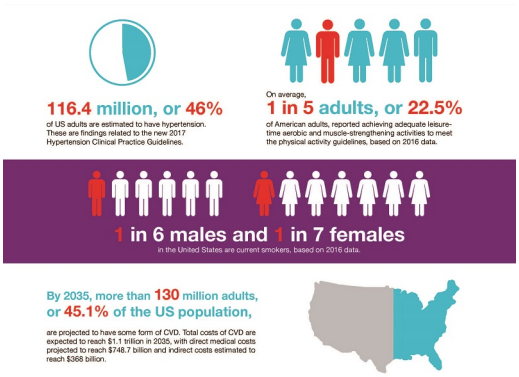
BDA Project - Heart Failure

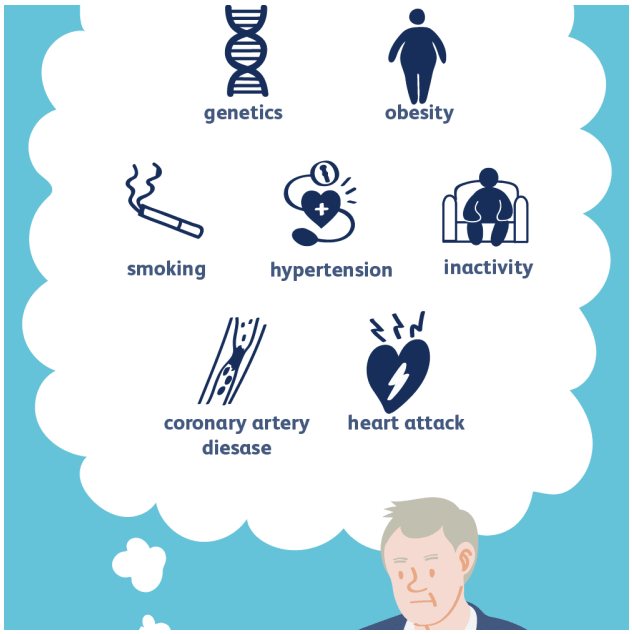
Khoa Nguyen & Phi Dang

1. Introduction

a) Problem

CVDs (or Cardiovascular diseases) is a general term indicating a set of heart and blood vessels disorders. Anually, CVDs causes about 18 million deaths, accounting for a third of total deaths all over the words. 85% of the number are due to heart attack and stroke which becomes one of the most common causes of death. People with or at the risk of CVDs require diagnosis as early as possible to receive treatment by means of therapy and pharmaceutical drugs as needed.





b) Motivation

The project aims to propose a clear understanding what is the cause of heart failure. First, it presents several data exploration processes with explanatory variable analysis and visualizations. Second, it indentifies important features to build up a prediction model together with some diagnostics. Finally, it compares and analyse model results with different settings accompanied by some potential improvement recommendations.

c) Main modeling idea

First, we perform some feature selection with several models. From acquired sets of feature, we plug them in a Logistic Regression model and see the compare the y_pred with the y_true. This is a classification task as the response variable is Boolean. We also use Bernoulli distribution to predict y_pred value.

2. Dataset

a) Description

The data is from Kaggle (https://www.kaggle.com/andrewmvd/heart-failure-clinical-data). There are several papers that uses the same dataset with a Machine Learning approach to predict heart failure such as Decision Tree, Neural Network, Naive Bayes and so on. We use a more statistical approach to the problem and see whether we can predict the heart failure rate with the same level of accuracy as the other papers. However, since the prediction model that we use here is Logistics Regression, there is a high chance that other paper uses it as well. Also, for feature selection, beside Bayesian Inference and LASSO, we also apply a Machine Learning model which is a Tree Classifier in order to test several combinations of features so that we can compare them together. In conclusion, for this project, we used 3 variable selection model: TreeClassifier, Bayesian Inference and LASSO and 1 non-linear model which is Logistics Regression.

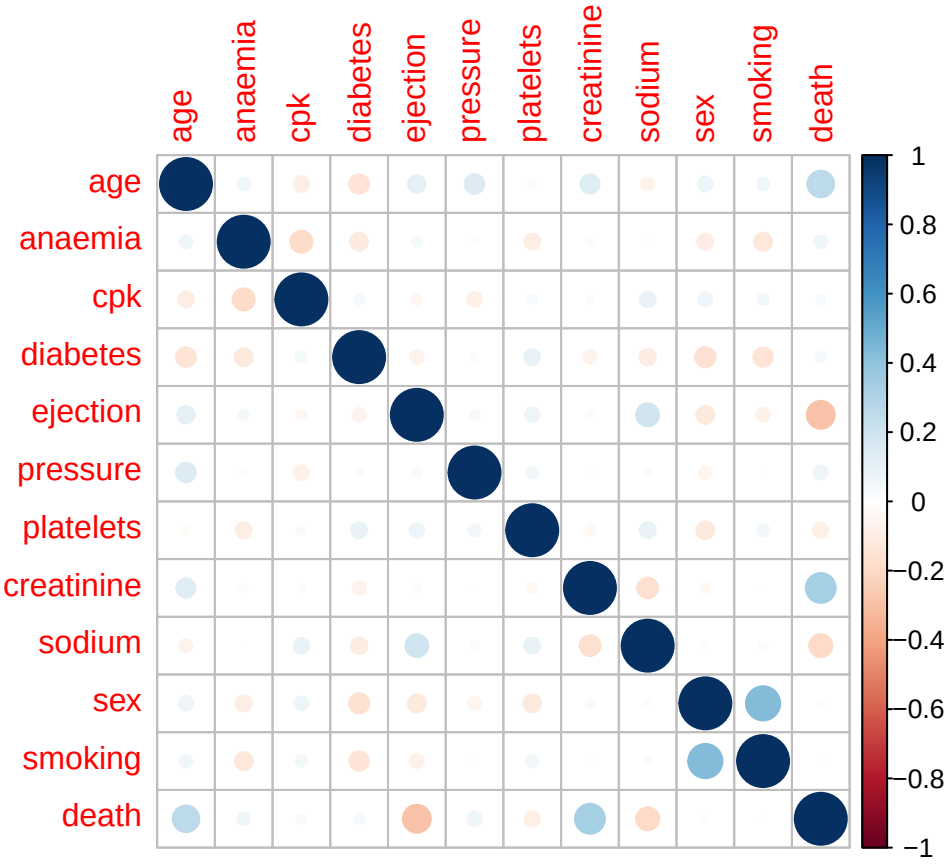
The dataset consists of the following features:
* age - Age of the patient
+ mean = 60.8
+ std = 11.9
* anaemia - Decrease of red blood cells
+ boolean (0 = No, 1 = Yes)
* cpk - Level of the CPK enzyme in the blood
+ mean = 582
+ std = 969
* diabetes - If the patient has diabetes
+ boolean (0 = No, 1 = Yes)
* ejection - Percentage of blood leaving the heart at each contraction
+ mean = 38.1%
+ std = 11.8%
* pressure - If the patient has hypertension
+ boolean (0 = No, 1 = Yes)
* platelets - Platelets in the blood
+ mean = 283k
+ std = 97.6k
* creatinine - Level of serum creatinine in the blood
+ mean = 1.39
+ std = 1.03
* sodium - Level of serum sodium in the blood
+ mean = 137
+ std = 4.41
* sex - Woman or man
+ binary (Male = 1, Female =0)
* smoking - If the patient smokes
+ boolean (0 = No, 1 = Yes)
* time - Follow-up period
+ mean = 130, std = 77.5
* death - (Response variable)
+ boolean (0 = No, 1 = Yes)

Although the feature “time”, according to the correlation plot below, has a very high correlation with our dependent variable, it is considered not to be one of the feature that can cause heart failure and therefore be omitted from the dataset.

b) Visualization

From the correlation plot, we see that the variables age, ejection_fraction, serum_creatine and serum_sodium seems to have high correlation with our independent variable which is DEATH_EVENT. Therefore, we decided to split the logistic regression model into 2 versions, one with our proposed variable only and another with all variable in order to evaluate both model's performance.

```
corrplot(cor(data))
```

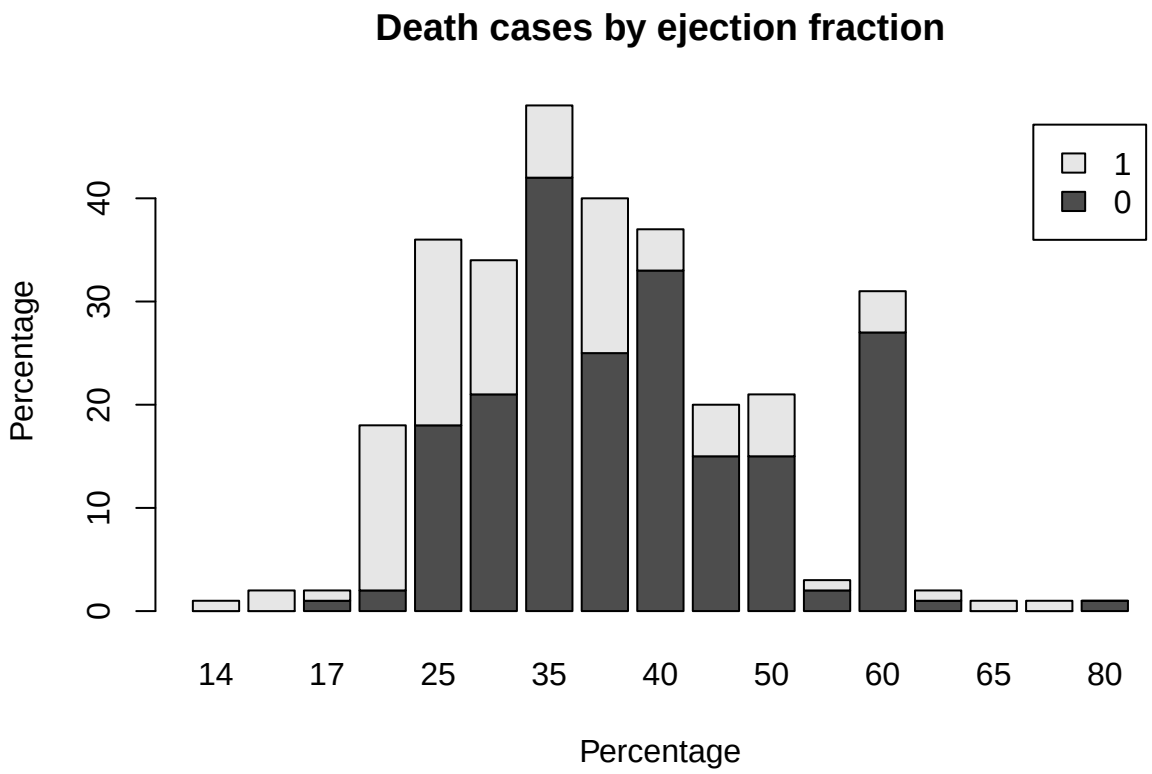


Besides correlation relationship between the variable, we also want to investigate the distributions of the features as well as the percentage of death by each feature.

```
barplot(table(death, age),
        main = "Death cases by Age",
        xlab = "Age", ylab = "Number of People",
        legend = rownames(table(death, age)))
```

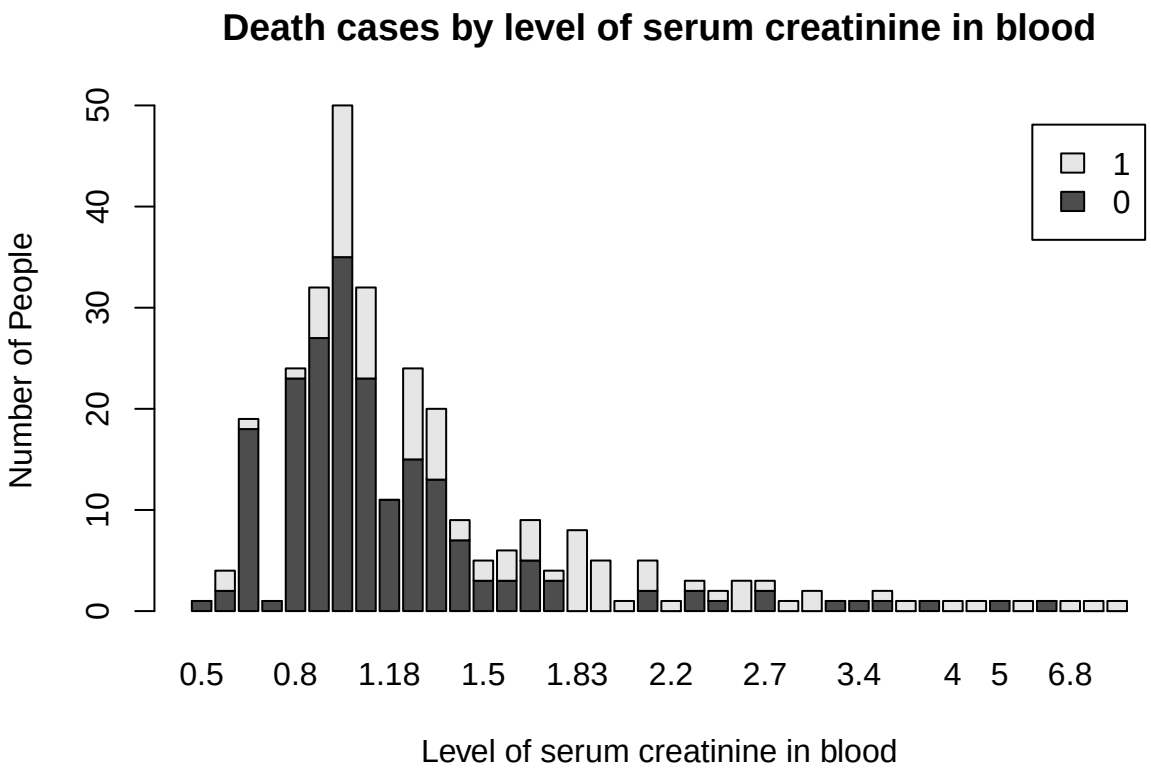
The age range from around 40-95 which is quite accurate since Cardiovascular disease is commonly seen in middle-age group. Although the ages seems to vary, the age still resembles normal distribution, especially when grouping the ages as a group.

```
barplot(table(death, ejection),
        main = "Death cases by ejection fraction",
        xlab = "Percentage", ylab = "Percentage",
        legend = rownames(table(death, ejection)))
```



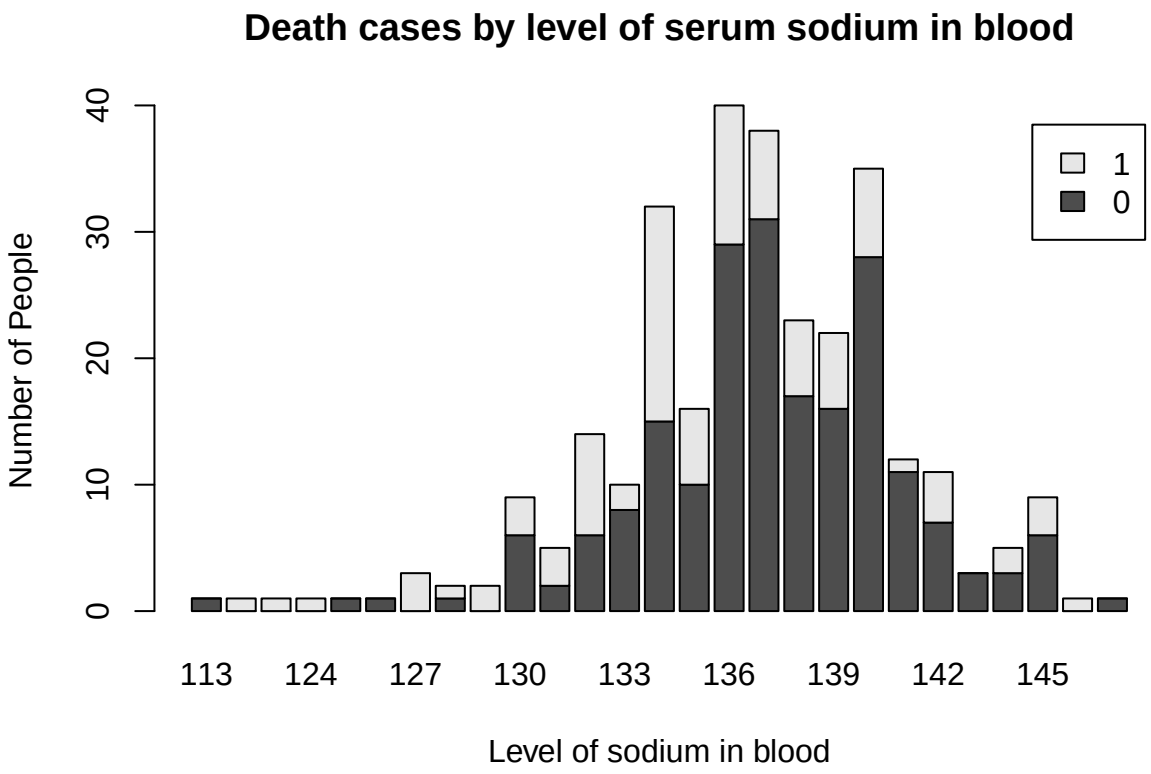
The percentage of ejection fraction is focus around 20-50 which is a common phenomenon of Cardiovascular disease with blood cells get blocked and cannot travel freely inside the vein. The graph resembles a normal distribution as well.

```
barplot(table(death, creatinine),
        main = "Death cases by level of serum creatinine in blood",
        xlab = "Level of serum creatinine in blood", ylab = "Number of People",
        legend = rownames(table(death, creatinine)))
```



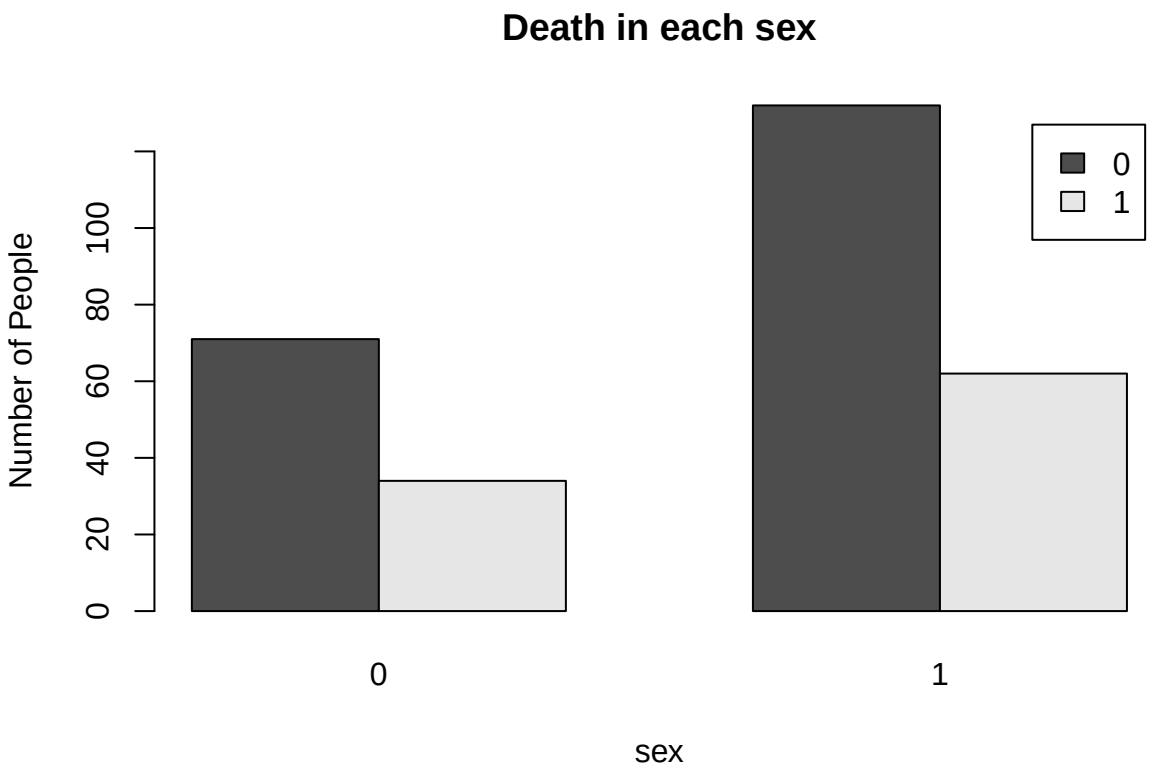
According to medical research, level of creatinine in blood can also increase the mortality rate of CVD patient. The distribution seen on the graph can be considered normal since the normal range for creatinine level is the blood is 0.84-1.21 mg/dL. The percentage of death is also reflected quite accurately on the graph. The distribution of creatinine feature can be considered left skewed normal distribution.

```
barplot(table(death, sodium),
        main = "Death cases by level of serum sodium in blood",
        xlab = "Level of sodium in blood", ylab = "Number of People",
        legend = rownames(table(death, sodium)))
```



According to medical research, sodium level to high in blood can cause the blood to thicken and harder to flow which results in higher blood pressure. This can become more severe if one has CVD. Although in the data, the sodium level concentrate around 135-145 mEq/L which is normal sodium level in blood, the feature can still be considered as a factor that causes heart failure. From the plot, we can also see that the level of sodium in blood follows a normal distribution but a little right skewed.

```
barplot(table(death, sex),
        main = "Death in each sex",
        xlab = "sex", ylab = "Number of People",
        legend = rownames(table(death, sex)), beside = TRUE)
```



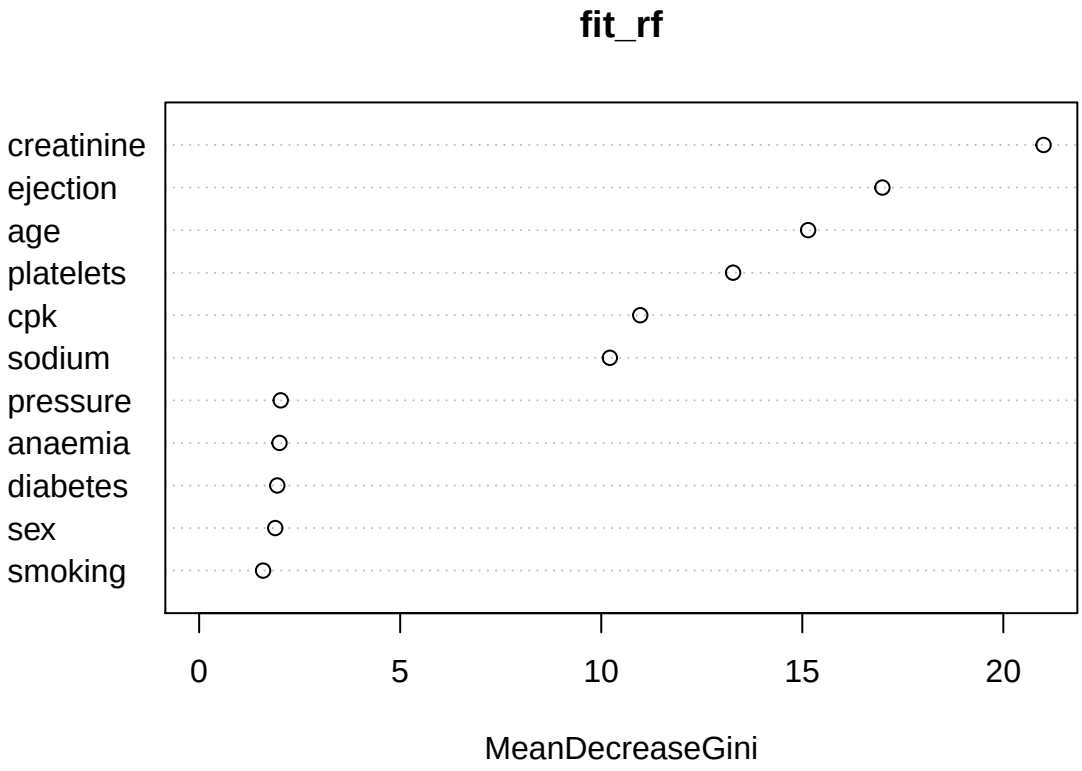
This plot shows the death ration in each sex. We can see that both men and women has the same alive to death ratio.

3. Feature selection

Beside using correlation matrix for feature selection, we decided to also implement TreeClassifier and variable selection based on Bayesian inference. The goal is to compare the results from these model with the full model and the selected variable from correlation matrix.
Ultimately, we want to study what can be considered "direct" causes to heart failure.

TreeClassifier

```
fit_rf = randomForest(factor(death)~., data=data)
varImpPlot(fit_rf)
```



From the plot, we can see that the first 3 features selected by TreeClassifier and the last one which is sodium is similar to the 4 features selected by looking at the correlation matrix. The are 2 different features which are platelets and cpk.

Bayesian Inference

Firstly, we fit the full model using brm function and analyze the performance of each feature:

```
fit = stan_glm(death~age+anaemia+cpk+diabetes+ejection+pressure+platelets+creatinine+sodium+sex+smoking
          data = data,
          family = binomial("logit"),
          prior = default_prior_coef(family),
          refresh=0,
          verbose = FALSE)
summary(fit)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      death ~ age + anaemia + cpk + diabetes + ejection + pressure +
##     platelets + creatinine + sodium + sex + smoking
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
```

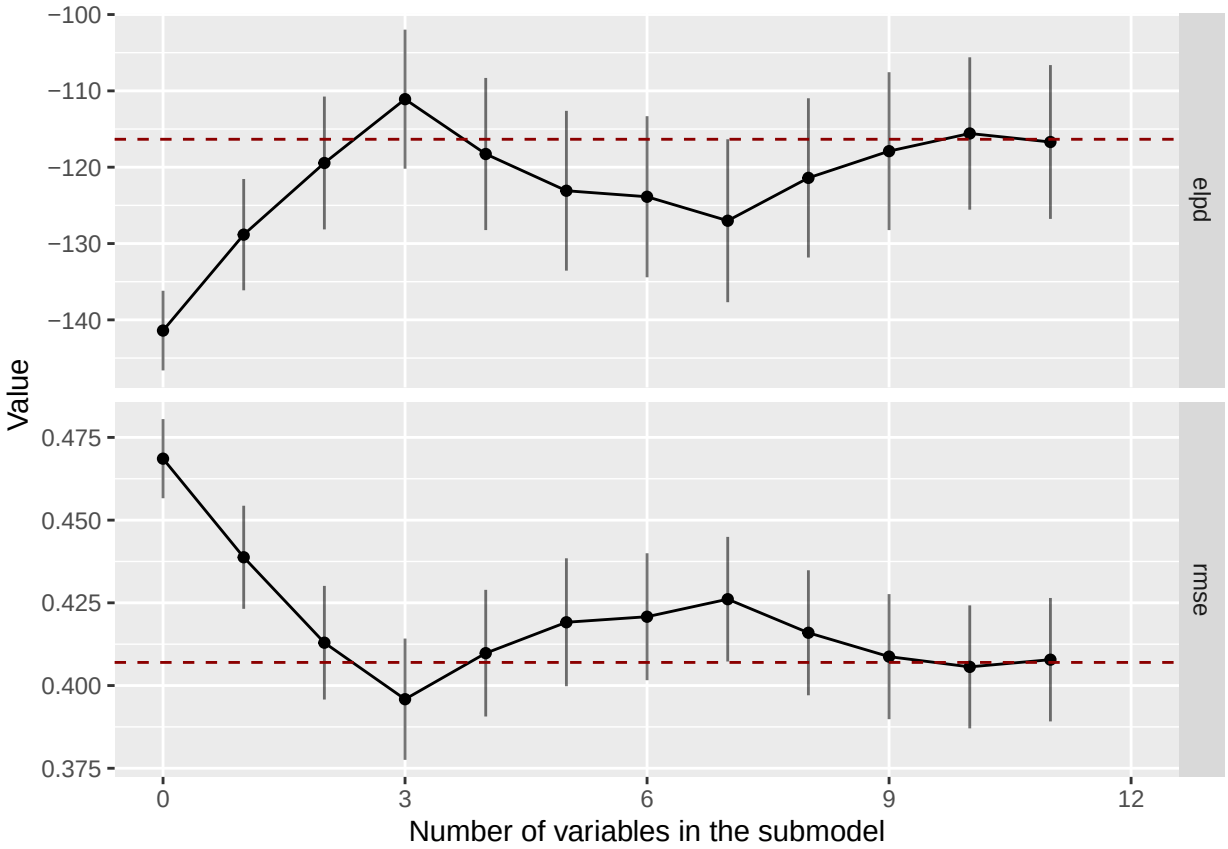
```
##  observations: 224
##  predictors:   12
##
## Estimates:
##               mean   sd   10%   50%   90%
## (Intercept)  2.2    5.7 -5.1   2.2   9.5
## age          0.1    0.0  0.0   0.1   0.1
## anaemia      0.5    0.4  0.0   0.5   1.0
## cpk          0.0    0.0  0.0   0.0   0.0
## diabetes     0.4    0.4  0.0   0.4   0.9
## ejection    -0.1    0.0 -0.1  -0.1  -0.1
## pressure     0.4    0.4 -0.1   0.4   0.9
## platelets    0.0    0.0  0.0   0.0   0.0
## creatinine   0.9    0.2  0.7   0.9   1.2
## sodium       0.0    0.0 -0.1   0.0   0.0
## sex         -0.3    0.4 -0.9  -0.3   0.2
## smoking      0.0    0.4 -0.5   0.0   0.6
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.3    0.0  0.3   0.3   0.4
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.1  1.0  4441
## age           0.0  1.0  3350
## anaemia       0.0  1.0  3766
## cpk           0.0  1.0  4225
## diabetes      0.0  1.0  4054
## ejection      0.0  1.0  3744
## pressure      0.0  1.0  4235
## platelets     0.0  1.0  3903
## creatinine    0.0  1.0  4063
## sodium        0.0  1.0  4461
## sex           0.0  1.0  3194
## smoking       0.0  1.0  3759
## mean_PPD      0.0  1.0  4929
## log-posterior 0.1  1.0  2031
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

The model Rhat and n_eff looks good. However, we want to perform feature selection to not only reduce the computing cost but also to investigate whether the model performs the same but with fewer features.

Perform feature selection:

After fit_cv, we plot it to see the optimal number of features, and find out what they are:

```
plot(fit_cv, stats = c('elpd', 'rmse'))
```



According to the plot, it seems like with only 3 variables, the model performs even better than with 11 variables. Now we investigate what are the 3 variables:

```
fit_cv$solution_terms[1:3]
```

```
## [1] "creatinine" "ejection"   "age"
```

The features proposed by fit_cv are very similar to those from the correlation matrix. We will see if their performance is better than the features previously selected.

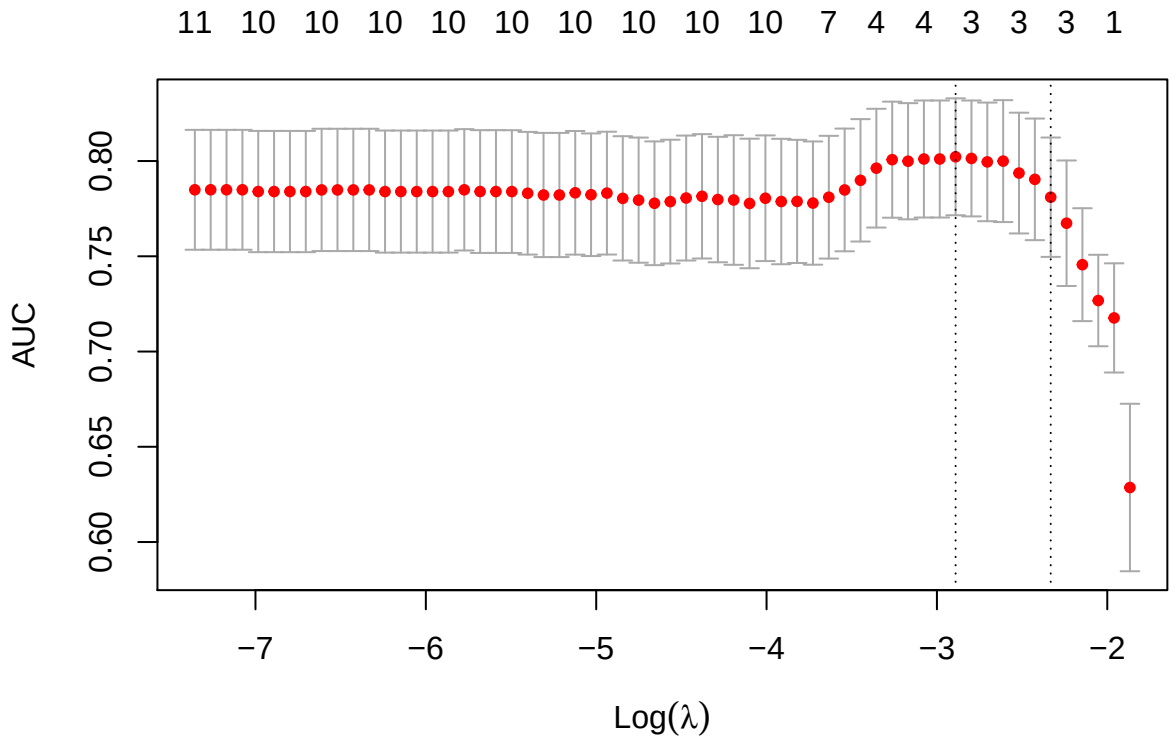
LASSO

Here, we also test the output of LASSO to see whether the combination of variables is different.

