ProjectFirst

Teemu Sormunen, Abdullah Günay, Nicola Brazzale

11/18/2020

Contents

1 Introduction

1.1 The problem

Write about heart diseases, whats the problem? (not knowing when patient is dying?) [1]

1.2 The motivation

Why do we need to solve this problem of predicting deaths? [1]

1.3 Modeling idea

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.

- **Ejection fraction (EF)**
  EF is a measurement which describes how much blood left ventricle pumps out of heart with each contraction. Low EF might indicate potential heart issues.

- **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.
- **Serum creatinine**
  ...
- **serum sodium**
  ...

2.2 Dataset introduction

The dataset of 299 patients was produced as a result of study [1] 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. EF, serum creatinine and platelets are categorical variables, and age, serum sodium and CPK are continuous variables.
Statistical analysis by [1] found age, creatinine, sodium, anemia and BP as significant variables.

3 Packages

Load data

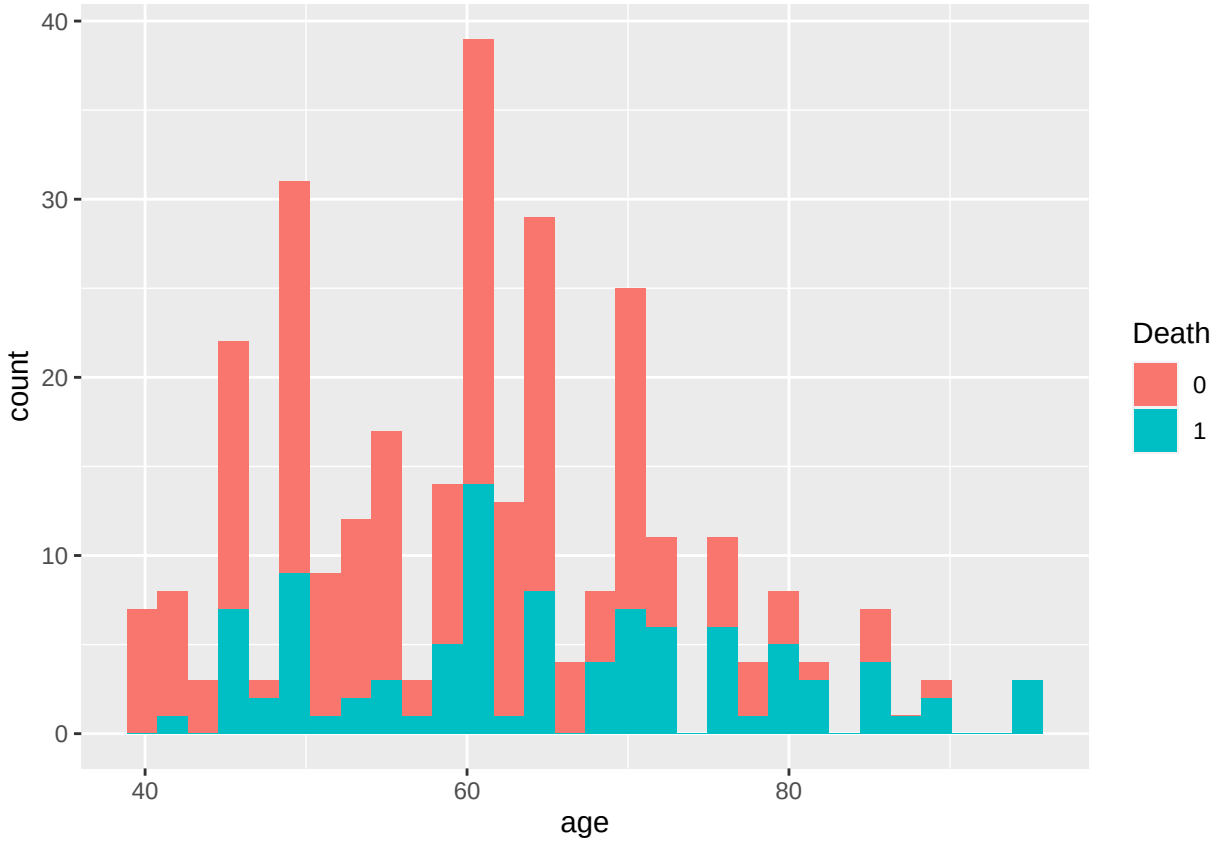
```
file.name <- './data/heart_failure_clinical_records_dataset.csv'
heart <- read_csv(file.name)
```

