data preprocessing and visualization | 7 |
| 4.1 | Plot histograms | 7 |
| 4.2 | Correlation matrix | 12 |
| 5 | Models | 14 |
| 5.1 | Model fitting | 15 |
| 5.2 | Prior choices | 18 |
| 5.3 | \hat{R} convergence | 19 |
| 5.4 | HMC specific convergence diagnostics (divergences, tree depth) with interpretation of the results | 23 |
| 5.5 | Effective sample size diagnostic (<code>n_eff</code> or ESS) | 24 |
| 5.6 | Posterior predictive checking | 28 |
| 5.7 | Model comparison and interpretation of the results | 33 |
| 5.8 | Predictive performance assessment (classification) | 37 |
| 5.9 | Prior sensitivity analysis (alternative prior tested) | 38 |
| 6 | Conclusion - Discussion problems and further improvements | 40 |
| 6.1 | Self-reflection | 40 |

| | |
|---|-----------|
| Appendix | 41 |
| A. Stan code of full model | 41 |
| B. Stan feature selected model | 43 |
| C. Stan hierarchical model | 44 |
| 6.2 D. Stan death time analysis model | 47 |

1 Introduction

In this project we analyse the data from Heart disease dataset [2]. We perform a Bayesian analysis of the data using this sequence of operations:

- Overview of analysis problem and of the dataset
- Data preprocessing and visualisation
- Prior choice discussion
- Models used in our analysis
- \hat{R} convergence
- HMC specific convergence diagnostics
- Effective sample size diagnostic (n_{eff} or ESS)
- Model comparison
- Prior sensitivity analysis
- Discussion and conclusion

In our study, we are observing the dataset from two distinct point of views. The first approach was based on the simple predictive analysis whether patient dies, or survives the research follow up period (4 - 285 days), and on the other hand we want to create an actual survival analysis, which takes time and censored variables in to consideration. Different models are used to carry out this analysis and a model comparison is performed in order to asses the quality of the outcome. More informations in upcoming chapters.

1.1 The problem

Cardiovascular Heart Disease (CHD) is the top reason causing 31% of deaths globally. Pakistan is one of the countries where CHD is increasing significantly, and previous studies do not directly apply to Pakistani area due to different diet patterns. [2]

1.2 The motivation

With this project we aim to estimate death events and the major risk factors for heart failure with, possibly, high accuracy [2].

1.3 Modeling idea

We created 4 models which are then compared based on \hat{r} , n_{eff} , using the loo package and the classification accuracy. The three first models ignore the time feature, and simply predict whether patient has died (1) during the experiment duration, or survived (0). We chose this approach to practise survival analysis with binary outcome.

We also created fourth model, which predicts the time with respect to death event. In this case, the death event 1 means the patient has died, and death event 0 means that the patient is censored from the study. Censoring practically means, that the patient has opted out of the study, and the researches couldn't reach him anymore. In this context, it doesn't necessarily mean that the patient has survived, but we just don't know the outcome.

The 1st model is the reduced model and consists in fewer variables which are selected base on their correlation with the death event. The 2nd model consists in all variables except for the variable "time" as we believe that doesn't represent an important factor in the death event scenario. The 3rd model used is a hierarchical model where we treated age class patients in a group with respect to the other selected variables.

The 4th model is similar type of linear model, but this time we consider time as outcome variable with respect to death event. We use correlation matrix to select the strongest variable that correspond with

time, and we also use BRMS internal function cens() for taking the censored variable DEATH_EVENT to consideration. More clear explanation is given later.

2 Dataset

2.1 Term explanation

Some of the terms in the dataset might not be familiar, and they are opened briefly here.

- **Creatine phosphokinase (CPK)**

CPK is an enzyme, which helps to regulate the concentration of adenosine triphosphate (ATP) in cells. ATP is responsible for carrying energy. If the CPK level is high, it often means that there has been an injury or stress on a muscle tissue. Although CPK is one the oldest markers of heart attack, high CPK might also indicate of acute muscle injury along with acute heart problems.

Normal level of CPK ranges from 20 to 200 IU/L [6]

- **Ejection fraction (EF)**

EF is a measurement in percentage which describes how much blood left ventricle pumps out of heart with each contraction. Low EF might indicate potential heart issues.

Normal EF is 50 to 70 percent, while measurement under 40 percent might be an indicator of heart failure or cardiomyopathy. [1]

- **Platelets**

Platelets are small cell fragments which can form clots. Too many platelets can lead to clotting of blood vessels, which in turn can lead to heart attack. Too Normal range of platelets is from 150 000 to 450 000. [5]

- **Serum creatinine**

When creatine breaks down, it forms a waste product called creatinine. Kidneys normally remove creatinine from body. Serum creatinine measures level of creatinine in the blood, indicating the kidney health. High levels of creatinine might indicate a kidney dysfunction.

Normal level of creatinine range from 0.9 to 1.3 mg/dL in men and 0.6 to 1.1 mg/dL in women who are 18 to 60 years old. [7]

- **Serum sodium**

Serum sodium measures the amount of sodium in blood. Sodium enters blood through food and drink, and leaves by urine, stool and sweat. Too much sodium can cause blood pressure, while too little sodium can cause nausea, vomiting, exhaustion or dizziness.

Normal levels of serum sodium are 135 to 145 mEq/L, according to Mayo Clinic. There are however different interpretations of “normal”. [4]

- **Time** Time variable indicates the time since the research has started for that person (the time of ventricular systolic dysfunction). We have time variable included, because we have to inspect when the death events are happening. This variable is ignored in the first three models, because we wanted to also interpret this dataset from binary survival approach, so predict whether patient dies or not.

2.2 Dataset introduction

The dataset of 299 patients was produced as a result of study [2] from Pakistani’s city Faisalabad. All of the patients were over 40 years old, each having ventricular systolic dysfunction. This means that patient has poor left ventricular ejection fraction. The follow up period was 4 to 285 days, with average of 130 days. This has to be taken in to consideration when doing the survival analysis.

The dataset has 105 women, and 194 men. Some features such as: Ejection fraction, serum creatinine

and platelets are categorical variables, while age, serum sodium and creatine phosphokinase are continuous variables.

Statistical analysis by [2] found age, creatinine, sodium, anemia and BP as significant variables.

3 Packages

Load data

```
file.name <- './data/heart_failure_clinical_records_dataset.csv'  
heart <- read_csv(file.name)
```

```
## Parsed with column specification:  
## cols(  
##   age = col_double(),  
##   anaemia = col_double(),  
##   creatinine_phosphokinase = col_double(),  
##   diabetes = col_double(),  
##   ejection_fraction = col_double(),  
##   high_blood_pressure = col_double(),  
##   platelets = col_double(),  
##   serum_creatinine = col_double(),  
##   serum_sodium = col_double(),  
##   sex = col_double(),  
##   smoking = col_double(),  
##   time = col_double(),  
##   DEATH_EVENT = col_double()  
## )
```

Prevent text overflow on PDF

```
library(knitr)  
opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)  
rstan_options(auto_write=TRUE) # Save stan models automatically
```

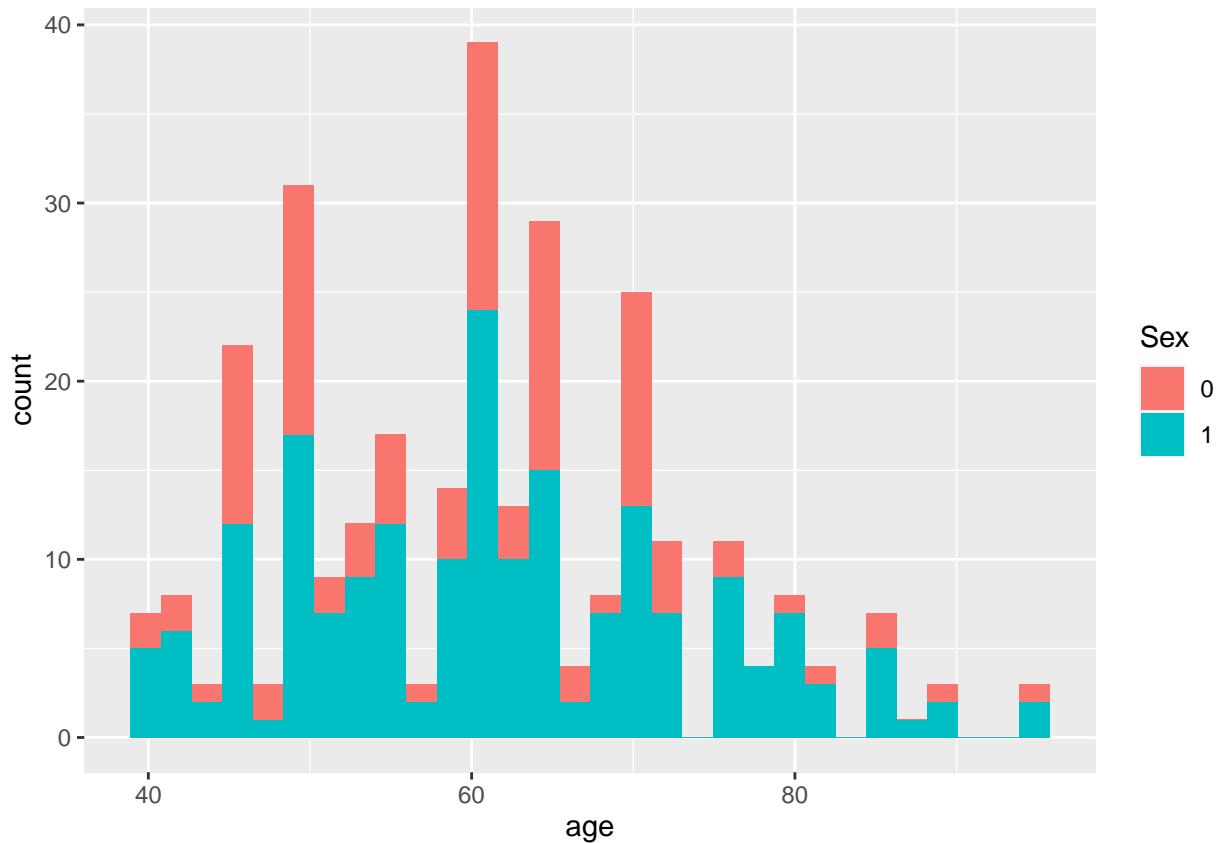
4 Data preprocessing and visualization

4.1 Plot histograms

We are first plotting the histograms to get an overview of the dataset.

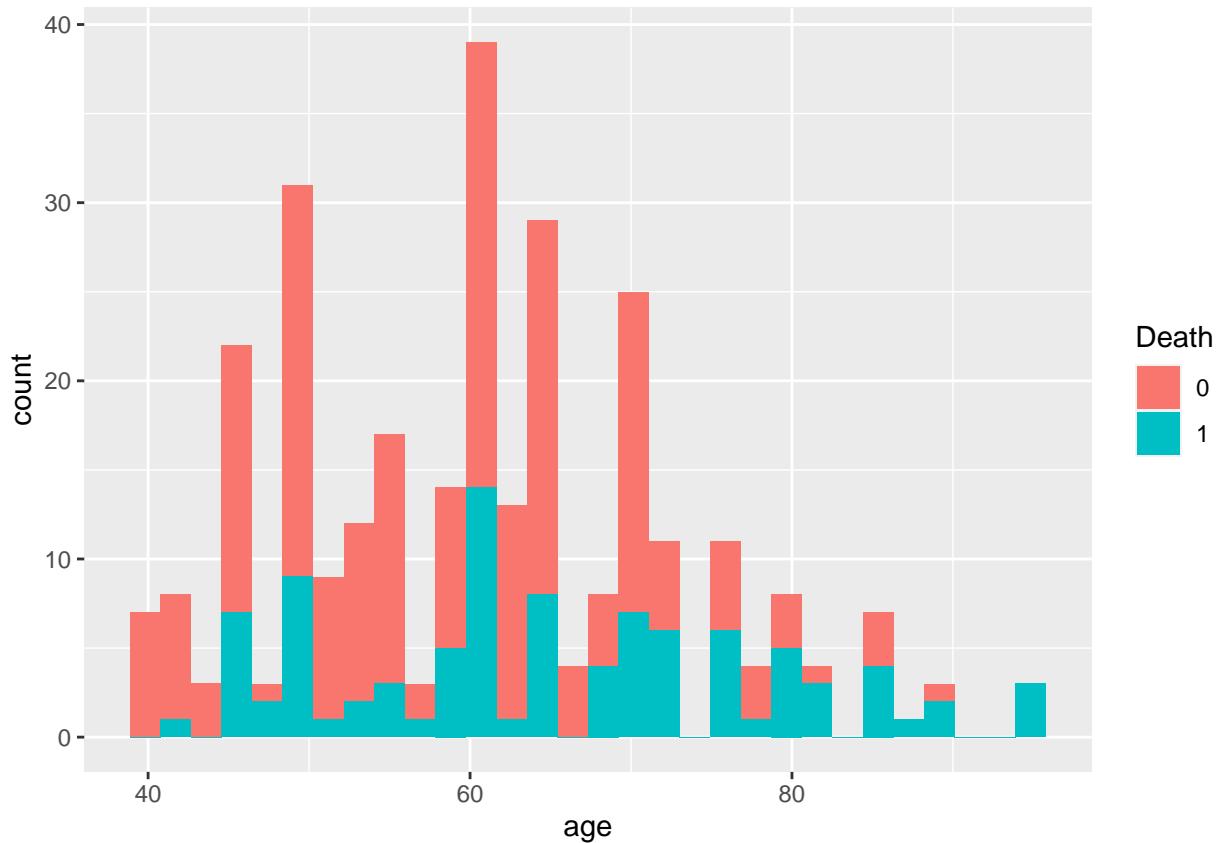
It seems like the sex between ages is distributed quite evenly, there's slightly more patients from the 50-60.

```
ggplot(heart, aes(x = age)) + geom_histogram(aes(fill = as.character(sex)),  
    bins = 30) + labs(fill = "Sex")
```



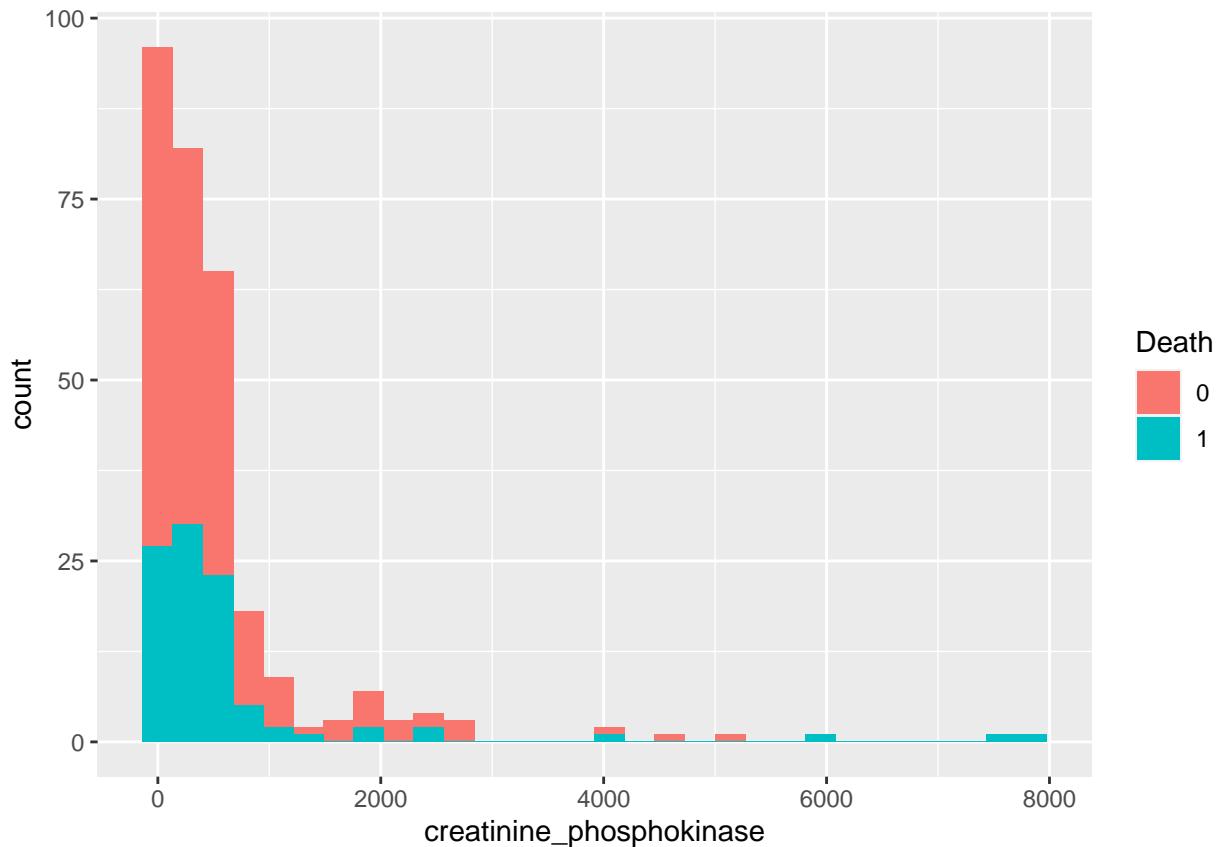
This histogram might suggest us that older people die during this follow up period with higher probability, and younger people either survive or opt-out of the study.

```
ggplot(heart, aes(x = age)) + geom_histogram(aes(fill = as.character(DEATH_EVENT)),  
bins = 30) + labs(fill = "Death")
```