```
x = as.matrix(data[,-12])
y = data$death
set.seed(123)
cv.lasso = cv.glmnet(x, y, family='binomial', alpha=1, parallel=TRUE, standardize=TRUE, type.measure='au
```

```
## Warning: executing %dopar% sequentially: no parallel backend registered
```

```
plot(cv.lasso)
```




```
cat('Min Lambda: ', cv.lasso$lambda.min, '\n 1Sd Lambda: ', cv.lasso$lambda.1se, '\n')
```

```
## Min Lambda:  0.05555967
##  1Sd Lambda:  0.0970921
```

```
df_coef = round(as.matrix(coef(cv.lasso, s=cv.lasso$lambda.min)), 2)
df_coef[df_coef[, 1] != 0, ]
```

```
## (Intercept)         age    ejection  creatinine
##       -1.50        0.03       -0.04        0.41
```

Since LASSO has the same combination of Bayesian Inference, we treat the outcome of the two as one combination only.

4. Priors

For the model, we decided that for $\beta_i, i \geq 1$ to have prior obtain from the multivariate normal distribution with mean vector and covariance matrix calculated based on the number of feature selected.

5. Models

After several feature selection method, we apply our different combinations of features into the model. First combination being the features chosen from Bayesian Inference analysis, second are the ones chosen from

correlation matrix and the thrid one is chosen by TreeClasifier. Finally, our last one is the full model, where we use every features as a control.

For modeling, we work on Logistics Regression model. Since our independent variable has Boolean type, we use Bernoulli distribution for the independent variable and Logistics Regression is good at modeling binary dependent variable.

```
data {
  int<lower=1> N; //number of observation
  int<lower=1> M; //number of explanatory variable
  int Y[N];       //independent variable
  matrix[N, M] X; //explanatory variables

  int<lower=1> N_test; //number of test observation
  int Y_true[N_test]; //true Y
  matrix[N_test, M] X_test; //test train

  vector[M] mu; //mean vector
  cov_matrix[M] sigma; //covariance matrix
}
parameters {
  vector[M] beta;
}
model {
  beta ~ multi_normal(mu, sigma); //prior based on distribution of the explanatory variables
  Y ~ bernoulli_logit(X*beta); //bernoulli logit since Y has boolean type
}
generated quantities {
  vector[N_test] Y_pred = inv_logit(X_test * beta); //Y_pred
  vector[N] log_lik;
  for (n in 1:N) {
    log_lik[n] = bernoulli_logit_lpmf(Y[n] |X[n, ] * beta);
  }
}
```

6. Convergence diagnostics

```
print("Rhat values: ")
```

```
## [1] "Rhat values: "
```

```
rhat(fit_selected_1)
```

```
##      beta[1]      beta[2]      beta[3]    Y_pred[1]    Y_pred[2]    Y_pred[3]
##    1.0010238    1.0014955    0.9997227    1.0006613    1.0007860    1.0008227
##    Y_pred[4]    Y_pred[5]    Y_pred[6]    Y_pred[7]    Y_pred[8]    Y_pred[9]
##    1.0006972    1.0008394    1.0004119    1.0010953    1.0006692    1.0002154
##   Y_pred[10]   Y_pred[11]   Y_pred[12]   Y_pred[13]   Y_pred[14]   Y_pred[15]
##    1.0011500    1.0010390    0.9999931    1.0004085    1.0004830    1.0003932
##   Y_pred[16]   Y_pred[17]   Y_pred[18]   Y_pred[19]   Y_pred[20]   Y_pred[21]
```

```
##    1.0008605    1.0001242    1.0003344    1.0011637    0.9997245    1.0007178
##   Y_pred[22]   Y_pred[23]   Y_pred[24]   Y_pred[25]   Y_pred[26]   Y_pred[27]
##    1.0009372    1.0003363    0.9999183    1.0005590    1.0002565    1.0001242
##   Y_pred[28]   Y_pred[29]   Y_pred[30]   Y_pred[31]   Y_pred[32]   Y_pred[33]
##    1.0012391    1.0010002    1.0008260    1.0012077    1.0006006    1.0004907
##   Y_pred[34]   Y_pred[35]   Y_pred[36]   Y_pred[37]   Y_pred[38]   Y_pred[39]
##    1.0000157    1.0007477    1.0002039    1.0002536    1.0008601    0.9994696
##   Y_pred[40]   Y_pred[41]   Y_pred[42]   Y_pred[43]   Y_pred[44]   Y_pred[45]
##    1.0003114    1.0002955    1.0002270    1.0007227    0.9993080    1.0002284
##   Y_pred[46]   Y_pred[47]   Y_pred[48]   Y_pred[49]   Y_pred[50]   Y_pred[51]
##    0.9994368    1.0002555    1.0003780    1.0001582    1.0002416    1.0004488
##   Y_pred[52]   Y_pred[53]   Y_pred[54]   Y_pred[55]   log_lik[1]   log_lik[2]
##    1.0008155    1.0003322    1.0011124    1.0009238    1.0008109    1.0008340
##   log_lik[3]   log_lik[4]   log_lik[5]   log_lik[6]   log_lik[7]   log_lik[8]
##    1.0007601    0.9997749    1.0002480    1.0000941    1.0005153    1.0008105
##   log_lik[9]  log_lik[10]  log_lik[11]  log_lik[12]  log_lik[13]  log_lik[14]
##    1.0009304    1.0003507    1.0002407    1.0006787    1.0003433    1.0007589
##  log_lik[15]  log_lik[16]  log_lik[17]  log_lik[18]  log_lik[19]  log_lik[20]
##    1.0008984    1.0010186    1.0005587    1.0003588    1.0014306    1.0007439
##  log_lik[21]  log_lik[22]  log_lik[23]  log_lik[24]  log_lik[25]  log_lik[26]
##    1.0001649    1.0005520    1.0002829    1.0002109    1.0002659    1.0012069
##  log_lik[27]  log_lik[28]  log_lik[29]  log_lik[30]  log_lik[31]  log_lik[32]
##    1.0002741    1.0006356    1.0002625    1.0005966    1.0005301    1.0010725
##  log_lik[33]  log_lik[34]  log_lik[35]  log_lik[36]  log_lik[37]  log_lik[38]
##    1.0002754    1.0007964    1.0003476    1.0004284    1.0006162    1.0001792
##  log_lik[39]  log_lik[40]  log_lik[41]  log_lik[42]  log_lik[43]  log_lik[44]
##    1.0012108    1.0009463    1.0006732    1.0004385    1.0005251    1.0011158
##  log_lik[45]  log_lik[46]  log_lik[47]  log_lik[48]  log_lik[49]  log_lik[50]
##    1.0010595    1.0007540    0.9999722    1.0008266    1.0003822    1.0003164
##  log_lik[51]  log_lik[52]  log_lik[53]  log_lik[54]  log_lik[55]  log_lik[56]
##    1.0001845    1.0007107    1.0004608    1.0004666    1.0004257    1.0004287
##  log_lik[57]  log_lik[58]  log_lik[59]  log_lik[60]  log_lik[61]  log_lik[62]
##    1.0001738    1.0008469    1.0004092    1.0000406    1.0006956    1.0003739
##  log_lik[63]  log_lik[64]  log_lik[65]  log_lik[66]  log_lik[67]  log_lik[68]
##    1.0012591    1.0009724    1.0008423    1.0006545    1.0011976    1.0000966
##  log_lik[69]  log_lik[70]  log_lik[71]  log_lik[72]  log_lik[73]  log_lik[74]
##    1.0009577    1.0008434    1.0003051    1.0008973    1.0002834    1.0007601
##  log_lik[75]  log_lik[76]  log_lik[77]  log_lik[78]  log_lik[79]  log_lik[80]
##    1.0008254    1.0004543    1.0005185    1.0008756    1.0006919    1.0012072
##  log_lik[81]  log_lik[82]  log_lik[83]  log_lik[84]  log_lik[85]  log_lik[86]
##    1.0012282    0.9997534    1.0001899    0.9999712    1.0008823    1.0004092
##  log_lik[87]  log_lik[88]  log_lik[89]  log_lik[90]  log_lik[91]  log_lik[92]
##    1.0001714    1.0009119    1.0006334    1.0008607    1.0011605    1.0007121
##  log_lik[93]  log_lik[94]  log_lik[95]  log_lik[96]  log_lik[97]  log_lik[98]
##    1.0001654    1.0002961    1.0003734    1.0009567    1.0003455    1.0009011
##  log_lik[99] log_lik[100] log_lik[101] log_lik[102] log_lik[103] log_lik[104]
##    0.9999489    1.0008096    1.0003581    0.9997366    1.0002129    1.0005252
## log_lik[105] log_lik[106] log_lik[107] log_lik[108] log_lik[109] log_lik[110]
##    1.0009059    1.0005024    1.0006762    1.0007192    0.9994784    1.0004482
## log_lik[111] log_lik[112] log_lik[113] log_lik[114] log_lik[115] log_lik[116]
##    1.0002333    1.0003793    1.0007852    1.0000966    1.0000904    1.0008029
## log_lik[117] log_lik[118] log_lik[119] log_lik[120] log_lik[121] log_lik[122]
##    1.0011865    1.0004301    1.0010078    1.0008023    1.0012943    0.9996746
## log_lik[123] log_lik[124] log_lik[125] log_lik[126] log_lik[127] log_lik[128]
```

```
##    1.0011168    1.0007481    1.0006276    1.0006027    1.0008041    0.9993679
## log_lik[129] log_lik[130] log_lik[131] log_lik[132] log_lik[133] log_lik[134]
##    1.0004715    1.0006726    1.0001334    1.0007766    0.9999921    1.0011802
## log_lik[135] log_lik[136] log_lik[137] log_lik[138] log_lik[139] log_lik[140]
##    1.0003729    1.0009474    1.0008823    1.0007435    1.0007121    1.0009158
## log_lik[141] log_lik[142] log_lik[143] log_lik[144] log_lik[145] log_lik[146]
##    1.0004994    1.0010298    1.0004202    1.0007164    1.0003682    1.0002733
## log_lik[147] log_lik[148] log_lik[149] log_lik[150] log_lik[151] log_lik[152]
##    1.0011868    1.0002275    1.0000305    1.0008497    1.0003361    1.0005404
## log_lik[153] log_lik[154] log_lik[155] log_lik[156] log_lik[157] log_lik[158]
##    1.0002867    1.0003500    1.0007373    1.0010503    1.0000492    1.0011969
## log_lik[159] log_lik[160] log_lik[161] log_lik[162] log_lik[163] log_lik[164]
##    1.0003727    0.9994637    1.0007076    0.9995051    1.0009259    1.0011215
## log_lik[165] log_lik[166] log_lik[167] log_lik[168] log_lik[169] log_lik[170]
##    1.0011655    1.0002210    1.0003844    1.0003027    1.0002642    1.0007430
## log_lik[171] log_lik[172] log_lik[173] log_lik[174] log_lik[175] log_lik[176]
##    1.0001925    1.0003095    1.0011976    0.9998537    1.0004263    1.0006970
## log_lik[177] log_lik[178] log_lik[179] log_lik[180] log_lik[181] log_lik[182]
##    0.9993881    1.0008918    1.0004276    1.0012457    1.0011282    1.0008053
## log_lik[183] log_lik[184] log_lik[185] log_lik[186] log_lik[187] log_lik[188]
##    1.0002113    0.9993894    1.0008108    1.0002265    1.0003476    1.0008013
## log_lik[189] log_lik[190] log_lik[191] log_lik[192] log_lik[193] log_lik[194]
##    1.0006082    1.0007998    1.0006362    1.0008395    1.0005492    1.0001644
## log_lik[195] log_lik[196] log_lik[197] log_lik[198] log_lik[199] log_lik[200]
##    1.0011196    1.0002036    1.0006196    1.0009231    1.0000299    1.0007601
## log_lik[201] log_lik[202] log_lik[203] log_lik[204] log_lik[205] log_lik[206]
##    1.0002924    1.0004297    1.0002724    1.0008519    1.0000951    1.0004229
## log_lik[207] log_lik[208] log_lik[209] log_lik[210] log_lik[211] log_lik[212]
##    1.0001984    1.0008041    1.0011241    1.0004199    1.0006124    1.0002891
## log_lik[213] log_lik[214] log_lik[215] log_lik[216] log_lik[217] log_lik[218]
##    1.0004386    1.0010816    1.0006882    1.0006468    1.0002107    1.0012308
## log_lik[219] log_lik[220] log_lik[221] log_lik[222] log_lik[223] log_lik[224]
##    1.0008773    1.0006486    1.0001243    1.0006047    1.0007946    1.0008816
##         lp__
##    1.0018883
```

```
print("Check divergences: ")
```

```
## [1] "Check divergences: "
```

```
check_divergences(fit_selected_1)
```

```
## 0 of 4000 iterations ended with a divergence.
```

```
print("Check treedepth:")
```

```
## [1] "Check treedepth:"
```

```
check_treedepth(fit_selected_1)
```

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

```r
print("n_eff values: ")
```

```
## [1] "n_eff values: "
```

```r
loo(fit_selected_1)$diagnostics$n_eff
```

```
##   [1] 1615.2973 2173.2919 1977.7044 2516.3602  903.6192 2979.9637 1926.3191
##   [8] 1747.8146 2147.9616 2581.7191 4525.6713 1911.6346 1823.8658 1734.5198
##  [15] 2767.0445 1307.0777 2464.7713 3156.6037 1209.2981 1650.9808 5160.9257
##  [22] 2951.7143 3856.8508 2751.3874 3696.9216 1696.3204 2819.6739 1992.6337
##  [29] 3900.9048 2559.4462 2293.7366 1552.0326 3343.0938 1464.1982 3506.3679
##  [36] 2393.8145 2379.3070 5056.1160 1424.5160 1377.6429 1950.4125 3150.9965
##  [43] 2597.3434 1897.9251 2228.6949 1931.4233 4168.6221 1626.9438 2873.9213
##  [50] 2255.5569 3457.0865 2034.6024 2768.6814 2805.6250 3857.0984 3314.6636
##  [57] 4866.0347 1738.8958 3236.2200 3066.7803 1662.2736 2555.8236 1660.2610
##  [64] 2486.8456 1722.7950 2204.5407 1801.9593 2892.3718 1918.1843 1592.9948
##  [71] 3717.8898 1574.0880 3466.0829 1791.1405 1553.2687 2230.6327 3458.6469
##  [78] 2101.7676 2211.9209 1784.3210 1582.8554 3143.3311 3544.2218 2259.6477
##  [85] 1963.0385 3236.2200 3593.6574 1583.6804 2204.0817 1530.5335 1937.7149
##  [92] 1935.1084 3402.1259 3189.8345 1433.8176 1767.3211 1880.2986 1443.5368
##  [99] 2217.3792 2131.2719 2866.1531 2752.8993 3996.4836 2772.0794 1435.8235
## [106] 2729.5640 2047.0358 2100.4485 1088.8245 2762.3326 3418.3371 2642.7661
## [113] 1652.6632 2892.3718 2513.5700 1555.2833 1700.7963 2933.1790 1341.9870
## [120] 1939.2431 1665.8496 2931.8319 1389.0032 2472.0432 1790.2482 2300.8768
## [127] 1734.8476 1433.6955 2421.4597 1919.7453 3196.9318 2073.8225 2361.1194
## [134] 1399.6159 3280.1574 1378.6408 1963.0385 1753.0601 1935.1084 1125.2636
## [141] 2612.9085 1333.7906 2646.6019 2145.9684 2753.5338 4833.9095 1862.2115
## [148] 3904.1468 1927.9956 2036.9883 3198.9796 2236.6298 2987.6863 3108.2399
## [155] 2017.2398 2072.7377 1958.6374 1405.1388 2710.8333 1901.2986 2562.7676
## [162] 2167.7354 1767.5943 2067.3967 1119.4859 3842.5080 5235.1744 3729.3054
## [169] 3789.9791 2035.9574 3836.3357 2214.5959 1801.9593 1728.0272 3014.8204
## [176] 2149.3759 1350.7570 1451.5740 2726.6651 1544.6074 2081.1540 1782.6215
## [183] 3210.8041 1749.2116 1616.7571 4168.0710 3506.3679 1694.1459 2221.7962
## [190] 1637.1711 2313.3251 1549.0904 1766.8780 1912.1125 1660.2180 4266.7608
## [197] 2370.9734 1418.8886 4576.3157 1791.1405 3593.6761 2381.8951 3141.7408
## [204] 1425.1996 5017.8338 1772.6261 3230.3190 1734.8476 1375.9176 2545.4257
## [211] 2367.2858 2224.8287 2827.5653 1818.2051 1925.2696 2160.1747 2669.9828
## [218] 1866.1014 1828.3806 3101.9080 2457.8611 1627.2591 2501.2532 1553.7148
```

```r
print("Performance of k_hat values and elpd_loo")
```

```
## [1] "Performance of k_hat values and elpd_loo"
```

```r
loo(fit_selected_1)
```

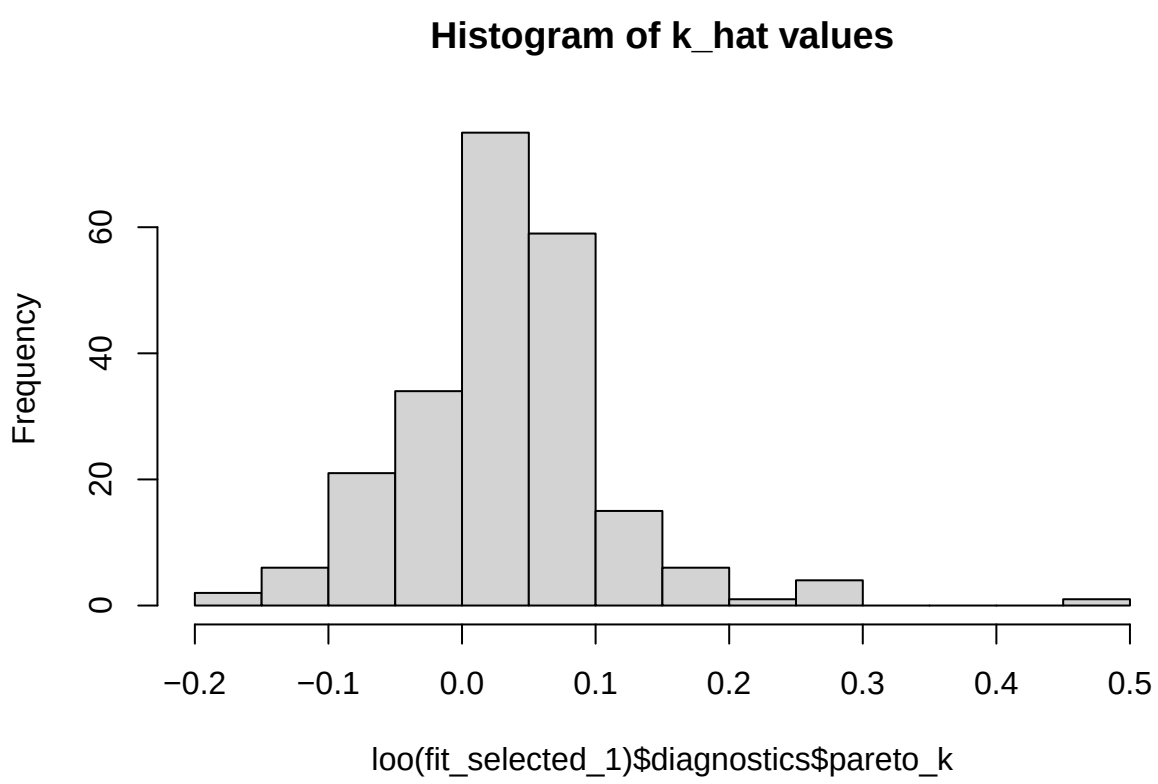
```
##
## Computed from 4000 by 224 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -113.3  9.1
## p_loo         3.7  0.7
```

```
## looic       226.5 18.3
## ------
## 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.
```

```
print("Distribution of k_hat")
```

```
## [1] "Distribution of k_hat"
```

```
hist(loo(fit_selected_1)$diagnostics$pareto_k, main = "Histogram of k_hat values")
```




```
print("Rhat values: ")
```

```
## [1] "Rhat values: "
```

```
rhat(fit_selected_2)
```

```
##      beta[1]      beta[2]      beta[3]      beta[4]    Y_pred[1]    Y_pred[2]
##    1.0004015    1.0000824    0.9991950    1.0003615    1.0001613    1.0001855
##    Y_pred[3]    Y_pred[4]    Y_pred[5]    Y_pred[6]    Y_pred[7]    Y_pred[8]
##    1.0001062    1.0002470    1.0000303    0.9997981    0.9999942    1.0000223
```

```
##    Y_pred[9]   Y_pred[10]   Y_pred[11]   Y_pred[12]   Y_pred[13]   Y_pred[14]
##    0.9999116    1.0000653    1.0000559    0.9996372    0.9998976    0.9998003
##   Y_pred[15]   Y_pred[16]   Y_pred[17]   Y_pred[18]   Y_pred[19]   Y_pred[20]
##    0.9994460    1.0001038    0.9994118    1.0000132    1.0002009    0.9995294
##   Y_pred[21]   Y_pred[22]   Y_pred[23]   Y_pred[24]   Y_pred[25]   Y_pred[26]
##    0.9995648    1.0001092    0.9999142    0.9999878    0.9995118    0.9996776
##   Y_pred[27]   Y_pred[28]   Y_pred[29]   Y_pred[30]   Y_pred[31]   Y_pred[32]
##    0.9994242    1.0000197    1.0004769    1.0001214    1.0000586    1.0000601
##   Y_pred[33]   Y_pred[34]   Y_pred[35]   Y_pred[36]   Y_pred[37]   Y_pred[38]
##    0.9997954    0.9995892    0.9996573    1.0001748    0.9998444    1.0001039
##   Y_pred[39]   Y_pred[40]   Y_pred[41]   Y_pred[42]   Y_pred[43]   Y_pred[44]
##    0.9992636    1.0001075    0.9995074    0.9995951    1.0000594    0.9991148
##   Y_pred[45]   Y_pred[46]   Y_pred[47]   Y_pred[48]   Y_pred[49]   Y_pred[50]
##    1.0001380    0.9993098    0.9997458    1.0000075    0.9998212    0.9995503
##   Y_pred[51]   Y_pred[52]   Y_pred[53]   Y_pred[54]   Y_pred[55]   log_lik[1]
##    0.9995724    1.0001535    0.9996875    1.0001412    0.9999220    0.9999255
##   log_lik[2]   log_lik[3]   log_lik[4]   log_lik[5]   log_lik[6]   log_lik[7]
##    1.0002278    1.0003072    0.9998350    0.9993693    0.9994413    1.0002664
##   log_lik[8]   log_lik[9]  log_lik[10]  log_lik[11]  log_lik[12]  log_lik[13]
##    1.0001100    1.0001375    0.9993928    1.0000189    1.0002386    1.0000480
##  log_lik[14]  log_lik[15]  log_lik[16]  log_lik[17]  log_lik[18]  log_lik[19]
##    0.9996417    1.0000202    1.0000548    1.0000002    0.9998220    1.0002374
##  log_lik[20]  log_lik[21]  log_lik[22]  log_lik[23]  log_lik[24]  log_lik[25]
##    1.0002700    0.9996703    0.9996272    0.9999974    0.9996659    0.9994954
##  log_lik[26]  log_lik[27]  log_lik[28]  log_lik[29]  log_lik[30]  log_lik[31]
##    1.0000659    0.9995200    1.0001144    0.9999860    0.9999909    0.9994464
##  log_lik[32]  log_lik[33]  log_lik[34]  log_lik[35]  log_lik[36]  log_lik[37]
##    0.9999713    0.9997697    0.9999474    1.0000064    0.9999177    0.9999974
##  log_lik[38]  log_lik[39]  log_lik[40]  log_lik[41]  log_lik[42]  log_lik[43]
##    0.9999013    1.0000458    1.0003806    1.0001600    0.9998311    0.9998443
##  log_lik[44]  log_lik[45]  log_lik[46]  log_lik[47]  log_lik[48]  log_lik[49]
##    0.9999832    1.0000041    1.0001005    0.9996633    1.0000952    0.9998707
##  log_lik[50]  log_lik[51]  log_lik[52]  log_lik[53]  log_lik[54]  log_lik[55]
##    0.9996533    1.0000191    1.0000425    0.9997696    1.0000513    1.0001326
##  log_lik[56]  log_lik[57]  log_lik[58]  log_lik[59]  log_lik[60]  log_lik[61]
##    0.9998148    1.0001178    1.0000459    1.0004518    1.0000069    0.9997997
##  log_lik[62]  log_lik[63]  log_lik[64]  log_lik[65]  log_lik[66]  log_lik[67]
##    0.9994352    1.0002150    0.9999788    1.0000924    0.9998630    1.0001482
##  log_lik[68]  log_lik[69]  log_lik[70]  log_lik[71]  log_lik[72]  log_lik[73]
##    0.9994073    0.9997804    0.9996811    0.9999986    1.0001801    0.9999443
##  log_lik[74]  log_lik[75]  log_lik[76]  log_lik[77]  log_lik[78]  log_lik[79]
##    1.0000933    1.0000104    1.0001861    1.0000781    1.0004151    1.0000762
##  log_lik[80]  log_lik[81]  log_lik[82]  log_lik[83]  log_lik[84]  log_lik[85]
##    1.0001668    1.0003792    0.9994948    0.9994299    0.9996152    1.0001088
##  log_lik[86]  log_lik[87]  log_lik[88]  log_lik[89]  log_lik[90]  log_lik[91]
##    1.0004609    0.9997350    0.9999134    0.9996551    1.0002569    1.0000815
##  log_lik[92]  log_lik[93]  log_lik[94]  log_lik[95]  log_lik[96]  log_lik[97]
##    0.9995452    0.9994425    0.9994414    0.9994546    1.0000609    0.9999245
##  log_lik[98]  log_lik[99] log_lik[100] log_lik[101] log_lik[102] log_lik[103]
##    1.0002744    0.9998781    0.9997155    0.9994980    1.0001334    0.9996633
## log_lik[104] log_lik[105] log_lik[106] log_lik[107] log_lik[108] log_lik[109]
##    0.9999772    1.0001382    1.0001307    1.0000686    1.0000461    0.9996107
## log_lik[110] log_lik[111] log_lik[112] log_lik[113] log_lik[114] log_lik[115]
##    1.0001497    0.9996124    1.0001291    0.9998434    0.9994178    0.9995454
```

