Untitled

2023-04-08

```
set.seed(1)
library(rstan)
## Loading required package: StanHeaders
## Loading required package: ggplot2
## rstan (Version 2.21.8, GitRev: 2e1f913d3ca3)
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
library(Rlab)
## Rlab 4.0 attached.
## Attaching package: 'Rlab'
## The following objects are masked from 'package:stats':
##
##
       dexp, dgamma, dweibull, pexp, pgamma, pweibull, qexp, qgamma,
       qweibull, rexp, rgamma, rweibull
##
## The following object is masked from 'package:datasets':
##
##
       precip
train <- read.csv("train_new.csv")</pre>
test <- read.csv("test_new.csv")</pre>
#train
data_train \leftarrow train[-c(1,4,11)]
data_test \leftarrow test[-c(1,3,10)]
```

```
\#data\_train["SibSp"] = ceiling(data\_train["SibSp"]/10)
#data_train["Parch"] = ceiling(data_train["Parch"]/10)
data_train["Sex"] = (data_train$Sex=="male")*1
#data_test["SibSp"] = ceiling(data_test["SibSp"]/10)
#data_test["Parch"] = ceiling(data_test["Parch"]/10)
data_test["Sex"] = (data_test$Sex=="male")*1
data_train["Age"] = (data_train$Age-mean(data_train$Age))/sqrt(var(data_train$Age))
data_test["Age"] = (data_test$Age-mean(data_test$Age))/sqrt(var(data_test$Age))
data_train["family"] = data_train$Parch+data_train$SibSp
data_test["family"] = data_test$Parch+data_test$SibSp
data_train["family"] = (data_train$family-mean(data_train$family))/sqrt(var(data_train$family))
data_test["family"] = (data_test$family-mean(data_test$family))/sqrt(var(data_test$family))
data_stan <- list(</pre>
 N = 889,
 n = 417
  vive_train = data_train$Survived,
  class1_train = (data_train$Pclass==1)*1,
  class1_test = (data_test$Pclass==1)*1,
  class2_train = (data_train$Pclass==2)*1,
  class2_test = (data_test$Pclass==2)*1,
  sex_train = data_train$Sex,
  sex_test = data_test$Sex,
  #young_train = (data_train$Age<10)*1,</pre>
  #young_test = (data_test$Age<10)*1,</pre>
  \#old\_train = (data\_train\$Age>60)*1,
  \#old\_test = (data\_test\$Age>60)*1
  age_train = data_train$Age,
  age_test = data_test$Age
  #family_train = data_train$family,
  #family_test = data_test$family
fit <- stan(file = "first.stan", data = data_stan, iter = 5000, chain = 4)
## Trying to compile a simple C file
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG
                                                                                       -I"/Library/Frame
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen
## In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen
## /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/src/Cor
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/src/Cor
## namespace Eigen {
##
##
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen
## /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/Core:96
```

```
## #include <complex>
##
            ^~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
## SAMPLING FOR MODEL 'first' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000192 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.92 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:
                          1 / 5000 [ 0%]
                                            (Warmup)
## Chain 1: Iteration: 500 / 5000 [ 10%]
                                            (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]
                                            (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]
                                            (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]
                                            (Warmup)
## Chain 1: Iteration: 2500 / 5000 [ 50%]
                                            (Warmup)
## Chain 1: Iteration: 2501 / 5000 [ 50%]
                                            (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]
                                            (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]
                                            (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]
                                            (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]
                                            (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]
                                            (Sampling)