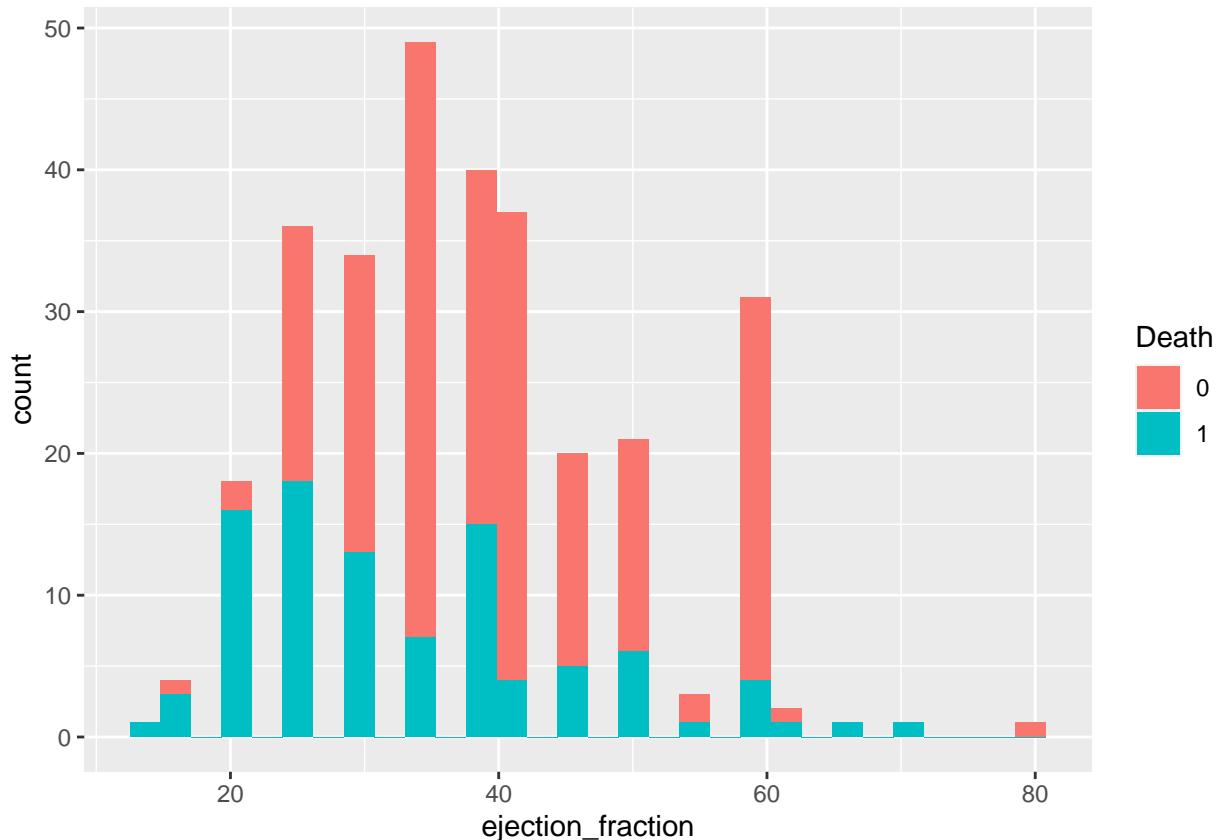
Creatinine phosphokinase doesn't bring too much information based on the histogram, although it shows us that people with high creatinine phosphokinase might tend to die more often. Because almost everyone who attended this study had already increased phosphokinase levels, we should take this interpretation with a slight grain of salt.

```
ggplot(heart, aes(x = creatinine_phosphokinase)) + geom_histogram(aes(fill = as.character(DEATH_EVENT)), bins = 30) + labs(fill = "Death")
```



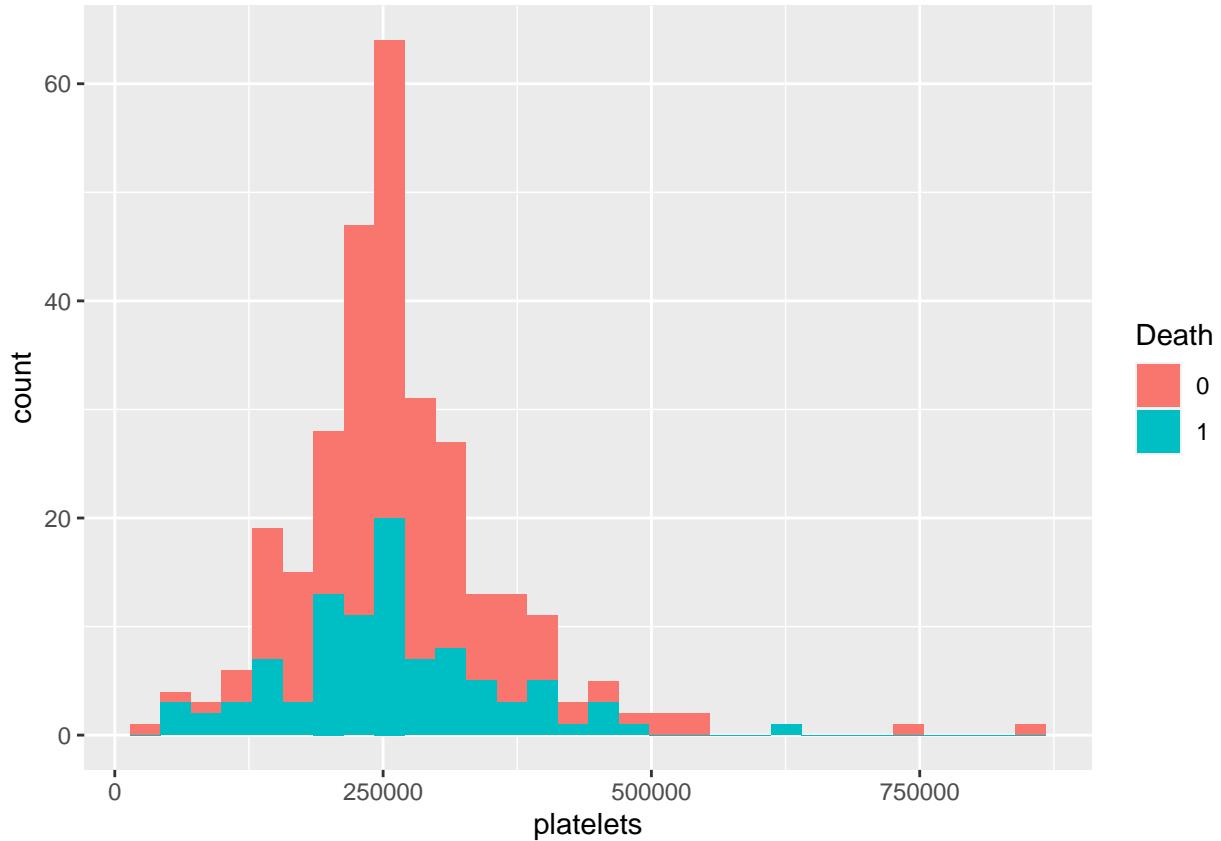
Ejection fraction seems to correlate strongly with death. This is logical, because ejection fraction measures the hearts ability to pump blood. Here we can again see that most of the people fall under normal levels of EF.

```
ggplot(heart, aes(x = ejection_fraction)) + geom_histogram(aes(fill = as.character(DEATH_EVENT)),  
bins = 30) + labs(fill = "Death")
```



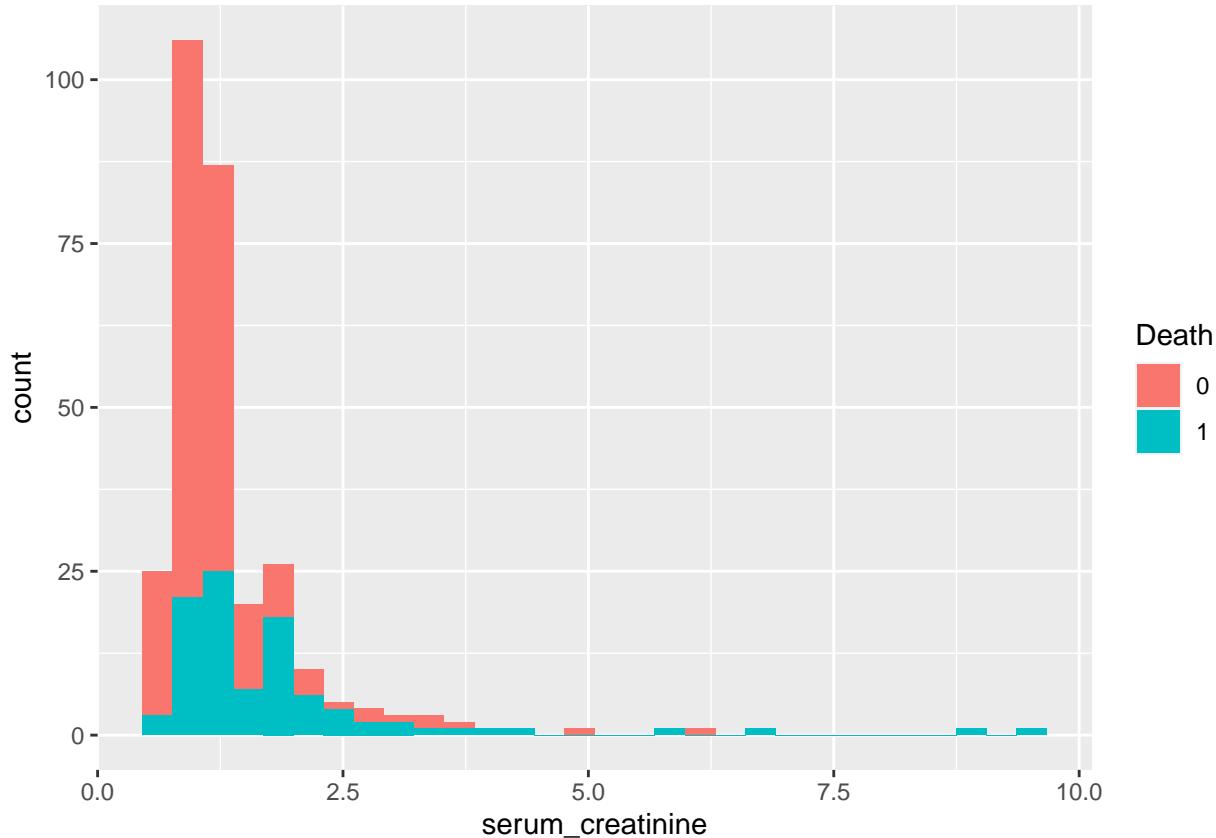
Platelets seems to follow quite even distribution with respect to death event, and there isn't too much information available only based on the histogram.

```
ggplot(heart, aes(x = platelets)) + geom_histogram(aes(fill = as.character(DEATH_EVENT)), bins = 30) + labs(fill = "Death")
```



Serum creatinine seems to correlate quite highly with death. When the upper bound for serum creatinine 1.3 is passed, it seems that probability for death becomes high.

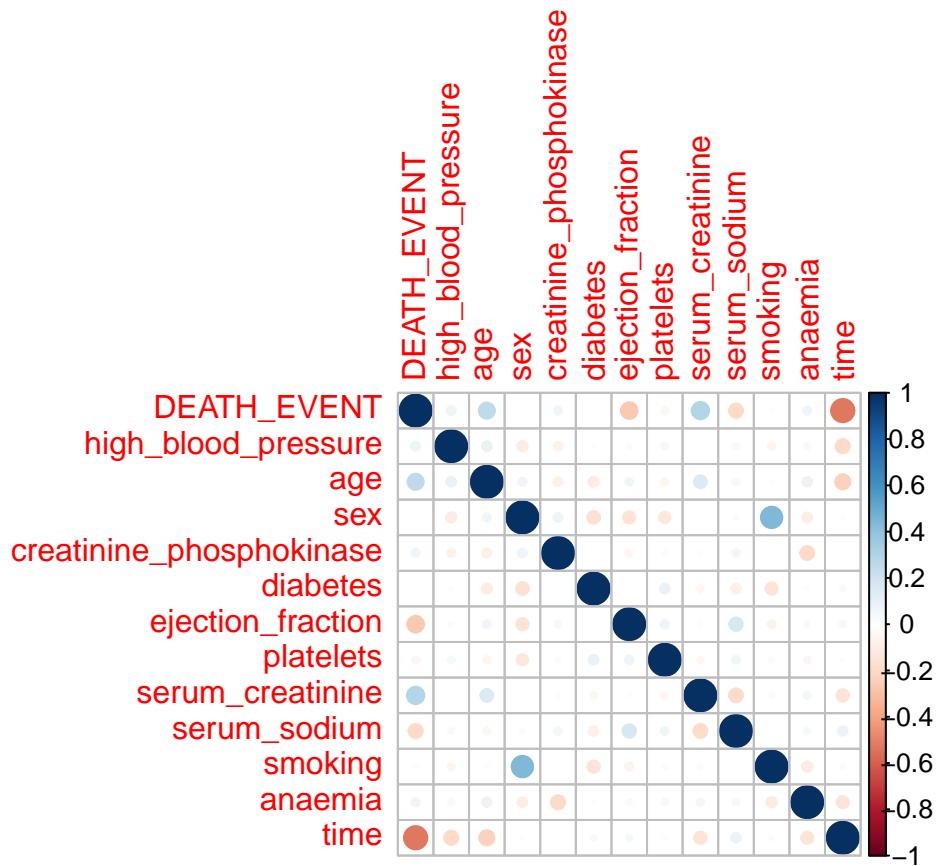
```
ggplot(heart, aes(x = serum_creatinine)) + geom_histogram(aes(fill = as.character(DEATH_EVENT)), bins = 30) + labs(fill = "Death")
```



4.2 Correlation matrix

By plotting the correlation matrix we can see the correlations.

```
pred <- c("high_blood_pressure", "age", "sex", "creatinine_phosphokinase",
         "diabetes", "ejection_fraction", "platelets", "serum_creatinine",
         "serum_sodium", "smoking", "anaemia", "time")
target <- c("DEATH_EVENT")
# formula <- paste('DEATH_EVENT ~', paste(pred, collapse =
# '+'))
p <- length(pred)
n <- nrow(heart)
x = cor(heart[, c(target, pred)])
corrplot(x)
```



Looking at the correlations only by eye might not be enough, so let's list the correlations with respect to death_event in sorted order.

```
sort(x[1, ])
```

| | | | |
|----|---------------------|--------------------------|------------------|
| ## | time | ejection_fraction | serum_sodium |
| ## | -0.526963779 | -0.268603312 | -0.195203596 |
| ## | platelets | smoking | sex |
| ## | -0.049138868 | -0.012623153 | -0.004316376 |
| ## | diabetes | creatinine_phosphokinase | anaemia |
| ## | -0.001942883 | 0.062728160 | 0.066270098 |
| ## | high_blood_pressure | age | serum_creatinine |
| ## | 0.079351058 | 0.253728543 | 0.294277561 |
| ## | DEATH_EVENT | | |
| ## | 1.000000000 | | |

We see that highly positively correlating variables are age and serum creatinine, which found to be true already earlier. Strongest negative correlations are time, ejection fraction and serum sodium. Time clearly indicates that people tend to die early after the ventricular systolic dysfunction has happened. Serum sodium also seems to correlate negatively, because the low sodium levels are usually occurring after heart failure. If we have to make conclusions, then we could conclude that the sodium levels are lower in the people who have had more severe ventricular systolic dysfunction.