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

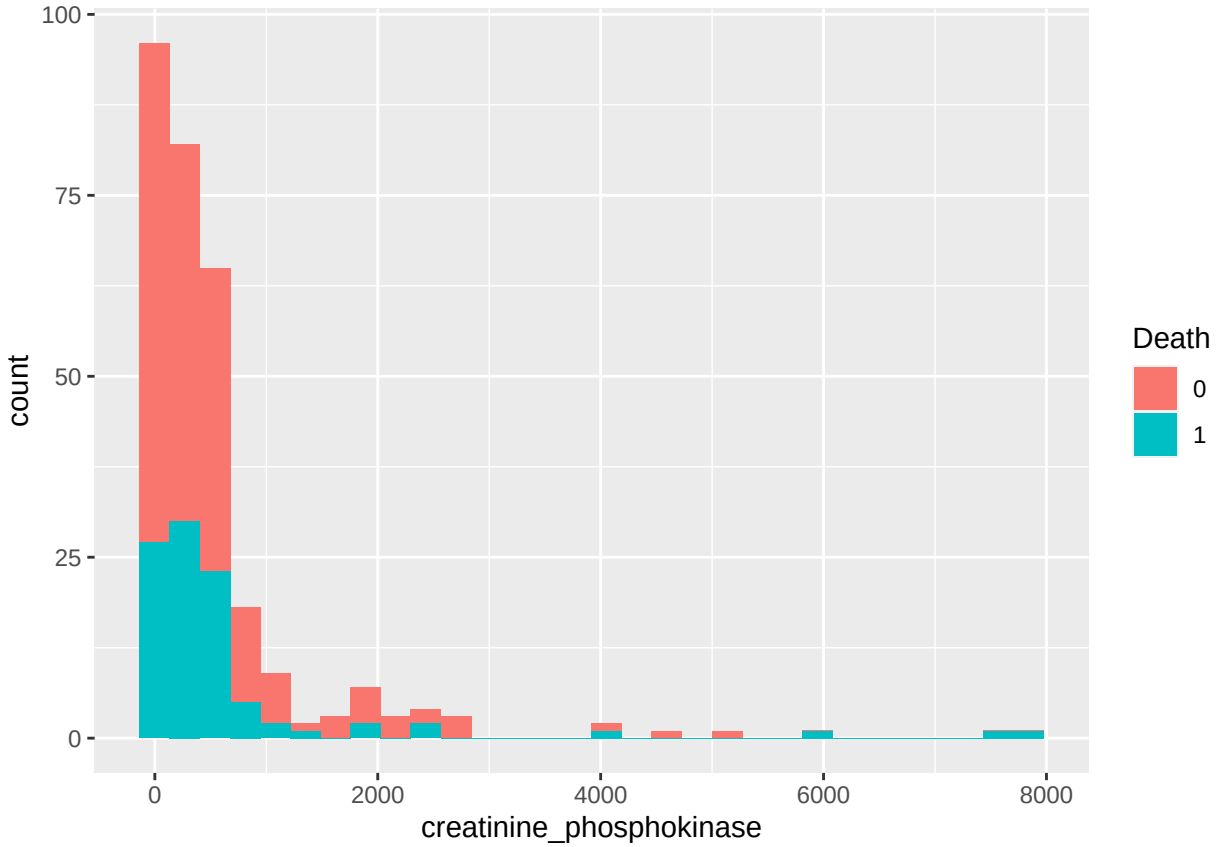
Plot histograms

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

