ProjectFirst

Teemu Sormunen, Abdullah Günay, Nicola Brazzale

11/18/2020

Contents

1 Introduction

In this project we analyse the data from Heart desease dataset [2]. We perform a Bayesian analysis of the data using this sequence of operations: • Overview of analysis problem and of the dataset • Data preprocsessing and visualisation • Prior choice discussion • Models used in our analysis • $\hat{R}$ convergence • HMC specific convergence diagnostics • Effective sample size diagnostic (n_eff or ESS) • Model comparison • Prior sensitivity analysis • Discussion and conclusion

1.1 The problem

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

1.2 The motivation

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

1.3 Modeling idea

We created 3 models which are then compared based on $\hat{r}$, n_{eff}, using the loo package and the classification accuracy. The 1st model is the reduced model and consists in fewer varibles which are selected base on their corrolection with the death event. The 2nd model consists in all varibles except for the varibale "time" as we believe that doens't represent an important factor in the death event scenario. The 3rd model used is a hierarchical model where we treated age class patients in a group with respect to the other selected variables. The 4th model is a non-linear model, as the hierarchical one we took in considertion only the selected varibales in the first model. Modeling is done with package brms, which is a interface for non-linear multivariate multilevel models in Stan.

2 Dataset

2.1 Term explanation

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

- **Creatine phosphokinase (CPK)**
  CPK is an enzyme, which helps to regulate the concentration of adenosine triphosphate (ATP) in cells. ATP is responsible for carrying energy. If the CPK level is high, it often means that there has been an injury or stress on a muscle tissue. Although CPK is one the oldest markers of heart attack, high CPK might also indicate of acute muscle injury along with acute heart problems.
  Normal level of CPK ranges from 20 to 200 IU/L [5]
- **Ejection fraction (EF)**
  EF is a measurement in percentage which describes how much blood left ventricle pumps out of heart with each contraction. Low EF might indicate potential heart issues.
  Normal EF is 50 to 70 percent, while measurement under 40 percept might be an indicator of heart failure or cardiomyopathy. [1]

- **Platelets**
  Platelets are small cell fragments which can form clots. Too many platelets can lead to clotting of blood vessels, which in turn can lead to heart attack. Too Normal range of platelets is from 150 000 to 450 000. [4]
- **Serum creatinine**
  When creatine breaks down, it forms a waste product called creatinine. Kidneys normally remove creatinine from body. Serum creatinine measures level of creatinine in the blood, indicating the kidney health. High levels of creatinine might indicate a kidney dysfunctioning.
  Normal level of creatinine range from 0.9 to 1.3 mg/dL in men and 0.6 to 1.1 mg/dL in women who are 18 to 60 years old. [6]
- **Serum sodium**
  Serum sodium measures the amount of sodium in blood. Sodium enters blood through food and drink, and leaves by urine, stool and sweat. Too much sodium can cause blood pressure, while too little sodium can cause nausea, vomiting, exhaustion or dizziness.
  Normal levels of serum sodium are 135 to 145 mEq/L, according to Mayo Clinic. There are however different interpretations of "normal".[3]

2.2 Dataset introduction

The dataset of 299 patients was produced as a result of study [2] from Pakistani's city Faisalabad. All of the patients were over 40 years old, each having ventricular systolic dysfunction. This means that patient has poor left ventricular ejection fraction. The dataset has 105 women, and 194 men. EF, serum creatinine and platelets are categorical variables, and age, serum sodium and CPK are continuous variables.
Statistical analysis by [2] found age, creatinine, sodium, anemia and BP as significant variables.

3 Models

3.1 Prior choices

The types of priors we considered in this project are uniformative priors and regularizing priors. Surely this dataset is not the only one about heart failure but since no prior knowledge are provided in the original papers we opted for using uninformative priors.

3.2 $\hat{R}$ **convergence**

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

3.4 Effective sample size diagnostic (n_eff or ESS) and an interpretation of the results

3.5 Posterior predictive checking and interpretation of the results

3.6 Model comparison and interpretation of the results

3.7 Predictive performance assessment (classification)

3.8 Prior sensitivity analysis (alternative prior tested)

3.9 Discussion of problems and further improvements

4 Packages

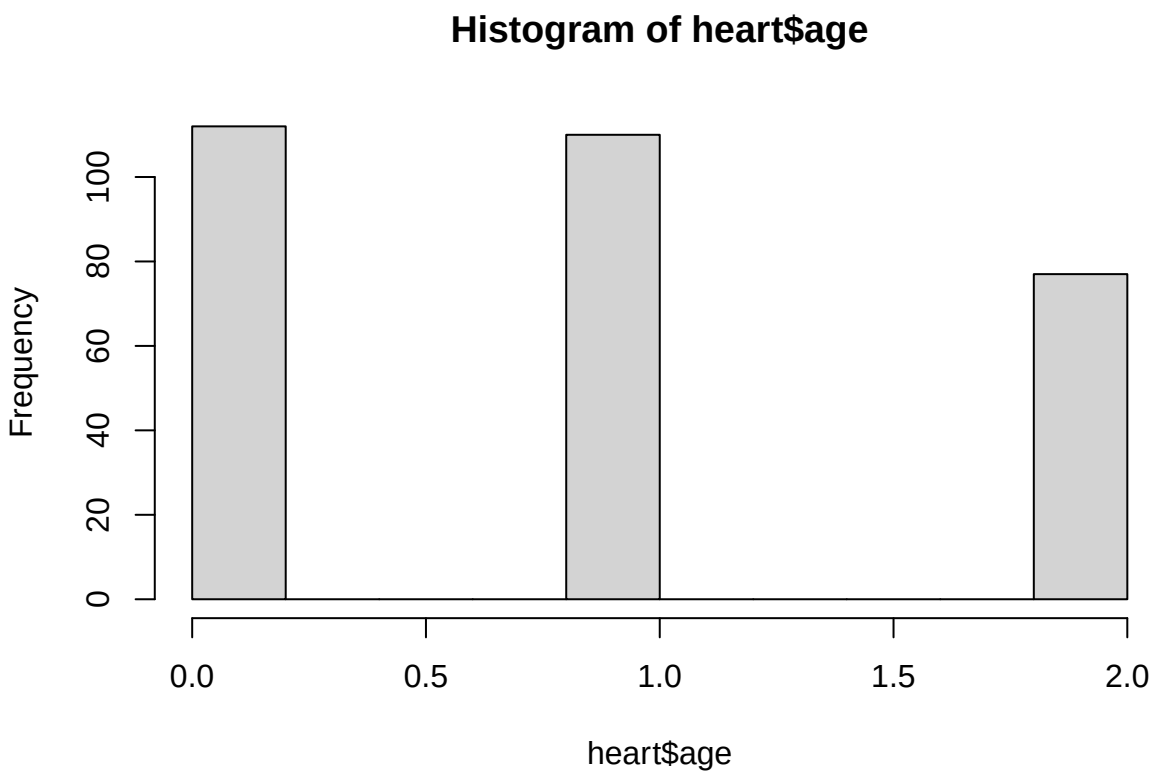
Load data

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

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   age = col_double(),
##   anaemia = col_double(),
##   creatinine_phosphokinase = col_double(),
##   diabetes = col_double(),
##   ejection_fraction = col_double(),
##   high_blood_pressure = col_double(),
##   platelets = col_double(),
##   serum_creatinine = col_double(),
##   serum_sodium = col_double(),
##   sex = col_double(),
##   smoking = col_double(),
##   time = col_double(),
##   DEATH_EVENT = col_double()
## )
```

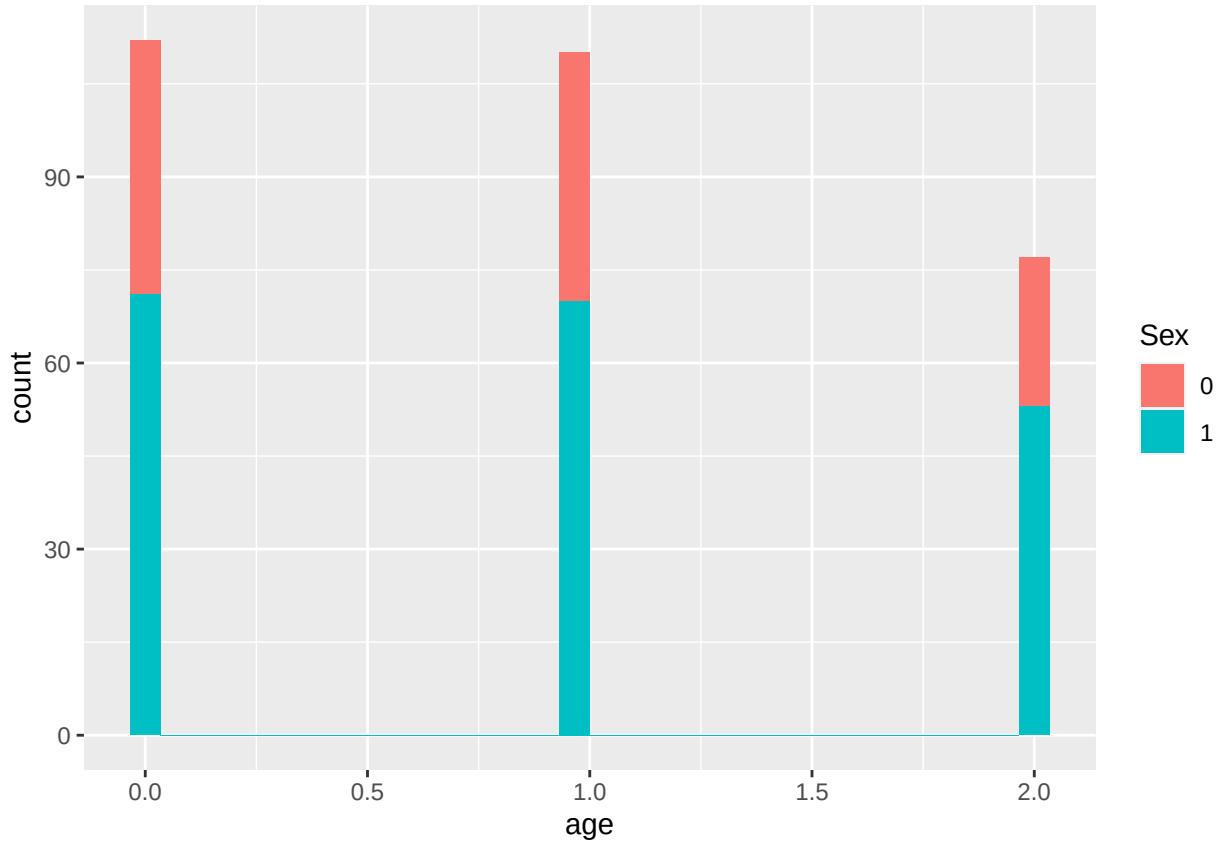
Data preprocessing - Age Bins