For the fourth survival analysis model we should also look at the highest correlations according to time.

```

sort(x[nrow(x), ])

##          DEATH_EVENT              age      high_blood_pressure
## -0.526963779 -0.224068420 -0.196439479
## serum_creatinine      anaemia      smoking
## -0.149315418 -0.141413982 -0.022838942
##           sex creatinine_phosphokinase platelets
## -0.015608220      -0.009345653 0.010513909
##      diabetes      ejection_fraction serum_sodium
## 0.033725509      0.041729235 0.087640000
##           time
## 1.000000000

```

According to time we see that highest negative correlation when death_event is discarded are age, high blood pressure, serum creatinine and anemia. This means that old people tend to die early, with high blood pressure, with high serum creatinine, with anemia. This seems to somewhat run hand in hand with the previous conclusions, although this time high blood pressure and anemia was introduced. High blood pressure sounds like it could lead to heart attack easily, which makes sense. Also anemia seems to go hand in hand with heart attack.

5 Models

We chose to use BRMS for modeling. It stands for Bayesian Regression Models for Stan. It's an interface to fit Bayesian generalized (non-)linear multivariate models using Stan. As we need to do analysis for binary response variables, we need some sort of a generalized linear model. We also chose BRMS due to its ease of use when fitting such complicated multilevel generalized linear models.

In BRMS modeling, the parameters are said to either be population level or group level. Population-level parameters means the same thing as regular parameters in our course, and group-level parameters mean hyperparameters in hierarchical case.

Family argument specifies the distribution family of the response variable.

Prior argument for each of the parameters. One can set different priors for each population level parameter, or group level parameter.

Before creating the test/train datasets, we need to preprocess the data little bit. Ejection fraction is described as percentage, and it can be given prior with beta distribution, which is constrained from 0 to 1. Let's normalize ejection fraction to be from 0 to 1.

```
heart$ejection_fraction = heart$ejection_fraction/100
```

Create general function for splitting the train and test data.

```

split.train.test <- function(data, test.size = 0.3) {
  train.indice <- sample(nrow(heart), nrow(heart) * (1 - test.size))
  train.data <- heart[train.indice, ]
  test.data <- heart[-train.indice, ]
  return(list(train = train.data, test = test.data))
}
set.seed(1)
new.data <- split.train.test(heart)
train.data <- new.data$train
test.data <- new.data$test

```

5.1 Model fitting

The Generalised Linear Model used for every parametrisation is Bernoulli-Logit Generalised Linear Model for the first three models, which is logistic regression. It's expressed in mathematical terms as following:

$$BernoulliLogitGLM(y|x, \alpha, \beta) = \prod_{1 \leq i \leq n} Bernoulli(y_i | logit^{-1}(\alpha_i + x_i \times \beta))$$

5.1.1 Full model

Stan code for full model can be found in Appendix A.

Full model includes all the parameters that are specified in the dataset.

```
fit.full <- brm(formula = DEATH_EVENT ~ age + ejection_fraction +
  serum_creatinine + serum_sodium + high_blood_pressure + creatinine_phosphokinase +
  diabetes + smoking + anaemia + sex, prior = c(set_prior("cauchy(40,20)",
  class = "b", coef = "age"), set_prior("inv_gamma(1,5)", class = "b",
  coef = "serum_creatinine"), set_prior("beta(6,4)", class = "b",
  coef = "ejection_fraction"), set_prior("cauchy(0,200)", class = "b",
  coef = "serum_sodium"), set_prior("cauchy(0,4000)", class = "b",
  coef = "creatinine_phosphokinase"), set_prior("normal(.5, .5)",
  class = "b", coef = "anaemia"), set_prior("normal(.5, .5)",
  class = "b", coef = "diabetes"), set_prior("normal(.5, .5)",
  class = "b", coef = "smoking"), set_prior("normal(.5, .5)",
  class = "b", coef = "high_blood_pressure"), set_prior("normal(.5, .5)",
  class = "b", coef = "sex")), data = train.data, family = bernoulli(),
  refresh = 0)
```

5.1.2 Feature selected model

Stan code for feature selected model can be found in Appendix B.

In feature selected model, we hand pick the features that seems to be the most promising with regards to fitting the model. As described above in correlation analysis, we saw that ejection_fraction, serum_creatinine, serum_sodium and age were correlating to death.

Based on this we can choose these variables to be the important ones, and build a model with them.

```
fit.feature_selected <- brm(formula = DEATH_EVENT ~ ejection_fraction +
  serum_creatinine + serum_sodium + age, data = train.data,
  family = bernoulli(), prior = c(set_prior("cauchy(40,20)",
  class = "b", coef = "age"), set_prior("inv_gamma(1,1)",
  class = "b", coef = "serum_creatinine"), set_prior("beta(6,4)",
  class = "b", coef = "ejection_fraction"), set_prior("cauchy(0,200)",
  class = "b", coef = "serum_sodium")), refresh = 0, control = list(adapt_delta = 0.99))
```

5.1.3 Hierarchical model

Stan code for hierarchical model can be found in Appendix C.

In hierarchical model, we choose age as hyperparameter, because by intuition we thought that different aged people tend to have different medical conditions by default. This intuition is supported by calculating the absolute row sums of the correlation matrix rows, and seeing absolute correlations of variables.

```

sort(rowSums(abs(x)))

##          platelets creatinine_phosphokinase      diabetes
##                1.651267           1.651341        1.710253
##    high_blood_pressure            anaemia      smoking
##                1.767362           1.908799        1.940394
##    ejection_fraction serum_creatinine serum_sodium
##                1.950445           1.999252        2.016403
##            age                  sex             time
##                2.243391           2.276192        2.459603
##    DEATH_EVENT
##                2.815147

```

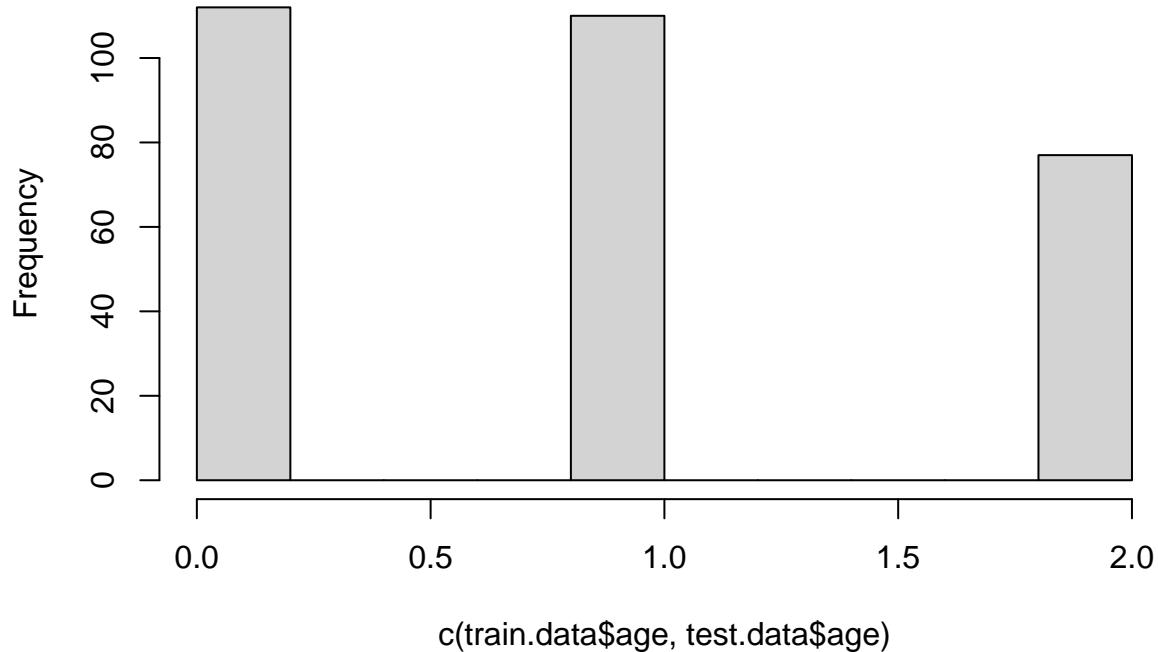
First we will discretize age data in to 3 equal depth bins.

```

discretize.variable <- function(variable.to.discretize, thresholds = c(55,
  70)) {
  variable.to.discretize[variable.to.discretize <= thresholds[1]] = 0
  variable.to.discretize[variable.to.discretize > thresholds[1] &
    variable.to.discretize < thresholds[2]] = 1
  variable.to.discretize[variable.to.discretize >= thresholds[2]] = 2
  return(variable.to.discretize)
}
set.seed(1)
test.data$age <- discretize.variable(test.data$age)
train.data$age <- discretize.variable(train.data$age)
hist(c(train.data$age, test.data$age))

```

Histogram of c(train.data\$age, test.data\$age)



Then we can fit the model

```
fit.hierarchical <- brm(formula = DEATH_EVENT ~ ejection_fraction +
  serum_creatinine + serum_sodium + (ejection_fraction + serum_creatinine +
  serum_sodium | age), data = train.data, family = bernoulli(),
  c(set_prior("cauchy(0,5)", class = "b", coef = "serum_creatinine"),
    set_prior("beta(6,4)", class = "b", coef = "ejection_fraction"),
    set_prior("normal(100,40)", class = "b", coef = "serum_sodium")),
  refresh = 0, control = list(adapt_delta = 0.99))
```

5.1.4 Model for death time analysis

As a fourth model we have model with response variable according to Weibull distribution, and features are selected based on correlation matrix.

In death time analysis model we are using different functions based on whether sample is censored or not. If sample is censored, then we are using log probability density function

$$Weibull(y|\alpha, \sigma) = \frac{\alpha}{\sigma} \left(\frac{y}{\sigma}\right)^{-\alpha-1} \exp\left(-\left(\frac{y}{\sigma}\right)^\alpha\right)$$

If sample is not censored, then we are using the cumulative function of previously mentioned probability density function.

First we need to retrieve the original data with original age values

```

set.seed(1)
new.data <- split.train.test(heart)
train.data <- new.data$train
test.data <- new.data$test

```

Then we can fit the model

```

fit.weibull <- brm(formula = time | cens(DEATH_EVENT) ~ age +
  anaemia + high_blood_pressure, data = train.data, family = weibull(),
  prior = c(set_prior("cauchy(40, 20)", class = "b", coef = "age"),
            prior_string("normal(.5, .5)", class = "b", coef = "anaemia"),
            prior_string("normal(.5, .5)", class = "b", coef = "high_blood_pressure")),
  refresh = 0)

```

5.2 Prior choices

We chose priors based on articles that we read about medical measurements. We used weakly informative priors, because there's only little indication in what type of prior we should use.

5.2.1 First models

Same priors were used more or less to every of first 3 models, and we will specify them here.

Coefficient distributions

- Age distribution is using half Cauchy distribution, which fits our population well. Cauchy is non-negative continuous distribution, which peaks at around 40, and decays to over 100. This is good, because it forces the prior to be non-negative, which is a must in age distribution. Also it centers around 40, which is convenient, because our age started from 40.

$$age \sim Cauchy(y|\mu, \sigma) = \frac{1}{\pi\sigma} \frac{1}{1 + ((y - \mu)/\sigma)^2}$$

$$age \sim Cauchy(40, 20)$$

- Serum creatinine used inverse gamma distribution. Creatinine levels also need non-negative distribution, and the normal level should be between 0.9 and 1.3. Severe symptoms start when creatinine reaches over 5, so we don't restrict the tail.

$$\text{serum creatinine} \sim InvGamma(y|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} y^{-(\alpha+1)} \exp\left(-\beta \frac{1}{y}\right)$$

$$\text{serum creatinine} \sim InvGamma(1, 1)$$

- Ejection fraction is constrained to be from 0 to 100, as its expressed in percentage. For this reason we need to use beta distribution. We chose to parametrize beta distribution as (6,4), because the ejection fraction should be between 50 and 70. (6,4) parametrization provides us mean of 0.6.

$$\text{ejection fraction} \sim Beta(y|\alpha, \beta) = \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1} (1-\theta)^{\beta-1}$$

$$\text{ejection fraction} \sim Beta(6, 4)$$

- Serum sodium should be around 135 and 145, but we gave it large variance with (half) $\text{cauchy}(0,200)$. Severe symptoms of too high sodium levels start above 160, so this prior gives a lot room to reach fatal levels.

$$\text{serum sodium} \sim \text{Cauchy}(y|\mu, \sigma) = \frac{1}{\pi\sigma} \frac{1}{1 + ((y - \mu)/\sigma)^2}$$

$$\text{serum sodium} \sim \text{Cauchy}(0, 200)$$

- Creatinine phosphokinase regular levels are between 20 and 200, but it can reach in certain coditions even hundreds of thousand. This is why we dont want to limit our priors. We chose

$$\text{serum sodium} \sim \text{Cauchy}(0, 4000)$$

- Priors for anemia, smoking, high blood pressure and sex were chosen as uniform $\text{beta}(1,1)$. This is because they might get values 0 or 1 with equal probabilities.

$$\text{anemia, smoking, high blood pressure, sex} \sim \mathcal{N}(.5, .5)$$

Intercept distribution Intercept distribution was `student_t` for every coefficient. The `student_t` distribution is parameterized by default as $\text{student_t}(3, 0, 2.5)$.

$$\text{Intercept} \sim \text{StudentT}(y|v, \mu, \sigma) = \frac{\Gamma((v+1)/2)}{\Gamma(v/2)} \left(1 + \frac{1}{v} \left(\frac{y - \mu}{\sigma}\right)^2\right)^{-(v+1)/2}$$

$$\text{Intercept} \sim \text{StudentT}(3, 0, 2.5)$$

By default BRMS uses improper flat prior over the reals for population level parameters. Group level parameter is assumed to come from multivariate normal distribution with zero mean and unknown covariance matrix.