```
## log_lik[116] log_lik[117] log_lik[118] log_lik[119] log_lik[120] log_lik[121]
##    0.9998763    0.9999592    0.9998308    1.0004942    1.0001054    1.0002592
## log_lik[122] log_lik[123] log_lik[124] log_lik[125] log_lik[126] log_lik[127]
##    0.9993505    0.9999978    1.0002861    0.9995943    1.0000111    1.0001350
## log_lik[128] log_lik[129] log_lik[130] log_lik[131] log_lik[132] log_lik[133]
##    0.9994000    0.9995979    0.9994808    0.9994755    0.9998798    0.9996320
## log_lik[134] log_lik[135] log_lik[136] log_lik[137] log_lik[138] log_lik[139]
##    1.0002121    0.9999640    1.0004131    1.0000499    0.9996520    0.9996008
## log_lik[140] log_lik[141] log_lik[142] log_lik[143] log_lik[144] log_lik[145]
##    0.9999259    1.0000384    1.0007167    0.9994339    1.0000953    0.9995921
## log_lik[146] log_lik[147] log_lik[148] log_lik[149] log_lik[150] log_lik[151]
##    1.0003775    1.0003412    0.9999121    0.9993167    1.0000952    0.9997071
## log_lik[152] log_lik[153] log_lik[154] log_lik[155] log_lik[156] log_lik[157]
##    0.9995260    0.9997289    0.9998712    0.9999626    1.0001562    0.9993045
## log_lik[158] log_lik[159] log_lik[160] log_lik[161] log_lik[162] log_lik[163]
##    1.0001209    0.9997945    0.9990590    1.0001167    0.9991931    0.9995539
## log_lik[164] log_lik[165] log_lik[166] log_lik[167] log_lik[168] log_lik[169]
##    1.0002287    0.9999456    0.9995002    1.0004698    0.9997905    0.9994835
## log_lik[170] log_lik[171] log_lik[172] log_lik[173] log_lik[174] log_lik[175]
##    1.0000502    0.9995807    0.9995579    1.0001482    0.9993003    0.9995711
## log_lik[176] log_lik[177] log_lik[178] log_lik[179] log_lik[180] log_lik[181]
##    1.0000482    0.9992317    1.0001448    0.9996717    1.0000741    1.0001722
## log_lik[182] log_lik[183] log_lik[184] log_lik[185] log_lik[186] log_lik[187]
##    1.0001113    1.0001337    0.9994794    0.9999555    0.9997415    0.9999864
## log_lik[188] log_lik[189] log_lik[190] log_lik[191] log_lik[192] log_lik[193]
##    0.9999804    0.9998567    0.9997963    1.0000373    0.9999271    1.0000605
## log_lik[194] log_lik[195] log_lik[196] log_lik[197] log_lik[198] log_lik[199]
##    0.9999233    1.0001728    0.9995348    1.0000736    1.0002528    0.9998658
## log_lik[200] log_lik[201] log_lik[202] log_lik[203] log_lik[204] log_lik[205]
##    1.0001916    0.9997344    0.9995644    0.9998878    1.0004730    0.9996296
## log_lik[206] log_lik[207] log_lik[208] log_lik[209] log_lik[210] log_lik[211]
##    1.0000627    0.9998425    1.0001594    1.0000373    0.9997236    0.9999742
## log_lik[212] log_lik[213] log_lik[214] log_lik[215] log_lik[216] log_lik[217]
##    0.9996836    0.9996360    1.0001033    0.9994935    0.9997062    0.9998593
## log_lik[218] log_lik[219] log_lik[220] log_lik[221] log_lik[222] log_lik[223]
##    1.0002002    0.9998594    0.9997978    0.9996051    0.9996068    1.0000613
## log_lik[224]         lp__
##    1.0001284    1.0005117
```

```
print("Check divergences: ")
```

```
## [1] "Check divergences: "
```

```
check_divergences(fit_selected_2)
```

```
## 0 of 4000 iterations ended with a divergence.
```

```
print("Check treedepth:")
```

```
## [1] "Check treedepth:"
```

```r
check_treedepth(fit_selected_2)
```

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

```r
print("n_eff values: ")
```

```
## [1] "n_eff values: "
```

```r
loo(fit_selected_2)$diagnostics$n_eff
```

```
##   [1] 2058.357 1924.256 2120.352 1788.863  876.288 2846.044 1996.544 2065.013
##   [9] 2335.836 2720.098 2135.117 1981.273 1937.840 2144.309 2264.985 2017.851
##  [17] 2333.661 2626.757 1916.257 1948.348 2875.225 2670.359 2184.947 2102.456
##  [25] 3262.159 2432.014 2947.835 2011.294 2073.099 2045.657 2541.633 2360.770
##  [33] 2525.161 2253.495 2351.924 2046.225 2127.730 3223.481 2184.888 1895.837
##  [41] 1944.141 2020.931 2409.220 2742.818 1949.973 2122.207 2459.371 2035.540
##  [49] 2214.019 2721.706 2036.946 2194.974 2524.350 2306.880 2384.758 3054.503
##  [57] 2145.864 2359.690 1848.174 2590.337 2495.012 2752.696 2267.330 3061.042
##  [65] 1966.948 2341.757 2195.341 3009.224 2174.973 2564.617 2540.447 1966.309
##  [73] 2096.045 1933.506 2006.164 1982.240 2170.709 1701.858 1979.900 2202.507
##  [81] 1974.599 2355.693 3184.938 2400.329 2386.929 1825.389 2497.277 2391.753
##  [89] 2420.918 1982.743 2442.033 2196.073 2948.253 3031.712 1888.714 2146.126
##  [97] 1967.527 1942.253 2199.555 2234.747 2839.156 2148.197 3243.927 2390.432
## [105] 1918.842 2326.830 2211.312 2123.914 2176.505 2116.390 2522.537 2288.585
## [113] 2074.370 3003.119 2671.463 2320.909 2565.834 2487.585 1821.088 1976.636
## [121] 2501.362 2734.884 2170.099 1875.839 2648.364 2304.836 2070.020 1621.262
## [129] 2426.338 2272.393 3099.155 1656.959 2408.480 2066.317 2297.705 1878.481
## [137] 2500.292 2164.516 2224.537 1526.870 1949.622 1778.178 2681.203 1971.294
## [145] 2410.696 2096.283 2101.471 2137.775 2343.579 2466.722 2337.371 2551.150
## [153] 2732.521 2801.594 2138.063 2527.301 2490.869 2119.037 2263.437 2533.457
## [161] 2129.490 2566.264 2126.634 2712.300 1385.804 3313.647 1850.636 2894.628
## [169] 3138.784 2040.016 3151.843 2341.982 2195.341 2457.613 2824.359 2091.080
## [177] 2318.264 1931.951 2581.848 2280.623 1933.025 1931.027 2324.540 2094.093
## [185] 2052.675 3289.380 2412.999 2099.820 2214.147 2077.852 1934.844 2009.780
## [193] 1917.037 2358.251 2489.209 3157.833 2108.735 1897.184 3198.773 1933.128
## [201] 2834.120 2529.465 2117.354 1932.827 3577.567 1847.493 2197.512 2051.464
## [209] 2123.851 2374.939 2236.727 2749.265 2707.910 1941.935 2242.855 2405.569
## [217] 1793.450 2628.733 2666.300 3175.338 2687.151 2533.497 3033.413 1939.100
```

```r
print("Performance of k_hat values and elpd_loo")
```

```
## [1] "Performance of k_hat values and elpd_loo"
```

```r
loo(fit_selected_2)
```

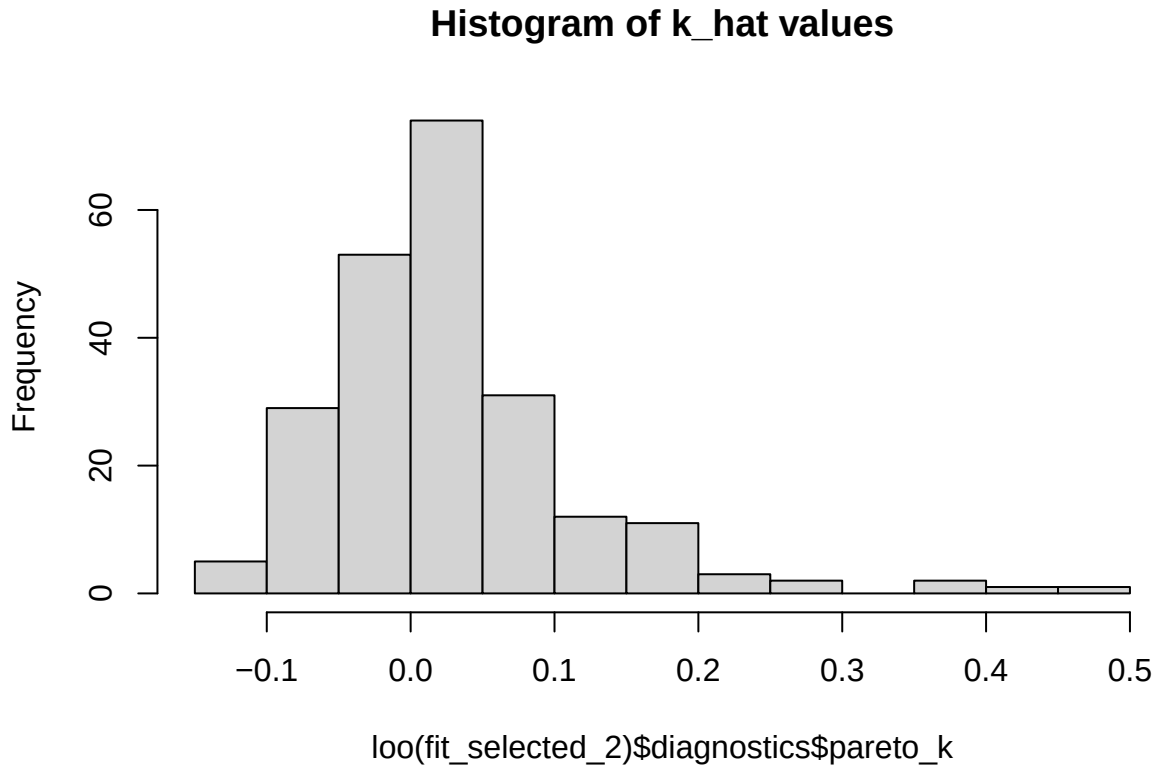
```
##
## Computed from 4000 by 224 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -111.6 10.0
```

```
## p_loo         5.4  1.0
## looic       223.2 20.0
## ------
## 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.
```

```
print("Distribution of k_hat")
```

```
## [1] "Distribution of k_hat"
```

```
hist(loo(fit_selected_2)$diagnostics$pareto_k, main = "Histogram of k_hat values")
```




```
print("Rhat values: ")
```

```
## [1] "Rhat values: "
```

```
rhat(fit_selected_3)
```

```
##      beta[1]      beta[2]      beta[3]      beta[4]      beta[5]      beta[6]
##     1.596538     1.817611    81.261519     1.551190     1.054120     1.601633
##    Y_pred[1]    Y_pred[2]    Y_pred[3]    Y_pred[4]    Y_pred[5]    Y_pred[6]
```

```
##     2.011020     1.174090     1.775969     1.812561     1.819148     1.194635
##    Y_pred[7]    Y_pred[8]    Y_pred[9]   Y_pred[10]   Y_pred[11]   Y_pred[12]
##     1.504598     1.320337     1.277021     1.441720     1.893721     2.745289
##   Y_pred[13]   Y_pred[14]   Y_pred[15]   Y_pred[16]   Y_pred[17]   Y_pred[18]
##     2.707002     1.072511     2.866089     1.814184     2.555801     2.101665
##   Y_pred[19]   Y_pred[20]   Y_pred[21]   Y_pred[22]   Y_pred[23]   Y_pred[24]
##     1.448817     2.776318     1.386817     1.966984     1.790472     1.896384
##   Y_pred[25]   Y_pred[26]   Y_pred[27]   Y_pred[28]   Y_pred[29]   Y_pred[30]
##     1.551909     1.540593     2.659595     1.302303     1.093328     1.598678
##   Y_pred[31]   Y_pred[32]   Y_pred[33]   Y_pred[34]   Y_pred[35]   Y_pred[36]
##     1.716511     1.277838     1.037967     1.781861     1.561160     1.742619
##   Y_pred[37]   Y_pred[38]   Y_pred[39]   Y_pred[40]   Y_pred[41]   Y_pred[42]
##     1.341513     1.227366     2.685865     1.224171     1.537218     2.218656
##   Y_pred[43]   Y_pred[44]   Y_pred[45]   Y_pred[46]   Y_pred[47]   Y_pred[48]
##     1.049614     3.026774     1.923692     2.891470     1.687484     2.004109
##   Y_pred[49]   Y_pred[50]   Y_pred[51]   Y_pred[52]   Y_pred[53]   Y_pred[54]
##     1.883394     1.173131     3.209885     1.634683     2.270977     1.631932
##   Y_pred[55]   log_lik[1]   log_lik[2]   log_lik[3]   log_lik[4]   log_lik[5]
##     2.043224     1.462888     1.849626     1.070929     1.182047     7.396253
##   log_lik[6]   log_lik[7]   log_lik[8]   log_lik[9]  log_lik[10]  log_lik[11]
##     1.187573     1.286854     1.714398     1.492483     2.032828     1.900539
##  log_lik[12]  log_lik[13]  log_lik[14]  log_lik[15]  log_lik[16]  log_lik[17]
##     1.752203     1.275199     1.433363     1.656866     1.417117     1.781660
##  log_lik[18]  log_lik[19]  log_lik[20]  log_lik[21]  log_lik[22]  log_lik[23]
##     1.103225     1.470175     1.570332     1.184862     1.287350     1.983242
##  log_lik[24]  log_lik[25]  log_lik[26]  log_lik[27]  log_lik[28]  log_lik[29]
##     2.769738     1.333656     1.581740     2.082839     1.656236     1.737151
##  log_lik[30]  log_lik[31]  log_lik[32]  log_lik[33]  log_lik[34]  log_lik[35]
##     1.629262     1.350154     1.402284     1.740365     1.308928     2.136351
##  log_lik[36]  log_lik[37]  log_lik[38]  log_lik[39]  log_lik[40]  log_lik[41]
##     1.690866     2.189586     1.246394     1.785586     1.394533     1.532919
##  log_lik[42]  log_lik[43]  log_lik[44]  log_lik[45]  log_lik[46]  log_lik[47]
##     1.103588     1.899852     1.258022     1.682240     1.764382     1.870659
##  log_lik[48]  log_lik[49]  log_lik[50]  log_lik[51]  log_lik[52]  log_lik[53]
##     1.497168     2.492074     2.732045     1.116108     1.817393     1.184981
##  log_lik[54]  log_lik[55]  log_lik[56]  log_lik[57]  log_lik[58]  log_lik[59]
##     1.309827     1.534382     1.068176     1.779809     1.091594     1.460655
##  log_lik[60]  log_lik[61]  log_lik[62]  log_lik[63]  log_lik[64]  log_lik[65]
##     1.506222     1.335510     2.704174     1.054365     1.247855     1.751720
##  log_lik[66]  log_lik[67]  log_lik[68]  log_lik[69]  log_lik[70]  log_lik[71]
##     1.722032     2.099865     2.501899     2.378918     1.665552     1.983416
##  log_lik[72]  log_lik[73]  log_lik[74]  log_lik[75]  log_lik[76]  log_lik[77]
##     1.627814     1.751244     1.590316     1.427350     1.080672     1.825200
##  log_lik[78]  log_lik[79]  log_lik[80]  log_lik[81]  log_lik[82]  log_lik[83]
##     1.683008     1.755834     1.407880     1.115338     3.070057     1.631703
##  log_lik[84]  log_lik[85]  log_lik[86]  log_lik[87]  log_lik[88]  log_lik[89]
##     1.329103     1.324436     1.168780     1.101754     1.464506     1.423246
##  log_lik[90]  log_lik[91]  log_lik[92]  log_lik[93]  log_lik[94]  log_lik[95]
##     1.439739     1.136265     1.188109     1.183841     1.887518     5.235818
##  log_lik[96]  log_lik[97]  log_lik[98]  log_lik[99] log_lik[100] log_lik[101]
##     1.764174     1.719497     1.466083     1.381124     1.833531     1.540411
## log_lik[102] log_lik[103] log_lik[104] log_lik[105] log_lik[106] log_lik[107]
##     1.758520     1.542811     1.262202     1.343682     1.301108     1.764490
## log_lik[108] log_lik[109] log_lik[110] log_lik[111] log_lik[112] log_lik[113]
```

```
##     2.063210     5.618526     1.369661     1.032371     1.689294     1.394329
## log_lik[114] log_lik[115] log_lik[116] log_lik[117] log_lik[118] log_lik[119]
##     2.618481     1.325176     2.722419     1.237811     2.346907     1.086915
## log_lik[120] log_lik[121] log_lik[122] log_lik[123] log_lik[124] log_lik[125]
##     1.549159     1.196558     2.349526     1.634226     1.311034     2.904886
## log_lik[126] log_lik[127] log_lik[128] log_lik[129] log_lik[130] log_lik[131]
##     1.928165     1.746625     2.714649     1.994963     1.417717     2.385424
## log_lik[132] log_lik[133] log_lik[134] log_lik[135] log_lik[136] log_lik[137]
##     1.039157     1.305559     1.684588     1.049538     1.197771     1.294781
## log_lik[138] log_lik[139] log_lik[140] log_lik[141] log_lik[142] log_lik[143]
##     1.524367     1.421958     1.655296     1.003710     1.314794     1.669903
## log_lik[144] log_lik[145] log_lik[146] log_lik[147] log_lik[148] log_lik[149]
##     1.560886     1.083152     1.504501     1.119339     1.096421     5.348777
## log_lik[150] log_lik[151] log_lik[152] log_lik[153] log_lik[154] log_lik[155]
##     1.214405     1.013548     1.341150     1.122083     1.570911     2.182746
## log_lik[156] log_lik[157] log_lik[158] log_lik[159] log_lik[160] log_lik[161]
##     1.127056     7.448916     1.709842     2.219966     7.002994     1.787640
## log_lik[162] log_lik[163] log_lik[164] log_lik[165] log_lik[166] log_lik[167]
##     2.630153    12.807276     1.038499     1.831925     1.079052     1.101650
## log_lik[168] log_lik[169] log_lik[170] log_lik[171] log_lik[172] log_lik[173]
##     1.477643     1.500722     1.966900     2.142337     3.116878     2.066939
## log_lik[174] log_lik[175] log_lik[176] log_lik[177] log_lik[178] log_lik[179]
##    10.393602     1.247597     1.047559     5.190942     1.358878     2.437912
## log_lik[180] log_lik[181] log_lik[182] log_lik[183] log_lik[184] log_lik[185]
##     1.748950     1.901284     1.658717     1.842154     2.637671     1.459569
## log_lik[186] log_lik[187] log_lik[188] log_lik[189] log_lik[190] log_lik[191]
##     1.641537     1.867246     1.686370     2.470305     1.471027     1.704933
## log_lik[192] log_lik[193] log_lik[194] log_lik[195] log_lik[196] log_lik[197]
##     1.445708     1.138533     1.840002     1.314300     1.149607     1.735754
## log_lik[198] log_lik[199] log_lik[200] log_lik[201] log_lik[202] log_lik[203]
##     1.393930     1.435369     1.720267     1.735260     2.949722     2.292862
## log_lik[204] log_lik[205] log_lik[206] log_lik[207] log_lik[208] log_lik[209]
##     1.482292     1.235410     1.556396     2.534482     1.593533     1.606826
## log_lik[210] log_lik[211] log_lik[212] log_lik[213] log_lik[214] log_lik[215]
##     2.580057     2.016937     2.299875     2.104466     1.507210     1.350771
## log_lik[216] log_lik[217] log_lik[218] log_lik[219] log_lik[220] log_lik[221]
##     1.706404     1.328015     1.245062     1.770223     1.339012     2.864950
## log_lik[222] log_lik[223] log_lik[224]         lp__
##     2.865006     1.223115     1.551520     2.005728
```

```
print("Check divergences: ")
```

```
## [1] "Check divergences: "
```

```
check_divergences(fit_selected_3)
```

```
## 1 of 4000 iterations ended with a divergence (0.025%).
## Try increasing 'adapt_delta' to remove the divergences.
```

```
print("Check treedepth:")
```

```
## [1] "Check treedepth:"
```

```
check_treedepth(fit_selected_3)
```

```
## 1595 of 4000 iterations saturated the maximum tree depth of 10 (39.875%).
## Try increasing 'max_treedepth' to avoid saturation.
```

```
print("n_eff values: ")
```

```
## [1] "n_eff values: "
```

```
loo(fit_selected_3)$diagnostics$n_eff
```

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

```
##   [1]   3.63881956   0.18946680   2.95488003   2.26306075   0.53533099
##   [6]   3.58770521   4.71095702   2.88983540   3.13903293   2.41506242
##  [11]   2.54506595   2.86113755   0.07459886   3.64214360   2.72283772
##  [16]   3.85113708   2.76977930   6.37942745   1.42832125   3.26341351
##  [21]   5.85675787   5.94643351   2.50366997   1.35058944   4.24875390
##  [26]   3.05228066   2.31786587   0.02936492   2.59482472   2.85799579
##  [31]   3.26989530   3.76037269   2.70299970   2.77385060   2.42316148
##  [36]   2.79208599   2.41693774   5.67434216   2.66011822   4.06620926
##  [41]   3.29991591  17.38910488   2.58009307   3.66438966   2.63287019
##  [46]   2.84492668   1.96291764   3.46194731   2.26031057   1.59227190
##  [51]   0.77642800   2.75918040   6.41379472   4.42113666   1.10151622
##  [56]  31.41496031   2.70020969   3.41317271   0.43924356   3.39666542
##  [61]   1.95620231   2.21648653   8.50084630   4.89219741   2.83842315
##  [66]   2.85593422   2.35223514   2.24932048   1.23616832   2.16724259
##  [71]   2.52658300   3.01192813   2.69752640   3.15938599   3.84650304
##  [76]  12.77930229   2.56900178   0.02572474   2.79624368   3.55637973
##  [81]   1.11698255   1.63959924   2.99607091   1.98712879   2.75047648
##  [86]   0.23544862   4.50600549   3.53819882   3.66989494   3.66746642
##  [91]   6.23165117   4.04939281   3.71778538   2.65073986   1.31252829
##  [96]   2.40592769   2.79958742   3.67596577   0.35234223   2.41056272
## [101]   3.13968171   2.83013482   3.29628146   2.24786474   4.41282691
## [106]   4.57909634   2.84789420   2.50847230   0.38311638   3.94063255
## [111]  12.25522277   2.93730844   4.00971783   2.22717578   2.86170124
## [116]   1.92667559   4.57815965   2.33219669  14.62874541   3.22340867
## [121]   6.07269145   1.96638642   2.98799991   1.45976306   1.76335773
## [126]   2.61825917   2.87734447   2.17507901   0.78432425   3.37015085
## [131]   2.30702590  37.23779144   2.11711533   2.85599543  47.96562903
## [136]   6.69616217   3.20900082   3.34884288   3.66467208   0.51944945
## [141] 490.00403052   4.94376493   2.83686268   3.11985035   5.64606900
## [146]   0.97268301   3.23049012   3.53576091   0.80584719   5.99218856
## [151] 105.01243496   4.05965827   5.93988191   3.12172198   2.43785097
## [156]   5.40065815   1.07349477   2.79461559   2.35993434   0.79069113
## [161]   0.54118839   1.07041859   0.42255091  76.76695296   0.65284296
## [166]  13.93831857   0.80192256   3.43934818   3.34949508   2.58423910
## [171]   2.43940947   1.58301699   2.38001885   0.50980655   5.19689548
## [176]  26.06178227   0.28877519   4.21004070   2.28764790   2.62458981
## [181]   0.22610388   2.99337972   2.68988054   2.21938404   3.67944884
## [186]   2.98019817   2.58905316   2.97124500   2.30338342   3.56369230
```

```
## [191]   2.85497930   3.72091646   6.08576926   2.77283890   4.38436110
## [196]   7.72579062   2.81717288   4.08419141   3.72695361   2.88933577
## [201]   2.72673366   2.15686522   2.30707792   3.67201915   5.52229395
## [206]   3.12046330   2.20370290   3.17035475   3.05933487   2.24336029
## [211]   2.50945057   2.29648614   2.44216875   3.13824312   3.56487287
## [216]   2.84678982   4.09845589   5.56900477   2.18052755   4.22284373
## [221]   2.11831836   1.70670373   5.46989884   3.28250487
```

```
print("Performance of k_hat values and elpd_loo")
```

```
## [1] "Performance of k_hat values and elpd_loo"
```

```
loo(fit_selected_3)
```

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

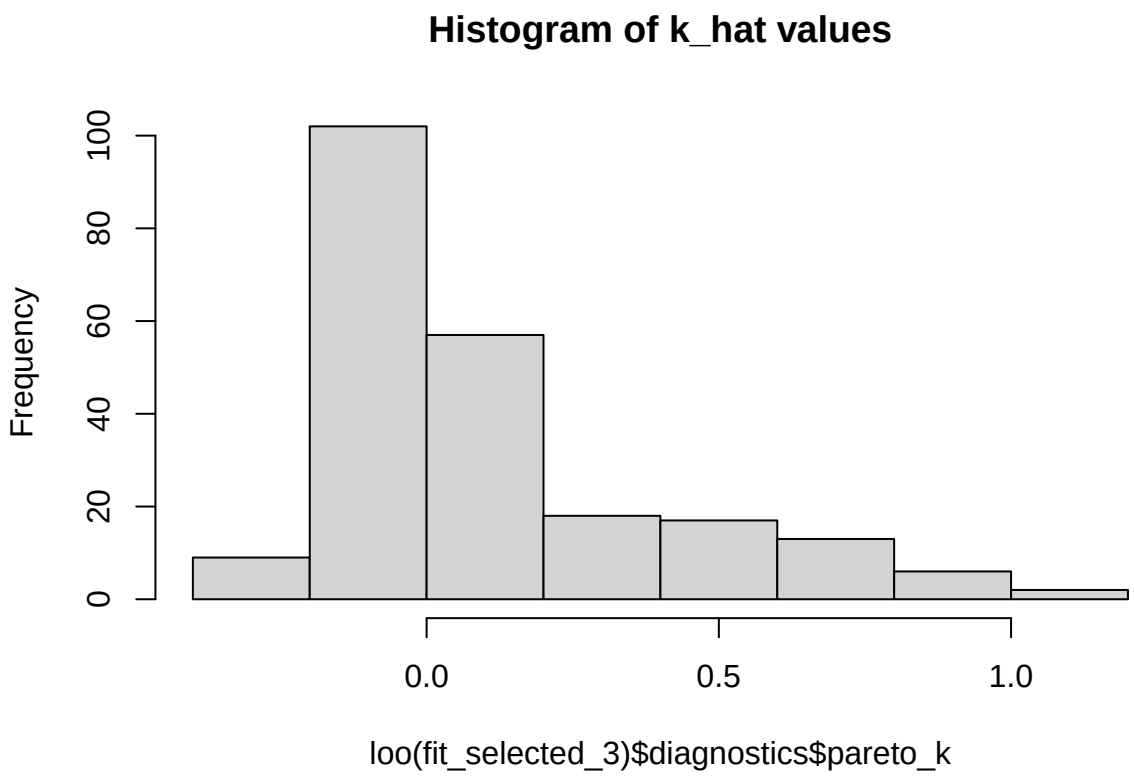
```
##
## Computed from 4000 by 224 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -163.6 18.0
## p_loo        49.3 12.9
## looic       327.1 36.1
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     193   86.2%   0
##  (0.5, 0.7]   (ok)        20    8.9%   1
##    (0.7, 1]   (bad)        9    4.0%   0
##    (1, Inf)   (very bad)   2    0.9%   0
## See help('pareto-k-diagnostic') for details.
```

```
print("Distribution of k_hat")
```

```
## [1] "Distribution of k_hat"
```

```
hist(loo(fit_selected_3)$diagnostics$pareto_k, main = "Histogram of k_hat values")
```

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




```
print("Rhat values: ")
```

```
## [1] "Rhat values: "
```

```
rhat(fit_full)
```

```
##      beta[1]      beta[2]      beta[3]      beta[4]      beta[5]      beta[6]
##     2.041319   243.050402     1.101865   151.693339     1.307572    32.326208
##      beta[7]      beta[8]      beta[9]     beta[10]     beta[11]     beta[12]
##     1.251276    84.597866     1.760298   157.970269    56.245538    57.283652
##    Y_pred[1]    Y_pred[2]    Y_pred[3]    Y_pred[4]    Y_pred[5]    Y_pred[6]
##     2.804601    10.542958     1.353619     2.807511     2.831758     8.947327
##    Y_pred[7]    Y_pred[8]    Y_pred[9]   Y_pred[10]   Y_pred[11]   Y_pred[12]
##     2.979902     1.151153     3.613487     6.396177     2.751921     3.674027
##   Y_pred[13]   Y_pred[14]   Y_pred[15]   Y_pred[16]   Y_pred[17]   Y_pred[18]
##     9.845967     2.817909     5.283624     1.639686     5.626987     2.588088
##   Y_pred[19]   Y_pred[20]   Y_pred[21]   Y_pred[22]   Y_pred[23]   Y_pred[24]
##     4.072607    14.698896     1.946549     2.403271     1.590658     3.941113
##   Y_pred[25]   Y_pred[26]   Y_pred[27]   Y_pred[28]   Y_pred[29]   Y_pred[30]
##     2.806750    10.570144     6.120067     1.912706     1.106631     1.464203
##   Y_pred[31]   Y_pred[32]   Y_pred[33]   Y_pred[34]   Y_pred[35]   Y_pred[36]
##     2.526911     2.389610     1.716661     5.134869     1.976235     7.547797
##   Y_pred[37]   Y_pred[38]   Y_pred[39]   Y_pred[40]   Y_pred[41]   Y_pred[42]
##     1.864670     1.592817     8.661988     1.057301     3.854410    10.646763
##   Y_pred[43]   Y_pred[44]   Y_pred[45]   Y_pred[46]   Y_pred[47]   Y_pred[48]
```