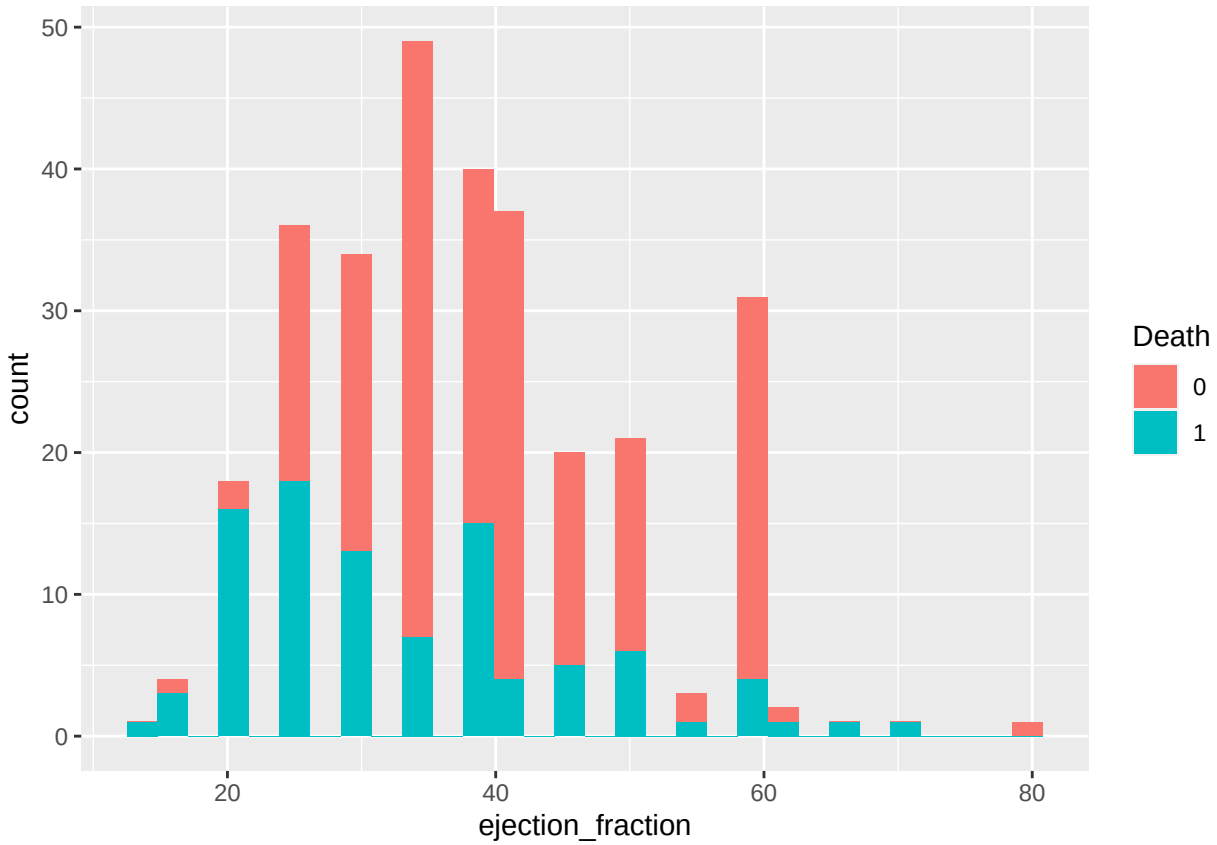
```
ggplot(heart, aes(x=age)) + geom_histogram(aes(fill=as.character(DEATH_EVENT)), bins = 30) + labs(fill =
```