```
heart$age[heart$age<=55] = 0
heart$age[heart$age>55 & heart$age<70] = 1
heart$age[heart$age>=70] = 2
hist(heart$age)
```

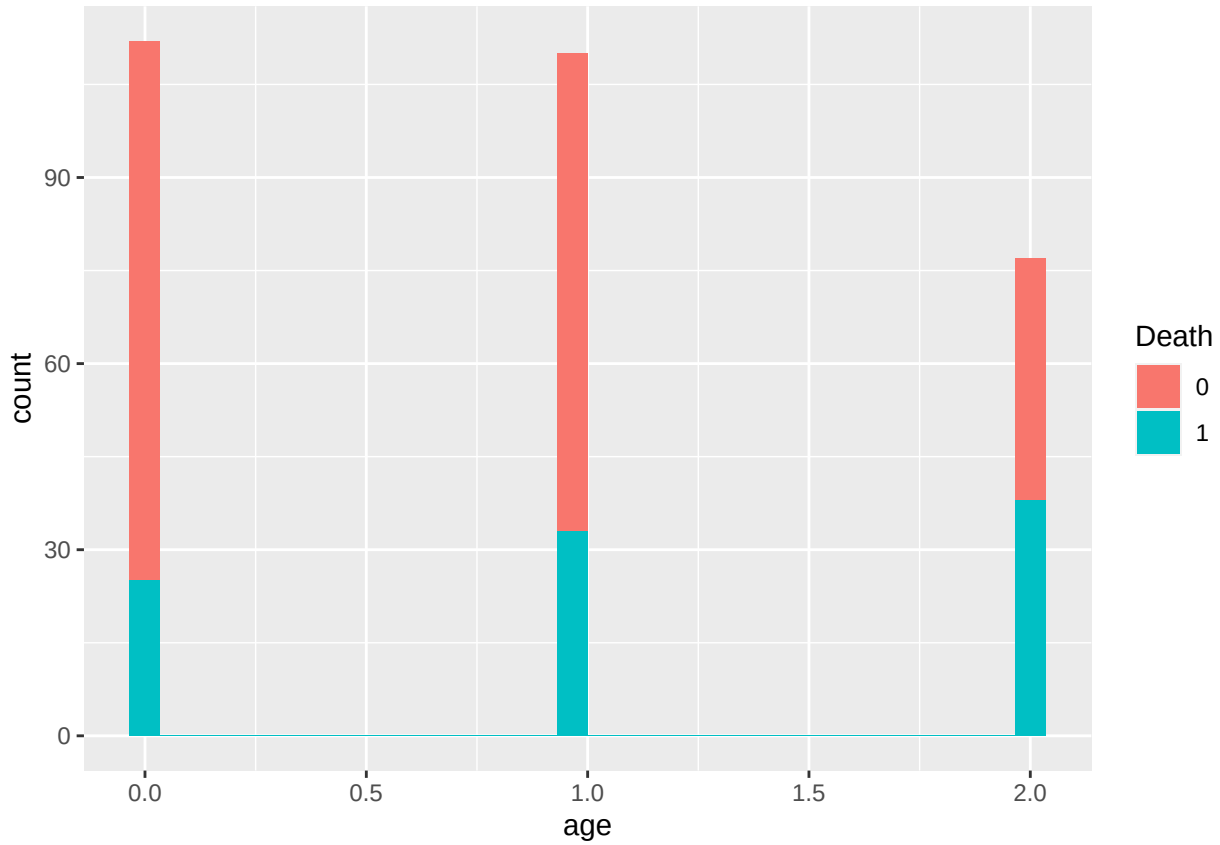


Plot histograms

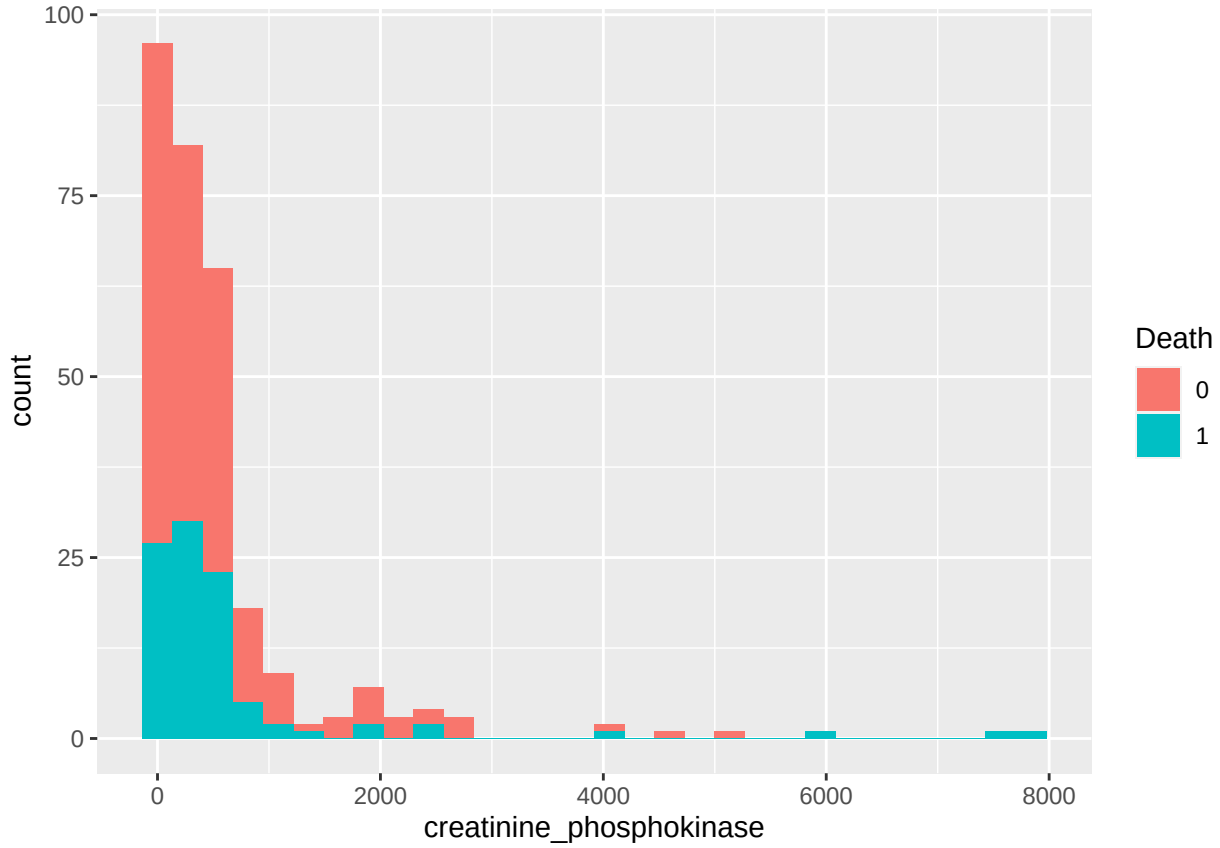
```
ggplot(heart, aes(x=age)) + geom_histogram(aes(fill=as.character(sex)), bins = 30) + labs(fill = "Sex")
```



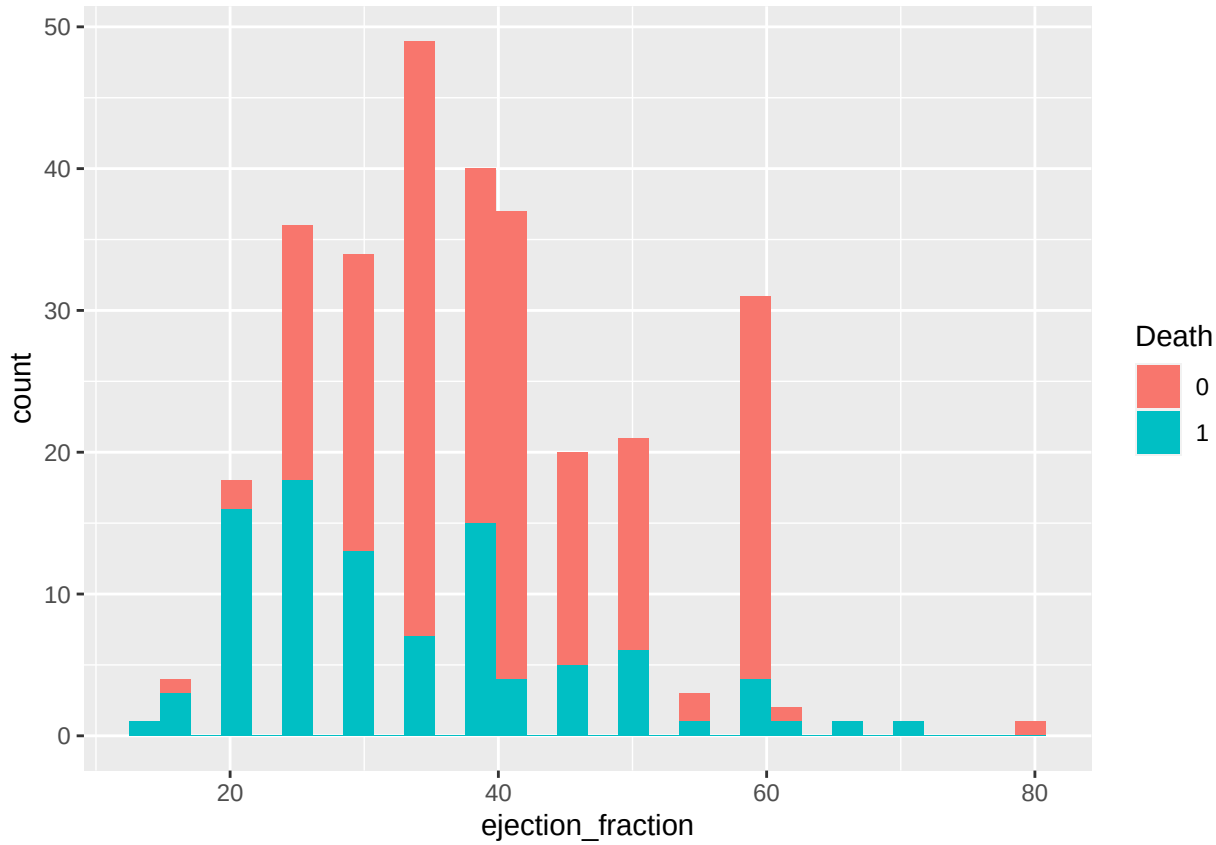