```
##     1.024699    23.956925     3.087501     4.244385     7.354153     2.924107
##   Y_pred[49]   Y_pred[50]   Y_pred[51]   Y_pred[52]   Y_pred[53]   Y_pred[54]
##     2.702778     4.792328     4.250186     2.275518     7.613801     1.306348
##   Y_pred[55]   log_lik[1]   log_lik[2]   log_lik[3]   log_lik[4]   log_lik[5]
##     2.254405     1.844562     5.901988     3.465484     1.708880    19.136263
##   log_lik[6]   log_lik[7]   log_lik[8]   log_lik[9]  log_lik[10]  log_lik[11]
##     3.858701     1.622475     1.346474     2.431125     5.159993     2.858034
##  log_lik[12]  log_lik[13]  log_lik[14]  log_lik[15]  log_lik[16]  log_lik[17]
##     2.710961     3.128975     1.618546     2.526739     1.297051     3.411204
##  log_lik[18]  log_lik[19]  log_lik[20]  log_lik[21]  log_lik[22]  log_lik[23]
##     6.590242     3.299853     1.498782     2.847998     3.007355     5.031136
##  log_lik[24]  log_lik[25]  log_lik[26]  log_lik[27]  log_lik[28]  log_lik[29]
##     3.822128     1.952597     2.610977     2.317344     1.209893     1.785946
##  log_lik[30]  log_lik[31]  log_lik[32]  log_lik[33]  log_lik[34]  log_lik[35]
##     2.862182     2.113182     3.814348     4.658045     1.391658     2.100204
##  log_lik[36]  log_lik[37]  log_lik[38]  log_lik[39]  log_lik[40]  log_lik[41]
##     2.115313     4.068766     3.003080     2.325141     1.705770     1.678171
##  log_lik[42]  log_lik[43]  log_lik[44]  log_lik[45]  log_lik[46]  log_lik[47]
##     1.825275     3.131889     6.299938     2.478238     2.272344     4.410924
##  log_lik[48]  log_lik[49]  log_lik[50]  log_lik[51]  log_lik[52]  log_lik[53]
##     1.450148     7.068921     2.186860     2.159573     2.382750     2.628402
##  log_lik[54]  log_lik[55]  log_lik[56]  log_lik[57]  log_lik[58]  log_lik[59]
##     1.922970     2.228220     2.303731     2.927164     6.066419     4.356355
##  log_lik[60]  log_lik[61]  log_lik[62]  log_lik[63]  log_lik[64]  log_lik[65]
##     3.175072     2.872390     4.558012     2.953226     3.933325     1.616562
##  log_lik[66]  log_lik[67]  log_lik[68]  log_lik[69]  log_lik[70]  log_lik[71]
##     2.144070     2.546865     4.807518     1.508426     2.172322     2.330534
##  log_lik[72]  log_lik[73]  log_lik[74]  log_lik[75]  log_lik[76]  log_lik[77]
##     1.916796     2.120068     1.664415     1.430036     1.134184     2.125770
##  log_lik[78]  log_lik[79]  log_lik[80]  log_lik[81]  log_lik[82]  log_lik[83]
##     3.466529     2.555692     3.248719     2.437656     6.121780     3.467116
##  log_lik[84]  log_lik[85]  log_lik[86]  log_lik[87]  log_lik[88]  log_lik[89]
##     4.024580     5.702718     1.180938     2.197429     2.620863     2.636884
##  log_lik[90]  log_lik[91]  log_lik[92]  log_lik[93]  log_lik[94]  log_lik[95]
##     1.871314     1.405370     1.851854     2.032557     4.281656    12.077967
##  log_lik[96]  log_lik[97]  log_lik[98]  log_lik[99] log_lik[100] log_lik[101]
##     2.193024     1.529377     1.368534     3.987551     4.688502     2.220701
## log_lik[102] log_lik[103] log_lik[104] log_lik[105] log_lik[106] log_lik[107]
##     2.065886     8.632608     3.019705     1.311210     1.795079     2.817697
## log_lik[108] log_lik[109] log_lik[110] log_lik[111] log_lik[112] log_lik[113]
##     4.186391    11.588710     2.777943     2.375848     2.560302     1.665609
## log_lik[114] log_lik[115] log_lik[116] log_lik[117] log_lik[118] log_lik[119]
##     5.188513     4.545924     2.567143     1.710235     2.390512     1.101140
## log_lik[120] log_lik[121] log_lik[122] log_lik[123] log_lik[124] log_lik[125]
##     1.454303     3.264716     3.763019     2.338532     3.925040     2.384858
## log_lik[126] log_lik[127] log_lik[128] log_lik[129] log_lik[130] log_lik[131]
##     3.876599     2.216788    10.090945     2.680411     1.991922     6.008444
## log_lik[132] log_lik[133] log_lik[134] log_lik[135] log_lik[136] log_lik[137]
##     1.722952     3.154342     1.889699     2.085896     1.210420     4.542783
## log_lik[138] log_lik[139] log_lik[140] log_lik[141] log_lik[142] log_lik[143]
##     1.857384     1.643042     1.999224     1.053945     1.238399     1.905608
## log_lik[144] log_lik[145] log_lik[146] log_lik[147] log_lik[148] log_lik[149]
##     2.604414     2.121848     3.415180     2.850123     1.724151    10.137709
## log_lik[150] log_lik[151] log_lik[152] log_lik[153] log_lik[154] log_lik[155]
```

```
##     1.442480     1.373195     2.179702     2.800023     4.715097     3.354666
## log_lik[156] log_lik[157] log_lik[158] log_lik[159] log_lik[160] log_lik[161]
##     2.644626     7.087376     1.861986     3.860320    12.483450     4.918584
## log_lik[162] log_lik[163] log_lik[164] log_lik[165] log_lik[166] log_lik[167]
##     5.126495    19.768816     2.644210     2.100765     1.693957     1.067236
## log_lik[168] log_lik[169] log_lik[170] log_lik[171] log_lik[172] log_lik[173]
##     2.484941     3.565800     2.449579     8.638261     5.170454     4.797446
## log_lik[174] log_lik[175] log_lik[176] log_lik[177] log_lik[178] log_lik[179]
##    18.984257     3.480082     1.023119     6.335550     1.575867     3.966131
## log_lik[180] log_lik[181] log_lik[182] log_lik[183] log_lik[184] log_lik[185]
##     3.137276     1.833346     1.845091     2.966581     3.105222     1.489525
## log_lik[186] log_lik[187] log_lik[188] log_lik[189] log_lik[190] log_lik[191]
##     5.115652     2.620953     1.207234     2.156574     1.308133     1.580997
## log_lik[192] log_lik[193] log_lik[194] log_lik[195] log_lik[196] log_lik[197]
##     1.666064     1.044458     2.534380     2.747674     4.088492     2.537159
## log_lik[198] log_lik[199] log_lik[200] log_lik[201] log_lik[202] log_lik[203]
##     1.644501     6.018488     1.750094     3.703242     3.506364     3.709813
## log_lik[204] log_lik[205] log_lik[206] log_lik[207] log_lik[208] log_lik[209]
##     1.773263     3.375594     2.191719     2.861518     2.146066     1.761489
## log_lik[210] log_lik[211] log_lik[212] log_lik[213] log_lik[214] log_lik[215]
##     4.414655     2.252438     6.447039     3.748481     1.287076     1.302731
## log_lik[216] log_lik[217] log_lik[218] log_lik[219] log_lik[220] log_lik[221]
##     1.528362     3.396891     3.952223     2.018390     3.321185     3.698381
## log_lik[222] log_lik[223] log_lik[224]         lp__
##     3.201965     6.516674     2.065030    21.525988
```

```
print("Check divergences: ")
```

```
## [1] "Check divergences: "
```

```
check_divergences(fit_full)
```

```
## 0 of 4000 iterations ended with a divergence.
```

```
print("Check treedepth:")
```

```
## [1] "Check treedepth:"
```

```
check_treedepth(fit_full)
```

```
## 3993 of 4000 iterations saturated the maximum tree depth of 10 (99.825%).
## Try increasing 'max_treedepth' to avoid saturation.
```

```
print("n_eff values: ")
```

```
## [1] "n_eff values: "
```

```
loo(fit_full)$diagnostics$n_eff
```

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

```
##   [1]  2.5443652  0.6775617  1.3203557  2.3528267  0.4551875  1.9638138
##   [7]  2.7871560  4.4239956  2.2029498  1.0402431  2.0036359  2.2959917
##  [13]  1.2490884  2.8934166  2.1891609  5.3499854  1.9274988  1.1435349
##  [19]  0.7119160  3.4462277  2.1154786  1.6746260  1.4801187  0.9972369
##  [25]  2.4461320  2.1654377  2.1644230  2.9185383  2.6066946  1.5751225
##  [31]  2.5501059  1.4450995  1.0612638  4.2222809  2.3670329  2.2747191
##  [37]  2.0150323  1.7894072  2.0867964  2.9578885  2.8616566  1.8074627
##  [43]  2.1446686  0.9120638  1.8628333  2.3541804  1.2164182  3.7234164
##  [49]  1.1941528  1.8111183  1.8959878  2.3030812  1.9727636  2.5895968
##  [55]  1.7430249  1.8332481  2.0737770  0.6346771  0.8009345  1.8196957
##  [61]  1.5748680  1.9794497  1.1335642  1.3669976  3.1920186  2.4629904
##  [67]  1.9519268  1.9572072  2.7467281  2.1138527  2.2343519  2.6001871
##  [73]  2.4337042  2.8897499  3.7194794  6.5980917  2.2492886  1.3229522
##  [79]  2.2936567  1.8397879  1.4325028  1.1594638  1.8993006  1.5453283
##  [85]  0.7904904  5.4023414  2.2810540  1.7657162  1.6832249  2.6980192
##  [91]  2.1294455  2.6577087  1.7135915  1.8220717  0.4899856  2.0737298
##  [97]  3.1764153  4.2457252  1.3110032  0.6046479  2.1579041  2.5041666
## [103]  0.9693597  1.9320305  4.5890932  2.6984065  2.1800607  1.8066477
## [109]  0.2801477  1.8520577  2.3048318  2.2278656  2.9968738  1.9722674
## [115]  1.8345511  2.0606606  2.4738767  2.3415536 12.9472369  3.6676648
## [121]  1.9963337  1.6715888  2.3159473  0.4865412  2.1927548  1.9097484
## [127]  2.4634336  0.4714174  1.2091528  2.5342978  1.3425305  2.0508284
## [133]  1.6546015  2.1874285  2.3613504  6.9844921  1.5661657  2.6214343
## [139]  2.9910587  1.3839701 18.3605772  6.5295534  2.6584311  1.9971447
## [145]  2.3821714  1.3318230  1.5810451  2.6928016  0.4752752  2.7100054
## [151]  3.5020096  2.1321890  1.7367548  1.1711408  1.9385542  2.1569447
## [157]  1.2057550  2.5042070  1.8853690  0.6170915  0.7881967  1.2756173
## [163]  0.4392737  1.8927542  1.2823562  2.6780040  1.0616992  2.0538696
## [169]  2.1047016  2.3195744  0.9506606  1.3035783  1.3340779  0.4537978
## [175]  2.0061991 72.1285642  0.2095471  3.1941807  2.0970239  1.6628473
## [181]  2.1723233  2.7094247  2.2113841  1.8675923  3.7323571  1.7074180
## [187]  2.2141798  7.0128837  2.4586291  4.8281501  3.0766899  2.9222284
## [193] 21.1115683  2.3123572  1.8566226  1.8603867  1.9827268  3.0749411
## [199]  1.4295667  2.8179337  1.6078136  1.9812750  1.8267618  2.8691347
## [205]  2.0855496  1.5316357  2.0578319  2.4261462  2.7636302  1.8016320
## [211]  2.3566139  1.2091916  2.0387896  4.6699677  5.0477405  3.2924651
## [217]  1.0175564  1.3494682  2.1189364  1.3690316  1.9586537  1.4871449
## [223]  1.2006695  2.5571756
```

```
print("Performance of k_hat values and elpd_loo")
```

```
## [1] "Performance of k_hat values and elpd_loo"
```

```
loo(fit_full)
```

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

```
##
## Computed from 4000 by 224 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -219.1 30.1
```

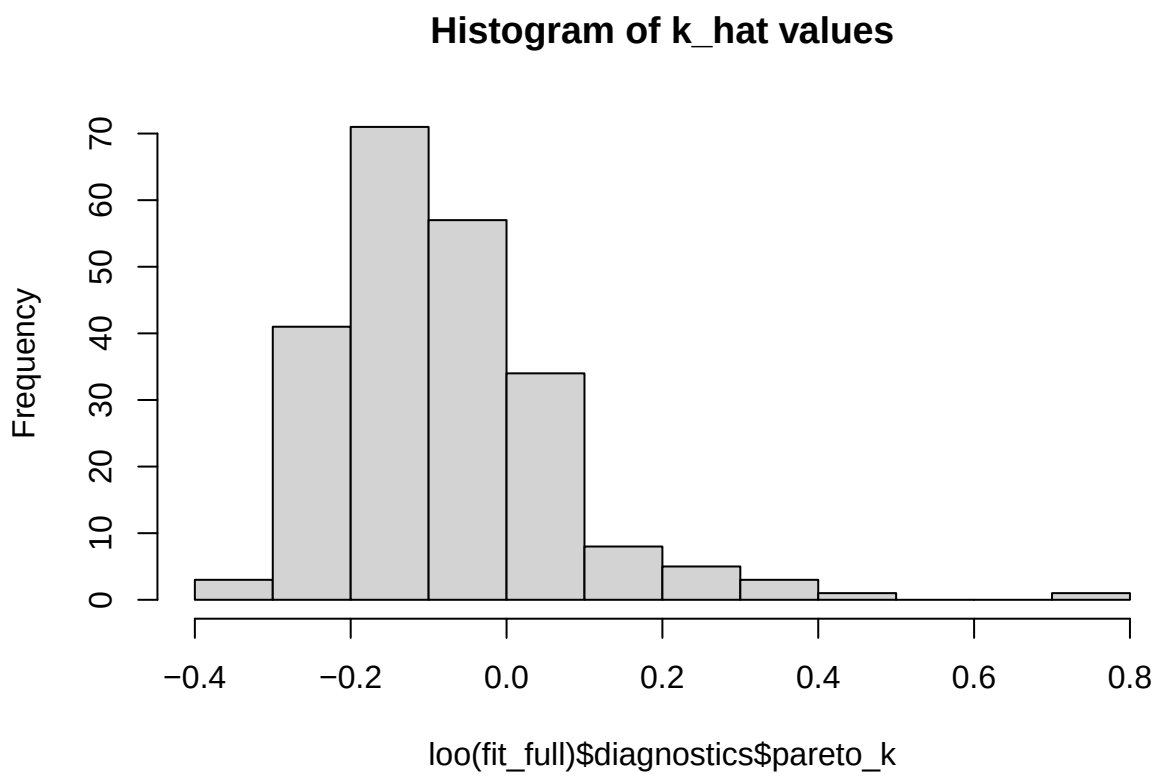
```
## p_loo       117.2 26.7
## looic       438.1 60.2
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     223   99.6%   0
##  (0.5, 0.7]   (ok)         0    0.0%   <NA>
##    (0.7, 1]   (bad)        1    0.4%   1
##    (1, Inf)   (very bad)   0    0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

```
print("Distribution of k_hat")
```

```
## [1] "Distribution of k_hat"
```

```
hist(loo(fit_full)$diagnostics$pareto_k, main = "Histogram of k_hat values")
```

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



From the results above, we can see that the 4 model sees no divergence. There are quite a lot diverging chains in the last two model indicating the unstability of the model. Rhat value from the first 2 is stable and very closed to 1. Tthe last 2 models' Rhat value seems to be very bad. In conclusion, the first 2 models are

very stable for analysis and reliable while the last 2 are very unstable, unreliable and more work is required to improve the last 2.
Also, from the $\hat{k}$ value plotted, we also see that the first 2 model is really good. The last model $\hat{k}$ value also show that the model is quite reliable although it does not seems to converge really well. However, the 3rd model $\hat{k}$ is very bad despite having elpd_loo value much higher than that from the full model, saying that not only it does not converge really well but also not very reliable.
Despite all the values above, we still compare all 4 prediction-accuracy-wise and elpd_loo-value-wise in the next section.

7. Model comparison

```
confusionMatrix(table(Y_pred_selected_1, Y_true))
```

```
## Confusion Matrix and Statistics
##
##                  Y_true
## Y_pred_selected_1  0  1
##                0 31  9
##                1  5 10
##
##               Accuracy : 0.7455
##                 95% CI : (0.61, 0.8533)
##    No Information Rate : 0.6545
##    P-Value [Acc > NIR] : 0.09897
##
##                  Kappa : 0.4077
##
##  Mcnemar's Test P-Value : 0.42268
##
##             Sensitivity : 0.8611
##             Specificity : 0.5263
##          Pos Pred Value : 0.7750
##          Neg Pred Value : 0.6667
##              Prevalence : 0.6545
##          Detection Rate : 0.5636
##    Detection Prevalence : 0.7273
##       Balanced Accuracy : 0.6937
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(table(Y_pred_selected_2, Y_true))
```

```
## Confusion Matrix and Statistics
##
##                  Y_true
## Y_pred_selected_2  0  1
##                0 32  9
##                1  4 10
##
```

```
##                Accuracy : 0.7636
##                  95% CI : (0.6298, 0.8677)
##     No Information Rate : 0.6545
##     P-Value [Acc > NIR] : 0.05633
##
##                   Kappa : 0.4427
##
##  Mcnemar's Test P-Value : 0.26726
##
##             Sensitivity : 0.8889
##             Specificity : 0.5263
##          Pos Pred Value : 0.7805
##          Neg Pred Value : 0.7143
##              Prevalence : 0.6545
##          Detection Rate : 0.5818
##    Detection Prevalence : 0.7455
##       Balanced Accuracy : 0.7076
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(table(Y_pred_selected_3, Y_true))
```

```
## Confusion Matrix and Statistics
##
##                  Y_true
## Y_pred_selected_3  0  1
##                 0 32  9
##                 1  4 10
##
##                Accuracy : 0.7636
##                  95% CI : (0.6298, 0.8677)
##     No Information Rate : 0.6545
##     P-Value [Acc > NIR] : 0.05633
##
##                   Kappa : 0.4427
##
##  Mcnemar's Test P-Value : 0.26726
##
##             Sensitivity : 0.8889
##             Specificity : 0.5263
##          Pos Pred Value : 0.7805
##          Neg Pred Value : 0.7143
##              Prevalence : 0.6545
##          Detection Rate : 0.5818
##    Detection Prevalence : 0.7455
##       Balanced Accuracy : 0.7076
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(table(Y_pred_full, Y_true))
```

```
## Confusion Matrix and Statistics
##
##            Y_true
## Y_pred_full  0  1
##           0 32 10
##           1  4  9
##
##                Accuracy : 0.7455
##                  95% CI : (0.61, 0.8533)
##     No Information Rate : 0.6545
##     P-Value [Acc > NIR] : 0.09897
##
##                   Kappa : 0.3918
##
##  Mcnemar's Test P-Value : 0.18145
##
##             Sensitivity : 0.8889
##             Specificity : 0.4737
##          Pos Pred Value : 0.7619
##          Neg Pred Value : 0.6923
##              Prevalence : 0.6545
##          Detection Rate : 0.5818
##    Detection Prevalence : 0.7636
##       Balanced Accuracy : 0.6813
##
##        'Positive' Class : 0
##
```

```
cat("\nSelected Bayesian Inference PSIS_LOO:" ,loo(fit_selected_1)$estimates[1], "\n");
```

```
##
## Selected Bayesian Inference PSIS_LOO: -113.2669
```

```
cat("\nSelected from correlation matrix PSIS_LOO:" , loo(fit_selected_2)$estimates[1], "\n");
```

```
##
## Selected from correlation matrix PSIS_LOO: -111.6207
```

```
cat("\nSelected from TreeClassifer PSIS_LOO:" ,loo(fit_selected_3)$estimates[1], "\n\n");
```

```
##
## Selected from TreeClassifer PSIS_LOO: -163.5729
```

```
cat("\nFull model PSIS_LOO:" ,loo(fit_full)$estimates[1], "\n\n");
```

```
##
## Full model PSIS_LOO: -219.0553
```

```
loo_compare(loo(fit_selected_1), loo(fit_selected_2), loo(fit_selected_3), loo(fit_full))
```

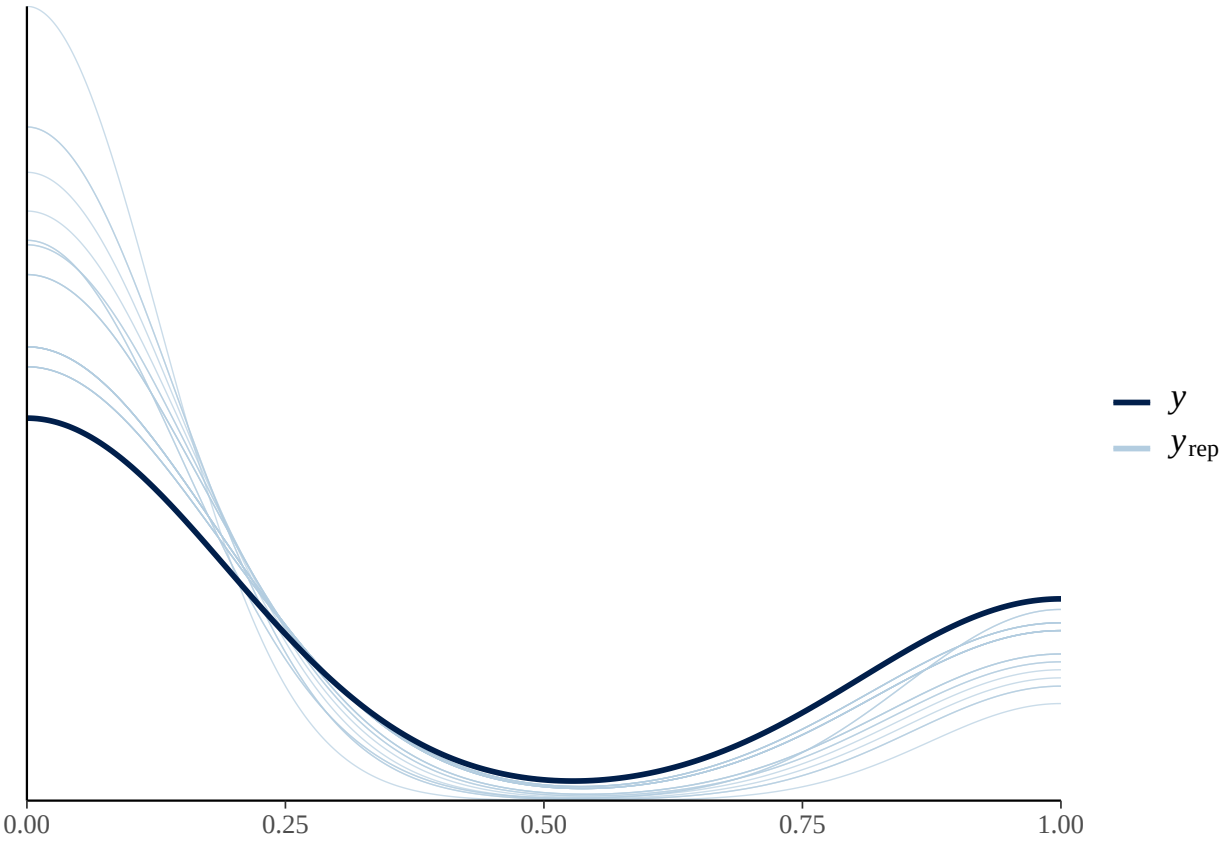
```
##        elpd_diff se_diff
## model2    0.0       0.0
## model1   -1.6       3.4
## model3  -52.0      16.3
## model4 -107.4      29.0
```

From both the comparison and confusion matrix results, we see that model 1 and 2 are quite reliable although accuracy is around 80% which can still be improved.

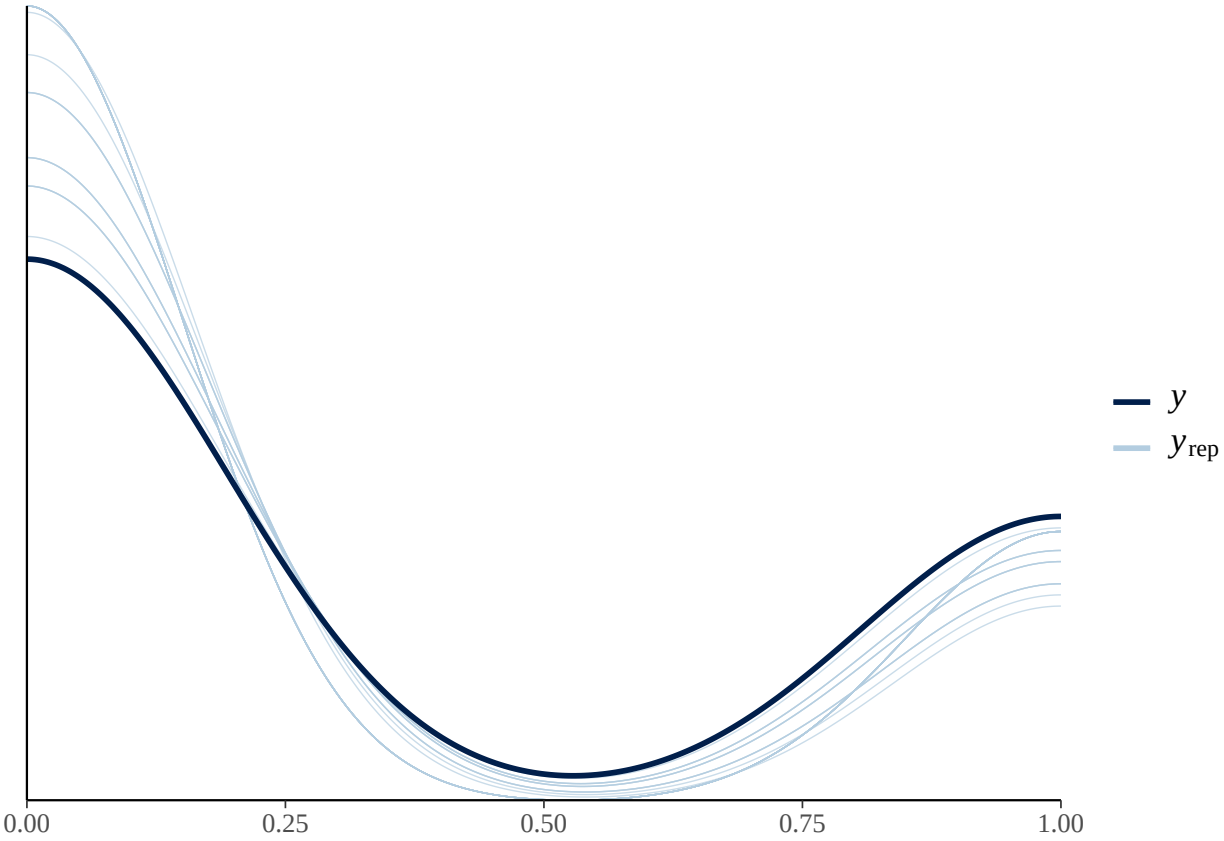
8. Posterior predictive assessment

We plot out the posterior predictive check with ppc_dens_overlay to see how the true values and the predicted values differ from each others.