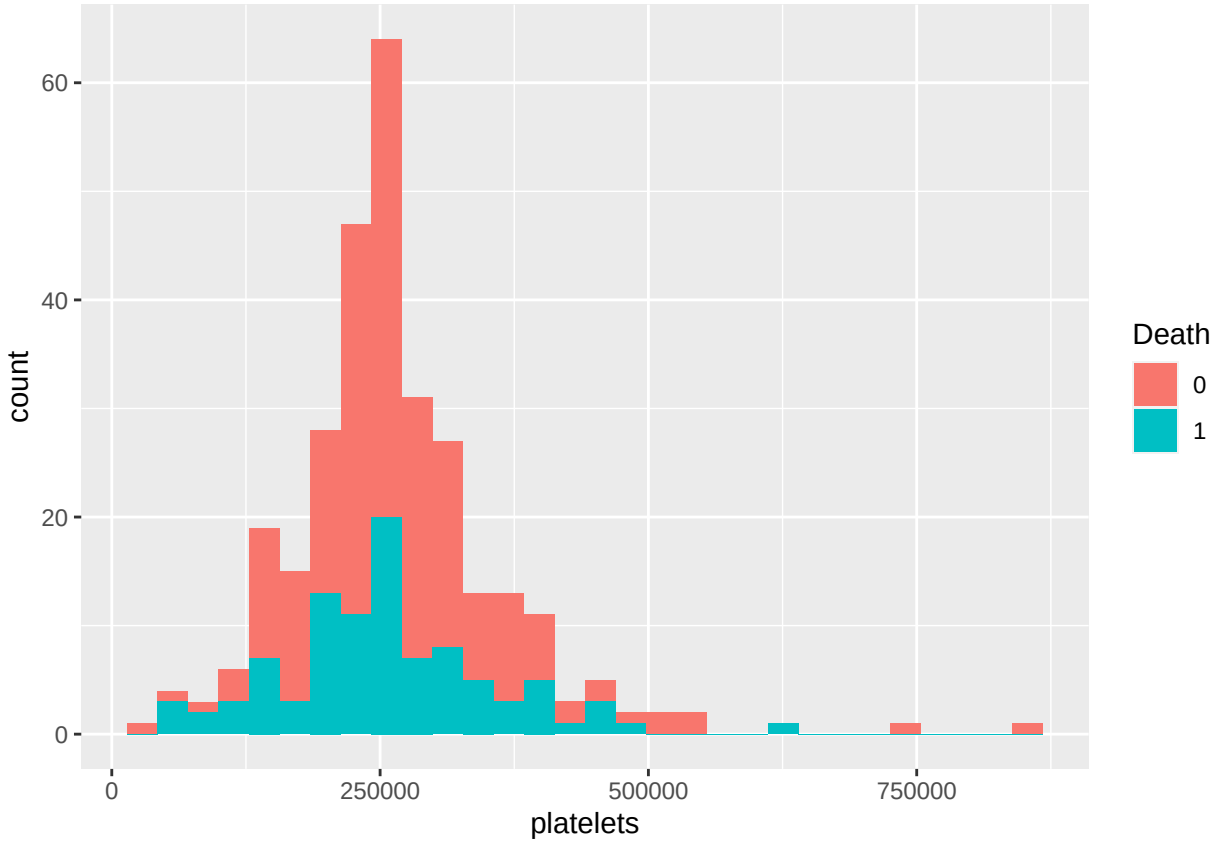

```
ggplot(heart, aes(x=creatinine_phosphokinase)) + geom_histogram(aes(fill=as.character(DEATH_EVENT)), bi
```