## Chain 1:
## Chain 1: Elapsed Time: 3.23787 seconds (Warm-up)
## Chain 1:
                           3.27902 seconds (Sampling)
## Chain 1:
                           6.51689 seconds (Total)
## Chain 1:
## SAMPLING FOR MODEL 'first' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000138 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.38 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:
                          1 / 5000 [ 0%]
                                            (Warmup)
## Chain 2: Iteration: 500 / 5000 [ 10%]
                                            (Warmup)
## Chain 2: Iteration: 1000 / 5000 [ 20%]
                                            (Warmup)
## Chain 2: Iteration: 1500 / 5000 [ 30%]
                                            (Warmup)
## Chain 2: Iteration: 2000 / 5000 [ 40%]
                                            (Warmup)
## Chain 2: Iteration: 2500 / 5000 [ 50%]
                                            (Warmup)
## Chain 2: Iteration: 2501 / 5000 [ 50%]
                                            (Sampling)
## Chain 2: Iteration: 3000 / 5000 [ 60%]
                                            (Sampling)
## Chain 2: Iteration: 3500 / 5000 [ 70%]
                                            (Sampling)
## Chain 2: Iteration: 4000 / 5000 [ 80%]
                                            (Sampling)
## Chain 2: Iteration: 4500 / 5000 [ 90%]
                                            (Sampling)
## Chain 2: Iteration: 5000 / 5000 [100%]
                                            (Sampling)
## Chain 2:
## Chain 2:
            Elapsed Time: 3.30282 seconds (Warm-up)
## Chain 2:
                           4.12386 seconds (Sampling)
## Chain 2:
                           7.42668 seconds (Total)
## Chain 2:
```

```
##
## SAMPLING FOR MODEL 'first' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000149 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.49 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:
                          1 / 5000 [ 0%]
                                            (Warmup)
## Chain 3: Iteration: 500 / 5000 [ 10%]
                                            (Warmup)
## Chain 3: Iteration: 1000 / 5000 [ 20%]
                                            (Warmup)
## Chain 3: Iteration: 1500 / 5000 [ 30%]
                                            (Warmup)
## Chain 3: Iteration: 2000 / 5000 [ 40%]
                                            (Warmup)
## Chain 3: Iteration: 2500 / 5000 [ 50%]
                                            (Warmup)
## Chain 3: Iteration: 2501 / 5000 [ 50%]
                                            (Sampling)
## Chain 3: Iteration: 3000 / 5000 [ 60%]
                                            (Sampling)
## Chain 3: Iteration: 3500 / 5000 [ 70%]
                                            (Sampling)
## Chain 3: Iteration: 4000 / 5000 [ 80%]
                                            (Sampling)
## Chain 3: Iteration: 4500 / 5000 [ 90%]
                                            (Sampling)
## Chain 3: Iteration: 5000 / 5000 [100%]
                                            (Sampling)
## Chain 3:
## Chain 3: Elapsed Time: 3.32905 seconds (Warm-up)
## Chain 3:
                           3.39368 seconds (Sampling)
## Chain 3:
                           6.72273 seconds (Total)
## Chain 3:
## SAMPLING FOR MODEL 'first' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000144 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.44 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:
                          1 / 5000 [ 0%]
                                            (Warmup)
## Chain 4: Iteration: 500 / 5000 [ 10%]
                                            (Warmup)
## Chain 4: Iteration: 1000 / 5000 [ 20%]
                                            (Warmup)
## Chain 4: Iteration: 1500 / 5000 [ 30%]
                                            (Warmup)
## Chain 4: Iteration: 2000 / 5000 [ 40%]
                                            (Warmup)
## Chain 4: Iteration: 2500 / 5000 [ 50%]
                                            (Warmup)
## Chain 4: Iteration: 2501 / 5000 [ 50%]
                                            (Sampling)
## Chain 4: Iteration: 3000 / 5000 [ 60%]
                                            (Sampling)