There is more than one group-level effect per grouping factor, so correlations between those effects have to be estimated. Group level correlation matrix is generated uniformly over all positive definite correlation matrices using LKJ distribution as shown below:

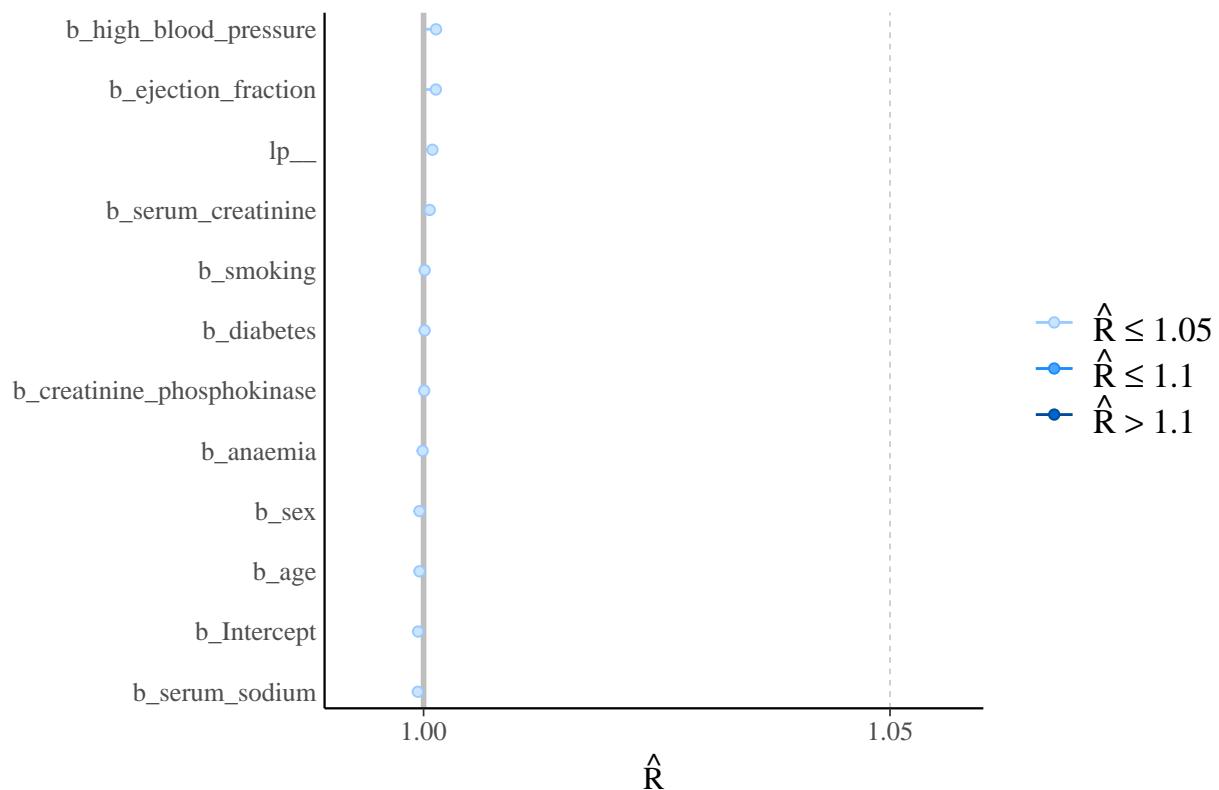
$$\text{LkjCholesky}(L|\eta) \propto |J| \det(LL^\top)^{(\eta-1)} = \prod_{k=2}^K L_{kk}^{K-k+2\eta-2}$$

5.3 \hat{R} convergence

5.3.1 Linear model with all variables

```
rhats <- rhat(fit.full)
color_scheme_set("brightblue")
mcmc_rhat(rhats) + yaxis_text(hjust = 1) + ggtitle("Rhat values for model with all variables") +
  theme(plot.title = element_text(hjust = 0.5))
```

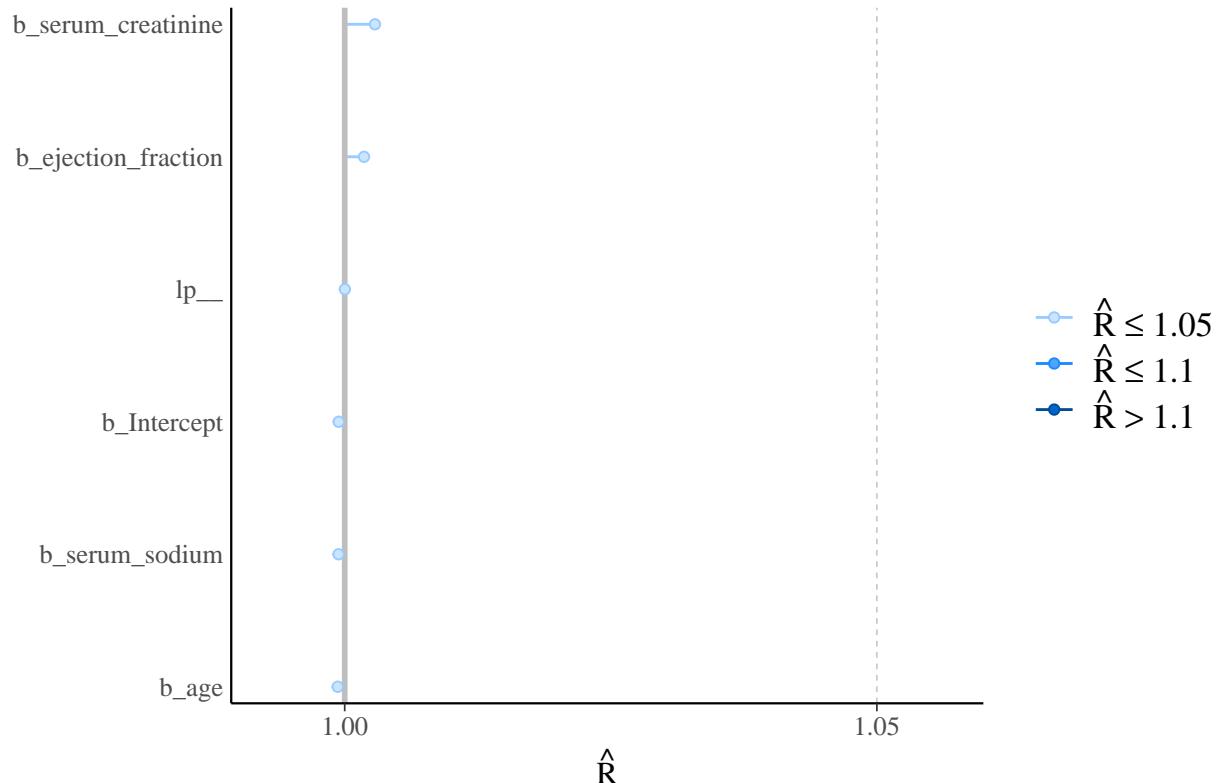
Rhat values for model with all variables



5.3.2 Linear model with selected variables

```
rhats <- rhat(fit.feature_selected)
color_scheme_set("brightblue")
mcmc_rhat(rhats) + yaxis_text(hjust = 1) + ggtitle("Rhat values for model with selected variables") +
  theme(plot.title = element_text(hjust = 0.5))
```

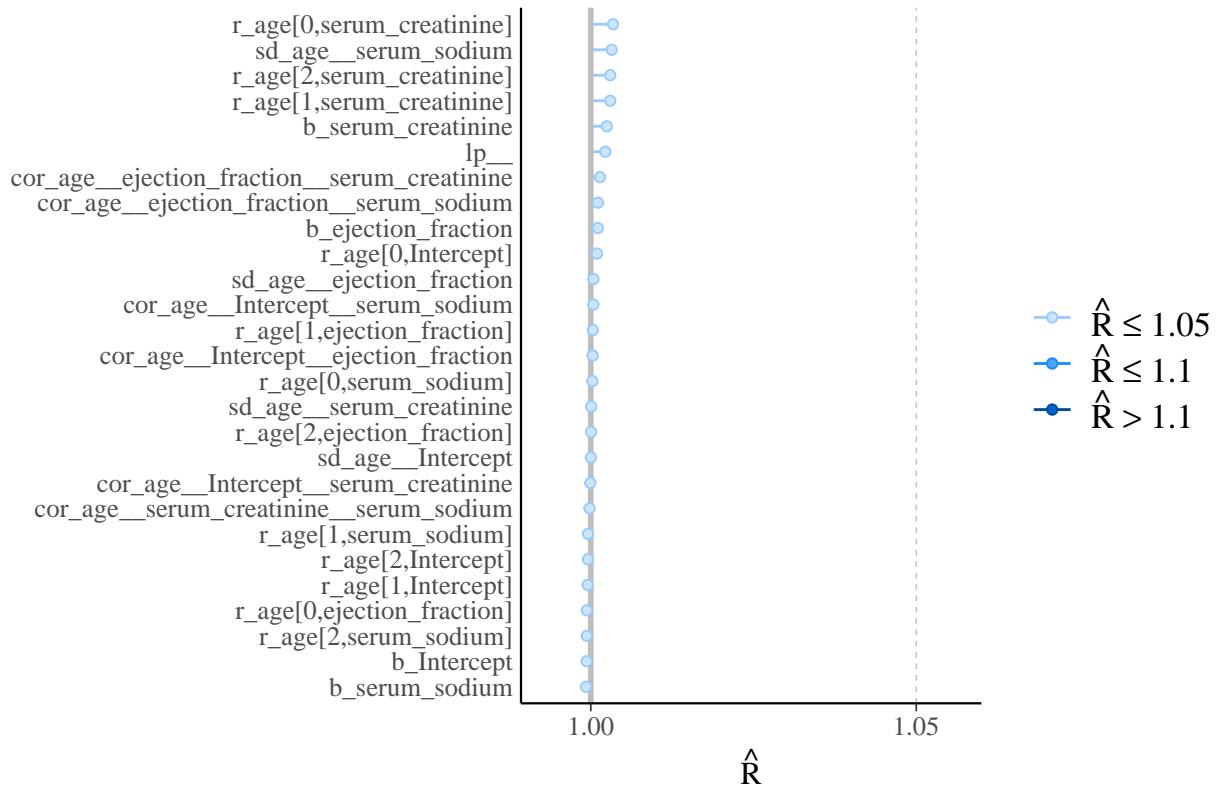
Rhat values for model with selected variables



5.3.3 Hierarchical model with all variables

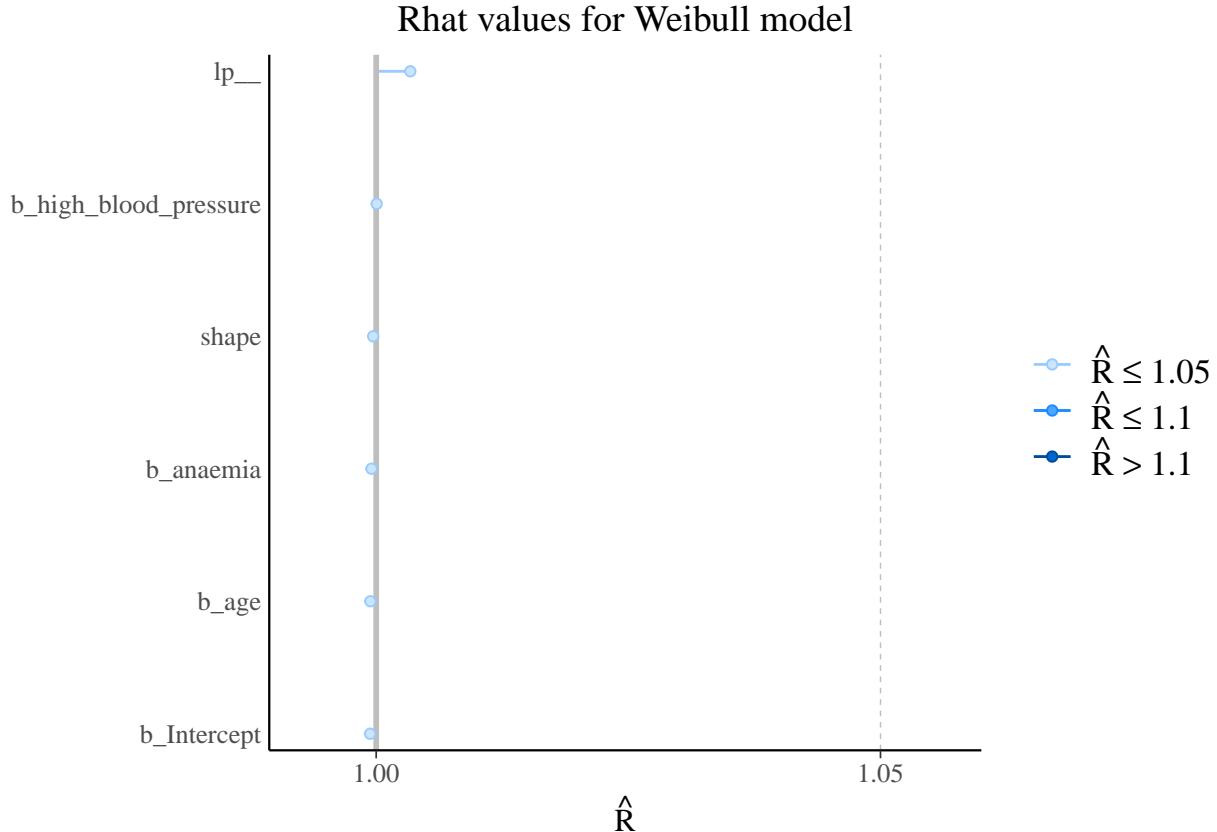
```
rhats <- rhat(fit.hierarchical)
color_scheme_set("brightblue")
mcmc_rhat(rhats) + yaxis_text(hjust = 1) + ggtitle("Rhat values for hierarchical model with all variables")
  theme(plot.title = element_text(hjust = 0.5))
```

Rhat values for hierarchical model with all variables



5.3.4 Weibull model

```
rhats <- rhat(fit.weibull)
color_scheme_set("brightblue")
mcmc_rhat(rhats) + yaxis_text(hjust = 1) + ggtitle("Rhat values for Weibull model") +
  theme(plot.title = element_text(hjust = 0.5))
```



From the model summaries, it is observable that \hat{R} values for all models are around 1.0 which is below the threshold value of 1.05 as mentioned in Stan documentation. With this information, we can interpret that the chains have mixed well and the samples are reliable.

5.4 HMC specific convergence diagnostics (divergences, tree depth) with interpretation of the results

5.4.1 Feature selected model

```
check_divergences(fit.feature_selected$fit)

## 0 of 4000 iterations ended with a divergence.

check_treedepth(fit.feature_selected$fit)

## 0 of 4000 iterations saturated the maximum tree depth of 10.
```

5.4.2 Full model

```
check_divergences(fit.full$fit)
```

```

## 0 of 4000 iterations ended with a divergence.

check_treedepth(fit.full$fit)

## 0 of 4000 iterations saturated the maximum tree depth of 10.

```

5.4.3 Hierarchical model

```

check_divergences(fit.hierarchical$fit)

## 0 of 4000 iterations ended with a divergence.

check_treedepth(fit.hierarchical$fit)

## 1 of 4000 iterations saturated the maximum tree depth of 10 (0.025%).
## Try increasing 'max_treedepth' to avoid saturation.

```

5.4.4 Weibull model

```

check_divergences(fit.weibull$fit)

## 0 of 4000 iterations ended with a divergence.

check_treedepth(fit.weibull$fit)

## 0 of 4000 iterations saturated the maximum tree depth of 10.

```

As we can see, we hardly observe any divergence. However we believe the ratio of the divergent iterations and the total iterations can be considered negligible and maybe can be fixed using different priors (more informative). More accurate analysis about the models can be found in the next sections where we investigate more on the quality of the latters.