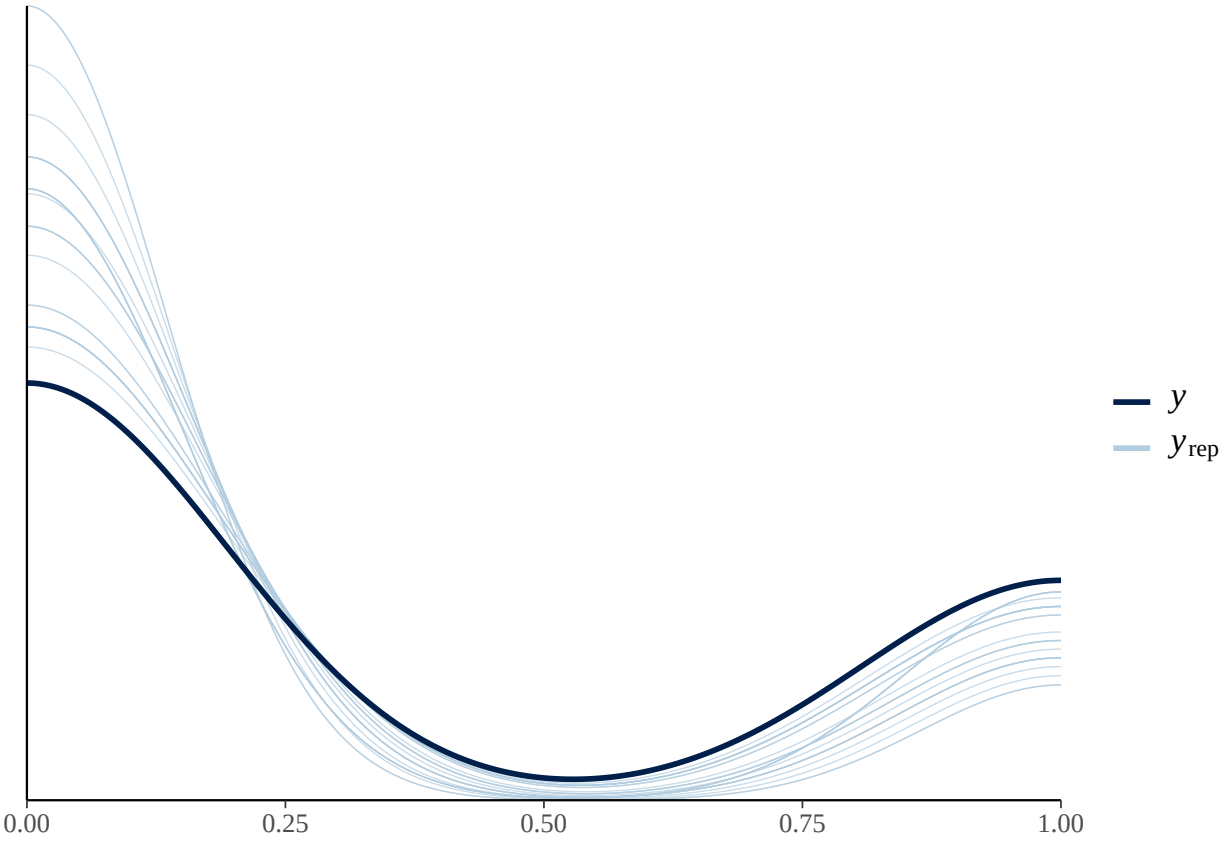
```
yrep = round(extract(fit_selected_1)$Y_pred)
ppc_dens_overlay(test$death, yrep[sample(nrow(yrep), 25), ])
```



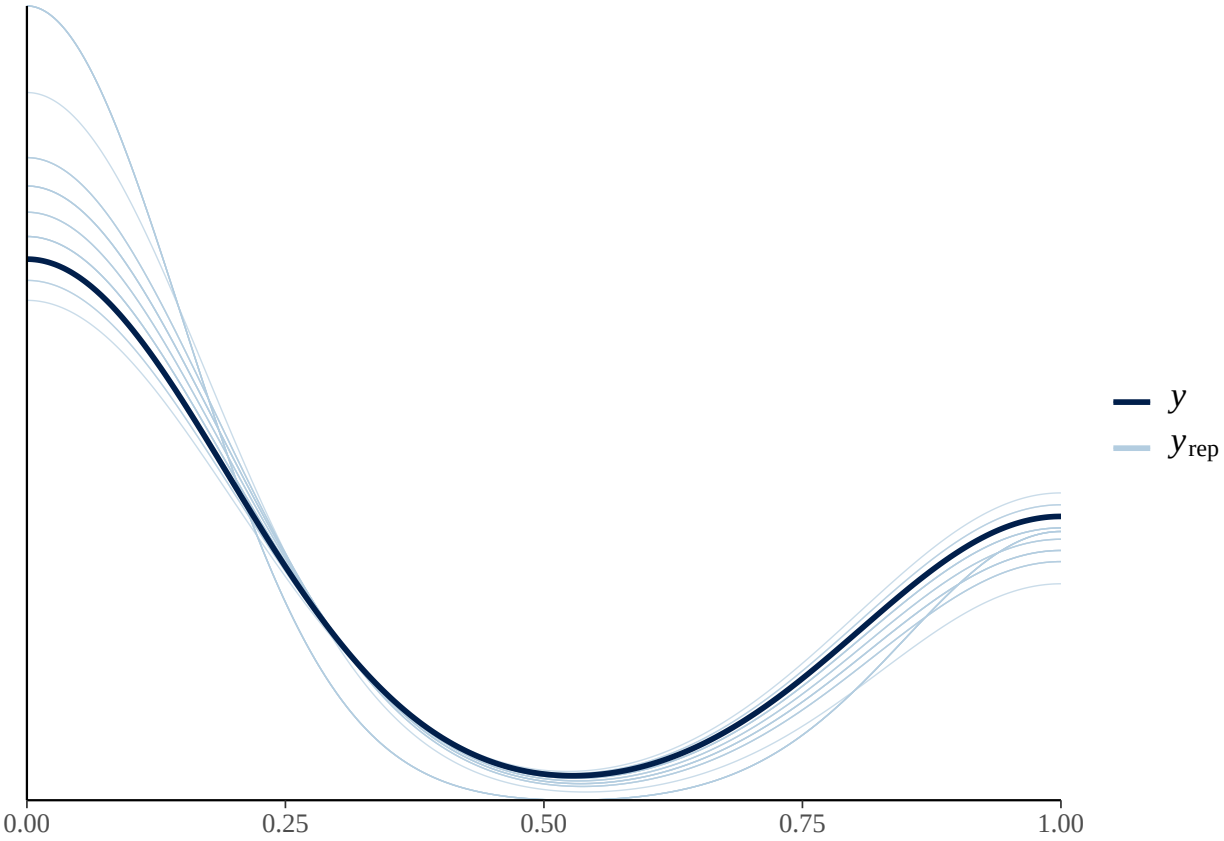

```
yrep = round(extract(fit_selected_2)$Y_pred)
ppc_dens_overlay(test$death, yrep[sample(nrow(yrep), 25), ])
```




```
yrep = round(extract(fit_selected_3)$Y_pred)
ppc_dens_overlay(test$death, yrep[sample(nrow(yrep), 25), ])
```




```
yrep = round(extract(fit_full)$Y_pred)
ppc_dens_overlay(test$death, yrep[sample(nrow(yrep), 25), ])
```



From the plots above, although the y_{rep} line is quite close to that of the Y_true line, the differences between the lines is still visible. We can say that the plots reflect the prediction accuracy well.

10. Sensitivity analysis

We test the sensitivity of the model by changing the prior from multi_normal to priors obtain by the normal distribution coefficient of each feature separately on the first 2 models.

```
data {
  int<lower=1> N; //number of observation
  int<lower=1> M; //number of explanatory variable
  int Y[N];       //independent variable
  matrix[N, M] X; //explanatory variables

  int<lower=1> N_test; //number of test observation
  int Y_true[N_test]; //true Y
  matrix[N_test, M] X_test; //test train

  vector[M] mu; //mean vector
  vector[M] sigma; //covariance matrix
}
parameters {
  vector[M] beta;
}
model {
```

```
  for (m in 1:M) {
    beta[m] ~ normal(mu[m], sigma[m]);//prior based on distribution of the explanatory variables
  }
  Y ~ bernoulli_logit(X*beta); //bernoulli logit since Y has boolean type
}
generated quantities {
  vector[N_test] Y_pred = inv_logit(X_test * beta); //Y_pred
  vector[N] log_lik;
  for (n in 1:N) {
    log_lik[n] = bernoulli_logit_lpmf(Y[n] |X[n, ] * beta);
  }
}
```

```
print("\nRhat values: ")
```

```
## [1] "\nRhat values: "
```

```
rhat(fit_selected_1)
```

```
##      beta[1]      beta[2]      beta[3]    Y_pred[1]    Y_pred[2]    Y_pred[3]
##    0.9996911    1.0007660    1.0028294    1.0005462    1.0017867    1.0009828
##    Y_pred[4]    Y_pred[5]    Y_pred[6]    Y_pred[7]    Y_pred[8]    Y_pred[9]
##    1.0003452    1.0000190    0.9996161    0.9998105    0.9994767    0.9998987
##   Y_pred[10]   Y_pred[11]   Y_pred[12]   Y_pred[13]   Y_pred[14]   Y_pred[15]
##    1.0008750    1.0000483    1.0004795    1.0009845    0.9995847    1.0013665
##   Y_pred[16]   Y_pred[17]   Y_pred[18]   Y_pred[19]   Y_pred[20]   Y_pred[21]
##    1.0011531    1.0003990    1.0001752    0.9997729    1.0007303    1.0011687
##   Y_pred[22]   Y_pred[23]   Y_pred[24]   Y_pred[25]   Y_pred[26]   Y_pred[27]
##    0.9994993    0.9996907    1.0001976    1.0012293    0.9998938    1.0003990
##   Y_pred[28]   Y_pred[29]   Y_pred[30]   Y_pred[31]   Y_pred[32]   Y_pred[33]
##    1.0008647    1.0009828    1.0010127    0.9998745    1.0008388    1.0012029
##   Y_pred[34]   Y_pred[35]   Y_pred[36]   Y_pred[37]   Y_pred[38]   Y_pred[39]
##    1.0004614    1.0014424    0.9999722    1.0000002    0.9995037    1.0017426
##   Y_pred[40]   Y_pred[41]   Y_pred[42]   Y_pred[43]   Y_pred[44]   Y_pred[45]
##    0.9998321    1.0003104    1.0002128    1.0010139    1.0010407    1.0000659
##   Y_pred[46]   Y_pred[47]   Y_pred[48]   Y_pred[49]   Y_pred[50]   Y_pred[51]
##    1.0018618    1.0000272    1.0003397    0.9999901    1.0000343    1.0015704
##   Y_pred[52]   Y_pred[53]   Y_pred[54]   Y_pred[55]   log_lik[1]   log_lik[2]
##    1.0008240    0.9997091    0.9997100    1.0016709    1.0009023    1.0007116
##   log_lik[3]   log_lik[4]   log_lik[5]   log_lik[6]   log_lik[7]   log_lik[8]
##    1.0017014    1.0002898    1.0031076    1.0004162    1.0000592    1.0009811
##   log_lik[9]  log_lik[10]  log_lik[11]  log_lik[12]  log_lik[13]  log_lik[14]
##    0.9995889    1.0012931    0.9999720    1.0003141    0.9995435    1.0013298
##  log_lik[15]  log_lik[16]  log_lik[17]  log_lik[18]  log_lik[19]  log_lik[20]
##    1.0000454    1.0000172    1.0004446    0.9995642    0.9997409    1.0002883
##  log_lik[21]  log_lik[22]  log_lik[23]  log_lik[24]  log_lik[25]  log_lik[26]
##    0.9998138    1.0008504    1.0002824    1.0013942    1.0003323    0.9999912
##  log_lik[27]  log_lik[28]  log_lik[29]  log_lik[30]  log_lik[31]  log_lik[32]
##    1.0001666    0.9994210    0.9998146    0.9996310    1.0013703    0.9996947
##  log_lik[33]  log_lik[34]  log_lik[35]  log_lik[36]  log_lik[37]  log_lik[38]
##    1.0005933    0.9995710    1.0002862    0.9997790    1.0010378    0.9998199
##  log_lik[39]  log_lik[40]  log_lik[41]  log_lik[42]  log_lik[43]  log_lik[44]
##    1.0006772    1.0009841    1.0003151    1.0007845    1.0008441    1.0008273
```

```
##  log_lik[45]  log_lik[46]  log_lik[47]  log_lik[48]  log_lik[49]  log_lik[50]
##    1.0000556    1.0007243    1.0004157    1.0012903    1.0009506    1.0021205
##  log_lik[51]  log_lik[52]  log_lik[53]  log_lik[54]  log_lik[55]  log_lik[56]
##    1.0004136    1.0006302    1.0008244    1.0003386    0.9998684    0.9995347
##  log_lik[57]  log_lik[58]  log_lik[59]  log_lik[60]  log_lik[61]  log_lik[62]
##    0.9998955    1.0018646    1.0003285    1.0000164    1.0022349    1.0013403
##  log_lik[63]  log_lik[64]  log_lik[65]  log_lik[66]  log_lik[67]  log_lik[68]
##    1.0006255    1.0001908    1.0011523    1.0008349    1.0003012    1.0003421
##  log_lik[69]  log_lik[70]  log_lik[71]  log_lik[72]  log_lik[73]  log_lik[74]
##    1.0013170    1.0019234    1.0001121    0.9995826    1.0005268    1.0004492
##  log_lik[75]  log_lik[76]  log_lik[77]  log_lik[78]  log_lik[79]  log_lik[80]
##    1.0004867    1.0000809    0.9995834    1.0007169    1.0007715    0.9997719
##  log_lik[81]  log_lik[82]  log_lik[83]  log_lik[84]  log_lik[85]  log_lik[86]
##    1.0012143    1.0008156    1.0003736    1.0004227    0.9995412    1.0003285
##  log_lik[87]  log_lik[88]  log_lik[89]  log_lik[90]  log_lik[91]  log_lik[92]
##    0.9998845    0.9995664    1.0006857    1.0008156    1.0001583    1.0011753
##  log_lik[93]  log_lik[94]  log_lik[95]  log_lik[96]  log_lik[97]  log_lik[98]
##    1.0004282    1.0006809    1.0030117    0.9994711    0.9998294    1.0007135
##  log_lik[99] log_lik[100] log_lik[101] log_lik[102] log_lik[103] log_lik[104]
##    1.0003293    1.0013837    1.0009513    1.0004323    0.9998380    0.9994857
## log_lik[105] log_lik[106] log_lik[107] log_lik[108] log_lik[109] log_lik[110]
##    1.0011728    1.0013213    1.0004388    1.0009787    1.0001735    1.0002265
## log_lik[111] log_lik[112] log_lik[113] log_lik[114] log_lik[115] log_lik[116]
##    1.0004878    1.0000703    1.0005475    1.0003421    1.0000281    1.0019229
## log_lik[117] log_lik[118] log_lik[119] log_lik[120] log_lik[121] log_lik[122]
##    1.0008258    1.0007067    1.0009923    1.0010030    0.9998231    1.0009094
## log_lik[123] log_lik[124] log_lik[125] log_lik[126] log_lik[127] log_lik[128]
##    0.9998948    1.0009769    1.0020225    1.0004329    1.0008184    1.0024056
## log_lik[129] log_lik[130] log_lik[131] log_lik[132] log_lik[133] log_lik[134]
##    1.0015843    1.0015569    1.0001631    1.0013844    1.0003836    1.0006182
## log_lik[135] log_lik[136] log_lik[137] log_lik[138] log_lik[139] log_lik[140]
##    1.0004970    1.0007470    0.9995412    1.0014540    1.0011753    1.0001767
## log_lik[141] log_lik[142] log_lik[143] log_lik[144] log_lik[145] log_lik[146]
##    1.0002816    1.0005364    1.0010284    1.0007528    1.0009905    1.0001335
## log_lik[147] log_lik[148] log_lik[149] log_lik[150] log_lik[151] log_lik[152]
##    1.0000329    0.9999353    1.0019624    0.9994752    1.0004812    1.0015499
## log_lik[153] log_lik[154] log_lik[155] log_lik[156] log_lik[157] log_lik[158]
##    0.9996404    0.9995999    1.0013374    0.9997703    1.0020264    1.0005848
## log_lik[159] log_lik[160] log_lik[161] log_lik[162] log_lik[163] log_lik[164]
##    1.0011633    1.0015376    1.0007740    1.0015729    1.0007691    0.9999735
## log_lik[165] log_lik[166] log_lik[167] log_lik[168] log_lik[169] log_lik[170]
##    1.0007601    1.0002391    0.9999352    1.0002212    1.0002584    1.0011500
## log_lik[171] log_lik[172] log_lik[173] log_lik[174] log_lik[175] log_lik[176]
##    1.0001601    1.0021549    1.0003012    1.0011446    1.0004207    1.0009990
## log_lik[177] log_lik[178] log_lik[179] log_lik[180] log_lik[181] log_lik[182]
##    1.0005458    1.0012911    1.0010077    1.0003187    1.0005931    1.0006336
## log_lik[183] log_lik[184] log_lik[185] log_lik[186] log_lik[187] log_lik[188]
##    0.9999905    1.0022500    1.0007756    0.9999541    1.0002862    1.0013475
## log_lik[189] log_lik[190] log_lik[191] log_lik[192] log_lik[193] log_lik[194]
##    1.0014252    1.0012617    1.0004852    1.0012912    0.9996276    0.9999334
## log_lik[195] log_lik[196] log_lik[197] log_lik[198] log_lik[199] log_lik[200]
##    0.9995468    0.9999763    1.0007371    1.0009360    1.0001949    1.0004492
## log_lik[201] log_lik[202] log_lik[203] log_lik[204] log_lik[205] log_lik[206]
##    1.0003739    1.0015509    1.0007788    1.0003353    0.9999462    0.9997426
```

```
## log_lik[207] log_lik[208] log_lik[209] log_lik[210] log_lik[211] log_lik[212]
##    1.0006340    1.0008184    0.9999731    1.0013154    1.0008382    0.9996473
## log_lik[213] log_lik[214] log_lik[215] log_lik[216] log_lik[217] log_lik[218]
##    1.0008131    0.9997442    1.0014343    1.0010143    1.0008584    1.0009885
## log_lik[219] log_lik[220] log_lik[221] log_lik[222] log_lik[223] log_lik[224]
##    1.0015943    1.0006639    0.9999672    1.0022085    0.9995947    1.0008497
##         lp__
##    1.0030837
```

```
print("\nCheck divergences: ")
```

```
## [1] "\nCheck divergences: "
```

```
check_divergences(fit_selected_1)
```

```
## 0 of 4000 iterations ended with a divergence.
```

```
print("Check treedepth:")
```

```
## [1] "Check treedepth:"
```

```
check_treedepth(fit_selected_1)
```

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

```
print("\nn_eff values: ")
```

```
## [1] "\nn_eff values: "
```

```
loo(fit_selected_1)$diagnostics$n_eff
```

```
##   [1] 1336.7206 1667.3388 1892.6258 2161.6018  921.4662 2907.6288 1591.7525
##   [8] 1347.1888 2198.3889 2197.1988 3922.8648 1523.1782 1933.2057 1335.9580
##  [15] 2807.6707 1260.4121 1904.6699 3103.2947 1213.7161 1394.8158 4491.4412
##  [22] 2475.1475 3339.7777 2836.0615 3226.1068 1596.5238 2812.6655 2058.5402
##  [29] 3539.7031 2587.8037 1761.3626 1549.1382 2992.9533 1536.9371 2923.6650
##  [36] 2444.3467 1825.4353 4394.8315 1261.2349 1234.9907 1544.1364 2826.5912
##  [43] 2044.0762 1751.7093 2188.3299 1441.2314 3710.1239 1281.5135 2423.7813
##  [50] 2339.4358 3140.1015 1519.0432 2232.5461 2223.7658 3546.4660 3238.0996
##  [57] 4042.6863 1618.7207 2606.4558 2505.0903 1562.1461 2157.4913 1507.7748
##  [64] 2495.8690 1318.1467 1649.0858 1664.7178 2879.7333 1434.5710 1437.2397
##  [71] 3050.6591 1627.2691 3061.4029 1443.3537 1347.5721 1779.2399 3355.4622
##  [78] 1605.7389 1665.7642 1730.2797 1412.2072 2824.3874 3202.9748 2433.3427
##  [85] 2026.9371 2606.4558 3348.6484 1644.7614 1657.9636 1287.0691 1825.0343
##  [92] 1409.5265 3106.9311 2795.3855 1456.7268 1834.3784 1978.5566 1249.7777
##  [99] 1880.1309 1650.6651 2430.5190 2370.9551 3583.8617 2782.1756 1255.6303
## [106] 2817.2736 1558.2518 1548.3246 1320.9483 2198.4356 3057.2495 2130.2888
## [113] 1381.4881 2879.7333 2592.5212 1392.7958 1538.7043 2398.0310 1235.4464
## [120] 1408.5740 1603.1484 2666.9811 1349.0375 1934.1916 1715.6686 1765.3991
```

```
## [127] 1362.5132 1597.5874 2103.4918 1397.0146 3075.8481 1630.6802 2518.9446
## [134] 1247.0116 2766.5286 1247.7306 2026.9371 1332.8816 1409.5265 1083.8397
## [141] 2060.1089 1254.2264 2155.6503 1606.6581 2298.0477 4018.1255 1770.6702
## [148] 3189.0919 2147.3972 2112.0617 2682.8212 1723.0214 2961.6169 3055.4972
## [155] 1467.7809 2060.2753 2151.1389 1253.6708 2299.0751 2073.3847 2007.8671
## [162] 2225.3050 1805.9709 2017.2768  986.1295 3404.9623 4465.9696 3141.6893
## [169] 3279.4702 1485.2123 3436.5457 2309.9041 1664.7178 1999.0187 2435.6380
## [176] 1592.4756 1549.4249 1252.9574 2240.6149 1398.6016 1977.2120 1406.9622
## [183] 2579.1699 1895.1026 1346.5975 3415.7355 2923.6650 1301.5080 1681.9218
## [190] 1309.4643 1776.3755 1279.1017 1845.6747 1700.0606 1686.7066 3773.8223
## [197] 1817.1082 1264.5971 3968.6710 1443.3537 3108.6383 1939.3089 2813.6757
## [204] 1293.3043 4233.2250 1863.7434 2939.8422 1362.5132 1318.8275 2098.1395
## [211] 1809.1465 2292.5518 2301.6926 1757.5942 1399.3224 1600.4563 2628.7305
## [218] 1726.1783 1703.4219 3128.4191 2523.9232 1503.9281 2545.3928 1277.1130
```

```
print("\nPerformance of k_hat values and elpd_loo")
```

```
## [1] "\nPerformance of k_hat values and elpd_loo"
```

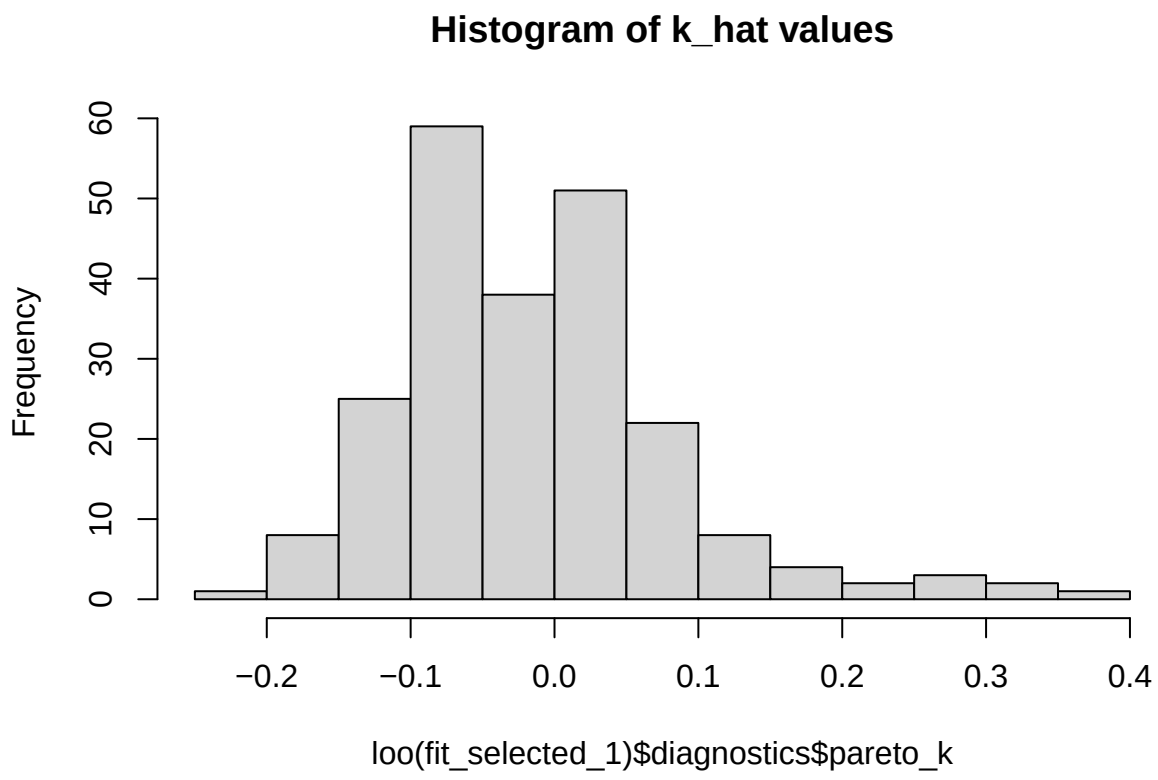
```
loo(fit_selected_1)
```

```
##
## Computed from 4000 by 224 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -113.3  8.7
## p_loo         3.5  0.7
## looic       226.6 17.4
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
print("\nDistribution of k_hat")
```

```
## [1] "\nDistribution of k_hat"
```

```
hist(loo(fit_selected_1)$diagnostics$pareto_k, main = "Histogram of k_hat values")
```




```
print("\nRhat values: ")
```

```
## [1] "\nRhat values: "
```

```
rhat(fit_selected_2)
```

```
##      beta[1]      beta[2]      beta[3]      beta[4]    Y_pred[1]    Y_pred[2]
##    1.0004855    1.0009878    1.0002104    1.0002035    1.0001965    1.0004341
##    Y_pred[3]    Y_pred[4]    Y_pred[5]    Y_pred[6]    Y_pred[7]    Y_pred[8]
##    1.0003251    1.0003813    0.9992429    0.9993961    1.0012941    0.9999114
##    Y_pred[9]   Y_pred[10]   Y_pred[11]   Y_pred[12]   Y_pred[13]   Y_pred[14]
##    0.9997491    1.0000337    0.9995256    0.9994984    0.9999498    0.9994190
##   Y_pred[15]   Y_pred[16]   Y_pred[17]   Y_pred[18]   Y_pred[19]   Y_pred[20]
##    0.9994362    1.0001425    0.9996512    0.9999473    1.0007779    0.9999320
##   Y_pred[21]   Y_pred[22]   Y_pred[23]   Y_pred[24]   Y_pred[25]   Y_pred[26]
##    1.0000085    1.0008270    1.0000317    1.0007584    0.9995704    0.9996993
##   Y_pred[27]   Y_pred[28]   Y_pred[29]   Y_pred[30]   Y_pred[31]   Y_pred[32]
##    0.9996965    0.9998824    1.0010268    1.0001339    1.0009551    1.0000610
##   Y_pred[33]   Y_pred[34]   Y_pred[35]   Y_pred[36]   Y_pred[37]   Y_pred[38]
##    0.9996650    0.9997775    1.0000256    1.0002620    0.9996277    1.0007049
##   Y_pred[39]   Y_pred[40]   Y_pred[41]   Y_pred[42]   Y_pred[43]   Y_pred[44]
##    1.0004322    1.0004214    0.9992135    0.9995838    1.0000306    1.0003216
##   Y_pred[45]   Y_pred[46]   Y_pred[47]   Y_pred[48]   Y_pred[49]   Y_pred[50]
##    1.0002391    1.0005645    0.9996149    0.9999487    1.0012597    0.9991431
##   Y_pred[51]   Y_pred[52]   Y_pred[53]   Y_pred[54]   Y_pred[55]   log_lik[1]
```

```
##    0.9995551    1.0004134    1.0003727    0.9997251    0.9999905    1.0005722
##   log_lik[2]   log_lik[3]   log_lik[4]   log_lik[5]   log_lik[6]   log_lik[7]
##    1.0003036    1.0002353    1.0004180    0.9998356    0.9994991    1.0004420
##   log_lik[8]   log_lik[9]  log_lik[10]  log_lik[11]  log_lik[12]  log_lik[13]
##    1.0003191    1.0007659    0.9993808    1.0002561    1.0003569    1.0001609
##  log_lik[14]  log_lik[15]  log_lik[16]  log_lik[17]  log_lik[18]  log_lik[19]
##    1.0000931    0.9992457    1.0016846    1.0000782    0.9994331    1.0009697
##  log_lik[20]  log_lik[21]  log_lik[22]  log_lik[23]  log_lik[24]  log_lik[25]
##    1.0004681    0.9991318    0.9994219    1.0001213    0.9995574    0.9992943
##  log_lik[26]  log_lik[27]  log_lik[28]  log_lik[29]  log_lik[30]  log_lik[31]
##    1.0002468    0.9997665    0.9999155    1.0004035    0.9994967    0.9995008
##  log_lik[32]  log_lik[33]  log_lik[34]  log_lik[35]  log_lik[36]  log_lik[37]
##    1.0008583    0.9998076    1.0011439    0.9999612    0.9996157    0.9999328
##  log_lik[38]  log_lik[39]  log_lik[40]  log_lik[41]  log_lik[42]  log_lik[43]
##    0.9997371    1.0004101    1.0008038    1.0002256    0.9995233    0.9997584
##  log_lik[44]  log_lik[45]  log_lik[46]  log_lik[47]  log_lik[48]  log_lik[49]
##    0.9999264    0.9995068    1.0003662    0.9994990    1.0002764    0.9999198
##  log_lik[50]  log_lik[51]  log_lik[52]  log_lik[53]  log_lik[54]  log_lik[55]
##    0.9998190    0.9995672    1.0003454    0.9996676    1.0000150    1.0001826
##  log_lik[56]  log_lik[57]  log_lik[58]  log_lik[59]  log_lik[60]  log_lik[61]
##    0.9994582    1.0002517    1.0000633    1.0003196    1.0002788    0.9999406
##  log_lik[62]  log_lik[63]  log_lik[64]  log_lik[65]  log_lik[66]  log_lik[67]
##    0.9994225    1.0003082    0.9995975    1.0001298    0.9999954    0.9997227
##  log_lik[68]  log_lik[69]  log_lik[70]  log_lik[71]  log_lik[72]  log_lik[73]
##    0.9996518    0.9999512    0.9997308    0.9999184    1.0003982    1.0001454
##  log_lik[74]  log_lik[75]  log_lik[76]  log_lik[77]  log_lik[78]  log_lik[79]
##    1.0001750    1.0010850    1.0002551    1.0005425    1.0002979    1.0000862
##  log_lik[80]  log_lik[81]  log_lik[82]  log_lik[83]  log_lik[84]  log_lik[85]
##    1.0008636    1.0002704    0.9999036    0.9993397    0.9998838    1.0006043
##  log_lik[86]  log_lik[87]  log_lik[88]  log_lik[89]  log_lik[90]  log_lik[91]
##    1.0003193    0.9993707    1.0009666    1.0000591    1.0006494    0.9997411
##  log_lik[92]  log_lik[93]  log_lik[94]  log_lik[95]  log_lik[96]  log_lik[97]
##    0.9999950    0.9993257    0.9993215    0.9998478    1.0008247    0.9999913
##  log_lik[98]  log_lik[99] log_lik[100] log_lik[101] log_lik[102] log_lik[103]
##    1.0005531    1.0009625    0.9997493    0.9994226    1.0003266    0.9997167
## log_lik[104] log_lik[105] log_lik[106] log_lik[107] log_lik[108] log_lik[109]
##    0.9994941    1.0006712    1.0004230    1.0004838    1.0000476    1.0005128
## log_lik[110] log_lik[111] log_lik[112] log_lik[113] log_lik[114] log_lik[115]
##    1.0001471    0.9993354    1.0002627    1.0008046    0.9996925    0.9998021
## log_lik[116] log_lik[117] log_lik[118] log_lik[119] log_lik[120] log_lik[121]
##    0.9997336    0.9998658    0.9997337    1.0010359    1.0001158    1.0005636
## log_lik[122] log_lik[123] log_lik[124] log_lik[125] log_lik[126] log_lik[127]
##    1.0003030    1.0011512    1.0001467    0.9996993    1.0002187    1.0004321
## log_lik[128] log_lik[129] log_lik[130] log_lik[131] log_lik[132] log_lik[133]
##    1.0004954    0.9996803    0.9997784    0.9996611    0.9999083    0.9998201
## log_lik[134] log_lik[135] log_lik[136] log_lik[137] log_lik[138] log_lik[139]
##    1.0005282    0.9999433    1.0011014    1.0005983    1.0000184    1.0000371
## log_lik[140] log_lik[141] log_lik[142] log_lik[143] log_lik[144] log_lik[145]
##    1.0012435    1.0001578    1.0015460    0.9994052    1.0001049    0.9994650
## log_lik[146] log_lik[147] log_lik[148] log_lik[149] log_lik[150] log_lik[151]
##    1.0002578    1.0004286    0.9996236    0.9997771    1.0007529    0.9994832
## log_lik[152] log_lik[153] log_lik[154] log_lik[155] log_lik[156] log_lik[157]
##    0.9995020    0.9995564    1.0002416    0.9999334    1.0003967    0.9998161
## log_lik[158] log_lik[159] log_lik[160] log_lik[161] log_lik[162] log_lik[163]
```

```
##    1.0005455    0.9998456    1.0003920    1.0000982    1.0005351    1.0013321
## log_lik[164] log_lik[165] log_lik[166] log_lik[167] log_lik[168] log_lik[169]
##    1.0005090    1.0007256    0.9993406    1.0003230    0.9996488    0.9992195
## log_lik[170] log_lik[171] log_lik[172] log_lik[173] log_lik[174] log_lik[175]
##    1.0000184    0.9995891    0.9995363    0.9997227    1.0004249    0.9995533
## log_lik[176] log_lik[177] log_lik[178] log_lik[179] log_lik[180] log_lik[181]
##    1.0000102    1.0003785    1.0005683    0.9995854    1.0002879    0.9993731
## log_lik[182] log_lik[183] log_lik[184] log_lik[185] log_lik[186] log_lik[187]
##    1.0001733    1.0001874    1.0004302    1.0006905    0.9995912    0.9999227
## log_lik[188] log_lik[189] log_lik[190] log_lik[191] log_lik[192] log_lik[193]
##    1.0001909    0.9997729    1.0002775    1.0001311    1.0003973    1.0000467
## log_lik[194] log_lik[195] log_lik[196] log_lik[197] log_lik[198] log_lik[199]
##    1.0013209    1.0007854    0.9991679    1.0000294    1.0008962    0.9998224
## log_lik[200] log_lik[201] log_lik[202] log_lik[203] log_lik[204] log_lik[205]
##    1.0002694    0.9996345    0.9995371    1.0000935    1.0012088    0.9993642
## log_lik[206] log_lik[207] log_lik[208] log_lik[209] log_lik[210] log_lik[211]
##    0.9998996    1.0001077    1.0004072    1.0011559    0.9997052    0.9999285
## log_lik[212] log_lik[213] log_lik[214] log_lik[215] log_lik[216] log_lik[217]
##    1.0003411    0.9995334    0.9997041    0.9998421    0.9998841    0.9995799
## log_lik[218] log_lik[219] log_lik[220] log_lik[221] log_lik[222] log_lik[223]
##    1.0003800    0.9998851    0.9994097    1.0002375    0.9996921    0.9999951
## log_lik[224]         lp__
##    1.0002485    1.0007431
```

```
print("\nCheck divergences: ")
```

```
## [1] "\nCheck divergences: "
```

```
check_divergences(fit_selected_2)
```

```
## 0 of 4000 iterations ended with a divergence.
```

```
print("Check treedepth:")
```

```
## [1] "Check treedepth:"
```

```
check_treedepth(fit_selected_2)
```