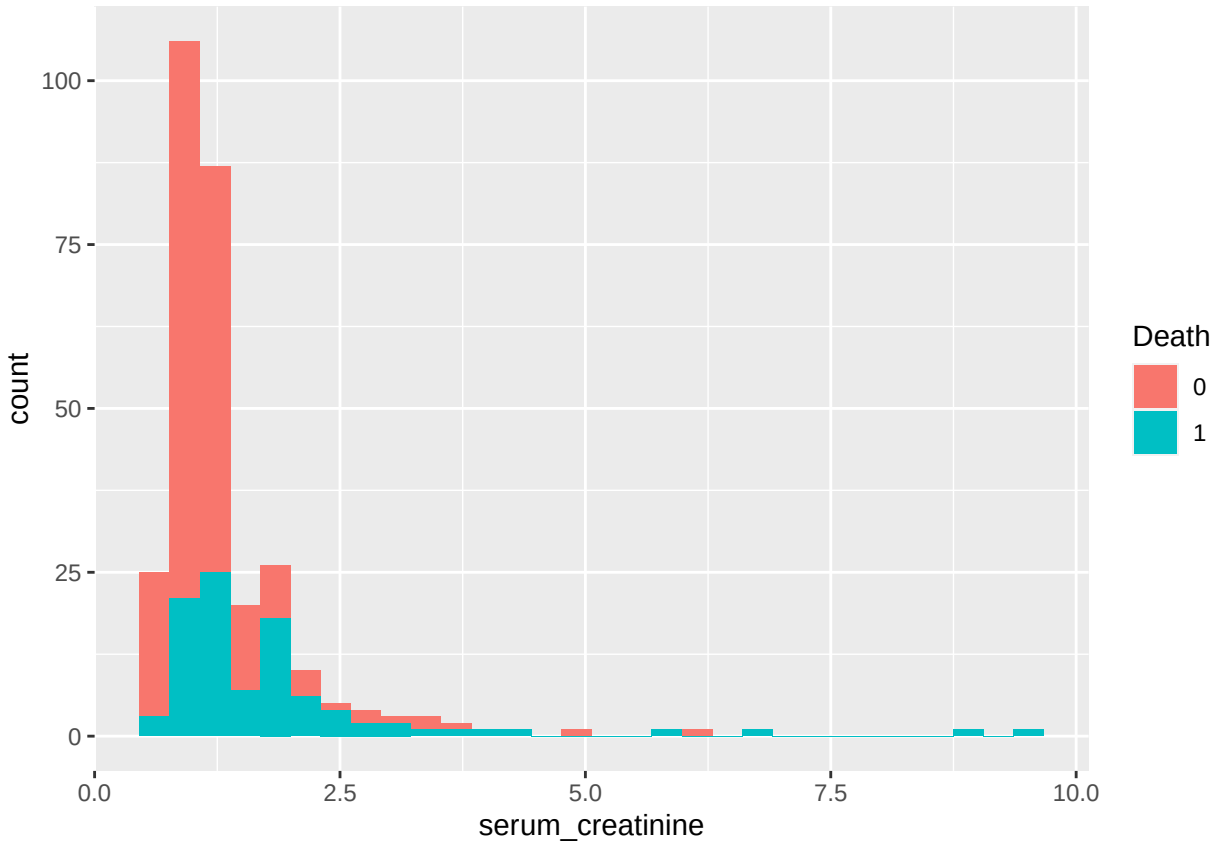

```
ggplot(heart, aes(x=ejection_fraction)) + geom_histogram(aes(fill=as.character(DEATH_EVENT)), bins = 30)
```