## Chain 4: Iteration: 3500 / 5000 [ 70%]
                                            (Sampling)
## Chain 4: Iteration: 4000 / 5000 [ 80%]
                                            (Sampling)
## Chain 4: Iteration: 4500 / 5000 [ 90%]
                                            (Sampling)
## Chain 4: Iteration: 5000 / 5000 [100%]
                                            (Sampling)
## Chain 4:
## Chain 4: Elapsed Time: 3.20046 seconds (Warm-up)
## Chain 4:
                           3.35466 seconds (Sampling)
## Chain 4:
                           6.55512 seconds (Total)
## Chain 4:
#print(fit)
cor(data_train[,-c(1,8)])
```

```
##
                                                  SibSp
                             Sex
                                        Age
                                                              Parch
## Pclass 1.00000000 0.12774090 -0.36528727 0.08165562 0.01682449 -0.5481933
          0.12774090 1.00000000 0.07262202 -0.11634817 -0.24750798 -0.1799575
         ## Age
## SibSp 0.08165562 -0.11634817 -0.25298934 1.00000000 0.41454164 0.1608869
## Parch 0.01682449 -0.24750798 -0.16628774 0.41454164 1.00000000 0.2175320
## Fare -0.54819329 -0.17995753 0.10821751 0.16088685 0.21753204 1.0000000
## family 0.06422053 -0.20319145 -0.25600999 0.89065367 0.78298776 0.2186582
##
              family
## Pclass 0.06422053
## Sex
         -0.20319145
         -0.25600999
## Age
## SibSp 0.89065367
## Parch
          0.78298776
## Fare
          0.21865817
## family 1.0000000
#plot(fit, pars = c("pred"))
fit_ss <- extract(fit, pars = "pred", permuted = TRUE)$pred</pre>
pred_stan <- 1:417</pre>
for(i in 1:417) {
 pred_stan[i] <- mean(fit_ss[,i])</pre>
 pred_stan[i] <- (pred_stan[i] > 0.5) * 1
 \#pred\_stan[i] \leftarrow rbern(1, pred\_stan[i])
  \#pret\_stan[i] \leftarrow rbern(1, pret\_stan[i])
}
freit <- glm(Survived ~ Pclass+Sex+Age, family=binomial(link='logit'), data=data_train)</pre>
summary(freit)
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age, family = binomial(link = "logit"),
##
      data = data train)
##
## Deviance Residuals:
##
      Min
               1Q
                    Median
                                 3Q
                                         Max
## -2.6523 -0.6482 -0.4363 0.6269
                                      2.4471
##
## Coefficients:
              Estimate Std. Error z value Pr(>|z|)
##
## (Intercept) 3.77737
                          0.32699 11.552 < 2e-16 ***
## Pclass
              -1.18690
                          0.12077 -9.828 < 2e-16 ***
              -2.61053
                          0.18663 -13.988 < 2e-16 ***
## Sex
## Age
              -0.44723
                          0.09714 -4.604 4.14e-06 ***
## ---
## Signif. codes: 0 '*** 0.001 '** 0.01 '* 0.05 '.' 0.1 ' 1
## (Dispersion parameter for binomial family taken to be 1)
##
      Null deviance: 1182.82 on 888 degrees of freedom
## Residual deviance: 804.34 on 885 degrees of freedom
```

```
## AIC: 812.34
##
## Number of Fisher Scoring iterations: 5
pred <- predict(freit, data_test, type="response")</pre>
pred <- (pred > 0.5)*1
result <- read.csv("submission.csv")</pre>
#result
real <- c(result[1:152,]$Survived,result[154:418,]$Survived)</pre>
mean(pred == real)
## [1] 0.7673861
mean(pred_stan == real)
## [1] 0.7697842
plot(fit, pars = "beta")
## ci_level: 0.8 (80% intervals)
## outer_level: 0.95 (95% intervals)
 beta[1]
 beta[2]
 beta[3]
 beta[4]
 beta[5]
                            <u>-2</u>
                                                    Ö
```

```
betas <- extract(fit, pars = "beta", permuted = TRUE)$beta
par(mfrow=c(2,3))
plot(density(betas[,1]))
plot(density(betas[,2]))
plot(density(betas[,3]))
plot(density(betas[,4]))
plot(density(betas[,5]))
#plot(density(betas[,6]))</pre>
```

density.default(x = betas[, 1])

Density

1.0

8.0

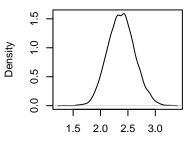
N = 10000 Bandwidth = 0.0225

0.4

0.0

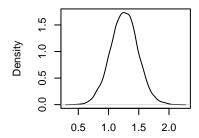
-0.4

density.default(x = betas[, 2])



N = 10000 Bandwidth = 0.03523

density.default(x = betas[, 3])



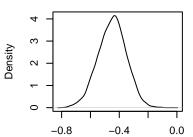
N = 10000 Bandwidth = 0.03207

density.default(x = betas[, 4])

Oensity -3.5 -3.0 -2.5 -2.0

N = 10000 Bandwidth = 0.02639

density.default(x = betas[, 5])



N = 10000 Bandwidth = 0.01404