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

```
print("\nn_eff values: ")
```

```
## [1] "\nn_eff values: "
```

```
loo(fit_selected_2)$diagnostics$n_eff
```

```
##   [1] 1956.176 2206.290 2441.256 2092.152 1069.666 3379.631 2122.590 2052.409
##   [9] 2784.337 3166.601 2650.278 2120.549 2294.859 2170.420 2471.850 1492.843
##  [17] 2664.531 3165.983 1496.662 1976.225 3456.571 3300.932 2716.830 2269.325
```

```
##  [25] 3920.460 2073.795 3496.210 2308.194 2566.718 2550.740 3056.276 1971.009
##  [33] 3117.057 2170.763 2872.234 2520.131 2514.266 3736.574 1772.018 1683.362
##  [41] 2170.424 2625.274 2866.033 2584.115 2168.383 2127.307 2788.386 1990.588
##  [49] 2688.185 2736.959 2612.539 2212.474 3024.462 2739.623 2971.555 3539.655
##  [57] 2626.556 2551.086 2242.939 2836.153 2545.514 3180.172 2529.388 3089.415
##  [65] 2064.641 2548.864 2024.339 3572.910 2426.329 2278.751 3060.160 1901.113
##  [73] 2588.977 2129.000 1779.608 2284.687 2672.377 1990.383 2277.690 2577.817
##  [81] 2322.350 2533.654 3830.159 2801.349 2743.132 2215.873 3111.621 2069.735
##  [89] 2510.682 1856.317 2198.227 2290.790 3555.847 3640.568 1968.796 2517.229
##  [97] 2375.838 1824.123 2209.684 2770.474 3374.612 2454.185 3924.021 2903.848
## [105] 1787.849 2666.928 2177.733 2339.883 2314.376 2516.630 3162.382 2630.538
## [113] 1915.137 3577.304 3135.346 1957.418 2168.236 3000.929 1594.235 2152.533
## [121] 2153.777 2716.243 1698.524 2252.918 2397.212 2477.793 2028.187 1962.924
## [129] 2846.354 2442.330 3719.173 2243.690 2833.734 1662.675 2809.926 1623.098
## [137] 2723.099 2212.108 2265.604 1204.611 2312.110 1448.622 3210.126 2260.008
## [145] 2983.878 2571.258 2495.931 2675.164 2391.898 2863.112 3019.713 2899.414
## [153] 3264.461 3386.902 2335.371 2783.382 2494.507 1701.099 2725.779 2597.550
## [161] 2546.581 2531.133 1918.445 2800.194 1331.149 3998.477 2284.368 3486.826
## [169] 3803.989 2269.423 3849.369 2296.548 2024.339 2633.115 3425.850 2373.445
## [177] 2499.821 1815.118 3076.106 1884.370 2019.576 2116.426 2703.926 2300.387
## [185] 1919.342 3803.680 2942.848 2074.419 2568.673 2041.936 2248.179 1940.221
## [193] 2267.964 1933.028 2289.539 3875.392 2468.505 1717.585 3475.476 2099.333
## [201] 3452.800 2904.529 2610.851 1615.408 4002.521 2244.080 2729.093 2027.128
## [209] 1646.672 2797.906 2632.456 3190.287 3245.943 2133.869 2390.879 2611.405
## [217] 2326.074 2691.277 2678.849 3258.578 3236.648 2215.986 3299.503 1970.839
```

```
print("\nPerformance of k_hat values and elpd_loo")
```

```
## [1] "\nPerformance of k_hat values and elpd_loo"
```

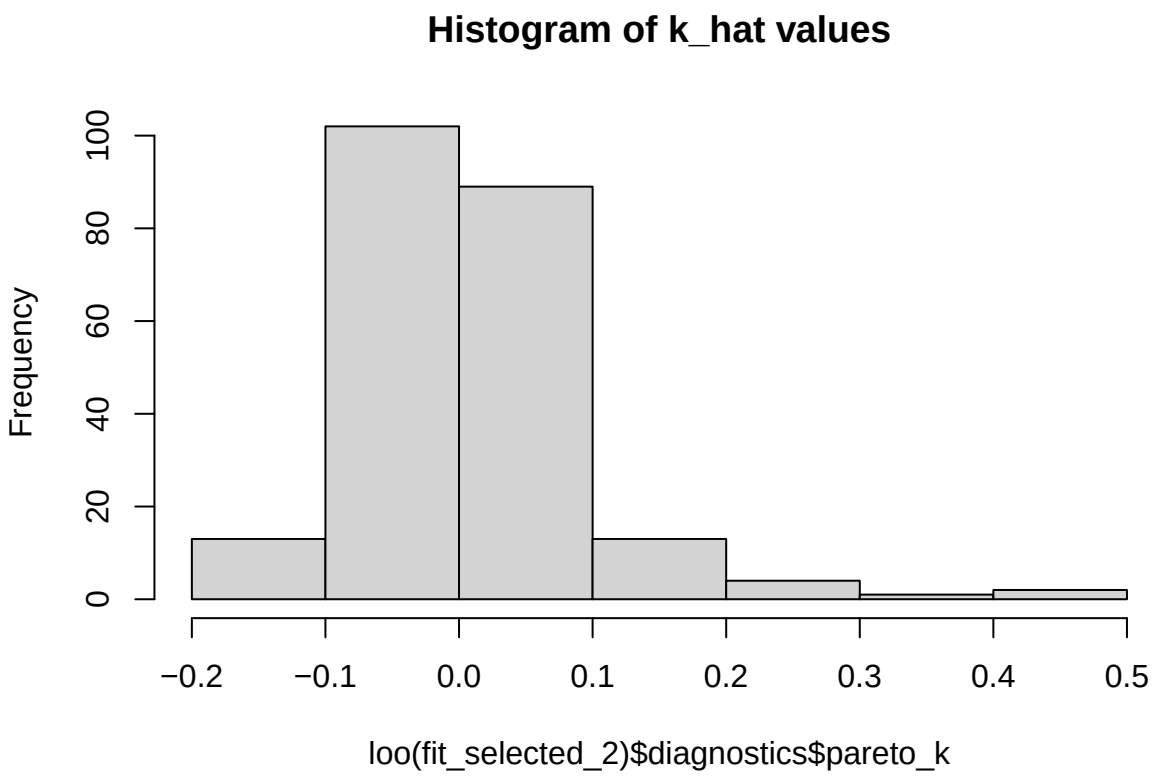
```
loo(fit_selected_2)
```

```
##
## Computed from 4000 by 224 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -110.8  8.8
## p_loo         4.8  0.8
## looic       221.5 17.5
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
print("\nDistribution of k_hat")
```

```
## [1] "\nDistribution of k_hat"
```

```
hist(loo(fit_selected_2)$diagnostics$pareto_k, main = "Histogram of k_hat values")
```




```
print("\nRhat values: ")
```

```
## [1] "\nRhat values: "
```

```
rhat(fit_full)
```

```
##      beta[1]      beta[2]      beta[3]      beta[4]      beta[5]      beta[6]
##    1.2248960   11.9907513    1.1656289    7.7420231    1.0051637    5.5047267
##      beta[7]      beta[8]      beta[9]     beta[10]     beta[11]    Y_pred[1]
##    1.2521907    3.9883796    1.3513733    4.7374154   12.1898046    1.0002091
##    Y_pred[2]    Y_pred[3]    Y_pred[4]    Y_pred[5]    Y_pred[6]    Y_pred[7]
##          NaN          NaN    1.0000216    1.0156816    1.0305025          NaN
##    Y_pred[8]    Y_pred[9]   Y_pred[10]   Y_pred[11]   Y_pred[12]   Y_pred[13]
##          NaN    1.0140931    1.0014743          NaN    1.0022999    1.0011752
##   Y_pred[14]   Y_pred[15]   Y_pred[16]   Y_pred[17]   Y_pred[18]   Y_pred[19]
##    1.0005165    1.0880622    1.0004074    1.0091766    1.0047231          NaN
##   Y_pred[20]   Y_pred[21]   Y_pred[22]   Y_pred[23]   Y_pred[24]   Y_pred[25]
##          NaN          NaN          NaN    1.0052221    1.0003822          NaN
##   Y_pred[26]   Y_pred[27]   Y_pred[28]   Y_pred[29]   Y_pred[30]   Y_pred[31]
##    1.0286953    1.0139554          NaN    1.0006660    1.0009031          NaN
##   Y_pred[32]   Y_pred[33]   Y_pred[34]   Y_pred[35]   Y_pred[36]   Y_pred[37]
##    1.0191072          NaN    1.0354551          NaN    1.0035384          NaN
##   Y_pred[38]   Y_pred[39]   Y_pred[40]   Y_pred[41]   Y_pred[42]   Y_pred[43]
##          NaN    1.0565854    1.0013640    1.0190408    1.0225400    1.0166564
##   Y_pred[44]   Y_pred[45]   Y_pred[46]   Y_pred[47]   Y_pred[48]   Y_pred[49]
```

```
##          NaN    1.0006475    1.0126146    1.0368685    1.0089092          NaN
##   Y_pred[50]   Y_pred[51]   Y_pred[52]   Y_pred[53]   Y_pred[54]   Y_pred[55]
##    1.1259380    1.0721154    1.0006622    1.0096272    0.9989995          NaN
##   log_lik[1]   log_lik[2]   log_lik[3]   log_lik[4]   log_lik[5]   log_lik[6]
##    1.1933120    1.2457786    1.1118447    1.7579985    1.0066169    1.0772406
##   log_lik[7]   log_lik[8]   log_lik[9]  log_lik[10]  log_lik[11]  log_lik[12]
##    1.0410701    1.4306844    1.0013370    1.1869892    1.0008813    1.0000216
##  log_lik[13]  log_lik[14]  log_lik[15]  log_lik[16]  log_lik[17]  log_lik[18]
##    1.0061717    1.2056220    1.0144623    1.0328157    1.0025718    1.0305025
##  log_lik[19]  log_lik[20]  log_lik[21]  log_lik[22]  log_lik[23]  log_lik[24]
##          NaN    1.0001213    1.0853085    1.0277009    1.0008593    1.0001653
##  log_lik[25]  log_lik[26]  log_lik[27]  log_lik[28]  log_lik[29]  log_lik[30]
##    1.0705382    1.0772253    1.1519886    1.1911667    1.0013176    1.0033864
##  log_lik[31]  log_lik[32]  log_lik[33]  log_lik[34]  log_lik[35]  log_lik[36]
##    1.3595254    1.1384714    1.0101632    1.5362215    1.0014535    1.0026696
##  log_lik[37]  log_lik[38]  log_lik[39]  log_lik[40]  log_lik[41]  log_lik[42]
##    1.0053740    1.0140931    1.2694495    1.2793660    1.0003940          NaN
##  log_lik[43]  log_lik[44]  log_lik[45]  log_lik[46]  log_lik[47]  log_lik[48]
##    1.5555263    1.0744409    1.0004701    1.6246184    1.4011593    1.0116350
##  log_lik[49]  log_lik[50]  log_lik[51]  log_lik[52]  log_lik[53]  log_lik[54]
##    1.0011752    1.0013864    1.1887769    1.7970488    1.0023125    1.0316137
##  log_lik[55]  log_lik[56]  log_lik[57]  log_lik[58]  log_lik[59]  log_lik[60]
##    1.1579049    1.0005165    1.0006166    1.0943760    1.3690184    1.0045532
##  log_lik[61]  log_lik[62]  log_lik[63]  log_lik[64]  log_lik[65]  log_lik[66]
##    1.1287501    1.0880611    1.0563243    1.0867521    1.0004074    1.0056762
##  log_lik[67]  log_lik[68]  log_lik[69]  log_lik[70]  log_lik[71]  log_lik[72]
##    1.5219778    1.0091766    1.0352029    1.1350076    1.0047231    1.0882783
##  log_lik[73]  log_lik[74]  log_lik[75]  log_lik[76]  log_lik[77]  log_lik[78]
##    1.0125147    1.0001458    1.0470780          NaN          NaN    1.0133805
##  log_lik[79]  log_lik[80]  log_lik[81]  log_lik[82]  log_lik[83]  log_lik[84]
##    1.0004245          NaN    1.2438654    1.0173065    1.3709902    1.0152171
##  log_lik[85]  log_lik[86]  log_lik[87]  log_lik[88]  log_lik[89]  log_lik[90]
##    1.1142379    1.0575274    1.3658128    1.0802840    1.2319366    1.4270488
##  log_lik[91]  log_lik[92]  log_lik[93]  log_lik[94]  log_lik[95]  log_lik[96]
##    1.2868051    1.1370193    1.1947117    1.0362223    1.0075427          NaN
##  log_lik[97]  log_lik[98]  log_lik[99] log_lik[100] log_lik[101] log_lik[102]
##    1.0906714    0.9999032    1.1329134    1.0018769    1.0041184    1.0021727
## log_lik[103] log_lik[104] log_lik[105] log_lik[106] log_lik[107] log_lik[108]
##    1.0286953    1.0206796    1.0483083          NaN    1.1635612    1.0039015
## log_lik[109] log_lik[110] log_lik[111] log_lik[112] log_lik[113] log_lik[114]
##          NaN    1.0070433    1.4215401    1.0014901    1.2989136    1.0139554
## log_lik[115] log_lik[116] log_lik[117] log_lik[118] log_lik[119] log_lik[120]
##    1.0319782    1.6050007    1.3210037    1.0105410    1.1846099    1.0009031
## log_lik[121] log_lik[122] log_lik[123] log_lik[124] log_lik[125] log_lik[126]
##    1.0011678    1.6619390    1.0738975    1.2792937    1.2329739    1.1174080
## log_lik[127] log_lik[128] log_lik[129] log_lik[130] log_lik[131] log_lik[132]
##    1.1221650    1.0089591    1.0621548    1.0613737    1.0409373          NaN
## log_lik[133] log_lik[134] log_lik[135] log_lik[136] log_lik[137] log_lik[138]
##    1.0244992    1.5148741    1.0101496    1.1090874    1.6603521    1.1624491
## log_lik[139] log_lik[140] log_lik[141] log_lik[142] log_lik[143] log_lik[144]
##    1.2012910          NaN    0.9999081    1.0702857    1.2902060    1.0041169
## log_lik[145] log_lik[146] log_lik[147] log_lik[148] log_lik[149] log_lik[150]
##    1.2232790    1.4074326    1.0145943    1.1749176    2.0806565          NaN
## log_lik[151] log_lik[152] log_lik[153] log_lik[154] log_lik[155] log_lik[156]
```

```
##    1.3454566    1.1171508    1.0103240    1.0171160    1.0012988    1.4400085
## log_lik[157] log_lik[158] log_lik[159] log_lik[160] log_lik[161] log_lik[162]
##    2.5631576    1.4309439    1.0004059    1.0738474    1.1499580    2.1802253
## log_lik[163] log_lik[164] log_lik[165] log_lik[166] log_lik[167] log_lik[168]
##    3.0805517    1.0055260          NaN    1.0393213    1.1532466    1.0104711
## log_lik[169] log_lik[170] log_lik[171] log_lik[172] log_lik[173] log_lik[174]
##    1.0263709    1.0001416    1.0225400    1.0134126    1.3061123    1.4857496
## log_lik[175] log_lik[176] log_lik[177] log_lik[178] log_lik[179] log_lik[180]
##    1.0932634    1.0083928          NaN    1.1156549    1.0199915    1.1765883
## log_lik[181] log_lik[182] log_lik[183] log_lik[184] log_lik[185] log_lik[186]
##    1.0211055    1.0000608    1.0006475    1.0126146    1.0181634    1.0368685
## log_lik[187] log_lik[188] log_lik[189] log_lik[190] log_lik[191] log_lik[192]
##    1.0089092    1.3018001    1.0021837    1.2814640    1.0002364    1.2115433
## log_lik[193] log_lik[194] log_lik[195] log_lik[196] log_lik[197] log_lik[198]
##    0.9999383    1.8356773          NaN    1.1089354    1.0007946    1.1426052
## log_lik[199] log_lik[200] log_lik[201] log_lik[202] log_lik[203] log_lik[204]
##    1.0349260    1.0020372    1.0066833    1.0692447    1.0010549    1.3068157
## log_lik[205] log_lik[206] log_lik[207] log_lik[208] log_lik[209] log_lik[210]
##    1.1173136    1.0002256    1.0016411    1.0006622    1.1234321    1.0017097
## log_lik[211] log_lik[212] log_lik[213] log_lik[214] log_lik[215] log_lik[216]
##    1.0043885    1.0096272    1.4947844    1.0016719    1.1289446    1.2652319
## log_lik[217] log_lik[218] log_lik[219] log_lik[220] log_lik[221] log_lik[222]
##    1.0008848    1.0110147    1.3683653    1.0870804    1.0130931    1.3023440
## log_lik[223] log_lik[224]         lp__
##    1.0157804    1.0001925    2.9714838
```

```
print("\nCheck divergences: ")
```

```
## [1] "\nCheck divergences: "
```

```
check_divergences(fit_full)
```

```
## 0 of 4000 iterations ended with a divergence.
```

```
print("Check treedepth:")
```

```
## [1] "Check treedepth:"
```

```
check_treedepth(fit_full)
```

```
## 3880 of 4000 iterations saturated the maximum tree depth of 10 (97%).
## Try increasing 'max_treedepth' to avoid saturation.
```

```
print("\nn_eff values: ")
```

```
## [1] "\nn_eff values: "
```

```
loo(fit_full)$diagnostics$n_eff
```

```
##   [1] 1.133436e+00 2.331080e+01 3.655769e+00 5.901757e+00 2.000000e+03
##   [6] 1.168263e+02 4.656349e+00 2.236155e-01 2.000000e+03 2.955808e+00
##  [11] 2.000000e+03 2.000000e+03 2.000000e+03 1.843894e+00 2.683883e+02
##  [16] 3.548415e+00 2.000000e+03 2.000000e+03 2.000000e+03 2.000000e+03
##  [21] 1.470023e+01 2.000000e+03 2.000000e+03 2.000000e+03 2.000000e+03
##  [26] 4.367637e+00 3.948449e+00 1.386414e+00 2.000000e+03 2.000000e+03
##  [31] 4.091598e+01 7.448336e+00 2.000000e+03 9.145546e+00 2.000000e+03
##  [36] 3.865977e+03 2.000000e+03 2.000000e+03 2.005100e+00 1.977670e+00
##  [41] 2.000000e+03 2.000000e+03 3.226158e-01 8.578580e+01 2.000000e+03
##  [46] 1.233856e-01 4.325014e+01 4.519113e+00 2.000000e+03 2.000000e+03
##  [51] 5.483934e+00 2.240316e-01 3.922107e+03 2.000000e+03 8.879854e+00
##  [56] 2.000000e+03 2.000000e+03 2.329439e+00 4.052661e+01 2.188462e+03
##  [61] 3.294705e+00 1.515138e+01 1.049616e+00 1.890731e+00 2.000000e+03
##  [66] 3.329859e+03 4.512189e-01 2.000000e+03 2.000000e+03 5.347945e+00
##  [71] 2.000000e+03 8.226421e+00 2.000000e+03 2.000000e+03 9.581388e+00
##  [76] 2.000000e+03 2.000000e+03 4.137150e+00 2.000000e+03 2.000000e+03
##  [81] 3.133991e+00 2.000000e+03 6.364932e-02 1.179041e+02 1.118991e+01
##  [86] 5.556400e+00 5.588471e+01 1.239842e+01 9.791774e-01 7.239380e+00
##  [91] 1.309478e+00 9.520386e+00 2.181906e+01 3.254435e+01 2.000000e+03
##  [96] 2.000000e+03 2.655179e+00 4.009954e+03 1.298298e+02 2.000000e+03
## [101] 2.000000e+03 2.000000e+03 2.000000e+03 9.894237e+01 3.685942e+00
## [106] 2.000000e+03 1.637870e+00 1.921753e+03 2.000000e+03 2.000000e+03
## [111] 7.310856e+01 2.000000e+03 1.363910e+01 2.000000e+03 2.000000e+03
## [116] 1.979124e+00 1.766613e+00 2.000000e+03 1.436503e+00 2.000000e+03
## [121] 2.000000e+03 1.749049e-01 3.549342e+00 2.063217e+00 1.665496e+00
## [126] 3.204841e-01 1.161582e+01 2.000000e+03 1.421007e+01 3.211820e+01
## [131] 2.000000e+03 2.000000e+03 1.963623e+01 3.772983e+00 3.500093e+03
## [136] 1.336835e+00 4.031298e+00 1.214746e+00 2.227366e+00 2.000000e+03
## [141] 2.000000e+03 2.353077e+00 1.872892e-01 2.000000e+03 3.968559e+01
## [146] 2.029058e+01 3.953485e+00 2.780359e+00 7.538856e-01 2.000000e+03
## [151] 1.244809e+01 3.053724e+00 2.000000e+03 2.000000e+03 3.068512e+03
## [156] 1.831315e+01 6.860577e-01 2.399635e+00 2.000000e+03 2.500752e+01
## [161] 5.264939e+01 1.367364e+00 3.110563e+00 2.000000e+03 2.000000e+03
## [166] 3.570945e+01 2.351511e+00 3.816412e+02 2.117698e+01 2.000000e+03
## [171] 2.000000e+03 2.000000e+03 3.703399e-01 1.966237e-02 5.795542e+00
## [176] 1.209767e+01 2.000000e+03 9.519393e+00 2.000000e+03 6.974762e+00
## [181] 2.000000e+03 2.000000e+03 2.000000e+03 2.000000e+03 1.048446e+01
## [186] 2.000000e+03 2.000000e+03 2.156516e+00 2.000000e+03 4.883080e+00
## [191] 2.000000e+03 1.173092e+01 2.000000e+03 3.073348e-01 2.000000e+03
## [196] 1.288715e+01 2.000000e+03 6.786950e+00 2.000000e+03 2.000000e+03
## [201] 2.000000e+03 2.530573e+01 2.000000e+03 3.119482e+00 2.000000e+03
## [206] 2.000000e+03 2.000000e+03 4.008024e+03 2.189306e+00 2.000000e+03
## [211] 1.378617e+03 2.000000e+03 4.727478e-02 3.968847e+03 1.894094e+00
## [216] 2.729697e-01 2.000000e+03 2.000000e+03 3.692657e+00 5.651685e+01
## [221] 2.000000e+03 6.803334e+00 2.000000e+03 2.000000e+03
```

```
print("\nPerformance of k_hat values and elpd_loo")
```

```
## [1] "\nPerformance of k_hat values and elpd_loo"
```

```
loo(fit_full)
```

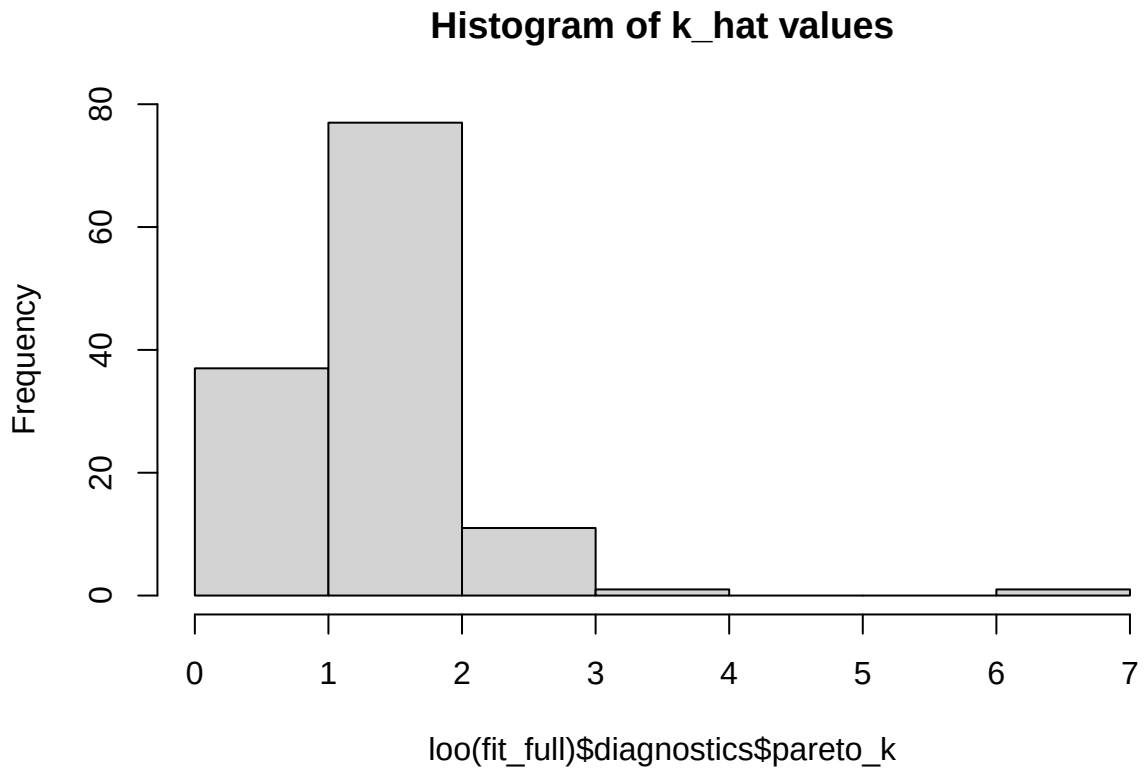
```
##
```

```
## Computed from 4000 by 224 log-likelihood matrix
##
##          Estimate     SE
## elpd_loo -25742.3 2496.0
## p_loo      1541.9  170.4
## looic     51484.6 4992.0
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)       1    0.4%   33
##  (0.5, 0.7]   (ok)        11    4.9%   13
##    (0.7, 1]   (bad)       25   11.2%   0
##    (1, Inf)   (very bad) 187   83.5%   0
## See help('pareto-k-diagnostic') for details.
```

```
print("\nDistribution of k_hat")
```

```
## [1] "\nDistribution of k_hat"
```

```
hist(loo(fit_full)$diagnostics$pareto_k, main = "Histogram of k_hat values")
```



From the results above, we can see that the first 2 model are not sensitive to changes. However, the full model is very sensitive and the result is much worse than what we obtained with the multivariate distribution.

11. Issues and potential improvement

The issue of this dataset is the fact that there are many categorical features alongside with numberical features that has large mean and standard deviation. This in fact affects a lot on the prediction accuracy. Another factor that is needed to be considered is the prior choice. We notice that better prior choice is needed by the result of the previous models.