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

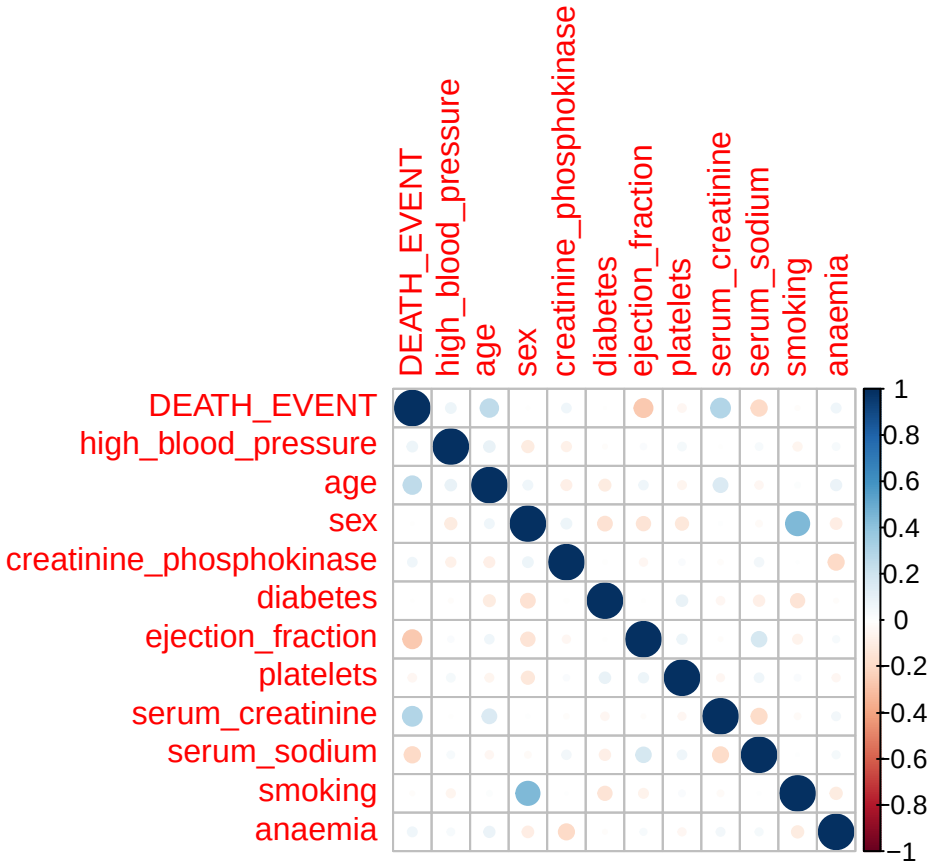



```
ggplot(heart, aes(x=serum_creatinine)) + geom_histogram(aes(fill=as.character(DEATH_EVENT)), bins = 30)
```



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)
```



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
fit <- brm(formula = DEATH_EVENT ~ age,
           data = train.data,
           family = bernoulli(),
           prior = c(set_prior("normal(50,50)", coef="age"))
           )
```

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

```
## Start sampling
```

```
##
```

```
## SAMPLING FOR MODEL '1e3e2736134c6119e65ed0f60927f367' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.074 seconds (Warm-up)
## Chain 1:                0.065 seconds (Sampling)
## Chain 1:                0.139 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '1e3e2736134c6119e65ed0f60927f367' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.074 seconds (Warm-up)
## Chain 2:                0.065 seconds (Sampling)
## Chain 2:                0.139 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '1e3e2736134c6119e65ed0f60927f367' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
```

```
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.068 seconds (Warm-up)
## Chain 3:                0.051 seconds (Sampling)
## Chain 3:                0.119 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '1e3e2736134c6119e65ed0f60927f367' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.078 seconds (Warm-up)
## Chain 4:                0.058 seconds (Sampling)
## Chain 4:                0.136 seconds (Total)
## Chain 4:
```

Analyze stan code

```
summary(fit)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: DEATH_EVENT ~ age
##    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    -3.47      0.82    -5.09    -1.96 1.00     3576     2845
## age           0.05      0.01     0.02     0.07 1.00     3755     2828
##
## 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(fit)
```

```
## // 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[1] | 50,50);
##   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

```
preds <- round(predict(fit, newdata = test.data))[1]
pred.corr <- preds == test.data$DEATH_EVENT
```

```r
acc <- length(pred.corr[pred.corr == TRUE])/nrow(test.data)
acc
```

```
## [1] 0.8111111
```

```r
#PRIORS ?
# 1 LINEAR MODEL WITH VARIABLE SELECTION
# 2 LINEAR MODEL WITH ALL VARIABLES
# HIERARCHICAL - in the Titanic one a hier. model have been used so we can use that one for reference

# All models with bernoulli outcome (1-0, death or not)
```

References

[1] 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.