5.5 Effective sample size diagnostic (n_eff or ESS)

5.5.1 Feature selected model

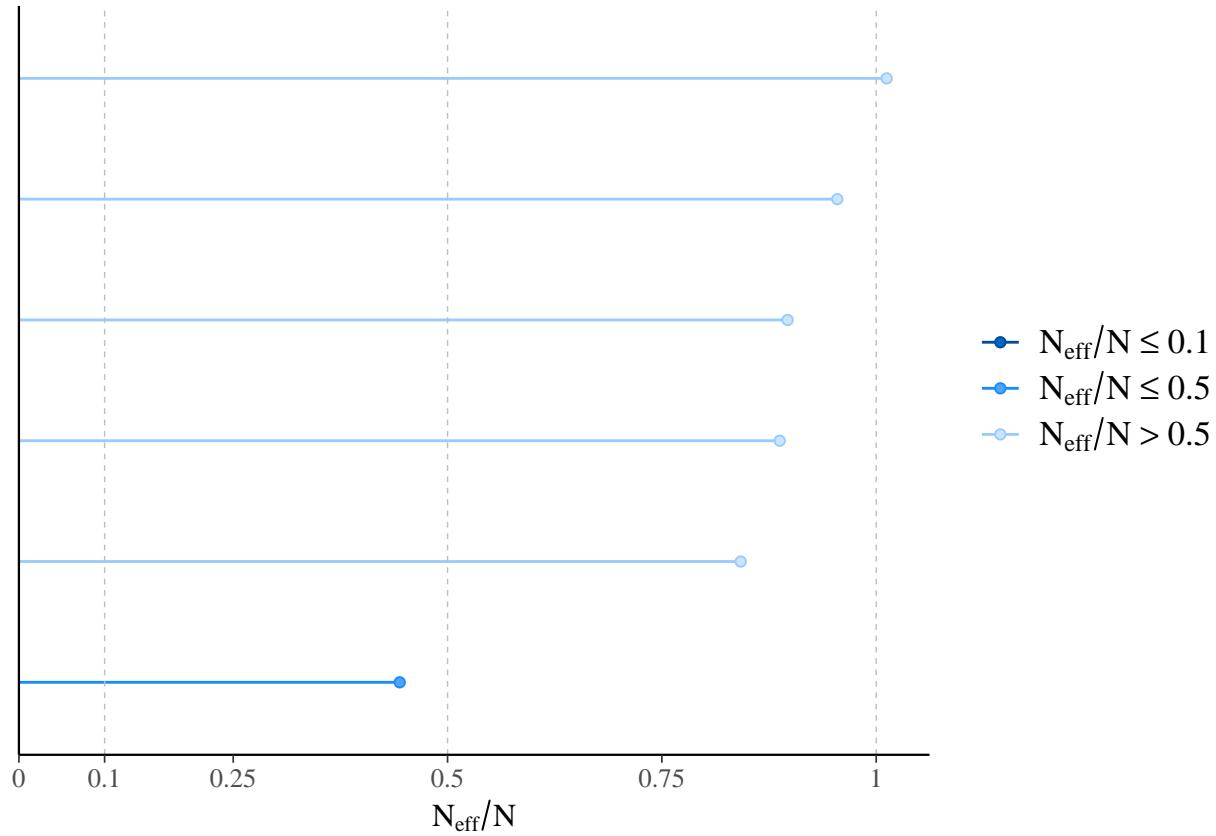
```

s <- summary(fit.feature_selected)
s[["fixed"]][, 6:7]

##           Bulk_ESS Tail_ESS
## Intercept      3662     2755
## ejection_fraction 3829     3141
## serum_creatinine   3808     2459
## serum_sodium       3624     2854
## age                 4103     2907

```

```
ratio <- neff_ratio(fit.feature_selected)
mcmc_neff(ratio)
```

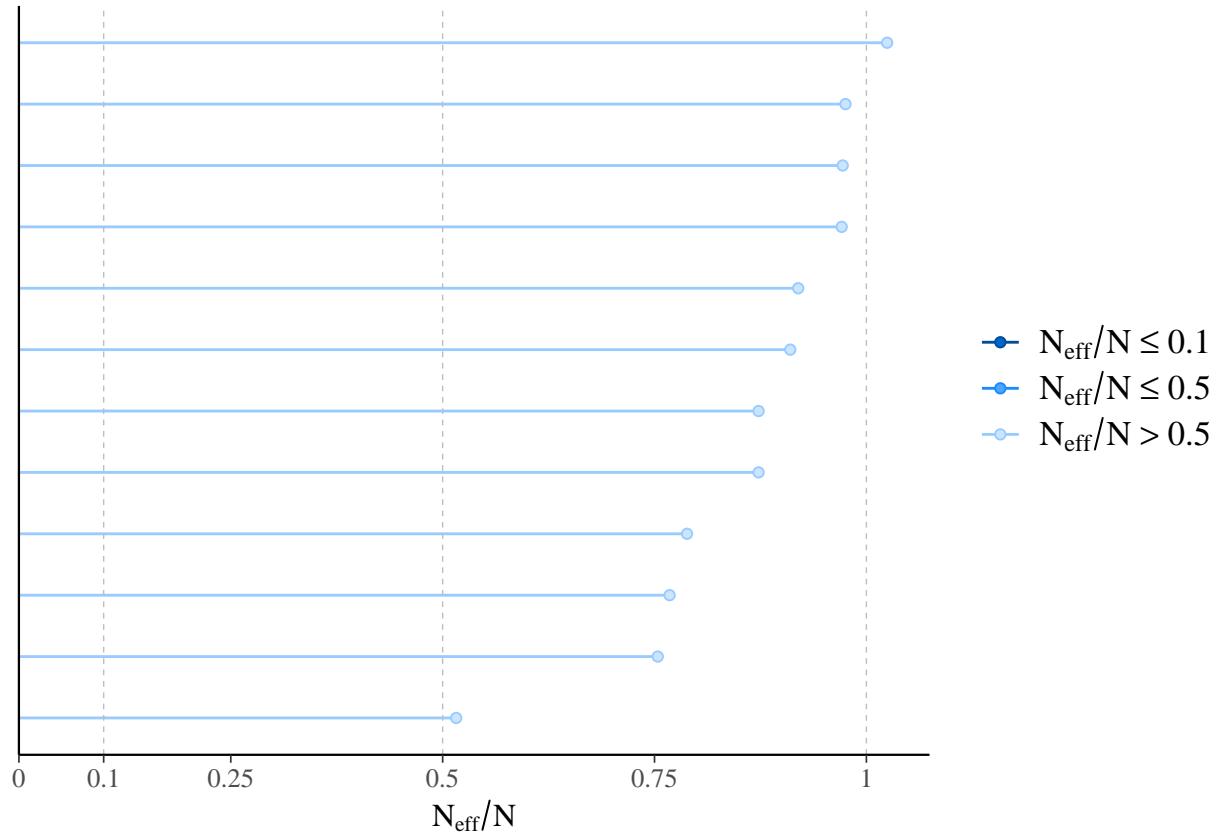


5.5.2 Full model

```
s <- summary(fit.full)
s[["fixed"]][, 6:7]
```

| | Bulk_ESS | Tail_ESS |
|-----------------------------|----------|----------|
| ## Intercept | 4104 | 3068 |
| ## age | 3675 | 3077 |
| ## ejection_fraction | 3573 | 3487 |
| ## serum_creatinine | 3310 | 2329 |
| ## serum_sodium | 3896 | 2949 |
| ## high_blood_pressure | 3542 | 2894 |
| ## creatinine_phosphokinase | 3901 | 2742 |
| ## diabetes | 3918 | 2937 |
| ## smoking | 3107 | 2882 |
| ## anaemia | 3711 | 2678 |
| ## sex | 3207 | 2695 |

```
ratio <- neff_ratio(fit.full)
mcmc_neff(ratio)
```

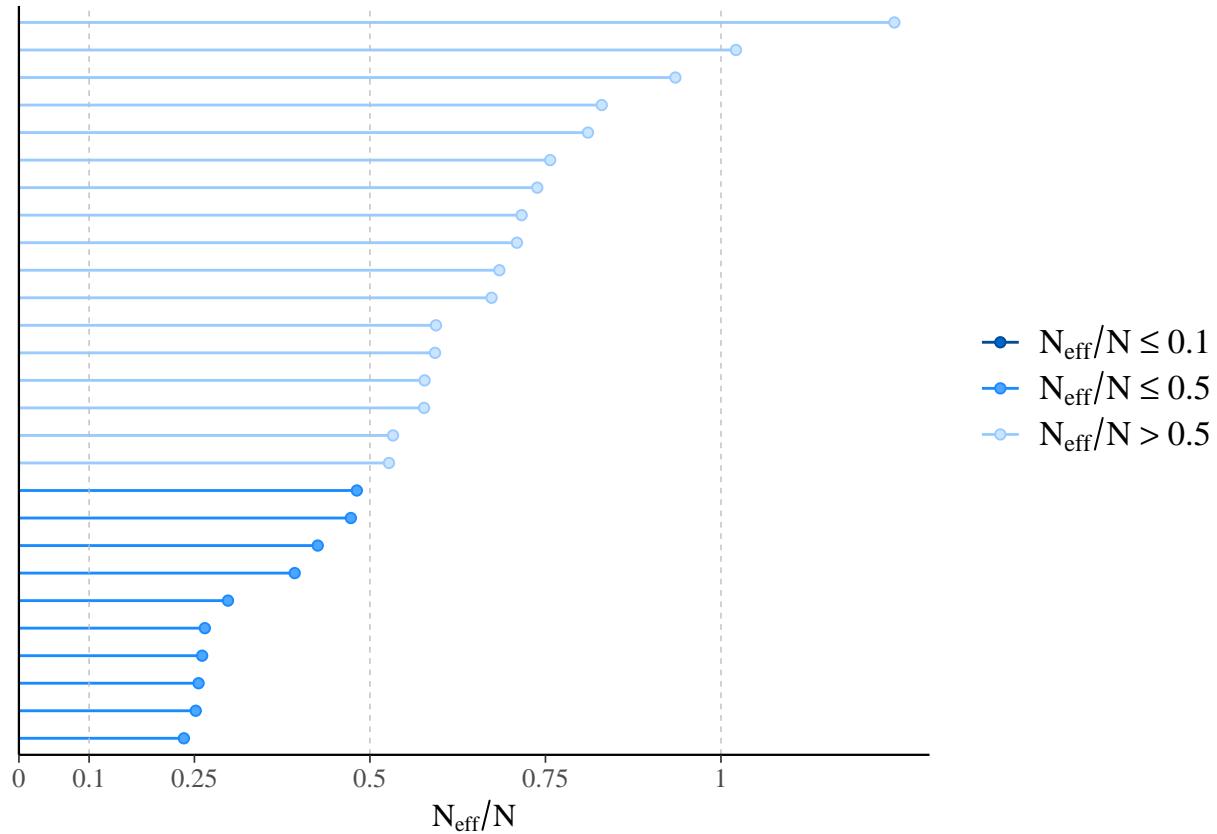


5.5.3 Hierarchical model

```
s <- summary(fit.hierarchical)
s[["fixed"]][, 6:7]

##           Bulk_ESS Tail_ESS
## Intercept      4187     2368
## ejection_fraction 5069     3226
## serum_creatinine   1452     1063
## serum_sodium       3818     2635

ratio <- neff_ratio(fit.hierarchical)
mcmc_neff(ratio)
```

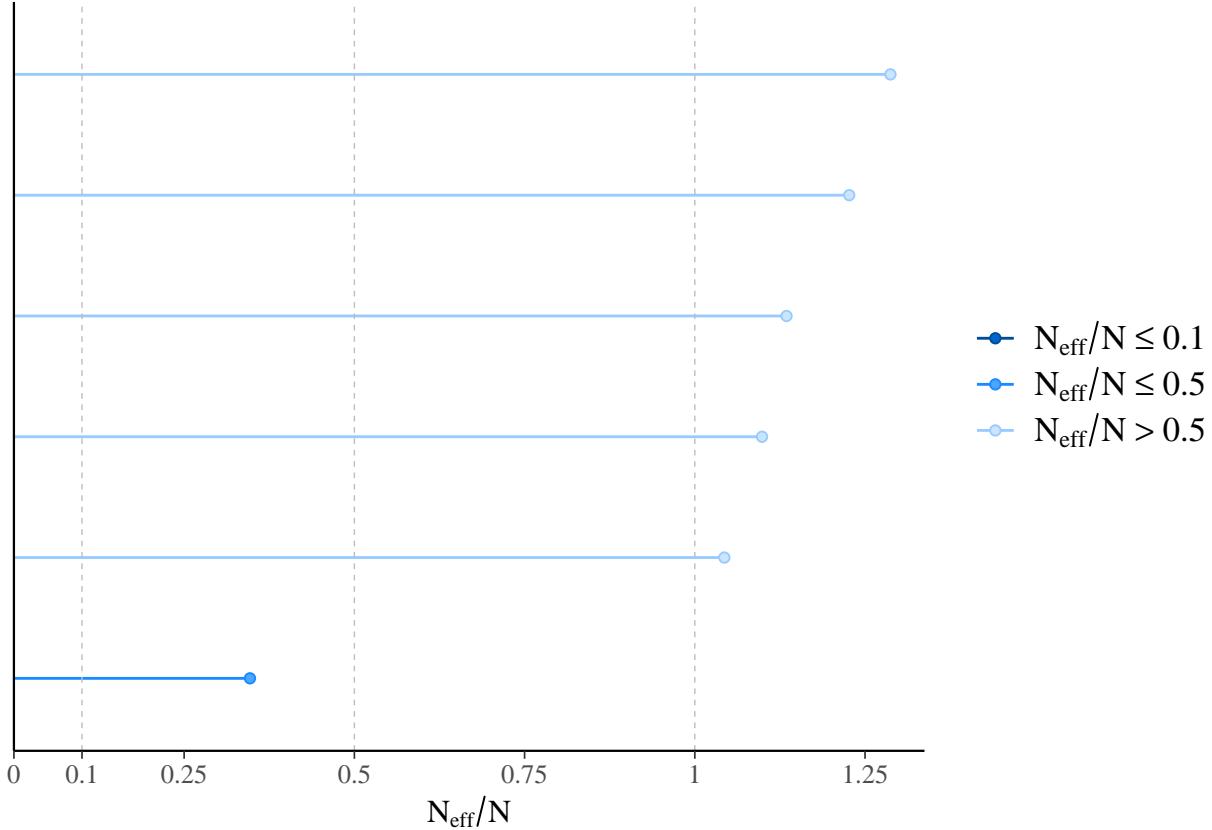


5.5.4 Weibull model

```
s <- summary(fit.weibull)
s[["fixed"]][, 6:7]

##                                     Bulk_ESS Tail_ESS
## Intercept                         5246     3083
## age                                5038     3015
## anaemia                            4494     2786
## high_blood_pressure                 4185     2859

ratio <- neff_ratio(fit.weibull)
mcmc_neff(ratio)
```



The effective sample size (ESS) measures the amount by which autocorrelation in samples increases uncertainty (standard errors) relative to an independent sample. In other words the effective sample size is an estimate of the number of independent draws from the posterior distribution of the estimand of interest. The n_{eff} metric used in Stan is based on the ability of the draws to estimate the true mean value of the parameter. Usually smaller than the total sample size N so the larger the ratio to N , the better. As we can see from the plots, mostly all the ratios of effective sample size to total sample size can be considered good. For the feature selected model only one chain out of 4 has a ration below 0.5. Concerning the full model, we obtained that all the chain has a good N_{eff}/N ratio but again we can find a chain whose ration is slightly below 0.5. A similar conclusion can be derive from the weibull model. Regarding the hierarchical one, we can find that nearly 50% of the chains present a ratio below 0.5, and then further evaluation need to be carried out.

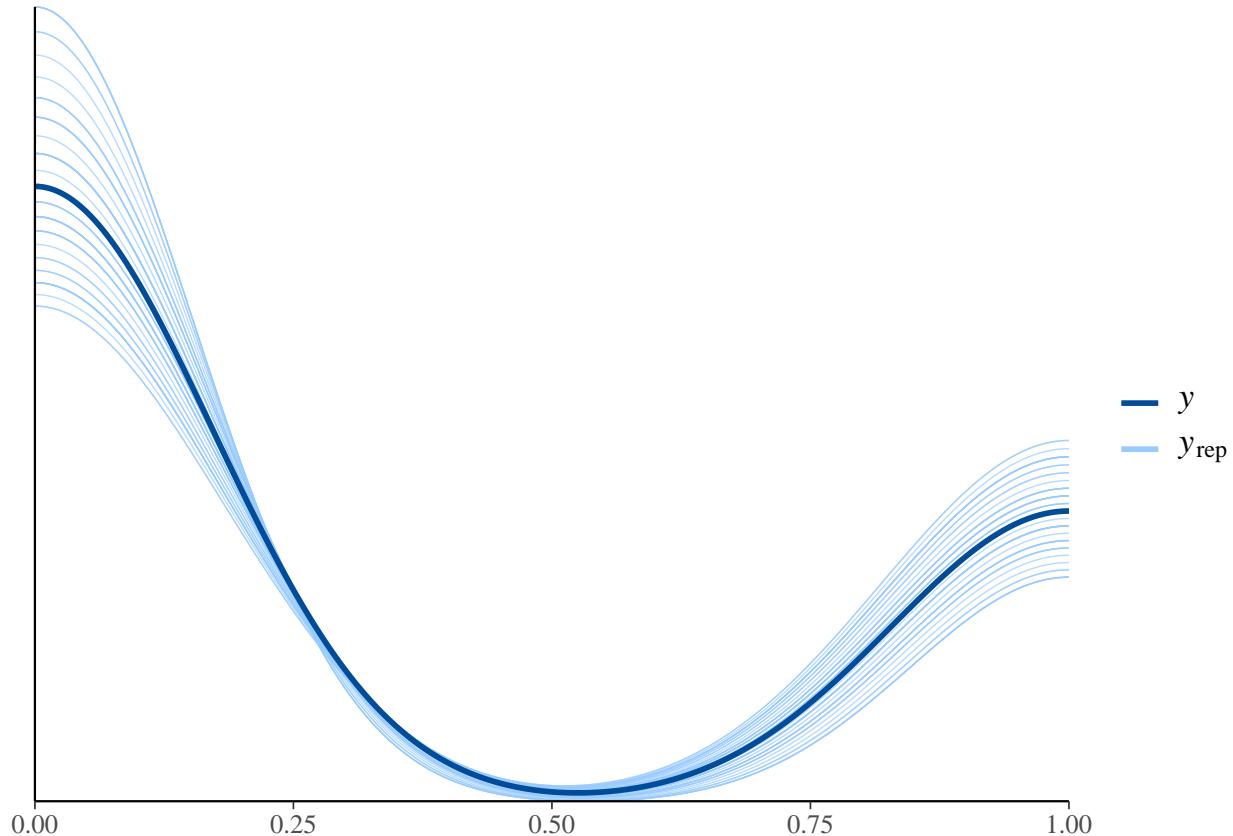
5.6 Posterior predictive checking

First three models seem to converge well to the actual test set. The death event is more predictable than the survival because the events leading to death are more predictable.