```
ggplot(heart, aes(x=age)) + geom_histogram(aes(fill=as.character(DEATH_EVENT)), bins = 30) + labs(fill =
```



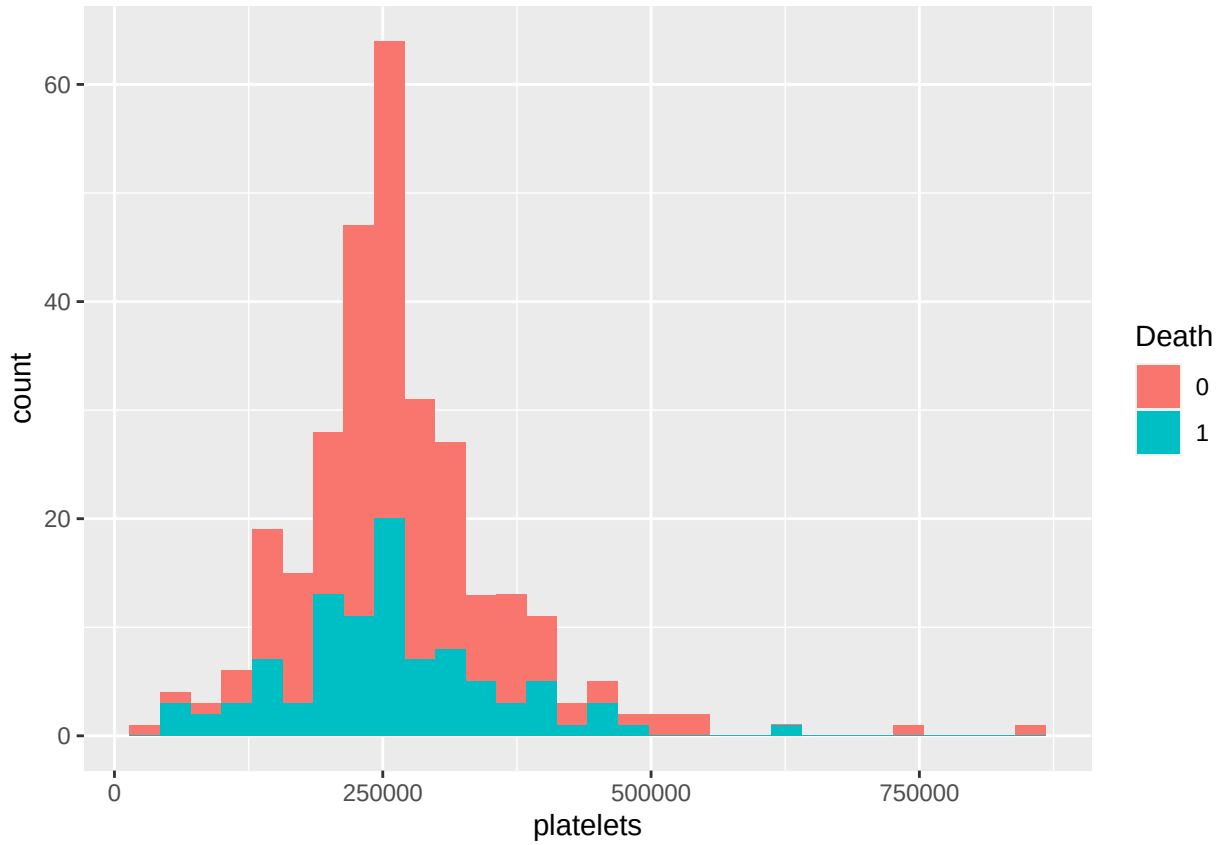

```
ggplot(heart, aes(x=creatinine_phosphokinase)) + geom_histogram(aes(fill=as.character(DEATH_EVENT)), bi
```



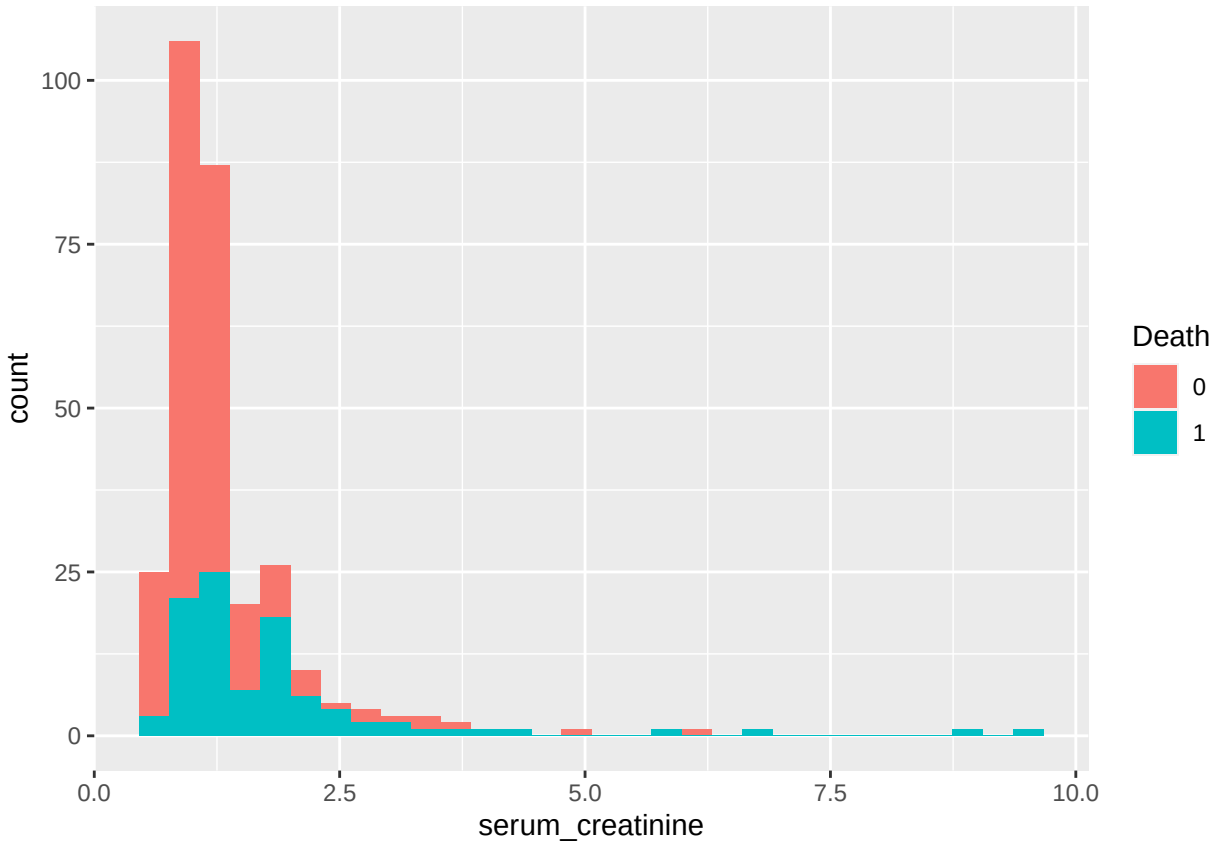