Testing whether normalization can improve the prediction accuracy with the first 2 combination of features:

```
print("\nRhat values: ")
```

```
## [1] "\nRhat values: "
```

```
rhat(fit_selected_1)
```

```
##      beta[1]      beta[2]      beta[3]    Y_pred[1]    Y_pred[2]    Y_pred[3]
##    1.0008081    1.0012824    0.9998877    1.0014404    0.9997627    1.0019762
##    Y_pred[4]    Y_pred[5]    Y_pred[6]    Y_pred[7]    Y_pred[8]    Y_pred[9]
##    1.0011687    1.0010972    1.0004854    1.0013288    1.0005717    1.0000835
##   Y_pred[10]   Y_pred[11]   Y_pred[12]   Y_pred[13]   Y_pred[14]   Y_pred[15]
##    1.0009464    1.0007229    0.9997320    1.0005198    1.0009334    0.9999551
##   Y_pred[16]   Y_pred[17]   Y_pred[18]   Y_pred[19]   Y_pred[20]   Y_pred[21]
##    1.0013054    0.9996615    1.0006939    1.0002716    1.0002309    1.0005186
##   Y_pred[22]   Y_pred[23]   Y_pred[24]   Y_pred[25]   Y_pred[26]   Y_pred[27]
##    1.0002949    1.0000010    0.9997032    0.9999502    0.9998892    0.9996615
##   Y_pred[28]   Y_pred[29]   Y_pred[30]   Y_pred[31]   Y_pred[32]   Y_pred[33]
##    1.0015844    1.0021265    1.0012172    1.0015708    1.0009055    0.9997285
##   Y_pred[34]   Y_pred[35]   Y_pred[36]   Y_pred[37]   Y_pred[38]   Y_pred[39]
##    0.9994523    1.0008628    1.0001539    1.0001912    1.0004902    0.9996434
##   Y_pred[40]   Y_pred[41]   Y_pred[42]   Y_pred[43]   Y_pred[44]   Y_pred[45]
##    1.0001956    0.9993909    0.9998532    1.0015480    0.9998713    1.0002318
##   Y_pred[46]   Y_pred[47]   Y_pred[48]   Y_pred[49]   Y_pred[50]   Y_pred[51]
##    0.9994293    1.0018326    1.0007505    1.0001923    0.9995373    1.0001481
##   Y_pred[52]   Y_pred[53]   Y_pred[54]   Y_pred[55]   log_lik[1]   log_lik[2]
##    1.0020931    1.0001189    1.0007171    1.0004879    1.0021627    1.0014746
##   log_lik[3]   log_lik[4]   log_lik[5]   log_lik[6]   log_lik[7]   log_lik[8]
##    1.0016181    0.9997816    1.0007138    1.0006138    1.0017819    1.0014091
##   log_lik[9]  log_lik[10]  log_lik[11]  log_lik[12]  log_lik[13]  log_lik[14]
##    1.0009611    0.9999051    0.9999863    1.0006082    1.0004040    1.0015475
##  log_lik[15]  log_lik[16]  log_lik[17]  log_lik[18]  log_lik[19]  log_lik[20]
##    1.0003238    0.9998572    1.0000876    1.0001021    1.0011220    1.0010992
##  log_lik[21]  log_lik[22]  log_lik[23]  log_lik[24]  log_lik[25]  log_lik[26]
##    1.0015381    1.0021094    0.9998534    1.0018923    1.0019259    1.0010001
##  log_lik[27]  log_lik[28]  log_lik[29]  log_lik[30]  log_lik[31]  log_lik[32]
##    1.0001128    0.9997337    0.9996244    1.0010150    1.0005854    1.0003311
##  log_lik[33]  log_lik[34]  log_lik[35]  log_lik[36]  log_lik[37]  log_lik[38]
##    1.0010461    1.0001381    1.0004499    0.9999522    0.9997857    0.9996669
##  log_lik[39]  log_lik[40]  log_lik[41]  log_lik[42]  log_lik[43]  log_lik[44]
##    1.0003635    0.9996577    1.0021016    1.0001187    0.9996168    0.9994844
##  log_lik[45]  log_lik[46]  log_lik[47]  log_lik[48]  log_lik[49]  log_lik[50]
##    1.0014091    0.9996346    1.0001919    1.0000429    1.0002951    1.0002399
##  log_lik[51]  log_lik[52]  log_lik[53]  log_lik[54]  log_lik[55]  log_lik[56]
##    1.0010789    1.0003111    0.9998254    0.9998452    0.9997079    1.0010252
```

```
##  log_lik[57]  log_lik[58]  log_lik[59]  log_lik[60]  log_lik[61]  log_lik[62]
##    1.0008959    0.9999512    1.0000928    1.0018656    1.0001633    0.9995415
##  log_lik[63]  log_lik[64]  log_lik[65]  log_lik[66]  log_lik[67]  log_lik[68]
##    1.0005576    1.0011423    1.0019743    1.0001459    0.9998221    1.0011431
##  log_lik[69]  log_lik[70]  log_lik[71]  log_lik[72]  log_lik[73]  log_lik[74]
##    1.0015854    1.0017617    1.0011354    1.0001447    1.0003518    1.0005048
##  log_lik[75]  log_lik[76]  log_lik[77]  log_lik[78]  log_lik[79]  log_lik[80]
##    1.0016592    1.0004800    1.0000809    1.0010379    0.9995533    1.0016878
##  log_lik[81]  log_lik[82]  log_lik[83]  log_lik[84]  log_lik[85]  log_lik[86]
##    1.0011682    1.0000788    1.0001996    1.0002829    1.0005704    1.0017889
##  log_lik[87]  log_lik[88]  log_lik[89]  log_lik[90]  log_lik[91]  log_lik[92]
##    1.0005576    1.0006471    1.0011678    1.0006120    0.9992871    1.0010111
##  log_lik[93]  log_lik[94]  log_lik[95]  log_lik[96]  log_lik[97]  log_lik[98]
##    1.0021012    1.0010997    1.0001642    1.0016067    1.0013879    1.0002197
##  log_lik[99] log_lik[100] log_lik[101] log_lik[102] log_lik[103] log_lik[104]
##    1.0009625    0.9994963    1.0011406    0.9996362    1.0013255    1.0003350
## log_lik[105] log_lik[106] log_lik[107] log_lik[108] log_lik[109] log_lik[110]
##    1.0000728    1.0021012    1.0021686    1.0007648    0.9994792    1.0004229
## log_lik[111] log_lik[112] log_lik[113] log_lik[114] log_lik[115] log_lik[116]
##    0.9999049    1.0004825    0.9999100    1.0005119    1.0009581    1.0008931
## log_lik[117] log_lik[118] log_lik[119] log_lik[120] log_lik[121] log_lik[122]
##    1.0011650    1.0003874    1.0007120    1.0007995    1.0013156    0.9994752
## log_lik[123] log_lik[124] log_lik[125] log_lik[126] log_lik[127] log_lik[128]
##    0.9999138    1.0002782    0.9996427    1.0017867    0.9995632    1.0014722
## log_lik[129] log_lik[130] log_lik[131] log_lik[132] log_lik[133] log_lik[134]
##    1.0000934    0.9996954    1.0018057    0.9998584    1.0012455    1.0010890
## log_lik[135] log_lik[136] log_lik[137] log_lik[138] log_lik[139] log_lik[140]
##    1.0008033    1.0000953    1.0007120    1.0018667    0.9998664    1.0003791
## log_lik[141] log_lik[142] log_lik[143] log_lik[144] log_lik[145] log_lik[146]
##    1.0002011    1.0007730    1.0003618    0.9999365    1.0022059    1.0003770
## log_lik[147] log_lik[148] log_lik[149] log_lik[150] log_lik[151] log_lik[152]
##    1.0003262    1.0008957    1.0019259    0.9999355    1.0002999    1.0010897
## log_lik[153] log_lik[154] log_lik[155] log_lik[156] log_lik[157] log_lik[158]
##    1.0018964    1.0006071    0.9996951    1.0011525    1.0000814    1.0002828
## log_lik[159] log_lik[160] log_lik[161] log_lik[162] log_lik[163] log_lik[164]
##    0.9998882    1.0015442    0.9994211    1.0007676    0.9995754    1.0010697
## log_lik[165] log_lik[166] log_lik[167] log_lik[168] log_lik[169] log_lik[170]
##    1.0004040    1.0004756    1.0018057    1.0004294    0.9993414    0.9998072
## log_lik[171] log_lik[172] log_lik[173] log_lik[174] log_lik[175] log_lik[176]
##    0.9998856    1.0013838    1.0002197    1.0003989    0.9997663    1.0013897
## log_lik[177] log_lik[178] log_lik[179] log_lik[180] log_lik[181] log_lik[182]
##    0.9994173    1.0021639    1.0010884    1.0007446    1.0003804    1.0005704
## log_lik[183] log_lik[184] log_lik[185] log_lik[186] log_lik[187] log_lik[188]
##    1.0014301    1.0006539    1.0007007    1.0013014    1.0020313    1.0006057
## log_lik[189] log_lik[190] log_lik[191] log_lik[192] log_lik[193] log_lik[194]
##    1.0000976    1.0015117    1.0018789    1.0009192    1.0011347    1.0012397
## log_lik[195] log_lik[196] log_lik[197] log_lik[198] log_lik[199] log_lik[200]
##    0.9996448    1.0004115    1.0009436    1.0002191    1.0003842    1.0001173
## log_lik[201] log_lik[202] log_lik[203] log_lik[204] log_lik[205] log_lik[206]
##    0.9999278    1.0021905    1.0001299    1.0002233    0.9995935    1.0020096
## log_lik[207] log_lik[208] log_lik[209] log_lik[210] log_lik[211] log_lik[212]
##    1.0011027    1.0001967    1.0003006    1.0007389    1.0013019    1.0020329
## log_lik[213] log_lik[214] log_lik[215] log_lik[216] log_lik[217] log_lik[218]
##    0.9998143    0.9995496    0.9996951    1.0010754    0.9995630    0.9996780
```

```
## log_lik[219] log_lik[220] log_lik[221] log_lik[222] log_lik[223] log_lik[224]
##    1.0002936    1.0003291    1.0021793    1.0003706    1.0006182    1.0003122
## log_lik[225] log_lik[226] log_lik[227] log_lik[228] log_lik[229] log_lik[230]
##    1.0005992    0.9999364    1.0008305    1.0005347    1.0002098    1.0015236
## log_lik[231] log_lik[232] log_lik[233] log_lik[234] log_lik[235] log_lik[236]
##    1.0008312    1.0003862    1.0006876    1.0006867    1.0021684    0.9994887
## log_lik[237] log_lik[238] log_lik[239] log_lik[240] log_lik[241] log_lik[242]
##    0.9994794    1.0012996    0.9993414    1.0022267    1.0011051    1.0015497
## log_lik[243] log_lik[244] log_lik[245] log_lik[246] log_lik[247] log_lik[248]
##    1.0005865    0.9995442    0.9995573    1.0021431    1.0003322    1.0010939
## log_lik[249] log_lik[250] log_lik[251] log_lik[252] log_lik[253] log_lik[254]
##    1.0006475    1.0017103    1.0003383    1.0003993    1.0020096    1.0007165
## log_lik[255] log_lik[256] log_lik[257] log_lik[258] log_lik[259] log_lik[260]
##    1.0020027    1.0003079    1.0012630    1.0005261    1.0003039    1.0016689
## log_lik[261] log_lik[262] log_lik[263] log_lik[264] log_lik[265] log_lik[266]
##    1.0014288    0.9997233    1.0018165    1.0003983    0.9997861    1.0007504
## log_lik[267] log_lik[268] log_lik[269] log_lik[270] log_lik[271] log_lik[272]
##    1.0003803    0.9995308    1.0010780    1.0006475    0.9997445    1.0014058
## log_lik[273] log_lik[274] log_lik[275] log_lik[276] log_lik[277] log_lik[278]
##    0.9999053    1.0011283    1.0004080    1.0010523    1.0003862    0.9998893
## log_lik[279] log_lik[280] log_lik[281] log_lik[282] log_lik[283] log_lik[284]
##    1.0003770    1.0011628    0.9994753    1.0006004    0.9995371    0.9994997
## log_lik[285] log_lik[286] log_lik[287] log_lik[288] log_lik[289] log_lik[290]
##    1.0012788    1.0018734    1.0001902    1.0021622    1.0001255    1.0003109
## log_lik[291] log_lik[292] log_lik[293] log_lik[294] log_lik[295] log_lik[296]
##    1.0020958    1.0007534    1.0012107    0.9997688    0.9995855    1.0013310
## log_lik[297] log_lik[298] log_lik[299]         lp__
##    1.0021455    1.0006184    1.0011337    1.0014349
```

```
print("\nCheck divergences: ")
```

```
## [1] "\nCheck divergences: "
```

```
check_divergences(fit_selected_1)
```

```
## 0 of 4000 iterations ended with a divergence.
```

```
print("Check treedepth:")
```

```
## [1] "Check treedepth:"
```

```
check_treedepth(fit_selected_1)
```

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

```
print("\nn_eff values: ")
```

```
## [1] "\nn_eff values: "
```

```r
loo(fit_selected_1)$diagnostics$n_eff
```

```
##   [1] 3246.472 3837.039 3113.501 3947.044 3618.234 3900.006 2900.058 3374.685
##   [9] 3323.800 3180.236 3537.884 2988.283 4391.882 3819.664 4374.082 3990.523
##  [17] 3072.316 3986.320 2756.915 2636.801 2992.433 3318.914 2741.841 3407.574
##  [25] 3366.227 3832.924 3151.549 4089.394 3271.762 2921.431 3647.513 3884.478
##  [33] 4138.964 4482.425 3712.417 3502.942 3584.594 3701.803 3378.729 3198.875
##  [41] 3308.582 4473.451 3905.076 3907.437 3374.685 2938.191 4010.817 3138.995
##  [49] 3509.460 3559.490 2783.851 3962.463 3115.354 3880.813 3090.346 3419.002
##  [57] 3383.166 3250.915 4121.974 2948.229 4372.945 2829.673 4162.785 4041.608
##  [65] 3123.254 3637.132 4343.291 2745.518 2835.719 3487.473 3741.792 3323.890
##  [73] 2961.841 3909.371 2927.231 3090.758 3885.606 3531.552 2862.573 3504.758
##  [81] 4083.304 3970.888 3589.440 4394.940 3268.462 3462.060 4162.785 3732.236
##  [89] 3773.942 3539.455 3419.074 3567.579 3057.077 3676.569 3312.580 3517.608
##  [97] 3107.726 3905.978 3281.845 3211.087 2875.179 3722.791 2667.282 4383.173
## [105] 2911.245 3274.615 3120.033 4099.116 2915.948 3750.524 4074.932 4423.530
## [113] 4108.943 3817.434 3485.407 3498.046 3675.708 3252.024 3736.887 3752.184
## [121] 3283.793 2903.648 3166.186 3079.896 3245.496 3076.888 3883.278 3579.220
## [129] 2959.189 2754.148 3442.423 1064.916 3548.306 3562.500 2991.034 3263.474
## [137] 3736.887 3381.145 2863.182 4112.073 3125.309 4107.632 4180.177 3226.498
## [145] 3204.356 3992.933 3813.641 3703.097 3366.227 3070.066 3683.695 3663.034
## [153] 3454.779 4211.386 2800.700 3633.153 3819.859 4215.134 3857.241 3557.285
## [161] 3299.646 4079.550 3129.917 4056.188 4391.882 3277.459 3442.423 3650.980
## [169] 3112.508 2750.245 3587.653 3700.935 3905.978 3855.370 2773.243 3576.884
## [177] 3103.336 3264.937 3608.883 3565.155 4354.414 3268.462 2954.569 3778.022
## [185] 3515.206 3592.748 3333.091 3617.107 3278.254 3444.538 3553.771 3504.025
## [193] 3730.143 2757.558 4171.583 4424.341 3723.605 2768.164 3970.478 3325.632
## [201] 3445.841 3157.830 3922.807 2867.671 3000.501 3383.925 3737.110 2969.177
## [209] 3329.436 4035.562 2732.924 3323.978 4203.081 3868.348 2800.700 3009.395
## [217] 3815.520 2989.976 2735.516 3534.782 3243.921 3827.826 4014.360 4173.894
## [225] 3356.812 4152.198 3584.908 3925.830 1796.305 2758.346 3659.085 2817.174
## [233] 4030.645 3375.675 3317.057 3916.891 3962.788 2751.499 3112.508 3266.875
## [241] 2949.736 2881.813 3932.974 3063.463 2894.045 3434.600 3786.642 3443.508
## [249] 4002.066 3524.826 3794.350 3620.959 3383.925 2866.555 3327.359 4178.202
## [257] 3741.661 3456.195 4244.010 3495.519 3643.059 3159.048 3139.538 3824.645
## [265] 3049.106 4087.515 3806.711 2898.984 3866.249 4002.066 3757.679 3711.775
## [273] 2995.190 3772.255 3135.024 3881.607 2817.174 2987.310 3992.933 3759.543
## [281] 3104.620 3365.977 2437.958 2883.427 3731.725 3522.913 3415.738 3181.308
## [289] 2711.762 2871.151 3276.540 3809.047 3863.696 2835.592 2968.248 3862.496
## [297] 3237.538 3382.034 2786.621
```

```r
print("\nPerformance of k_hat values and elpd_loo")
```

```
## [1] "\nPerformance of k_hat values and elpd_loo"
```

```r
loo(fit_selected_1)
```

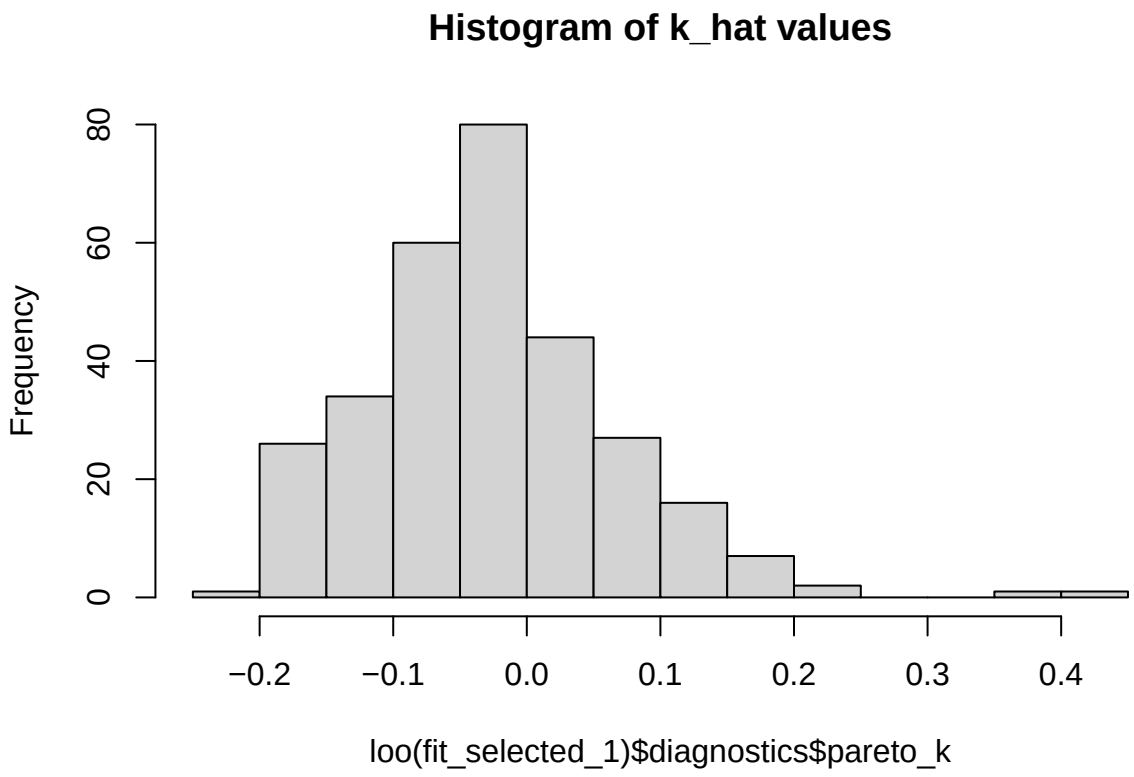
```
##
## Computed from 4000 by 299 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -178.9  8.5
```

```
## p_loo         4.3  1.1
## looic       357.9 16.9
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
print("\nDistribution of k_hat")
```

```
## [1] "\nDistribution of k_hat"
```

```
hist(loo(fit_selected_1)$diagnostics$pareto_k, main = "Histogram of k_hat values")
```




```
print("\nRhat values: ")
```

```
## [1] "\nRhat values: "
```

```
rhat(fit_selected_2)
```

```
##      beta[1]      beta[2]      beta[3]      beta[4]    Y_pred[1]    Y_pred[2]
##    1.0004256    0.9995062    1.0003368    0.9992510    0.9998677    0.9999967
##    Y_pred[3]    Y_pred[4]    Y_pred[5]    Y_pred[6]    Y_pred[7]    Y_pred[8]
```

```
##    0.9998275    1.0001205    0.9996663    0.9997617    0.9997375    1.0006390
##    Y_pred[9]   Y_pred[10]   Y_pred[11]   Y_pred[12]   Y_pred[13]   Y_pred[14]
##    0.9994803    0.9997737    0.9998164    0.9993077    0.9994403    1.0001966
##   Y_pred[15]   Y_pred[16]   Y_pred[17]   Y_pred[18]   Y_pred[19]   Y_pred[20]
##    1.0000037    0.9998137    1.0001196    0.9995536    0.9993802    0.9997441
##   Y_pred[21]   Y_pred[22]   Y_pred[23]   Y_pred[24]   Y_pred[25]   Y_pred[26]
##    0.9998138    0.9994293    1.0008149    1.0008419    1.0000987    0.9995942
##   Y_pred[27]   Y_pred[28]   Y_pred[29]   Y_pred[30]   Y_pred[31]   Y_pred[32]
##    1.0004047    0.9997865    1.0000793    0.9998328    0.9995994    0.9994697
##   Y_pred[33]   Y_pred[34]   Y_pred[35]   Y_pred[36]   Y_pred[37]   Y_pred[38]
##    0.9999544    1.0009794    0.9998707    0.9997933    0.9996436    0.9994388
##   Y_pred[39]   Y_pred[40]   Y_pred[41]   Y_pred[42]   Y_pred[43]   Y_pred[44]
##    1.0002232    0.9996696    0.9999893    1.0000303    0.9996062    1.0002828
##   Y_pred[45]   Y_pred[46]   Y_pred[47]   Y_pred[48]   Y_pred[49]   Y_pred[50]
##    1.0004098    1.0002527    0.9997939    0.9995783    0.9995967    0.9998132
##   Y_pred[51]   Y_pred[52]   Y_pred[53]   Y_pred[54]   Y_pred[55]   log_lik[1]
##    1.0003137    0.9998504    1.0000445    1.0001215    0.9997896    0.9995967
##   log_lik[2]   log_lik[3]   log_lik[4]   log_lik[5]   log_lik[6]   log_lik[7]
##    0.9995800    0.9993313    1.0001322    0.9995083    0.9998937    1.0000287
##   log_lik[8]   log_lik[9]  log_lik[10]  log_lik[11]  log_lik[12]  log_lik[13]
##    0.9995634    0.9994406    1.0011422    1.0004452    0.9996748    0.9998462
##  log_lik[14]  log_lik[15]  log_lik[16]  log_lik[17]  log_lik[18]  log_lik[19]
##    0.9998199    0.9996669    1.0006171    1.0010029    0.9992069    1.0000414
##  log_lik[20]  log_lik[21]  log_lik[22]  log_lik[23]  log_lik[24]  log_lik[25]
##    0.9994103    0.9995438    0.9996894    1.0003872    1.0000209    0.9997643
##  log_lik[26]  log_lik[27]  log_lik[28]  log_lik[29]  log_lik[30]  log_lik[31]
##    0.9998340    1.0010405    1.0004676    1.0006822    1.0000189    1.0002218
##  log_lik[32]  log_lik[33]  log_lik[34]  log_lik[35]  log_lik[36]  log_lik[37]
##    0.9998775    0.9992685    0.9998613    0.9998901    1.0005276    1.0008814
##  log_lik[38]  log_lik[39]  log_lik[40]  log_lik[41]  log_lik[42]  log_lik[43]
##    1.0002797    1.0003688    1.0001467    0.9997800    0.9997283    1.0004113
##  log_lik[44]  log_lik[45]  log_lik[46]  log_lik[47]  log_lik[48]  log_lik[49]
##    1.0002184    0.9994183    1.0007332    0.9992021    1.0002060    1.0010165
##  log_lik[50]  log_lik[51]  log_lik[52]  log_lik[53]  log_lik[54]  log_lik[55]
##    0.9994326    0.9999692    0.9996868    1.0003505    1.0003416    0.9999563
##  log_lik[56]  log_lik[57]  log_lik[58]  log_lik[59]  log_lik[60]  log_lik[61]
##    1.0000332    0.9998159    1.0003665    0.9995055    0.9998830    0.9996440
##  log_lik[62]  log_lik[63]  log_lik[64]  log_lik[65]  log_lik[66]  log_lik[67]
##    1.0001645    0.9994601    0.9994671    1.0002776    0.9999321    0.9997167
##  log_lik[68]  log_lik[69]  log_lik[70]  log_lik[71]  log_lik[72]  log_lik[73]
##    1.0001344    0.9998716    0.9997924    0.9997989    0.9996644    1.0003528
##  log_lik[74]  log_lik[75]  log_lik[76]  log_lik[77]  log_lik[78]  log_lik[79]
##    0.9997224    0.9996768    0.9996297    0.9999013    0.9999610    1.0002477
##  log_lik[80]  log_lik[81]  log_lik[82]  log_lik[83]  log_lik[84]  log_lik[85]
##    0.9996135    0.9996981    0.9999768    1.0001642    0.9996949    0.9995096
##  log_lik[86]  log_lik[87]  log_lik[88]  log_lik[89]  log_lik[90]  log_lik[91]
##    0.9998048    0.9995696    0.9998531    0.9998082    0.9993917    1.0007088
##  log_lik[92]  log_lik[93]  log_lik[94]  log_lik[95]  log_lik[96]  log_lik[97]
##    0.9997206    1.0001023    0.9997542    0.9996097    0.9996435    0.9993703
##  log_lik[98]  log_lik[99] log_lik[100] log_lik[101] log_lik[102] log_lik[103]
##    0.9998658    0.9994221    0.9996800    0.9995328    1.0005296    0.9997734
## log_lik[104] log_lik[105] log_lik[106] log_lik[107] log_lik[108] log_lik[109]
##    0.9998589    0.9995335    0.9999290    0.9997583    0.9998329    0.9996294
## log_lik[110] log_lik[111] log_lik[112] log_lik[113] log_lik[114] log_lik[115]
```

```
##    0.9998270    1.0003012    0.9995492    1.0003134    0.9997429    0.9993110
## log_lik[116] log_lik[117] log_lik[118] log_lik[119] log_lik[120] log_lik[121]
##    0.9996625    1.0000336    0.9993817    0.9997837    1.0000494    0.9996866
## log_lik[122] log_lik[123] log_lik[124] log_lik[125] log_lik[126] log_lik[127]
##    1.0007606    0.9999205    0.9993778    1.0004785    1.0001886    0.9994330
## log_lik[128] log_lik[129] log_lik[130] log_lik[131] log_lik[132] log_lik[133]
##    0.9999644    0.9998602    1.0002004    1.0000085    1.0002950    1.0001284
## log_lik[134] log_lik[135] log_lik[136] log_lik[137] log_lik[138] log_lik[139]
##    0.9993291    1.0004847    1.0006659    0.9999263    0.9994461    1.0001131
## log_lik[140] log_lik[141] log_lik[142] log_lik[143] log_lik[144] log_lik[145]
##    0.9995457    1.0005803    0.9996667    0.9994616    0.9996578    0.9999322
## log_lik[146] log_lik[147] log_lik[148] log_lik[149] log_lik[150] log_lik[151]
##    0.9994800    0.9995442    0.9998544    0.9997643    0.9999658    0.9997769
## log_lik[152] log_lik[153] log_lik[154] log_lik[155] log_lik[156] log_lik[157]
##    0.9998262    0.9998665    1.0000049    1.0004540    0.9997034    1.0004857
## log_lik[158] log_lik[159] log_lik[160] log_lik[161] log_lik[162] log_lik[163]
##    0.9995363    1.0002569    0.9995882    1.0000149    0.9997304    1.0006854
## log_lik[164] log_lik[165] log_lik[166] log_lik[167] log_lik[168] log_lik[169]
##    0.9994079    0.9998242    1.0005128    1.0000259    1.0002079    1.0006920
## log_lik[170] log_lik[171] log_lik[172] log_lik[173] log_lik[174] log_lik[175]
##    1.0002292    1.0007635    0.9995495    0.9994025    0.9994429    1.0005046
## log_lik[176] log_lik[177] log_lik[178] log_lik[179] log_lik[180] log_lik[181]
##    0.9998099    1.0006557    0.9996708    0.9993555    0.9997542    1.0000011
## log_lik[182] log_lik[183] log_lik[184] log_lik[185] log_lik[186] log_lik[187]
##    0.9994553    0.9996622    0.9999143    0.9994865    0.9996205    0.9996569
## log_lik[188] log_lik[189] log_lik[190] log_lik[191] log_lik[192] log_lik[193]
##    0.9997056    1.0002961    0.9998506    0.9996703    0.9996294    0.9996090
## log_lik[194] log_lik[195] log_lik[196] log_lik[197] log_lik[198] log_lik[199]
##    0.9997541    0.9999364    0.9996148    1.0001847    1.0000471    0.9994514
## log_lik[200] log_lik[201] log_lik[202] log_lik[203] log_lik[204] log_lik[205]
##    0.9992676    1.0005090    1.0000943    0.9999483    1.0003955    1.0008057
## log_lik[206] log_lik[207] log_lik[208] log_lik[209] log_lik[210] log_lik[211]
##    0.9999027    0.9997463    1.0006713    0.9994792    0.9995246    0.9996469
## log_lik[212] log_lik[213] log_lik[214] log_lik[215] log_lik[216] log_lik[217]
##    0.9998985    1.0000114    0.9995748    1.0001342    1.0001519    0.9999299
## log_lik[218] log_lik[219] log_lik[220] log_lik[221] log_lik[222] log_lik[223]
##    1.0003045    1.0001141    0.9996409    0.9998229    0.9993044    1.0000331
## log_lik[224] log_lik[225] log_lik[226] log_lik[227] log_lik[228] log_lik[229]
##    0.9995321    0.9994205    0.9995001    0.9992871    0.9994239    1.0002684
## log_lik[230] log_lik[231] log_lik[232] log_lik[233] log_lik[234] log_lik[235]
##    0.9997792    0.9992595    1.0002244    1.0000391    1.0004504    0.9998020
## log_lik[236] log_lik[237] log_lik[238] log_lik[239] log_lik[240] log_lik[241]
##    1.0003522    0.9993393    0.9996784    1.0007689    0.9997465    1.0000440
## log_lik[242] log_lik[243] log_lik[244] log_lik[245] log_lik[246] log_lik[247]
##    0.9996428    1.0002597    1.0002213    1.0006124    0.9993010    0.9994722
## log_lik[248] log_lik[249] log_lik[250] log_lik[251] log_lik[252] log_lik[253]
##    1.0000083    0.9997625    0.9994933    0.9996107    0.9994038    0.9998170
## log_lik[254] log_lik[255] log_lik[256] log_lik[257] log_lik[258] log_lik[259]
##    0.9997925    0.9998104    0.9994318    0.9994376    0.9996219    0.9996266
## log_lik[260] log_lik[261] log_lik[262] log_lik[263] log_lik[264] log_lik[265]
##    1.0000092    0.9996941    0.9999845    0.9992721    0.9997418    0.9993140
## log_lik[266] log_lik[267] log_lik[268] log_lik[269] log_lik[270] log_lik[271]
##    0.9995419    0.9998461    0.9999419    0.9997072    1.0000479    1.0001704
## log_lik[272] log_lik[273] log_lik[274] log_lik[275] log_lik[276] log_lik[277]
```

```
##    0.9997409    0.9998202    0.9997729    0.9995662    0.9998773    0.9997457
## log_lik[278] log_lik[279] log_lik[280] log_lik[281] log_lik[282] log_lik[283]
##    1.0001535    0.9995635    1.0000351    1.0008982    0.9997658    1.0007186
## log_lik[284] log_lik[285] log_lik[286] log_lik[287] log_lik[288] log_lik[289]
##    1.0005894    0.9995416    0.9997209    0.9997028    1.0000589    0.9995211
## log_lik[290] log_lik[291] log_lik[292] log_lik[293] log_lik[294] log_lik[295]
##    1.0000925    0.9997887    0.9994205    0.9995017    1.0002797    0.9993384
## log_lik[296] log_lik[297] log_lik[298] log_lik[299]         lp__
##    0.9994963    1.0000617    1.0001158    1.0003203    1.0003950
```

```
print("\nCheck divergences: ")
```

```
## [1] "\nCheck divergences: "
```

```
check_divergences(fit_selected_2)
```

```
## 0 of 4000 iterations ended with a divergence.
```

```
print("Check treedepth:")
```

```
## [1] "Check treedepth:"
```

```
check_treedepth(fit_selected_2)
```