Weibull model has a lot of variance but as the underlying distribution is multimodal, it is hard to modelled with small amount of data. The double peak is explained by the fact that some patients already had a high probability of death due to severe ventricular systolic dysfunction. The model does perform well with regards to detecting the death event at certain timestep. Due to the small amount of data, the distribution might be hard to interpret the actual scenario.

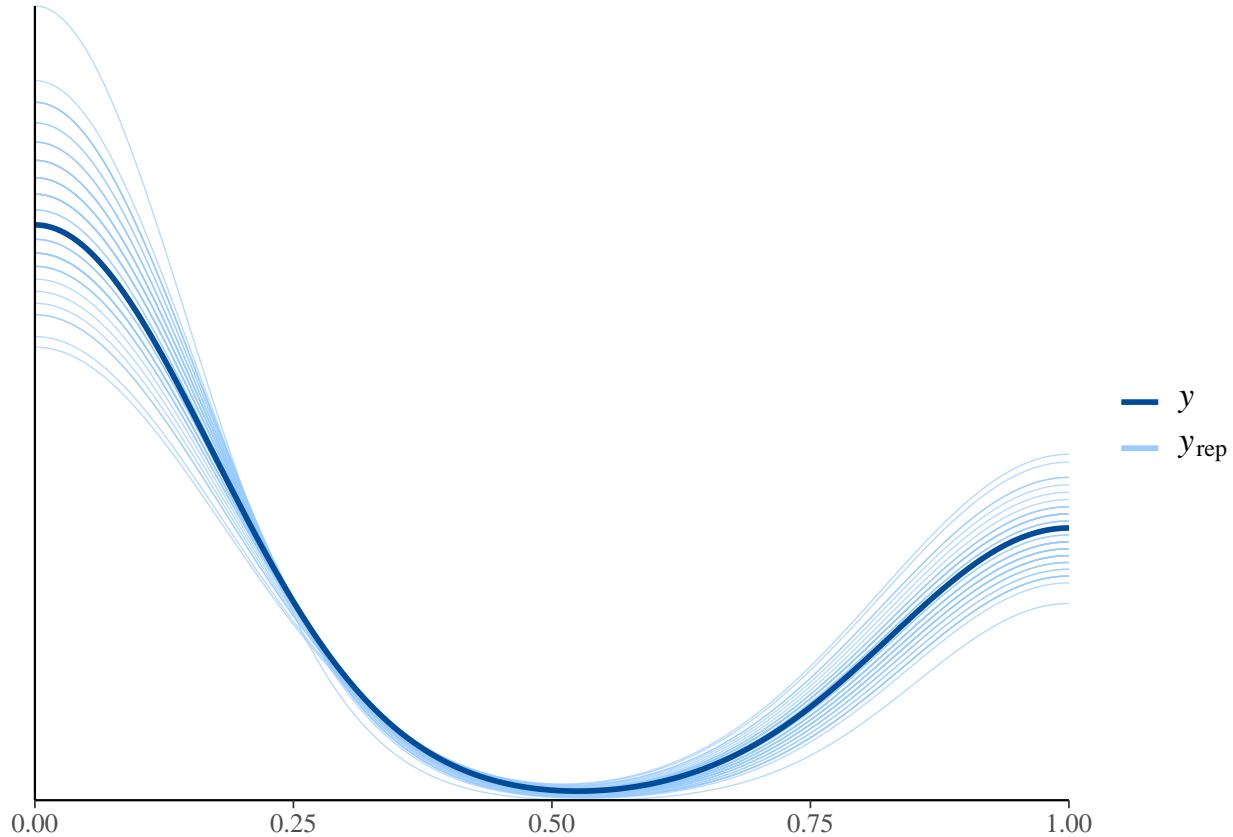
5.6.1 Feature selected model

```
pp_check(fit.feature_selected, nsamples = 50, newdata = test.data)
```



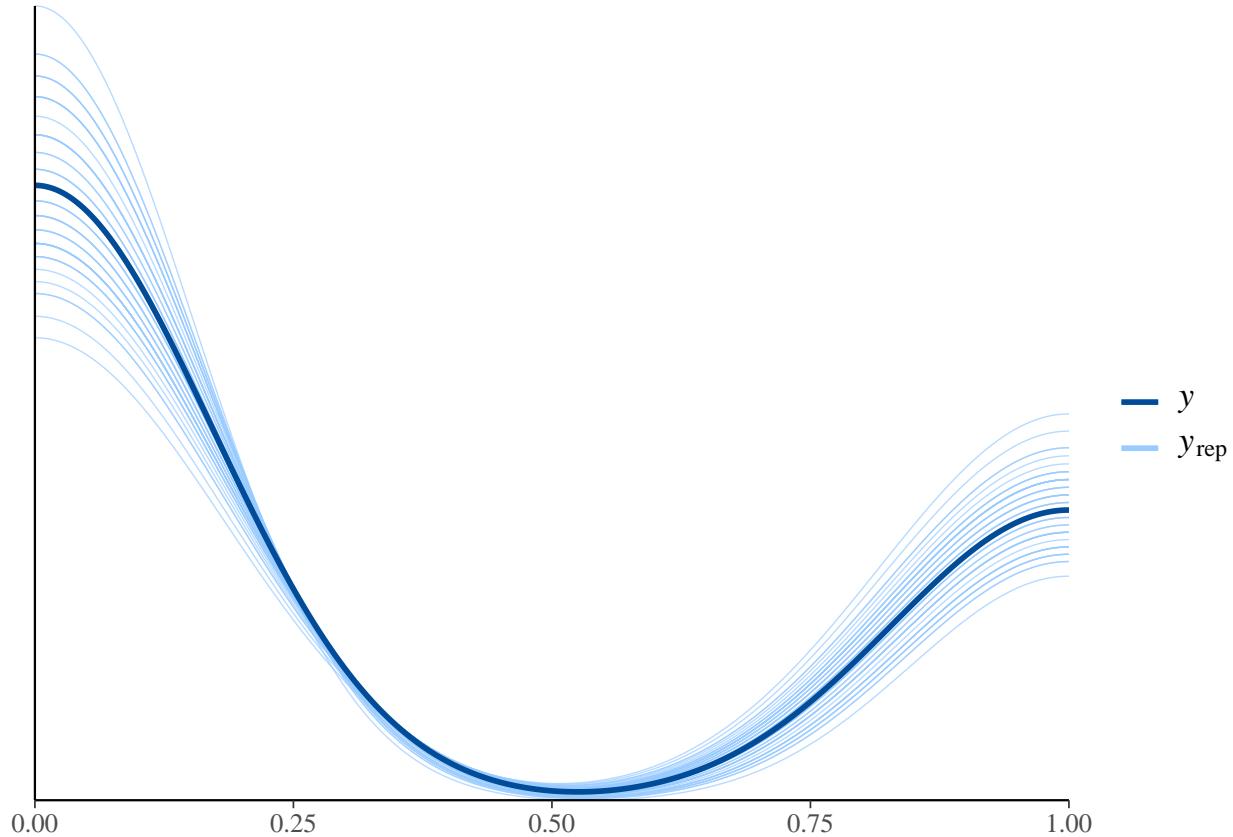
5.6.2 Full model

```
pp_check(fit.full, nsamples = 50, newdata = test.data)
```



5.6.3 Hierarchical model

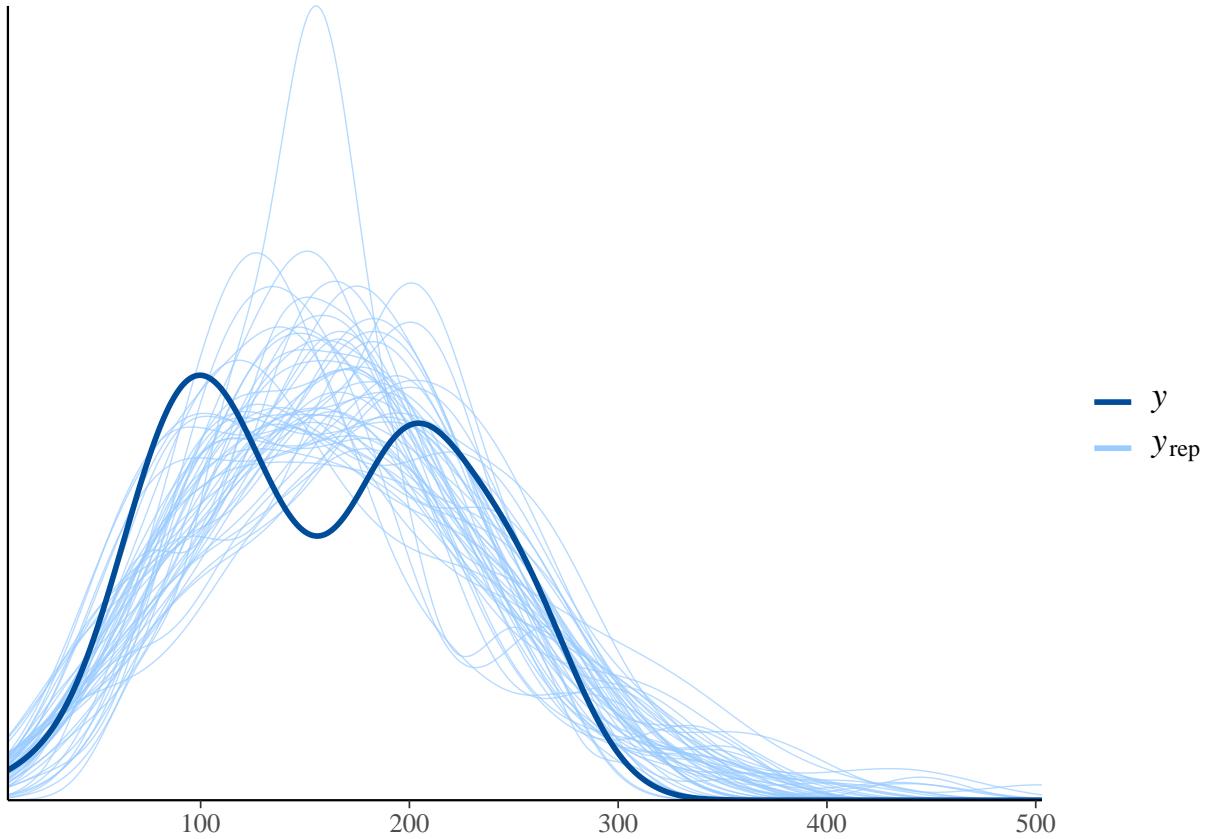
```
test.data$age <- discretize.variable(test.data$age)
train.data$age <- discretize.variable(train.data$age)
pp_check(fit.hierarchical, nsamples = 50, newdata = test.data)
```



5.6.4 Weibull model

```
set.seed(1)
new.data <- split.train.test(heart)
train.data <- new.data$train
test.data <- new.data$test
pp_check(fit.weibull, nsamples = 50, newdata = test.data)
```

Warning: Censored responses are not shown in 'pp_check'.



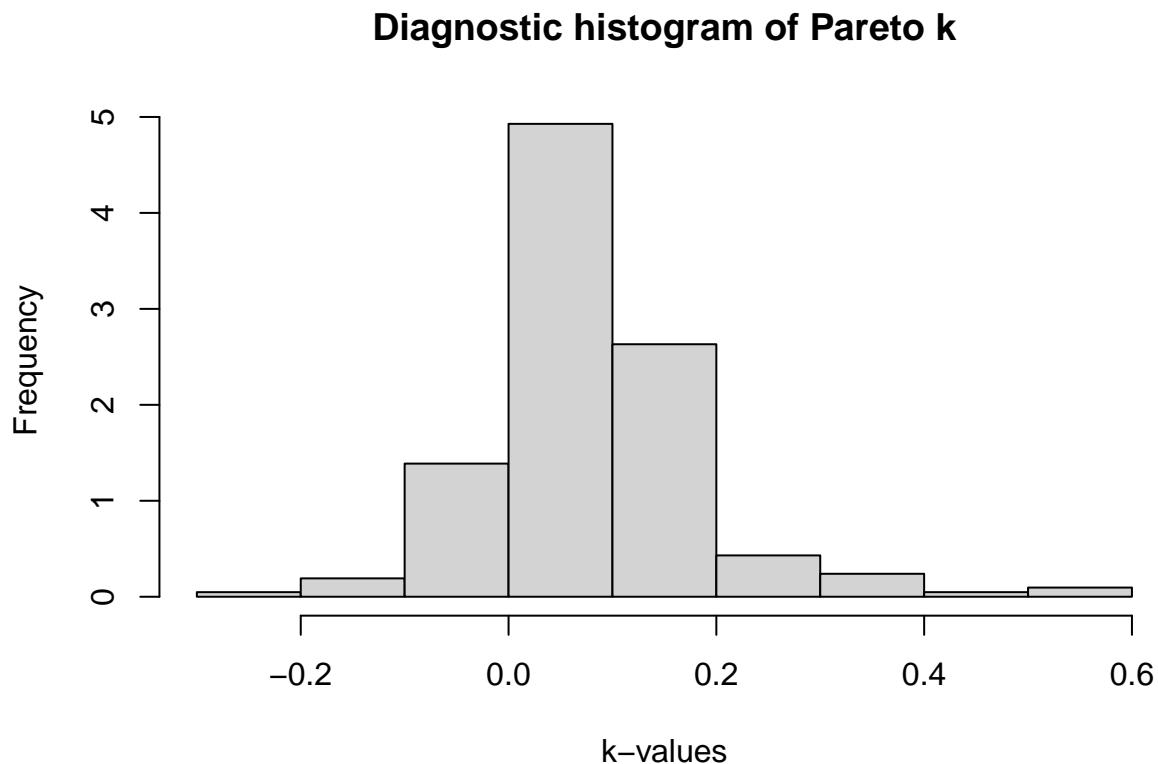
5.7 Model comparison and interpretation of the results

A detailed comparison is carried at the end of the “Model comparison and interpretation of the results” section.