```
ggplot(heart, aes(x=ejection_fraction)) + geom_histogram(aes(fill=as.character(DEATH_EVENT)), bins = 30)
```




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



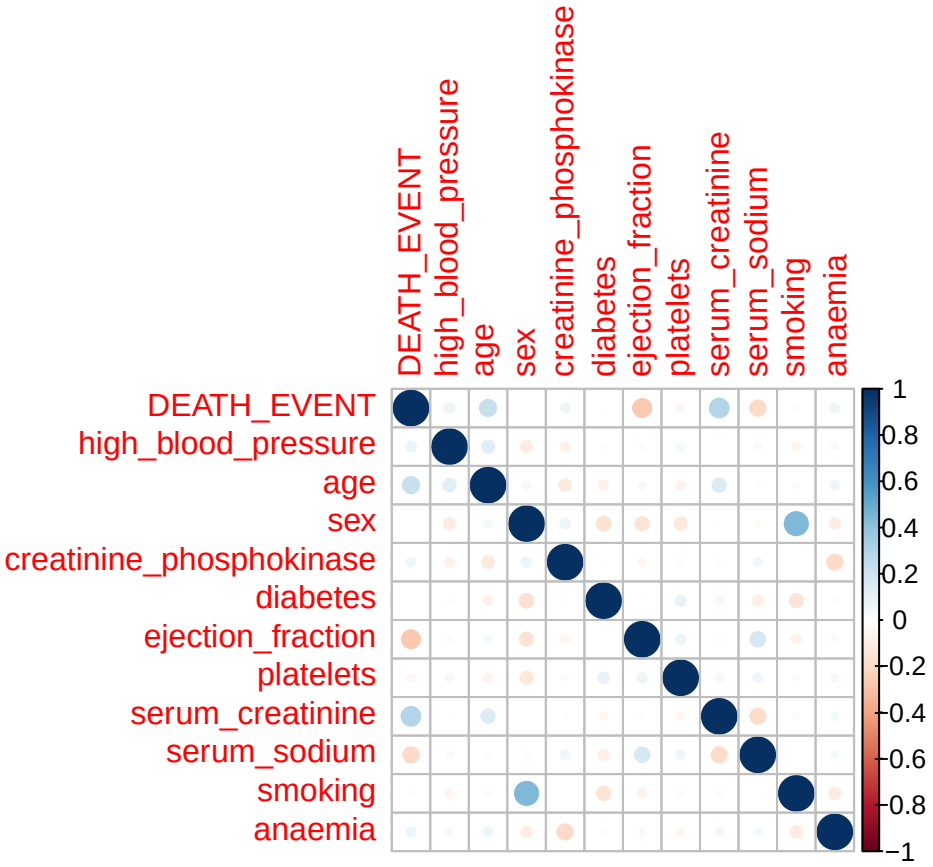

```
ggplot(heart, aes(x=serum_creatinine)) + geom_histogram(aes(fill=as.character(DEATH_EVENT)), bins = 30)
```



Plot individual

Correlation matrix

```
pred <- c("high_blood_pressure", "age", "sex", "creatinine_phosphokinase", "diabetes", "ejection_fracti
target <- c("DEATH_EVENT")
#formula <- paste("DEATH_EVENT ~", paste(pred, collapse = "+"))
p <- length(pred)
n <- nrow(heart)
x = cor(heart[, c(target,pred)])
corrplot(x)
```



5 BRMS modeling

In BRMS modeling, the parameters are said to either be population level or group level. Population level probably means the same thing as regular parameters in our course, and group level equals hyper parameters (??????????????????)

Brms example, investigate results based on age. **Family argument** specifies the distribution family of the output.

Prior argument for each of the parameters, in this case only age. One can set different priors for each population level parameter, or group level parameter.

```
# Split test and train data
test.size <- 0.3
train.indice <- sample(nrow(heart), nrow(heart)*(1-test.size))
train.data <- heart[train.indice,]
test.data <- heart[-train.indice,]
```

```
# Fit model based on age
fitFeatSel <- brm(formula = DEATH_EVENT ~ age + ejection_fraction + serum_creatinine + serum_sodium,
            data = train.data,
            family = bernoulli(),
            prior = set_prior('normal(0, 1000)'),
            refresh=0
)
```

```
## Compiling Stan program...
```

```
## Start sampling
```

Analyze stan code

```
summary(fitFeatSel)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: DEATH_EVENT ~ age + ejection_fraction + serum_creatinine + serum_sodium
##    Data: train.data (Number of observations: 209)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept             6.72      5.44    -3.82    17.46 1.00     4507     3058
## age                   0.65      0.23     0.22     1.10 1.00     4240     3042
## ejection_fraction    -0.07      0.02    -0.11    -0.04 1.00     4089     3363
## serum_creatinine      0.66      0.21     0.25     1.10 1.00     4288     2528
## serum_sodium         -0.05      0.04    -0.13     0.03 1.00     4485     3044
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
stancode(fitFeatSel)
```

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

```
##   }
##   // priors including all constants
##   target += normal_lpdf(b | 0, 1000);
##   target += student_t_lpdf(Intercept | 3, 0, 2.5);
## }
## generated quantities {
##   // actual population-level intercept
##   real b_Intercept = Intercept - dot_product(means_X, b);
## }
```

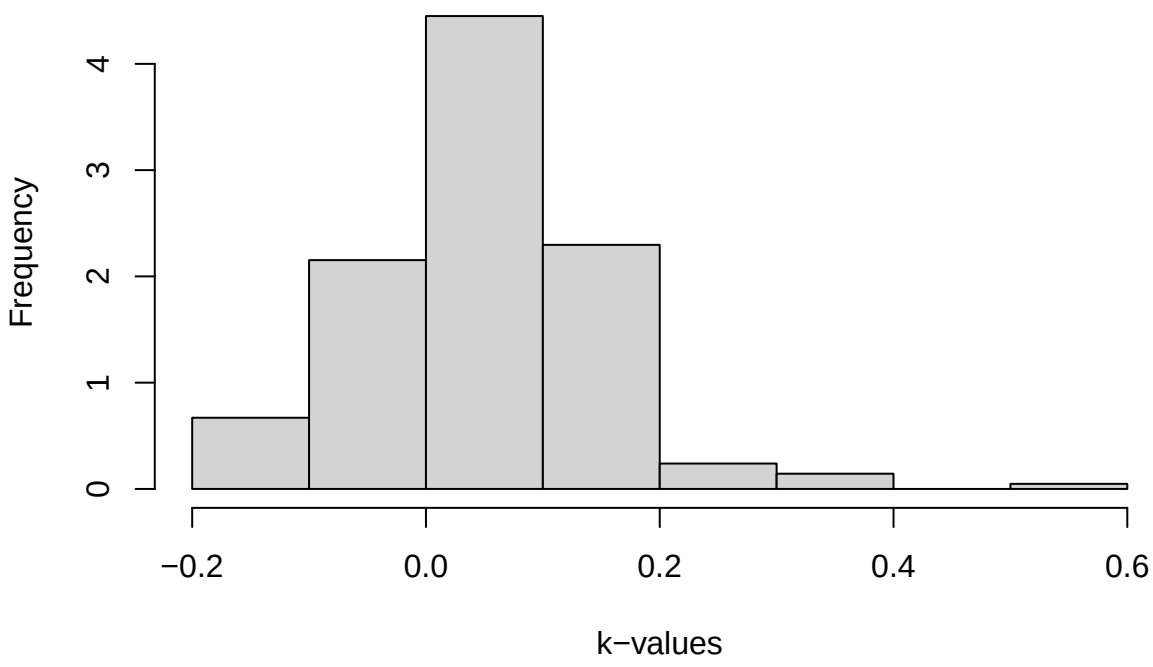
Predict survival