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

```
print("\nn_eff values: ")
```

```
## [1] "\nn_eff values: "
```

```
loo(fit_selected_2)$diagnostics$n_eff
```

```
##   [1] 4329.930 4531.474 4375.734 4503.795 4387.360 4130.883 3717.530 3684.386
##   [9] 4250.532 3180.558 4687.714 3824.042 4398.532 4286.602 4706.621 4266.570
##  [17] 3621.793 4791.617 3573.928 3480.321 3787.383 4101.966 3827.642 4024.900
##  [25] 4175.564 4169.668 3691.237 4470.543 4403.218 3778.112 3891.409 4606.191
##  [33] 4258.724 4442.879 5019.780 4821.792 4070.509 4508.155 4216.337 4094.240
##  [41] 4257.919 4707.911 4911.771 4147.192 4366.587 3700.600 4742.299 4520.817
##  [49] 4214.385 4232.749 3603.069 4229.583 4106.540 4089.769 4609.771 3855.738
##  [57] 4265.675 4657.909 4818.623 3780.360 4448.710 4017.138 4677.727 4034.212
##  [65] 3710.916 5026.542 4961.694 3554.234 3658.193 4391.328 4294.100 4564.940
##  [73] 3785.420 4671.461 3869.680 3961.586 4958.621 3940.002 3702.720 4820.975
##  [81] 4611.170 4706.527 5060.442 4306.933 3961.454 4542.849 4963.166 4833.287
##  [89] 4445.130 4012.642 4151.097 3843.116 3808.791 4618.765 4747.584 4580.698
##  [97] 4247.733 4535.241 4139.032 4788.330 3668.933 4195.089 3418.751 4340.449
## [105] 4068.762 3824.556 3825.765 4539.877 3938.533 4145.360 4213.638 4893.250
## [113] 4775.673 4577.920 4024.847 4972.341 4233.569 4171.843 4830.614 4040.983
## [121] 4056.085 3805.312 4681.269 4133.256 4698.717 3681.022 5046.743 4276.179
## [129] 4128.583 3303.129 4272.535 1444.480 3945.317 4746.128 3453.159 3758.038
```

```
## [137] 4470.920 4550.332 4034.060 4910.386 3778.127 4555.767 4823.782 4839.992
## [145] 3784.964 4643.936 4945.623 4444.917 4175.564 4327.504 4882.741 4608.648
## [153] 3969.486 4352.117 3857.108 3863.541 4218.369 4833.578 4096.335 4813.685
## [161] 3920.578 4611.099 4328.652 4511.179 4308.274 3783.268 3941.381 4742.732
## [169] 4342.621 3749.955 4075.439 4730.670 5093.983 4537.385 3780.400 4406.128
## [177] 3866.912 4252.165 4802.740 4919.391 4329.381 4323.457 3892.621 4776.014
## [185] 4365.588 4420.252 3812.973 3820.522 4155.212 4090.722 4134.502 4246.569
## [193] 4124.168 3627.389 4950.358 4538.119 4006.739 3712.385 5013.404 2748.975
## [201] 4423.014 3885.170 4740.358 3744.333 3671.514 3970.910 4277.224 3523.757
## [209] 4868.746 4372.422 3561.617 4280.735 4656.064 4865.128 3874.914 3769.494
## [217] 5160.488 3873.215 3798.296 4744.684 4161.084 4963.485 4125.419 4911.282
## [225] 4178.613 3800.014 4707.814 4648.704 2354.211 3617.261 4756.621 3779.667
## [233] 4350.123 3766.396 4020.078 4563.079 4838.753 3901.491 3987.184 3992.228
## [241] 3660.407 3798.040 4134.749 4095.544 4036.026 4202.097 4464.375 4383.180
## [249] 4147.530 4135.041 4827.971 4578.312 3893.689 3924.778 4242.499 4408.307
## [257] 4221.991 4839.930 4827.795 4041.631 4720.943 3921.707 4449.549 4938.361
## [265] 4538.063 4704.872 4893.002 3869.797 4425.948 4256.323 4572.502 4247.284
## [273] 3939.025 4391.555 4132.420 4435.592 3925.172 3970.513 5003.155 4098.754
## [281] 3814.154 4602.413 3249.898 3997.556 4707.277 4020.661 3868.933 3778.077
## [289] 4099.358 3574.692 4221.729 4062.740 4665.705 3969.066 4513.425 4550.306
## [297] 4038.782 3949.537 3467.867
```

```
print("\nPerformance of k_hat values and elpd_loo")
```

```
## [1] "\nPerformance of k_hat values and elpd_loo"
```

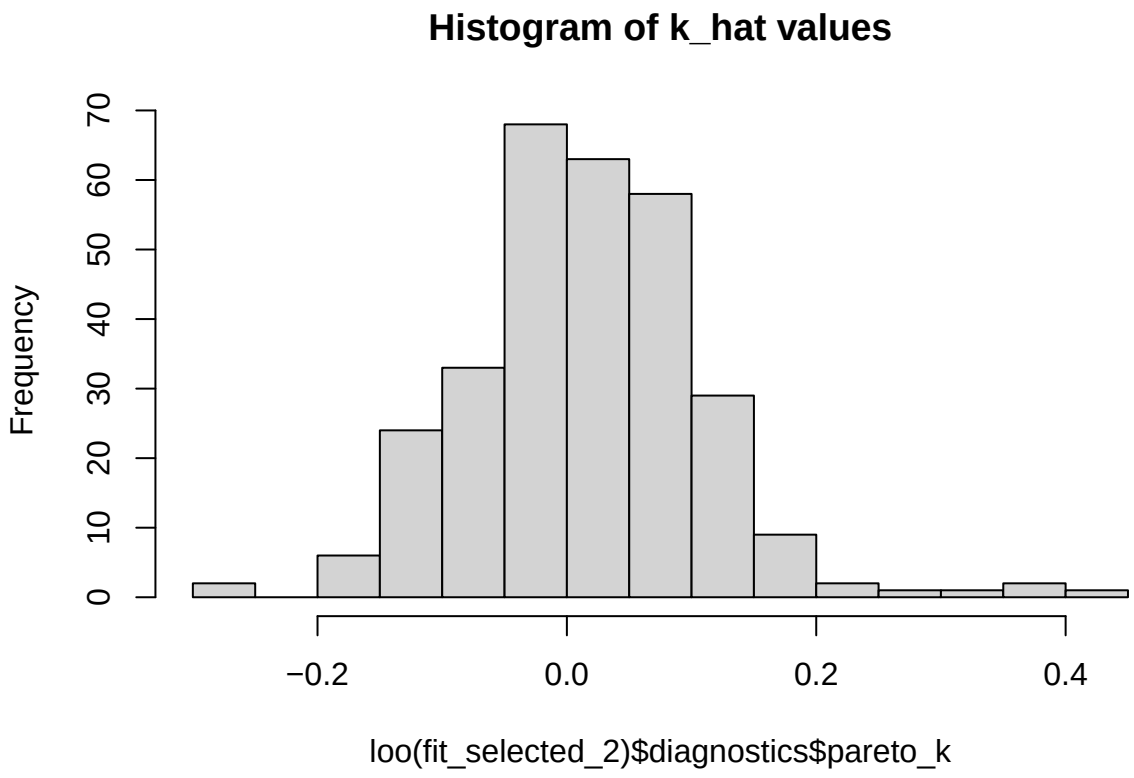
```
loo(fit_selected_2)
```

```
##
## Computed from 4000 by 299 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -179.0  8.7
## p_loo         5.6  1.1
## looic       358.0 17.4
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
print("\nDistribution of k_hat")
```

```
## [1] "\nDistribution of k_hat"
```

```
hist(loo(fit_selected_2)$diagnostics$pareto_k, main = "Histogram of k_hat values")
```




```
confusionMatrix(table(Y_pred_selected_1, Y_true))
```

```
## Confusion Matrix and Statistics
##
##                  Y_true
## Y_pred_selected_1  0  1
##                 0 25  7
##                 1 11 12
##
##               Accuracy : 0.6727
##                 95% CI : (0.5329, 0.7932)
##    No Information Rate : 0.6545
##    P-Value [Acc > NIR] : 0.4494
##
##                  Kappa : 0.3106
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 0.6944
##             Specificity : 0.6316
##          Pos Pred Value : 0.7812
##          Neg Pred Value : 0.5217
##              Prevalence : 0.6545
##          Detection Rate : 0.4545
##    Detection Prevalence : 0.5818
```

```
##       Balanced Accuracy : 0.6630
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(table(Y_pred_selected_2, Y_true))
```

```
## Confusion Matrix and Statistics
##
##                  Y_true
## Y_pred_selected_2  0  1
##                 0 26  7
##                 1 10 12
##
##                Accuracy : 0.6909
##                  95% CI : (0.5519, 0.8086)
##     No Information Rate : 0.6545
##     P-Value [Acc > NIR] : 0.3399
##
##                   Kappa : 0.3411
##
##  Mcnemar's Test P-Value : 0.6276
##
##             Sensitivity : 0.7222
##             Specificity : 0.6316
##          Pos Pred Value : 0.7879
##          Neg Pred Value : 0.5455
##              Prevalence : 0.6545
##          Detection Rate : 0.4727
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.6769
##
##        'Positive' Class : 0
##
```

```
cat("\nSelected Bayesian Inference PSIS_LOO:" ,loo(fit_selected_1)$estimates[1], "\n");
```

```
##
## Selected Bayesian Inference PSIS_LOO: -178.9285
```

```
cat("\nSelected from correlation matrix PSIS_LOO:" , loo(fit_selected_2)$estimates[1], "\n");
```

```
##
## Selected from correlation matrix PSIS_LOO: -179.0149
```

```
loo_compare(loo(fit_selected_1), loo(fit_selected_2))
```

```
##        elpd_diff se_diff
## model1  0.0       0.0
## model2 -0.1       1.9
```

From the result above, beside the fact that convergence seems to not be affected by the normalization, we can see that both the prediction accuracy and the elpd-loo value dropped by a lot despite of our previous beliefs and argument. However, we believe that normalization is still necessary for this dataset but it has to be done alongside with choosing better priors.

12. Conclusion

Although there are several factors or behavior that can consider to be the cause of hear failure and cardiovascular diseases, according to the analysis we made above, age, ejection contraction, level of creatinine in blood and level of sodium in heart are considered to be a more "direct" cause to heart failure. This conclusion is based on the fact that the second combination of features performs better than the rest, showed by elpd-loo value, prediction accuracy and so on. However, this result does not render other factors irrelevant. To our belief, other factors can be considered the cause for these 4 "direct" factors that ultimately causes heart failure. However, there are plenty of room for improvement with this analysis with choice of prior, model and so on to ultimately enhance credibility and accuracy of the prediction. As discussed in introduction, CVD is a dangerous disease as it can often lead to heart failure and death which is something we all want to prevent. Therefore, being able to predict the outcome and act fast can save many lives.

13. Reflection

Thanks to the project, we have a better insight how models can be applied to reality. There were some difficulties while determining which models to apply, especially for hierarchical model. The data was not initially clustered, and it is also hard to cluster them in an appropriate way to fit into the hierarchical model. At the end of the day, we decided to omit it and focus on feature selection.
Also, at first we split the data by taking first 80% to be trainset and last 20% to be the testset. This results in a testset with only 2 rows death = 1 over total 60 rows. We have to find another way to shuffle the dataset and split it again to create a better test set.
Another thing about this project is that we have to try choosing prior by ourselves. It was not very easy and although our prior choice is not necessarily the best, we still learnt a lot from it.
Finally, we realize that normalization is really important as explanatory variables whose values are very close to zero (in non-normalized data) tend to have regression coefficients with a large face value even though the outcome is not as positive as we have speculated.

14. Full R code

```
library(loo)
library(rstan)
library(aaltobda)
library(brms)
library(corrplot)
library(rstanarm)
library(bayesplot)
library(projpred)
library(rpart)
library(rpart.plot)
library(randomForest)
library(caret)
library(glmnet)
```

```r
set.seed(123)
smp_size = floor(0.75 * nrow(data))

data = read.csv(file = "heart_failure_clinical_records_dataset.csv")
data = subset (data, select = -time)
names(data) = c("age", "anaemia", "cpk", "diabetes", "ejection", "pressure", "platelets", "creatinine",

age = data$age #mean = 60.8, std = 11.9
anaemia = data$anaemia #boolean - decrease of red blood cells
cpk = data$cpk #mean = 582, std = 969 - Level of the CPK enzyme in the blood
diabetes = data$diabetes #boolean - has diabetes
ejection = data$ejection #mean = 38.1%, std = 11.8% Percentage of blood leaving the heart at each contr
pressure = data$pressure #boolean If the patient has hypertension
platelets = data$platelets #mean = 283k, std = 97.6k If the patient has hypertension
creatinine = data$creatinine #mean = 1.39, std = 1.03 Level of serum creatinine in the blood
sodium = data$sodium #mean = 137, std = 4.41, Level of serum sodium in the blood
sex = data$sex #binary
smoking = data$smoking #boolean
time = data$time #mean = 130, std = 77.5 Follow-up period
death = data$death #boolean (Response variable)

smp_size = floor(0.75 * nrow(data))
train_ind = sample(seq_len(nrow(data)), size = smp_size)
data = data[train_ind, ]
test = data[-train_ind, ]


full_data = list(N = nrow(data), M = ncol(data), Y = data$death, X = data,
                 N_test = nrow(test), Y_true = test$death, X_test = test, mu = colMeans(data), sigma = c

X = subset(data, select = c(age, ejection, creatinine, sodium))
N = nrow(X)
M = ncol(X)
Y = data$death
mu = c(60.8, 0.38, 1.39, 137)
sigma_selected = cov(subset(data, select = c(age, ejection, creatinine, sodium)))

X_test = subset(test, select = c(age, ejection, creatinine, sodium))
N_test = nrow(X_test)
Y_true = test$death

selected_data_2 = list(N = N, M = M, Y = Y, X = X,
                       N_test = N_test, Y_true = Y_true, X_test = X_test, mu = mu, sigma = sigma_selecte

X = subset(data, select = c(age, ejection, creatinine))
N = nrow(X)
M = ncol(X)
Y = data$death
mu = c(60.8, 0.38, 1.39)
sigma_full = cov(data)
sigma_selected = cov(subset(data, select = c(age, ejection, creatinine)))

X_test = subset(test, select = c(age, ejection, creatinine))
```

```
N_test = nrow(X_test)
Y_true = test$death

selected_data_1 = list(N = N, M = M, Y = Y, X = X,
                       N_test = N_test, Y_true = Y_true, X_test = X_test, mu = mu, sigma = sigma_selecte

X = subset(data, select = c(age, ejection, creatinine, cpk, platelets, sodium))
N = nrow(X)
M = ncol(X)
Y = data$death
mu = c(60.8, 0.38, 1.39, 582, 283000, 137)
sigma_selected = cov(subset(data, select = c(age, ejection, creatinine, cpk, platelets, sodium)))

X_test = subset(test, select = c(age, ejection, creatinine, cpk, platelets, sodium))
N_test = nrow(X_test)
Y_true = test$death
selected_data_3 = list(N = N, M = M, Y = Y, X = X,
                       N_test = N_test, Y_true = Y_true, X_test = X_test, mu = mu, sigma = sigma_selecte

corrplot(cor(data))

barplot(table(death, age),
        main = "Death cases by Age",
        xlab = "Age", ylab = "Number of People",
        legend = rownames(table(death, age)))

barplot(table(death, ejection),
        main = "Death cases by ejection fraction",
        xlab = "Percentage", ylab = "Percentage",
        legend = rownames(table(death, ejection)))

barplot(table(death, creatinine),
        main = "Death cases by level of serum creatinine in blood",
        xlab = "Level of serum creatinine in blood", ylab = "Number of People",
        legend = rownames(table(death, creatinine)))

barplot(table(death, sodium),
        main = "Death cases by level of serum sodium in blood",
        xlab = "Level of sodium in blood", ylab = "Number of People",
        legend = rownames(table(death, sodium)))

barplot(table(death, sex),
        main = "Death in each sex",
        xlab = "sex", ylab = "Number of People",
        legend = rownames(table(death, sex)), beside = TRUE)

fit_rf = randomForest(factor(death)~., data=data)
varImpPlot(fit_rf)

fit = stan_glm(death~age+anaemia+cpk+diabetes+ejection+pressure+platelets+creatinine+sodium+sex+smoking
          data = data,
          family = binomial("logit"),
          prior = default_prior_coef(family),
```

```
          refresh=0,
          verbose = FALSE)
summary(fit)

fit_cv = cv_varsel(fit, method = 'forward', cv_method='LOO')

plot(fit_cv, stats = c('elpd', 'rmse'))

fit_cv$solution_terms[1:3]

x = as.matrix(data[,-12])
y = data$death
cv.lasso = cv.glmnet(x, y, family='binomial', alpha=1, parallel=TRUE, standardize=TRUE, type.measure='au
plot(cv.lasso)
cat('Min Lambda: ', cv.lasso$lambda.min, '\n 1Sd Lambda: ', cv.lasso$lambda.1se, '\n')
df_coef = round(as.matrix(coef(cv.lasso, s=cv.lasso$lambda.min)), 2)
df_coef[df_coef[, 1] != 0, ]

fit_selected_1 = stan(file = "LogitRegression.stan", data = selected_data_1, refresh = 0)
Y_pred_selected_1 = round(colMeans(extract(fit_selected_1)$Y_pred))

fit_selected_2 = stan(file = "LogitRegression.stan", data = selected_data_2, refresh = 0)
Y_pred_selected_2 = round(colMeans(extract(fit_selected_2)$Y_pred))

fit_selected_3 = stan(file = "LogitRegression.stan", data = selected_data_3, refresh = 0)
Y_pred_selected_3 = round(colMeans(extract(fit_selected_2)$Y_pred))

fit_full = stan(file = "LogitRegression.stan", data = full_data, refresh = 0)
Y_pred_full = round(colMeans(extract(fit_full)$Y_pred))

print("Rhat values: ")
rhat(fit_selected_1)

print("Check divergences: ")
check_divergences(fit_selected_1)

print("n_eff values: ")
loo(fit_selected_1)$diagnostics$n_eff

print("Performance of k_hat values and elpd_loo")
loo(fit_selected_1)

print("Distribution of k_hat")
hist(loo(fit_selected_1)$diagnostics$pareto_k, main = "Histogram of k_hat values")

print("Rhat values: ")
rhat(fit_selected_2)

print("Check divergences: ")
check_divergences(fit_selected_2)

print("n_eff values: ")
loo(fit_selected_2)$diagnostics$n_eff
```

```
print("Performance of k_hat values and elpd_loo")
loo(fit_selected_2)

print("Distribution of k_hat")
hist(loo(fit_selected_2)$diagnostics$pareto_k, main = "Histogram of k_hat values")

print("Rhat values: ")
rhat(fit_selected_3)

print("Check divergences: ")
check_divergences(fit_selected_3)

print("n_eff values: ")
loo(fit_selected_3)$diagnostics$n_eff

print("Performance of k_hat values and elpd_loo")
loo(fit_selected_3)

print("Distribution of k_hat")
hist(loo(fit_selected_3)$diagnostics$pareto_k, main = "Histogram of k_hat values")

print("Rhat values: ")
rhat(fit_full)

print("Check divergences: ")
check_divergences(fit_full)

print("n_eff values: ")
loo(fit_full)$diagnostics$n_eff

print("Performance of k_hat values and elpd_loo")
loo(fit_full)

print("Distribution of k_hat")
hist(loo(fit_full)$diagnostics$pareto_k, main = "Histogram of k_hat values")

confusionMatrix(table(Y_pred_selected_1, Y_true))

confusionMatrix(table(Y_pred_selected_2, Y_true))

confusionMatrix(table(Y_pred_selected_3, Y_true))

confusionMatrix(table(Y_pred_full, Y_true))

cat("\nSelected Bayesian Inference PSIS_LOO:" ,loo(fit_selected_1)$estimates[1], "\n");
cat("\nSelected from correlation matrix PSIS_LOO:" , loo(fit_selected_2)$estimates[1], "\n");
cat("\nSelected from TreeClassifer PSIS_LOO:" ,loo(fit_selected_3)$estimates[1], "\n\n");
cat("\nFull model PSIS_LOO:" ,loo(fit_full)$estimates[1], "\n\n");
loo_compare(loo(fit_selected_1), loo(fit_selected_2), loo(fit_selected_3), loo(fit_full))

yrep = round(extract(fit_selected_1)$Y_pred)
ppc_dens_overlay(test$death, yrep[sample(nrow(yrep), 25), ])
```

```
yrep = round(extract(fit_selected_2)$Y_pred)
ppc_dens_overlay(test$death, yrep[sample(nrow(yrep), 25), ])

yrep = round(extract(fit_selected_3)$Y_pred)
ppc_dens_overlay(test$death, yrep[sample(nrow(yrep), 25), ])

yrep = round(extract(fit_full)$Y_pred)
ppc_dens_overlay(test$death, yrep[sample(nrow(yrep), 25), ])

full_data = list(N = nrow(data), M = 11, Y = data$death, X =subset(data, select = - death) ,
                 N_test = nrow(test), Y_true = test$death, X_test = subset(test, select = -death),
                 mu = colMeans(subset(data, select = - death)),
                 sigma = c(11.9, 0.5, 969, 0.49, 0.118, 0.48, 97600, 1.03, 4.41, 0.48, 0.47))

X = subset(data, select = c(age, ejection, creatinine, sodium))
N = nrow(X)
M = ncol(X)
Y = data$death
mu = c(60.8, 0.38, 1.39, 137)
sigma_selected = c(11.9, 0.118, 1.03, 4.41)

X_test = subset(test, select = c(age, ejection, creatinine, sodium))
N_test = nrow(X_test)
Y_true = test$death

selected_data_2 = list(N = N, M = M, Y = Y, X = X,
                       N_test = N_test, Y_true = Y_true, X_test = X_test, mu = mu, sigma = sigma_selecte

X = subset(data, select = c(age, ejection, creatinine))
N = nrow(X)
M = ncol(X)
Y = data$death
mu = c(60.8, 0.38, 1.39)
sigma_selected = c(11.9, 0.118, 1.03)

X_test = subset(test, select = c(age, ejection, creatinine))
N_test = nrow(X_test)
Y_true = test$death

selected_data_1 = list(N = N, M = M, Y = Y, X = X,
                       N_test = N_test, Y_true = Y_true, X_test = X_test, mu = mu, sigma = sigma_selecte

fit_selected_1 = stan(file = "LogitRegression2.stan", data = selected_data_1, refresh = 0)
Y_pred_selected_1 = round(colMeans(extract(fit_selected_1)$Y_pred))

fit_selected_2 = stan(file = "LogitRegression2.stan", data = selected_data_2, refresh = 0)
Y_pred_selected_2 = round(colMeans(extract(fit_selected_2)$Y_pred))

fit_full = stan(file = "LogitRegression2.stan", data = full_data, refresh = 0)
Y_pred_full = round(colMeans(extract(fit_full)$Y_pred))

print("\nRhat values: ")
rhat(fit_selected_1)
```

```
print("\nCheck divergences: ")
check_divergences(fit_selected_1)

print("\nn_eff values: ")
loo(fit_selected_1)$diagnostics$n_eff

print("\nPerformance of k_hat values and elpd_loo")
loo(fit_selected_1)

print("\nDistribution of k_hat")
hist(loo(fit_selected_1)$diagnostics$pareto_k, main = "Histogram of k_hat values")

print("\nRhat values: ")
rhat(fit_selected_2)

print("\nCheck divergences: ")
check_divergences(fit_selected_2)

print("\nn_eff values: ")
loo(fit_selected_2)$diagnostics$n_eff

print("\nPerformance of k_hat values and elpd_loo")
loo(fit_selected_2)

print("\nDistribution of k_hat")
hist(loo(fit_selected_2)$diagnostics$pareto_k, main = "Histogram of k_hat values")

print("\nRhat values: ")
rhat(fit_full)

print("\nCheck divergences: ")
check_divergences(fit_full)

print("\nn_eff values: ")
loo(fit_full)$diagnostics$n_eff

print("\nPerformance of k_hat values and elpd_loo")
loo(fit_full)

print("\nDistribution of k_hat")
hist(loo(fit_full)$diagnostics$pareto_k, main = "Histogram of k_hat values")

data = read.csv(file = "heart_failure_clinical_records_dataset.csv")
data = subset (data, select = -time)
names(data) = c("age", "anaemia", "cpk", "diabetes", "ejection", "pressure", "platelets", "creatinine",
smp_size = floor(0.75 * nrow(data))
train_ind = sample(seq_len(nrow(data)), size = smp_size)
org_data = data[train_ind, ]
org_test = data[-train_ind, ]
Y = org_data$death
Y_true = org_test$death

preproc1 = preProcess(data, method=c("center", "scale"))
```

```
norm1 = predict(preproc1, data)

data = norm1[train_ind, ]
test = norm1[-train_ind, ]

X = subset(data, select = c(age, ejection, creatinine, sodium))
N = nrow(X)
M = ncol(X)
mu = colMeans(X)
sigma_selected = cov(X)

X_test = subset(test, select = c(age, ejection, creatinine, sodium))
N_test = nrow(X_test)

selected_data_2 = list(N = N, M = M, Y = Y, X = X,
                       N_test = N_test, Y_true = Y_true, X_test = X_test, mu = mu, sigma = sigma_selecte

X = subset(data, select = c(age, ejection, creatinine))
N = nrow(X)
M = ncol(X)
mu = colMeans(X)
sigma_full = cov(X)
sigma_selected = cov(subset(data, select = c(age, ejection, creatinine)))

X_test = subset(test, select = c(age, ejection, creatinine))
N_test = nrow(X_test)

selected_data_1 = list(N = N, M = M, Y = Y, X = X,
                       N_test = N_test, Y_true = Y_true, X_test = X_test, mu = mu, sigma = sigma_selecte

fit_selected_1 = stan(file = "LogitRegression.stan", data = selected_data_1, refresh = 0)
Y_pred_selected_1 = round(colMeans(extract(fit_selected_1)$Y_pred))

fit_selected_2 = stan(file = "LogitRegression.stan", data = selected_data_2, refresh = 0)
Y_pred_selected_2 = round(colMeans(extract(fit_selected_2)$Y_pred))

print("\nRhat values: ")
rhat(fit_selected_1)

print("\nCheck divergences: ")
check_divergences(fit_selected_1)

print("\nn_eff values: ")
loo(fit_selected_1)$diagnostics$n_eff

print("\nPerformance of k_hat values and elpd_loo")
loo(fit_selected_1)

print("\nDistribution of k_hat")
hist(loo(fit_selected_1)$diagnostics$pareto_k, main = "Histogram of k_hat values")

print("\nRhat values: ")
rhat(fit_selected_2)
```

```r
print("\nCheck divergences: ")
check_divergences(fit_selected_2)

print("\nn_eff values: ")
loo(fit_selected_2)$diagnostics$n_eff

print("\nPerformance of k_hat values and elpd_loo")
loo(fit_selected_2)

print("\nDistribution of k_hat")
hist(loo(fit_selected_2)$diagnostics$pareto_k, main = "Histogram of k_hat values")

confusionMatrix(table(Y_pred_selected_1, Y_true))

confusionMatrix(table(Y_pred_selected_2, Y_true))

cat("\nSelected Bayesian Inference PSIS_LOO:" ,loo(fit_selected_1)$estimates[1], "\n");
cat("\nSelected from correlation matrix PSIS_LOO:" , loo(fit_selected_2)$estimates[1], "\n");
loo_compare(loo(fit_selected_1), loo(fit_selected_2))
```