5.7.1 Full model

```
loo.full <- loo(fit.full)

hist(loo.full$diagnostics$pareto_k, main = "Diagnostic histogram of Pareto k",
     xlab = "k-values", ylab = "Frequency", freq = FALSE)
```

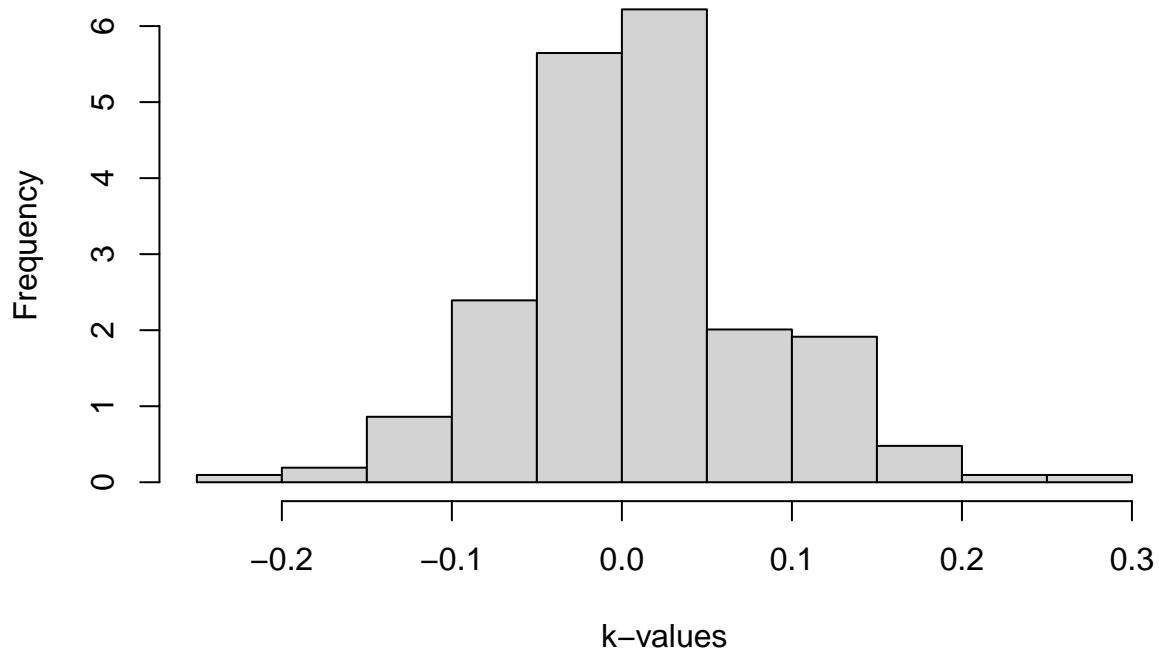


5.7.2 Feature selected model

```
loo.feature_selected <- loo(fit.feature_selected)

hist(loo.feature_selected$diagnostics$pareto_k, main = "Diagnostic histogram of Pareto k",
     xlab = "k-values", ylab = "Frequency", freq = FALSE)
```

Diagnostic histogram of Pareto k



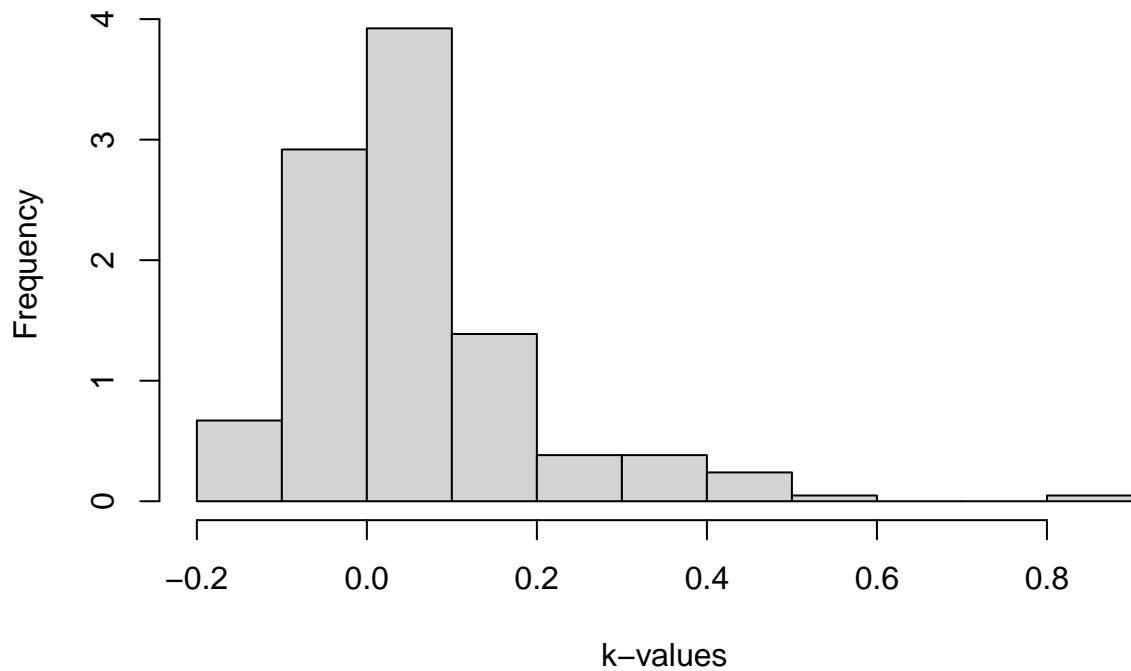
5.7.3 Hierarchical model

```
loo.hierarchical <- loo(fit.hierarchical)

## Warning: Found 1 observations with a pareto_k > 0.7 in model 'fit.hierarchical'.
## It is recommended to set 'moment_match = TRUE' in order to perform moment
## matching for problematic observations.

hist(loo.hierarchical$diagnostics$pareto_k, main = "Diagnostic histogram of Pareto k",
      xlab = "k-values", ylab = "Frequency", freq = FALSE)
```

Diagnostic histogram of Pareto k

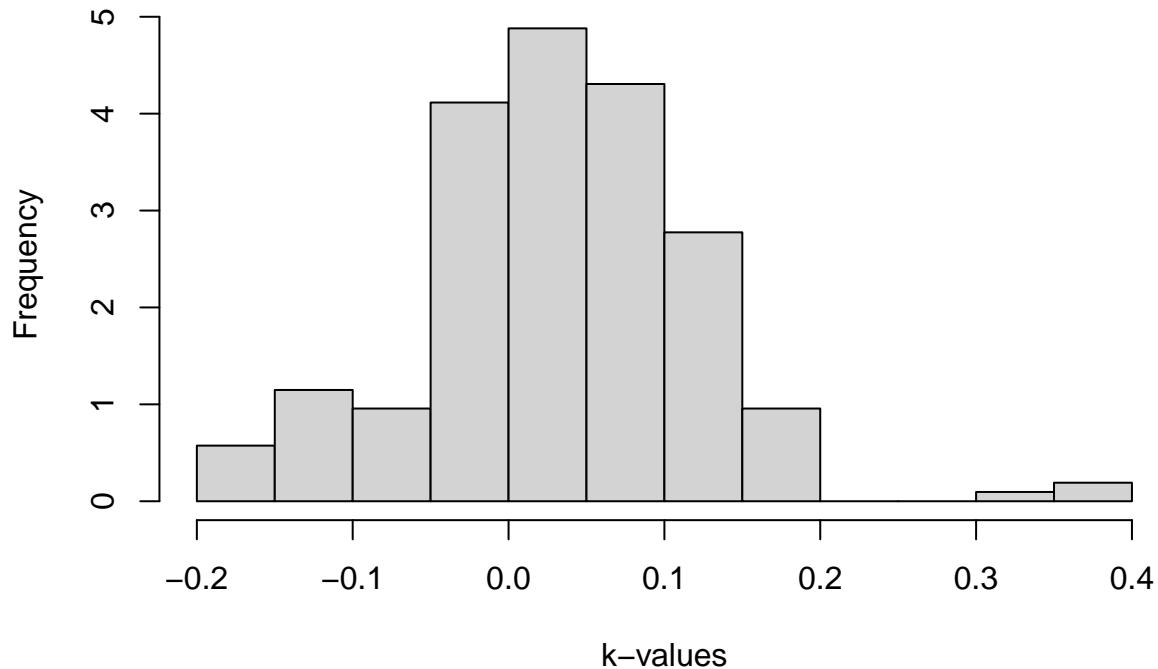


5.7.4 Weibull model

```
loo.weibull <- loo(fit.weibull)

hist(loo.weibull$diagnostics$pareto_k, main = "Diagnostic histogram of Pareto k",
     xlab = "k-values", ylab = "Frequency", freq = FALSE)
```

Diagnostic histogram of Pareto k



Model comparison:

```
loo_compare(list(loo.feature_selected, loo.full, loo.hierarchical))
```

```
##          elpd_diff se_diff
## fit.feature_selected  0.0      0.0
## fit.full            -1.7     4.2
## fit.hierarchical    -2.6     3.3
```

Leave-one-out cross-validation is a method for estimating pointwise out-of-sample prediction accuracy from a fitted Bayesian model using the log-likelihood evaluated at the posterior simulations of the parameter values [3].

From the PSIS-LOO we can see that the hierarchical model has the highest value compared to the other two models. This value is an indicator of model performance and the best model would be the one with the highest PSIS-LOO. Using the “loo” package we are able to estimate the difference in the models expected predictive accuracy and the function “loo_compare” creates a chart of the models with the one with highest ELPD (smallest LOOIC) first with zero values, meaning that is the best. We can see that the hierarchical model outstand the others, despite the difference is not huge. Pareto k-values estimates the tail shape which determines the convergence rate of PSIS. K-values less than 0.7 are considered to be a measure of reliability for the models in question. Values greater than 0.7 could represent a problem for the models, are further evaluation need to be done. As we can see from the k-values of the 4 models, none of the models return values below 0.7. We can state that the measure obtained using the loo package can be considere then reliable and the k-values help us to confirm our claim that the hierarhcical one is the best.

5.8 Predictive performance assessment (classification)

For actually seeing how the models perform, we split the data into train and test sets. We have to use different train and test sets for the hierarchical model, but the pointwise accuracy assessment is straightforward in discrete models by checking the prediction classes with test data labels.

Function for predicting pointwise accuracy for discrete models

```
predict.pointwise.accuracy <- function(fitted.model, test.data) {  
  preds <- round(predict(fitted.model, newdata = test.data)[,  
    1])  
  preds.correct <- preds == test.data$DEATH_EVENT  
  
  pointwise.accuracy <- length(preds.correct[preds.correct ==  
    TRUE])/nrow(test.data)  
}  
return(pointwise.accuracy)
```

For Weibull model, we can only find out the average days for death event to happen as it predicts the distribution for the number of days for death events. To assess its performance, predict function can be used.

5.8.1 Full model

```
predict.pointwise.accuracy(fit.full, test.data)
```

```
## [1] 0.7
```

5.8.2 Feature selected model

```
predict.pointwise.accuracy(fit.feature_selected, test.data)
```

```
## [1] 0.7222222
```

5.8.3 Hierarchical model

For hierarchical model, age column in the test data should be discretized since this feature is categorized to fit the model.

```
discretized.test.data <- test.data  
discretized.test.data$age <- discretize.variable(discretized.test.data$age)  
predict.pointwise.accuracy(fit.hierarchical, discretized.test.data)
```

```
## [1] 0.7222222
```

5.8.4 Weibull model

For Weibull model, predict function is used to get the estimate values for the number of days it takes for a death event to happen since the model is continuous.

```
colMeans(predict(fit.weibull, test.data))
```

```
## Estimate Est.Error      Q2.5      Q97.5
## 168.32531  68.20726  48.05475 310.58870
```

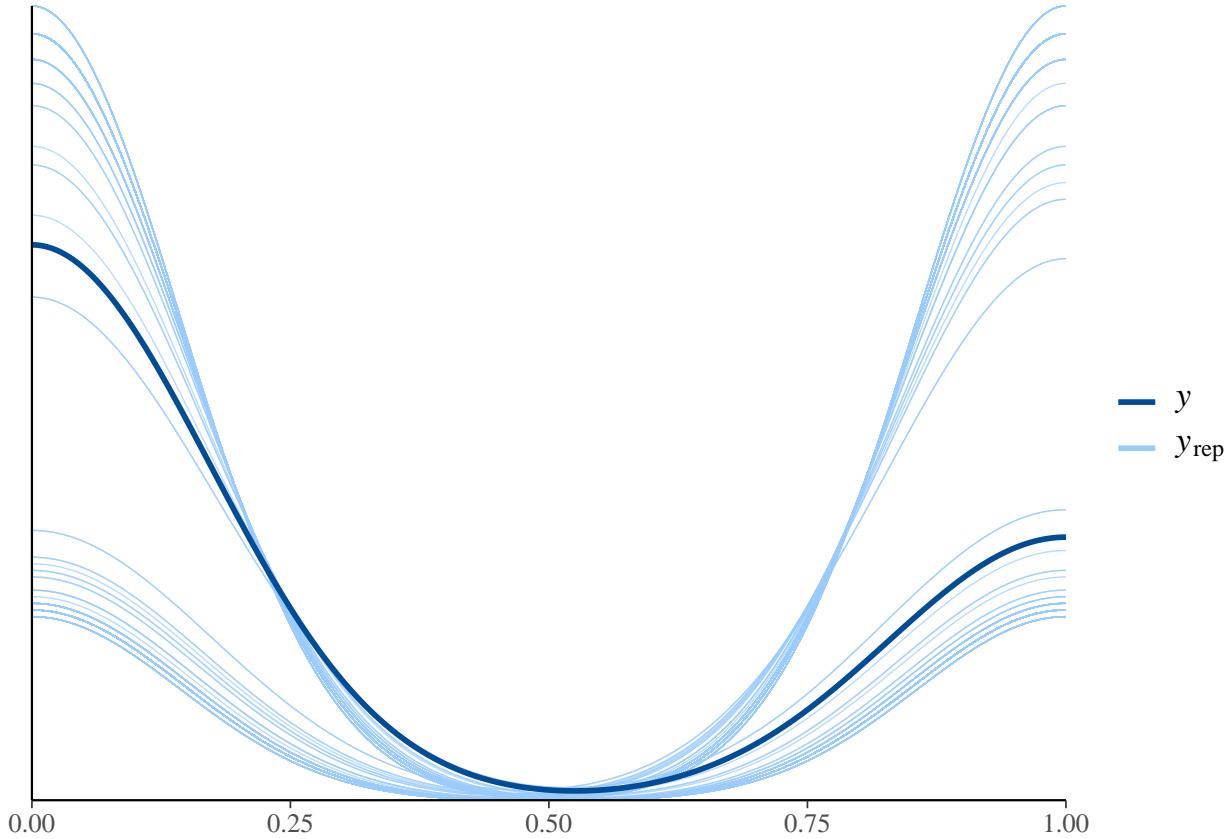
After the assessment, it seems that the hierarchical model performs better than the model with all variables and feature selected model when it comes to predict the correct labels (death or survival) for patients.

5.9 Prior sensitivity analysis (alternative prior tested)

We will use the model that has most priors for prior sensitivity analysis, the full model. This describes the meaning of prior well.

First let's use priors that we have chosen to be informative.