```
predsFeatSel <- round(predict(fitFeatSel, newdata = test.data)[,1])
predsFeatSel.corr <- predsFeatSel == test.data$DEATH_EVENT

accFeatSel <- length(predsFeatSel.corr[predsFeatSel.corr == TRUE])/nrow(test.data)
accFeatSel
```

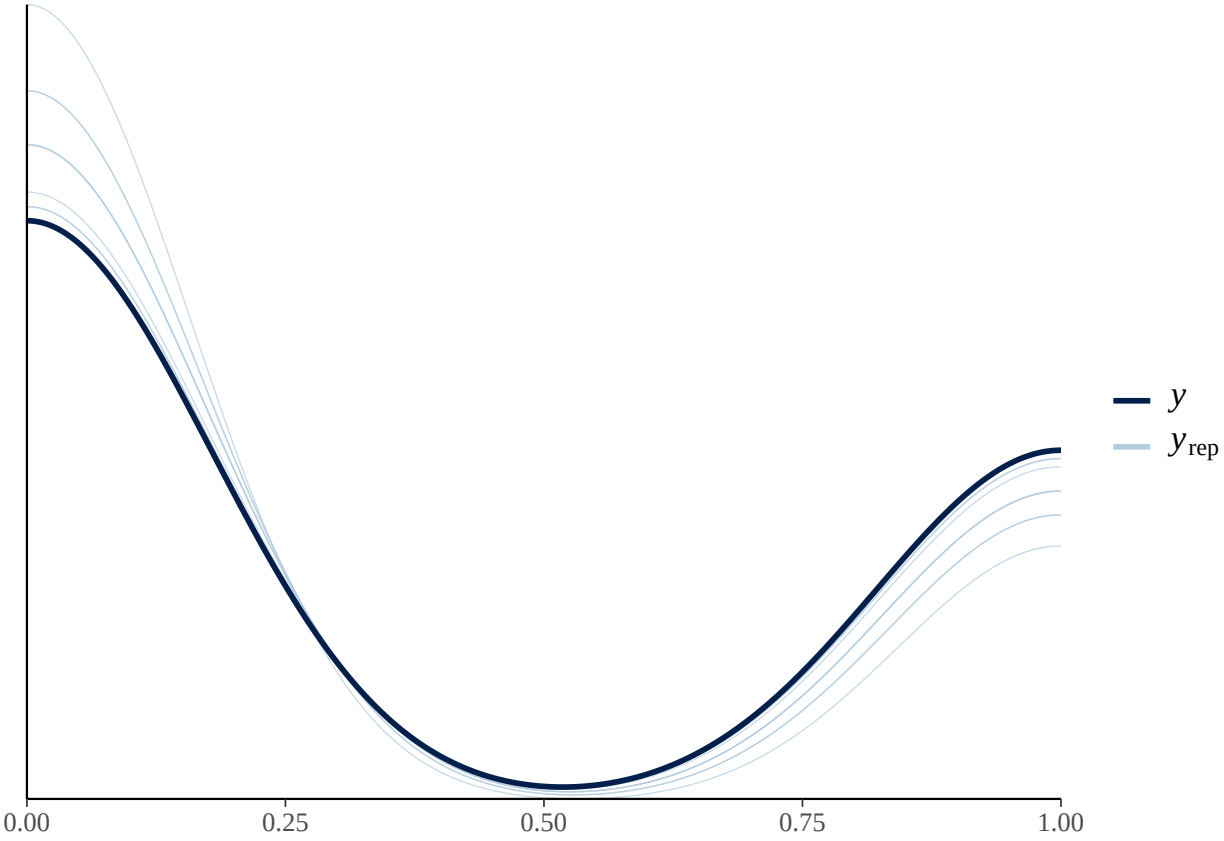
```
## [1] 0.7444444
```

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




```
pp_check(fitFeatSel, newdata = test.data)
```

```
## Using 10 posterior samples for ppc type 'dens_overlay' by default.
```



Full model

```
fitFull <- brm(formula = DEATH_EVENT ~ age + ejection_fraction + serum_creatinine + serum_sodium + high
           data = train.data,
           family = bernoulli(),
           prior = set_prior('normal(50, 1000)'),
           refresh=0
)
```

```
## Compiling Stan program...
```

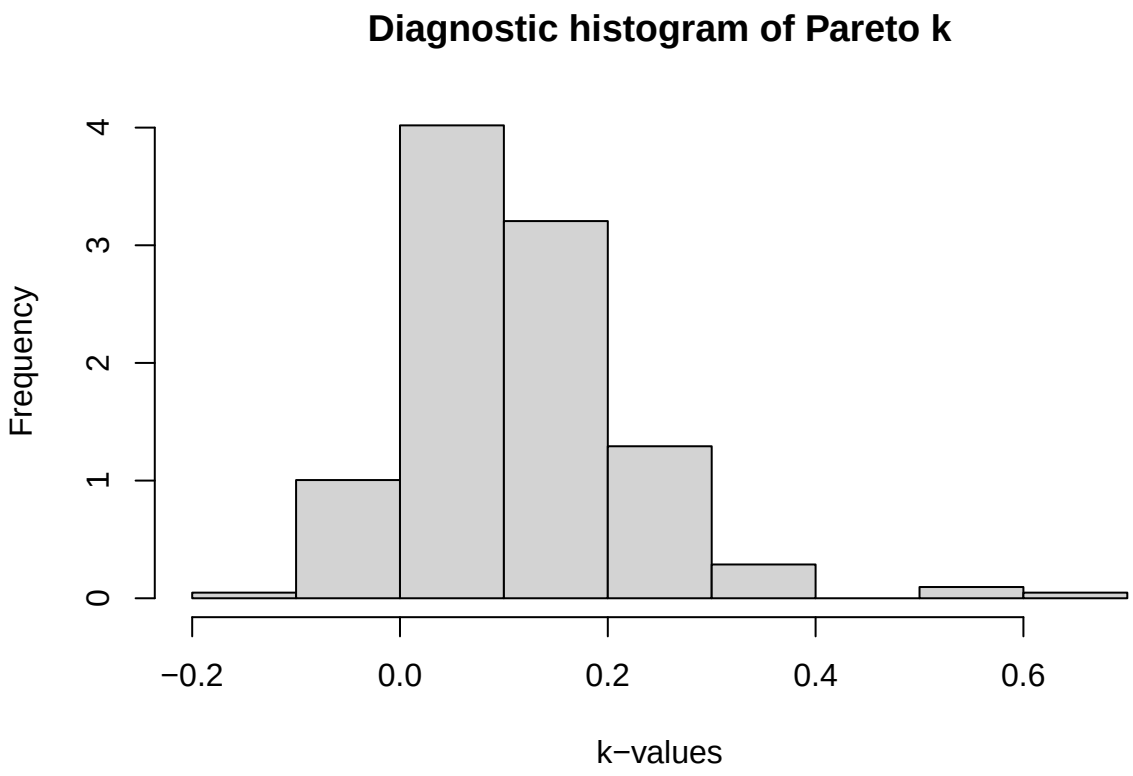
```
## Start sampling
```

```
predsFull <- round(predict(fitFull, newdata = test.data)[,1])
predFull.corr <- predsFull == test.data$DEATH_EVENT

accFull <- length(predFull.corr[predFull.corr == TRUE])/nrow(test.data)
accFull
```