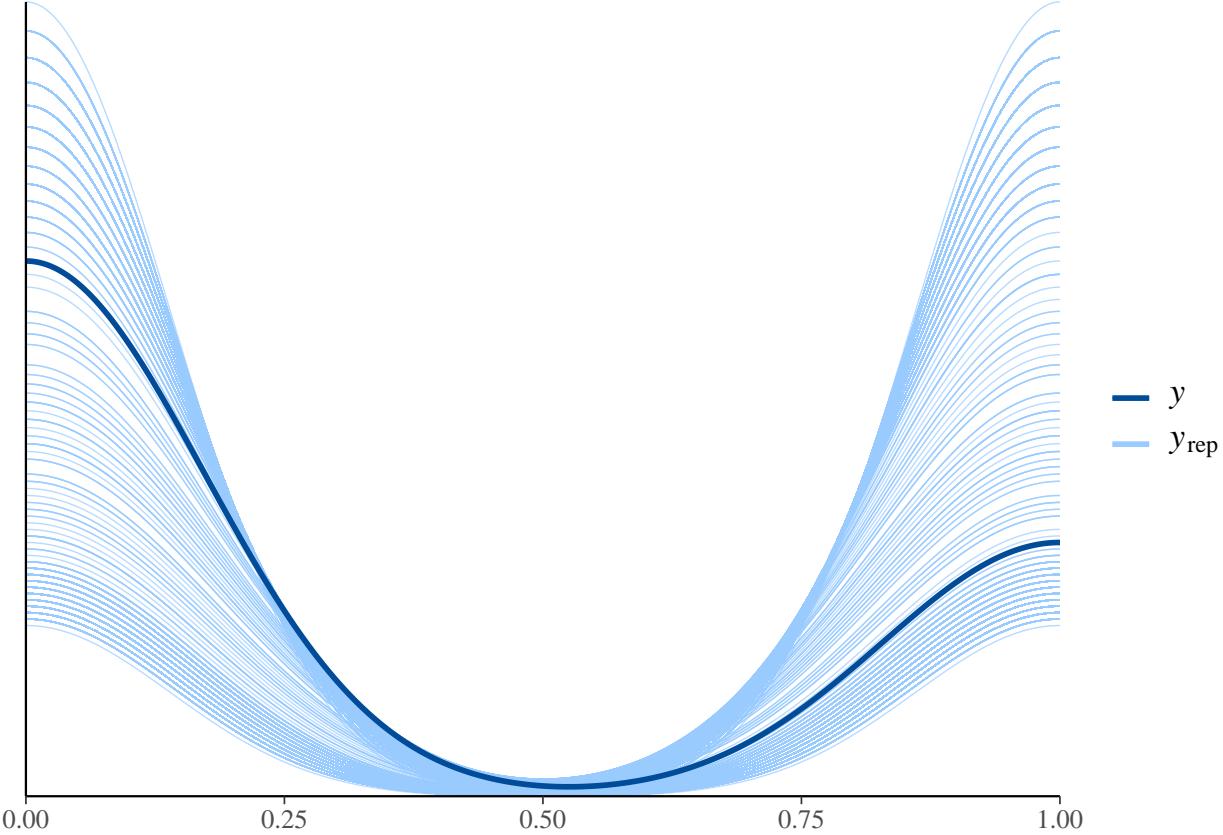
```
prior.feature_selected.basic <- brm(formula = DEATH_EVENT ~ age +
  ejection_fraction + serum_creatinine + serum_sodium + high_blood_pressure +
  creatinine_phosphokinase + diabetes + smoking + anaemia +
  sex, data = train.data, family = bernoulli(), prior = c(set_prior("cauchy(40,20)",
  class = "b", coef = "age"), set_prior("inv_gamma(1,5)", class = "b",
  coef = "serum_creatinine"), set_prior("beta(6,4)", class = "b",
  coef = "ejection_fraction"), set_prior("cauchy(0,4000)",
  class = "b", coef = "creatinine_phosphokinase"), set_prior("cauchy(0,200)",
  class = "b", coef = "serum_sodium"), set_prior("normal(.5, .5)",
  class = "b", coef = "anaemia"), set_prior("normal(.5, .5)",
  class = "b", coef = "diabetes"), set_prior("normal(.5, .5)",
  class = "b", coef = "smoking"), set_prior("normal(.5, .5)",
  class = "b", coef = "high_blood_pressure"), set_prior("normal(.5, .5)",
  class = "b", coef = "sex")), refresh = 0, control = list(adapt_delta = 0.99),
  sample_prior = "only")
yrep <- brms::posterior_predict(prior.feature_selected.basic,
  newdata = test.data, draws = 120)
ppc_dens_overlay(y = test.data$DEATH_EVENT, yrep = yrep)
```



Then use uninformative normal prior

```
prior.feature_selected.noninfo <- brm(formula = DEATH_EVENT ~
  age + ejection_fraction + serum_creatinine + serum_sodium +
  high_blood_pressure + creatinine_phosphokinase + diabetes +
  smoking + anaemia + sex, data = train.data, family = bernoulli(),
  prior = set_prior("normal(0,6000)", refresh = 0, control = list(adapt_delta = 0.99),
  sample_prior = "only")

yrep <- posterior_predict(prior.feature_selected.noninfo, newdata = test.data,
  draws = 120)
ppc_dens_overlay(y = test.data$DEATH_EVENT, yrep = yrep)
```



As we see above, we are first fitting the models by ignoring likelihood. After that we generate samples based on test data, and see how well our model generates data fitting for the test data. We see, that by using our chosen priors we get better fit, which is expected as we are using more informative priors.

6 Conclusion - Discussion problems and further improvements

In this project we aimed to discover how different values of features for patients could affect the death event given by an heart failure. We explored how different models and different approaches affect the results of our objective. Our first discovery were that some variables had an higher correlation to the death event and we looked at this relationship when selecting the variables for our reduced model. The second model consist on a full feature model. The third model is a hierarchical in which we used the same variables we used in the first model, the substantial change was represented that now patients are divided into 3 groups based on their age. The last model was created by using a different approach to the dataset, we considered time as outcome variable with respect to death event and we tried to predict the time with respect to death event. We compared the 4 models using different assessment criteria (accuracy, loo, \hat{R} and n_{eff}) and we can conclude that the hierarchical model performed the best. With the posterior performance assessment we can conclude that the model fairly good and we would recommended this model as an accurate predictor Future improvement could be looking at more advanced future engineering and differnt choice of priors.

6.1 Self-reflection

We learned to work as a group. We used GitHub for version management, and it was nice to see that with communication we had zero conflicts, even though we only worked on main-branch. Seems like we also stayed on time perfectly. At first we were having group meetings really early, but it was worth it because we had longer time to think about the problem and possible solutions.

Appendix

A. Stan code of full model

```
stancode(fit.full)

## // generated with brms 2.14.4
## functions {
## }
## data {
##   int<lower=1> N;  // total number of observations
##   int Y[N];  // response variable
##   int<lower=1> K;  // number of population-level effects
##   matrix[N, K] X;  // population-level design matrix
##   int prior_only;  // should the likelihood be ignored?
## }
## transformed data {
##   int Kc = K - 1;
##   matrix[N, Kc] Xc;  // centered version of X without an intercept
##   vector[Kc] means_X;  // column means of X before centering
##   for (i in 2:K) {
##     means_X[i - 1] = mean(X[, i]);
##     Xc[, i - 1] = X[, i] - means_X[i - 1];
##   }
## }
## parameters {
##   vector[Kc] b;  // population-level effects
##   real Intercept;  // temporary intercept for centered predictors
## }
## transformed parameters {
## }
## model {
##   // likelihood including all constants
##   if (!prior_only) {
##     target += bernoulli_logit_glm_lpmf(Y | Xc, Intercept, b);
##   }
##   // priors including all constants
##   target += cauchy_lpdf(b[1] | 40, 20);
##   target += beta_lpdf(b[2] | 6, 4);
##   target += inv_gamma_lpdf(b[3] | 1, 5);
##   target += cauchy_lpdf(b[4] | 0, 200);
##   target += normal_lpdf(b[5] | .5, .5);
##   target += cauchy_lpdf(b[6] | 0, 4000);
##   target += normal_lpdf(b[7] | .5, .5);
##   target += normal_lpdf(b[8] | .5, .5);
##   target += normal_lpdf(b[9] | .5, .5);
##   target += normal_lpdf(b[10] | .5, .5);
##   target += student_t_lpdf(Intercept | 3, 0, 2.5);
## }
## generated quantities {
##   // actual population-level intercept
##   real b_Intercept = Intercept - dot_product(means_X, b);
```

```
## }
```

B. Stan feature selected model

```
stancode(fit.feature_selected)

## // generated with brms 2.14.4
## functions {
## }
## data {
##   int<lower=1> N;  // total number of observations
##   int Y[N];  // response variable
##   int<lower=1> K;  // number of population-level effects
##   matrix[N, K] X;  // population-level design matrix
##   int prior_only;  // should the likelihood be ignored?
## }
## transformed data {
##   int Kc = K - 1;
##   matrix[N, Kc] Xc;  // centered version of X without an intercept
##   vector[Kc] means_X;  // column means of X before centering
##   for (i in 2:K) {
##     means_X[i - 1] = mean(X[, i]);
##     Xc[, i - 1] = X[, i] - means_X[i - 1];
##   }
## }
## parameters {
##   vector[Kc] b;  // population-level effects
##   real Intercept;  // temporary intercept for centered predictors
## }
## transformed parameters {
## }
## model {
##   // likelihood including all constants
##   if (!prior_only) {
##     target += bernoulli_logit_lpmf(Y | Xc, Intercept, b);
##   }
##   // priors including all constants
##   target += beta_lpdf(b[1] | 6,4);
##   target += inv_gamma_lpdf(b[2] | 1,1);
##   target += cauchy_lpdf(b[3] | 0,200);
##   target += cauchy_lpdf(b[4] | 40,20);
##   target += student_t_lpdf(Intercept | 3, 0, 2.5);
## }
## generated quantities {
##   // actual population-level intercept
##   real b_Intercept = Intercept - dot_product(means_X, b);
## }
```

C. Stan hierarchical model

```
stancode(fit.hierarchical)

## // generated with brms 2.14.4
## functions {
##   /* turn a vector into a matrix of defined dimension
##    * stores elements in row major order
##    * Args:
##    *   X: a vector
##    *   N: first dimension of the desired matrix
##    *   K: second dimension of the desired matrix
##    * Returns:
##    *   a matrix of dimension N x K
##   */
##   matrix as_matrix(vector X, int N, int K) {
##     matrix[N, K] Y;
##     for (i in 1:N) {
##       Y[i] = to_row_vector(X[((i - 1) * K + 1):(i * K)]);
##     }
##     return Y;
##   }
##   /* compute correlated group-level effects
##    * Args:
##    *   z: matrix of unscaled group-level effects
##    *   SD: vector of standard deviation parameters
##    *   L: cholesky factor correlation matrix
##    * Returns:
##    *   matrix of scaled group-level effects
##   */
##   matrix scale_r_cor(matrix z, vector SD, matrix L) {
##     // r is stored in another dimension order than z
##     return transpose(diag_pre_multiply(SD, L) * z);
##   }
## }
## data {
##   int<lower=1> N; // total number of observations
##   int Y[N]; // response variable
##   int<lower=1> K; // number of population-level effects
##   matrix[N, K] X; // population-level design matrix
##   // data for group-level effects of ID 1
##   int<lower=1> N_1; // number of grouping levels
##   int<lower=1> M_1; // number of coefficients per level
##   int<lower=1> J_1[N]; // grouping indicator per observation
##   // group-level predictor values
##   vector[N] Z_1_1;
##   vector[N] Z_1_2;
##   vector[N] Z_1_3;
##   vector[N] Z_1_4;
##   int<lower=1> NC_1; // number of group-level correlations
##   int prior_only; // should the likelihood be ignored?
## }
## transformed data {
```

```

##  int Kc = K - 1;
##  matrix[N, Kc] Xc; // centered version of X without an intercept
##  vector[Kc] means_X; // column means of X before centering
##  for (i in 2:K) {
##    means_X[i - 1] = mean(X[, i]);
##    Xc[, i - 1] = X[, i] - means_X[i - 1];
##  }
## }

## parameters {
##   vector[Kc] b; // population-level effects
##   real Intercept; // temporary intercept for centered predictors
##   vector<lower=0>[M_1] sd_1; // group-level standard deviations
##   matrix[M_1, N_1] z_1; // standardized group-level effects
##   cholesky_factor_corr[M_1] L_1; // cholesky factor of correlation matrix
## }

## transformed parameters {
##   matrix[N_1, M_1] r_1; // actual group-level effects
##   // using vectors speeds up indexing in loops
##   vector[N_1] r_1_1;
##   vector[N_1] r_1_2;
##   vector[N_1] r_1_3;
##   vector[N_1] r_1_4;
##   // compute actual group-level effects
##   r_1 = scale_r_cor(z_1, sd_1, L_1);
##   r_1_1 = r_1[, 1];
##   r_1_2 = r_1[, 2];
##   r_1_3 = r_1[, 3];
##   r_1_4 = r_1[, 4];
## }

## model {
##   // likelihood including all constants
##   if (!prior_only) {
##     // initialize linear predictor term
##     vector[N] mu = Intercept + rep_vector(0.0, N);
##     for (n in 1:N) {
##       // add more terms to the linear predictor
##       mu[n] += r_1_1[J_1[n]] * Z_1_1[n] + r_1_2[J_1[n]] * Z_1_2[n] + r_1_3[J_1[n]] * Z_1_3[n] + r_1_4[J_1[n]];
##     }
##     target += bernoulli_logit_glm_lpmf(Y | Xc, mu, b);
##   }
##   // priors including all constants
##   target += beta_lpdf(b[1] | 6,4);
##   target += cauchy_lpdf(b[2] | 0,5);
##   target += normal_lpdf(b[3] | 100,40);
##   target += student_t_lpdf(Intercept | 3, 0, 2.5);
##   target += student_t_lpdf(sd_1 | 3, 0, 2.5)
##   - 4 * student_t_lccdf(0 | 3, 0, 2.5);
##   target += std_normal_lpdf(to_vector(z_1));
##   target += lkj_corr_cholesky_lpdf(L_1 | 1);
## }

## generated quantities {
##   // actual population-level intercept
##   real b_Intercept = Intercept - dot_product(means_X, b);
##   // compute group-level correlations

```

```
##  corr_matrix[M_1] Cor_1 = multiply_lower_tri_self_transpose(L_1);
##  vector<lower=-1,upper=1>[NC_1] cor_1;
##  // extract upper diagonal of correlation matrix
##  for (k in 1:M_1) {
##    for (j in 1:(k - 1)) {
##      cor_1[choose(k - 1, 2) + j] = Cor_1[j, k];
##    }
##  }
## }
```

6.2 D. Stan death time analysis model

```
stancode(fit.weibull)

## // generated with brms 2.14.4
## functions {
## }
## data {
##   int<lower=1> N;    // total number of observations
##   vector[N] Y;      // response variable
##   int<lower=-1,upper=2> cens[N]; // indicates censoring
##   int<lower=1> K;    // number of population-level effects
##   matrix[N, K] X;   // population-level design matrix
##   int prior_only;   // should the likelihood be ignored?
## }
## transformed data {
##   int Kc = K - 1;
##   matrix[N, Kc] Xc; // centered version of X without an intercept
##   vector[Kc] means_X; // column means of X before centering
##   for (i in 2:K) {
##     means_X[i - 1] = mean(X[, i]);
##     Xc[, i - 1] = X[, i] - means_X[i - 1];
##   }
## }
## parameters {
##   vector[Kc] b; // population-level effects
##   real Intercept; // temporary intercept for centered predictors
##   real<lower=0> shape; // shape parameter
## }
## transformed parameters {
## }
## model {
##   // likelihood including all constants
##   if (!prior_only) {
##     // initialize linear predictor term
##     vector[N] mu = Intercept + Xc * b;
##     for (n in 1:N) {
##       // apply the inverse link function
##       mu[n] = exp(mu[n]) / tgamma(1 + 1 / shape);
##     }
##     for (n in 1:N) {
##       // special treatment of censored data
##       if (cens[n] == 0) {
##         target += weibull_lpdf(Y[n] | shape, mu[n]);
##       } else if (cens[n] == 1) {
##         target += weibull_lccdf(Y[n] | shape, mu[n]);
##       } else if (cens[n] == -1) {
##         target += weibull_lcdf(Y[n] | shape, mu[n]);
##       }
##     }
##   }
##   // priors including all constants
##   target += cauchy_lpdf(b[1] | 40, 20);
```

```
## target += normal_lpdf(b[2] | .5, .5);
## target += normal_lpdf(b[3] | .5, .5);
## target += student_t_lpdf(Intercept | 3, 4.8, 2.5);
## target += gamma_lpdf(shape | 0.01, 0.01);
## }
## generated quantities {
##   // actual population-level intercept
##   real b_Intercept = Intercept - dot_product(means_X, b);
## }
```

References

- [1] Ejection fraction heart failure measurement, 2017.
- [2] Ahmad T, Munir A, Bhatti SH, Aftab M, Raza MA. Survival analysis of heart failure patients: A case study. 2017. doi: <https://doi.org/10.1371/journal.pone.0181001>.
- [3] Aki Vehtari, Andrew Gelman, Jonah Gabry. Practical bayesian model evaluation using leave-one-out cross-validation and waic. 2017.
- [4] Christine Case-Lo. Blood sodium test, 2018. URL <https://www.healthline.com/health/sodium-blood>.
- [5] Gregg D, Goldschmidt-Clermont P. J. Platelets and cardiovascular disease. *Journal of the American Heart Association*, 108, 2003. doi: <https://doi.org/10.1161/01.CIR.0000086897.15588.4B>.
- [6] Roshan Patel Ravinder S. Aujla. Creatine phosphokinase. *StatPearls*, 2020. URL <https://www.ncbi.nlm.nih.gov/books/NBK546624/>.
- [7] Roth Erica. Creatinine blood test, 2019. URL <https://www.healthline.com/health/creatinine-blood>.