```
## [1] 0.7111111
```

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




```
summary(fitFull)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: DEATH_EVENT ~ age + ejection_fraction + serum_creatinine + serum_sodium + high_blood_pressu
##    Data: train.data (Number of observations: 209)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##                          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                    6.58      5.90    -4.69    18.65 1.00     4454
## age                          0.79      0.24     0.33     1.28 1.00     4648
## ejection_fraction           -0.07      0.02    -0.11    -0.04 1.00     4129
## serum_creatinine             0.72      0.23     0.32     1.21 1.00     4083
## serum_sodium                -0.05      0.04    -0.14     0.03 1.00     4409
## high_blood_pressure          0.39      0.38    -0.37     1.13 1.00     4277
## creatinine_phosphokinase     0.00      0.00     0.00     0.00 1.00     3654
## diabetes                     0.11      0.37    -0.61     0.85 1.00     4357
## smoking                      0.08      0.39    -0.69     0.86 1.00     4705
## anaemia                      0.37      0.37    -0.36     1.10 1.00     4732
##                          Tail_ESS
## Intercept                    2952
## age                          3135
## ejection_fraction            2809
## serum_creatinine             2673
## serum_sodium                 2742
## high_blood_pressure          2682
## creatinine_phosphokinase     3086
## diabetes                     3102
```

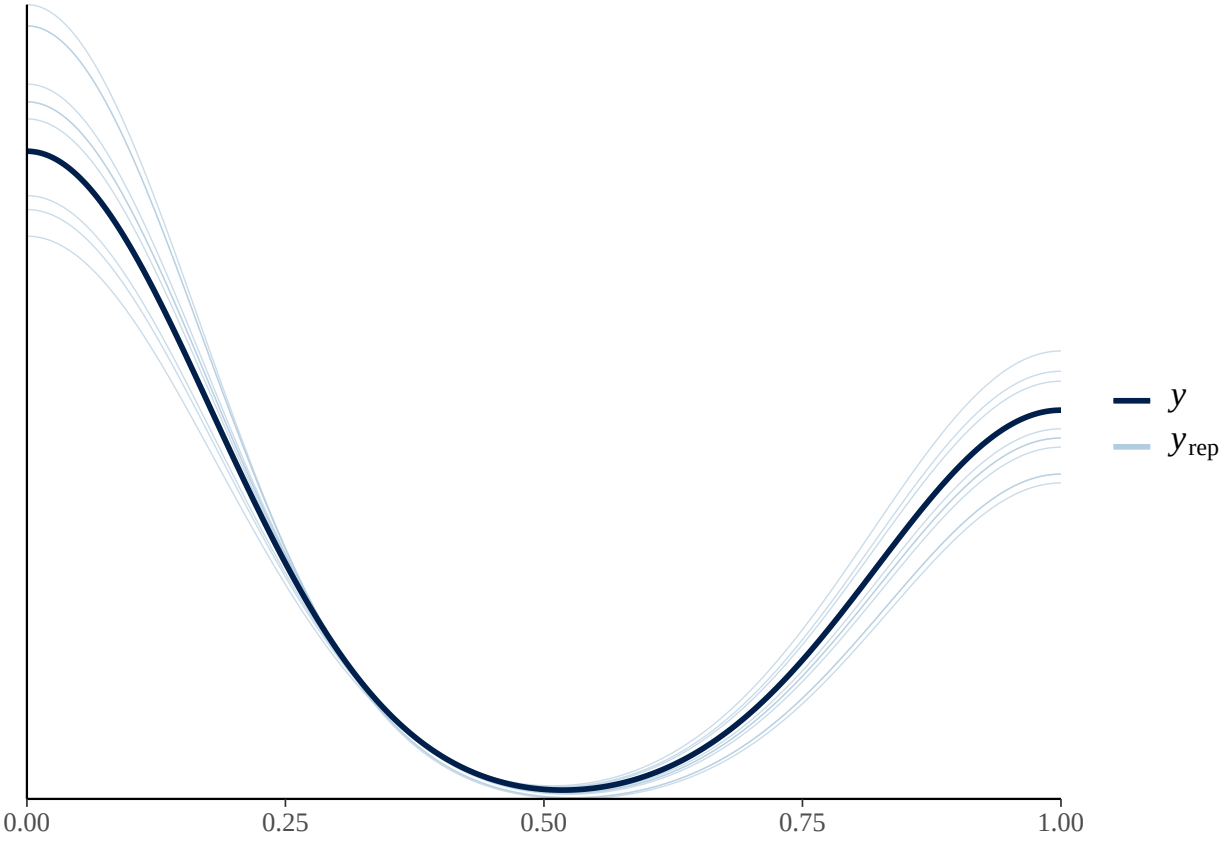
```
## smoking                      2508
## anaemia                      2938
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
stancode(fitFull)
```

```
## // generated with brms 2.14.4
## functions {
## }
## data {
##   int<lower=1> N;  // total number of observations
##   int Y[N];  // response variable
##   int<lower=1> K;  // number of population-level effects
##   matrix[N, K] X;  // population-level design matrix
##   int prior_only;  // should the likelihood be ignored?
## }
## transformed data {
##   int Kc = K - 1;
##   matrix[N, Kc] Xc;  // centered version of X without an intercept
##   vector[Kc] means_X;  // column means of X before centering
##   for (i in 2:K) {
##     means_X[i - 1] = mean(X[, i]);
##     Xc[, i - 1] = X[, i] - means_X[i - 1];
##   }
## }
## parameters {
##   vector[Kc] b;  // population-level effects
##   real Intercept;  // temporary intercept for centered predictors
## }
## transformed parameters {
## }
## model {
##   // likelihood including all constants
##   if (!prior_only) {
##     target += bernoulli_logit_glm_lpmf(Y | Xc, Intercept, b);
##   }
##   // priors including all constants
##   target += normal_lpdf(b | 50, 1000);
##   target += student_t_lpdf(Intercept | 3, 0, 2.5);
## }
## generated quantities {
##   // actual population-level intercept
##   real b_Intercept = Intercept - dot_product(means_X, b);
## }
```

```
pp_check(fitFull, newdata = test.data)
```

```
## Using 10 posterior samples for ppc type 'dens_overlay' by default.
```




```
#NON LINEAR
fitNonLinear <- brm(formula = DEATH_EVENT ~ s(ejection_fraction) + s(serum_creatinine) + s(serum_sodium
```

```
## Compiling Stan program...
```

```
## Start sampling
```

```
## Warning: There were 80 divergent transitions after warmup. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
summary(fitNonLinear)
```

```
## Warning: There were 80 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help. See http://mc-stan.org/misc/
## warnings.html#divergent-transitions-after-warmup
```

```
##  Family: poisson
##   Links: mu = log
## Formula: DEATH_EVENT ~ s(ejection_fraction) + s(serum_creatinine) + s(serum_sodium)
##    Data: train.data (Number of observations: 209)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Smooth Terms:
##                            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sds(sejection_fraction_1)      1.10      0.89     0.04     3.42 1.00     1288
## sds(sserum_creatinine_1)       1.74      1.49     0.05     5.52 1.00     1269
```

```
## sds(sserum_sodium_1)          1.35      1.13     0.05     4.31 1.00     1303
##                           Tail_ESS
## sds(sejection_fraction_1)     1848
## sds(sserum_creatinine_1)      1446
## sds(sserum_sodium_1)          1429
##
## Population-Level Effects:
##                      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept               -1.45      0.15    -1.75    -1.17 1.00     3512
## sejection_fraction_1    -3.10      2.77    -9.60     2.21 1.01     1538
## sserum_creatinine_1      5.93      4.47    -1.48    17.58 1.00     1269
## sserum_sodium_1          0.67      3.61    -5.53     9.61 1.00     1213
##                      Tail_ESS
## Intercept                2713
## sejection_fraction_1     1333
## sserum_creatinine_1      1072
## sserum_sodium_1          1027
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
stancode(fitNonLinear)
```

```
## // generated with brms 2.14.4
## functions {
## }
## data {
##   int<lower=1> N;  // total number of observations
##   int Y[N];  // response variable
##   // data for splines
##   int Ks;  // number of linear effects
##   matrix[N, Ks] Xs;  // design matrix for the linear effects
##   // data for spline s(ejection_fraction)
##   int nb_1;  // number of bases
##   int knots_1[nb_1];  // number of knots
##   // basis function matrices
##   matrix[N, knots_1[1]] Zs_1_1;
##   // data for spline s(serum_creatinine)
##   int nb_2;  // number of bases
##   int knots_2[nb_2];  // number of knots
##   // basis function matrices
##   matrix[N, knots_2[1]] Zs_2_1;
##   // data for spline s(serum_sodium)
##   int nb_3;  // number of bases
##   int knots_3[nb_3];  // number of knots
##   // basis function matrices
##   matrix[N, knots_3[1]] Zs_3_1;
##   int prior_only;  // should the likelihood be ignored?
## }
## transformed data {
## }
## parameters {
##   real Intercept;  // temporary intercept for centered predictors
##   vector[Ks] bs;  // spline coefficients
```

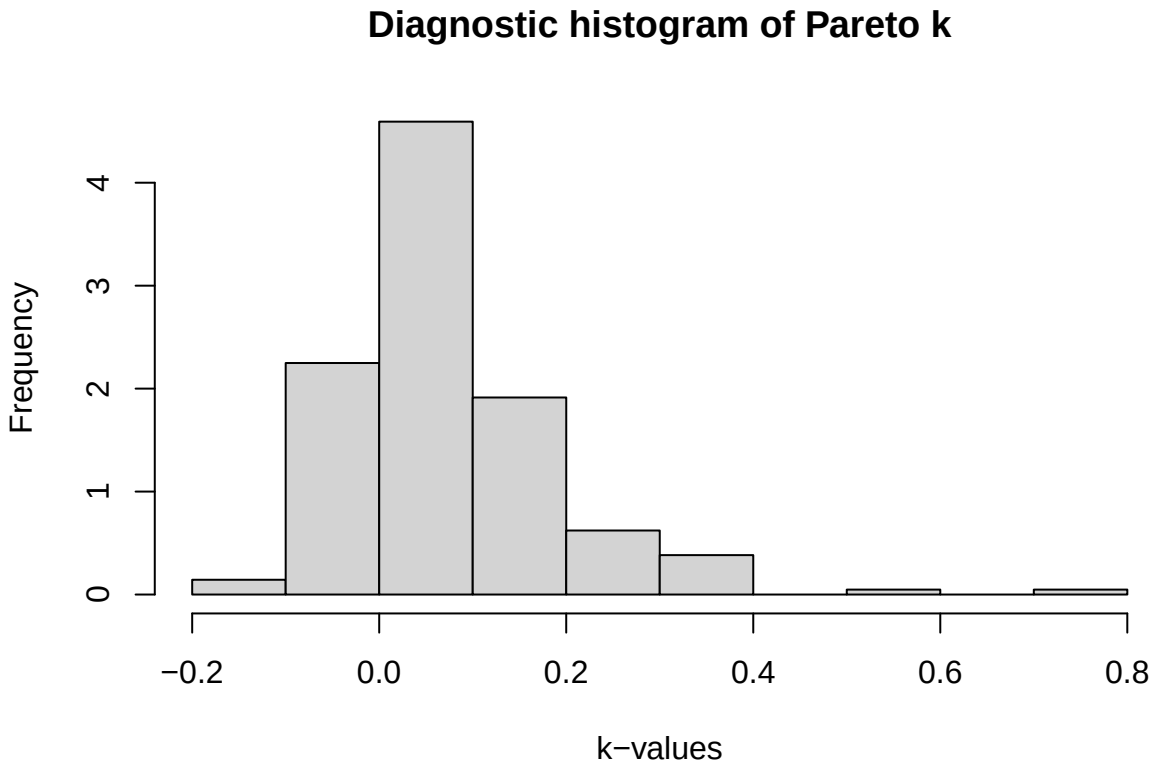
```
##   // parameters for spline s(ejection_fraction)
##   // standarized spline coefficients
##   vector[knots_1[1]] zs_1_1;
##   real<lower=0> sds_1_1;  // standard deviations of spline coefficients
##   // parameters for spline s(serum_creatinine)
##   // standarized spline coefficients
##   vector[knots_2[1]] zs_2_1;
##   real<lower=0> sds_2_1;  // standard deviations of spline coefficients
##   // parameters for spline s(serum_sodium)
##   // standarized spline coefficients
##   vector[knots_3[1]] zs_3_1;
##   real<lower=0> sds_3_1;  // standard deviations of spline coefficients
## }
## transformed parameters {
##   // actual spline coefficients
##   vector[knots_1[1]] s_1_1;
##   // actual spline coefficients
##   vector[knots_2[1]] s_2_1;
##   // actual spline coefficients
##   vector[knots_3[1]] s_3_1;
##   // compute actual spline coefficients
##   s_1_1 = sds_1_1 * zs_1_1;
##   // compute actual spline coefficients
##   s_2_1 = sds_2_1 * zs_2_1;
##   // compute actual spline coefficients
##   s_3_1 = sds_3_1 * zs_3_1;
## }
## model {
##   // likelihood including all constants
##   if (!prior_only) {
##     // initialize linear predictor term
##     vector[N] mu = Intercept + rep_vector(0.0, N) + Xs * bs + Zs_1_1 * s_1_1 + Zs_2_1 * s_2_1 + Zs_3
##     target += poisson_log_lpmf(Y | mu);
##   }
##   // priors including all constants
##   target += student_t_lpdf(Intercept | 3, -2.3, 2.5);
##   target += student_t_lpdf(sds_1_1 | 3, 0, 2.5)
##     - 1 * student_t_lccdf(0 | 3, 0, 2.5);
##   target += std_normal_lpdf(zs_1_1);
##   target += student_t_lpdf(sds_2_1 | 3, 0, 2.5)
##     - 1 * student_t_lccdf(0 | 3, 0, 2.5);
##   target += std_normal_lpdf(zs_2_1);
##   target += student_t_lpdf(sds_3_1 | 3, 0, 2.5)
##     - 1 * student_t_lccdf(0 | 3, 0, 2.5);
##   target += std_normal_lpdf(zs_3_1);
## }
## generated quantities {
##   // actual population-level intercept
##   real b_Intercept = Intercept;
## }
```

```
looNL<-loo(fitNonLinear)
```

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

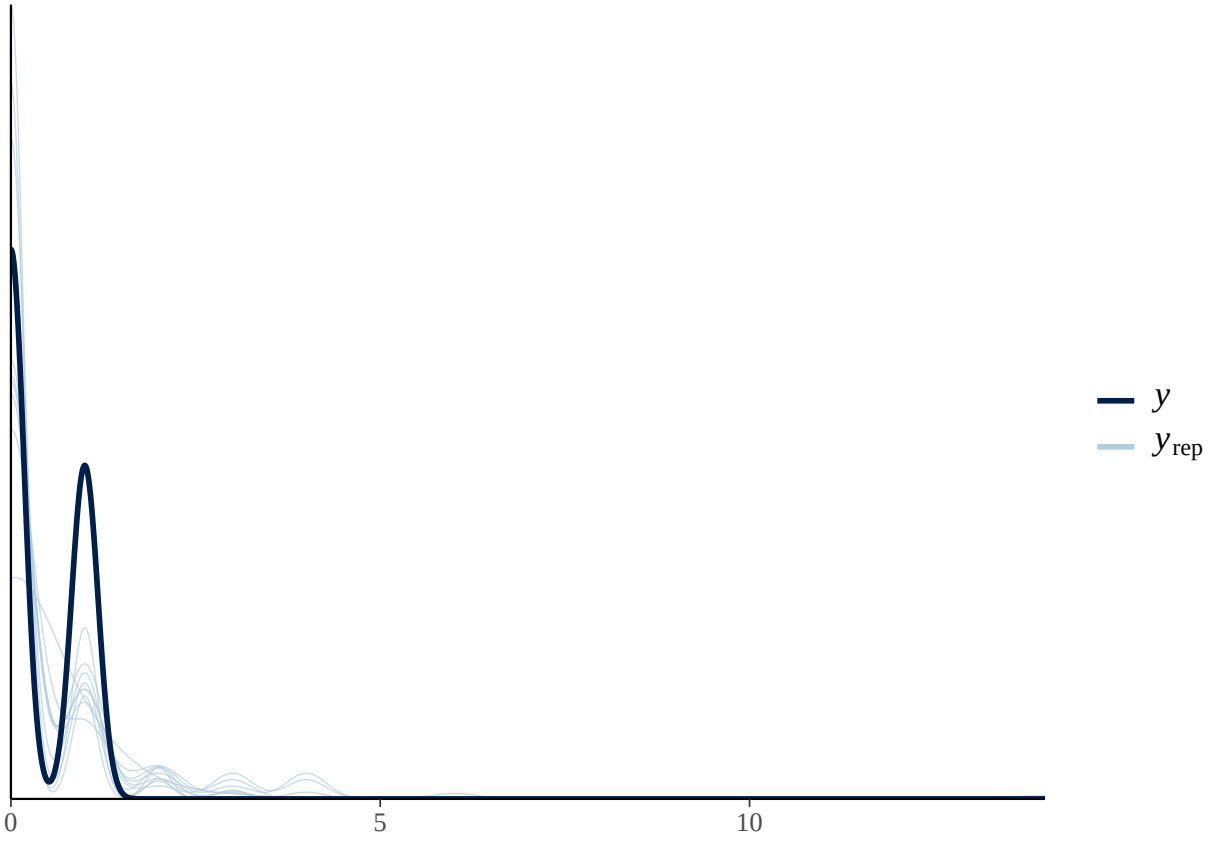
```
## for problematic observations.
```

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




```
pp_check(fitNonLinear, newdata = test.data)
```

```
## Using 10 posterior samples for ppc type 'dens_overlay' by default.
```



First try with Hierarchical model

```
fitHier <- brm(formula = DEATH_EVENT ~ ejection_fraction + serum_creatinine + serum_sodium + (ejection_
           data = train.data,
           family = bernoulli(),
           prior = set_prior('normal(0, 1000)'),
           refresh=0,
           control = list(adapt_delta = 0.99),
           save_all_pars=T,
           save_pars=save_pars(all = T)
)
```

```
## Warning: Argument 'save_all_pars' is deprecated. Please use argument 'all' in
## function 'save_pars()' instead.
```

```
## Compiling Stan program...
```

```
## Start sampling
```

```
## Warning: There were 561 transitions after warmup that exceeded the maximum treedepth. Increase max_t
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
summary(fitHier)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: DEATH_EVENT ~ ejection_fraction + serum_creatinine + serum_sodium + (ejection_fraction + se
##    Data: train.data (Number of observations: 209)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
```

```
##          total post-warmup samples = 4000
##
## Group-Level Effects:
## ~age (Number of levels: 3)
##                                     Estimate Est.Error l-95% CI u-95% CI
## sd(Intercept)                           1.46      1.42     0.06     5.22
## sd(ejection_fraction)                   0.09      0.13     0.00     0.43
## sd(serum_creatinine)                    0.66      0.82     0.02     2.68
## sd(serum_sodium)                        0.02      0.03     0.00     0.10
## cor(Intercept,ejection_fraction)       -0.05      0.45    -0.85     0.80
## cor(Intercept,serum_creatinine)        -0.04      0.46    -0.85     0.80
## cor(ejection_fraction,serum_creatinine)  0.02      0.46    -0.83     0.83
## cor(Intercept,serum_sodium)            -0.10      0.45    -0.84     0.77
## cor(ejection_fraction,serum_sodium)    -0.08      0.46    -0.87     0.81
## cor(serum_creatinine,serum_sodium)     -0.05      0.46    -0.85     0.80
##                                     Rhat Bulk_ESS Tail_ESS
## sd(Intercept)                       1.00     2588     2538
## sd(ejection_fraction)               1.00     1007     1354
## sd(serum_creatinine)                1.00     1817     2028
## sd(serum_sodium)                    1.00     1164     1962
## cor(Intercept,ejection_fraction)    1.00     2886     2794
## cor(Intercept,serum_creatinine)     1.00     3842     2965
## cor(ejection_fraction,serum_creatinine) 1.00 3429     2688
## cor(Intercept,serum_sodium)         1.00     3691     2962
## cor(ejection_fraction,serum_sodium) 1.00     3272     2665
## cor(serum_creatinine,serum_sodium)  1.00     2794     3012
##
## Population-Level Effects:
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept             7.09      5.55    -3.93    18.00 1.00     4928     2849
## ejection_fraction    -0.07      0.05    -0.17     0.04 1.00     1407      981
## serum_creatinine      0.78      0.55    -0.23     1.99 1.00     1451     1252
## serum_sodium         -0.04      0.04    -0.13     0.04 1.00     4363     2919
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
looHier<-loo(fitHier, moment_match = TRUE)
```

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

```
looHier
```

```
##
## Computed from 4000 by 209 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -114.0  9.0
## p_loo        10.7  1.8
## looic       228.1 17.9
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
```

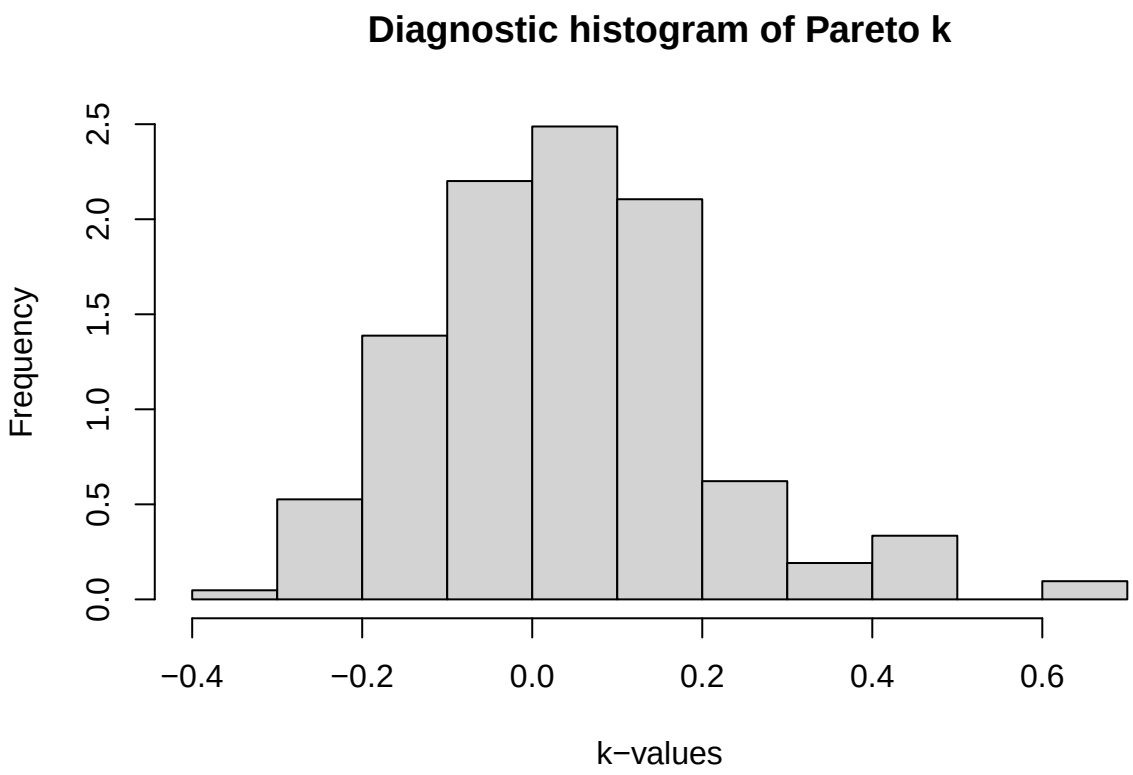
```
## (-Inf, 0.5]   (good)     207   99.0%   688
##  (0.5, 0.7]   (ok)         2    1.0%   232
##    (0.7, 1]   (bad)        0    0.0%   <NA>
##    (1, Inf)   (very bad)   0    0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
predsHier <- round(predict(fitHier, newdata = test.data)[,1])
predsHier.corr <- predsHier == test.data$DEATH_EVENT

accHier <- length(predsHier.corr[predsHier.corr == TRUE])/nrow(test.data)
accHier
```

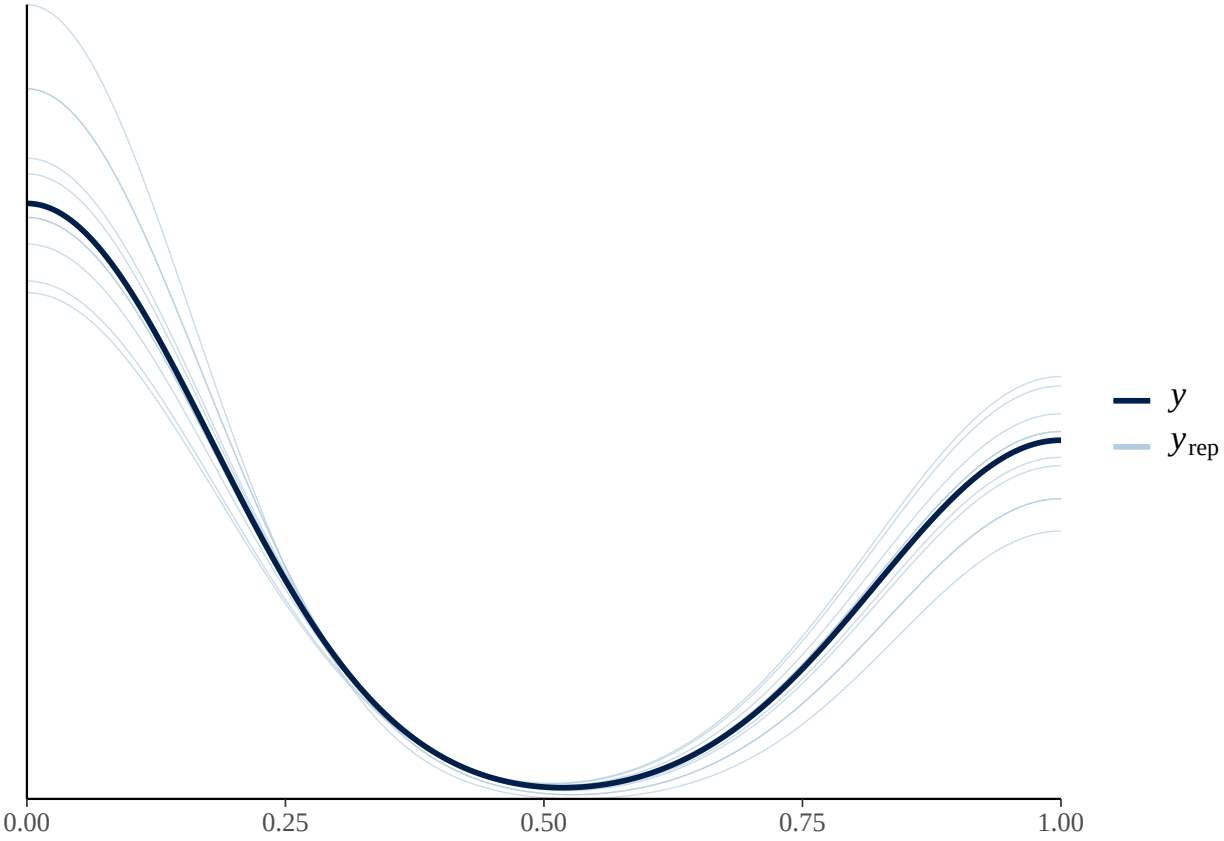
```
## [1] 0.7111111
```

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




```
pp_check(fitHier, newdata = test.data)
```

```
## Using 10 posterior samples for ppc type 'dens_overlay' by default.
```



6 Conclusion

References

[1] Ejection fraction heart failure measurement, 2017.

[2] Ahmad T, Munir A, Bhatti SH, Aftab M, Raza MA. Survival analysis of heart failure patients: A case study. 2017. doi: https://doi.org/10.1371/journal.pone.0181001.

[3] Christine Case-Lo. Blood sodium test, 2018. URL https://www.healthline.com/health/sodium-blood.

[4] Gregg D, Goldschmidt-Clermont P. J. Platelets and cardiovascular disease. *Journal of the American Heart Association*, 108, 2003. doi: https://doi.org/10.1161/01.CIR.0000086897.15588.4B.

[5] Roshan Patel Ravinder S. Aujla. Creatine phosphokinase. *StatPearls*, 2020. URL https://www.ncbi.nlm.nih.gov/books/NBK546624/.

[6] Roth Erica. Creatinine blood test, 2019. URL https://www.healthline.com/health/creatinine-blood.