```r
library(dplyr)
train <- read.csv("data/Train.csv")
head(train)
```

```r
# Factorize categorical variables
fac_train <- train %>% mutate_if(is.character, as.factor)
fac_train <- fac_train %>% mutate_if(is.logical, as.factor)
fac_train <- fac_train %>% mutate(ATTEND = factor(ATTEND), BFACIL = factor(BFACIL),
                                  DMAR = factor(DMAR), FEDUC = factor(FEDUC),
                                  FRACE6 = factor(FRACE6), MBSTATE_REC = factor(MBSTATE_REC),
                                  MEDUC = factor(MEDUC), MRAVE6 = factor(MRAVE6),
                                  NO_INFEC = factor(NO_INFEC),NO_MMORB = factor(NO_MMORB),
                                  NO_RISKS = factor(NO_RISKS), PAY_REC = factor(PAY_REC),
                                  PRECARE = factor(PRECARE), RDMETH_REC = factor(RDMETH_REC),
                                  RESTATUS = factor(RESTATUS))
head(fac_train)
```

```r
biggest.model = lm(DBWT ~ ., data = fac_train)
summary(biggest.model)

# Remove the columns causing singularity
fac_train <- fac_train %>% select(!c(RF_CESAR))
biggest.model = lm(DBWT ~ ., data = fac_train)
min.model = lm(DBWT ~ 1, data = fac_train)
summary(biggest.model)
```

```r
# Forward selection with BIC
forward.BIC = step(min.model, direction="forward", scope = formula(biggest.model),
                   k = log(nrow(fac_train)), trace = 0)
forward.BIC$anova
```

```r
# Backward selection with BIC
backward.BIC = step(biggest.model, direction="backward",
                  k = log(nrow(fac_train)), trace = 0)
backward.BIC
```

```r
# Forward selection with AIC
forward.AIC = step(min.model, direction="forward", scope = formula(biggest.model),
                  k = 2, trace = 0)
forward.AIC
```

```r
# Backward selection with AIC
backward.AIC = step(biggest.model, direction="backward",
                  k = 2, trace = 0)
backward.AIC
```

```r
# Compute the leave-one-out cross-validation errors
for_AIC.cv = mean((residuals(forward.AIC) / (1 - hatvalues(forward.AIC))) ^ 2)
back_AIC.cv = mean((residuals(backward.AIC) / (1 - hatvalues(backward.AIC))) ^ 2)
for_BIC.cv = mean((residuals(forward.BIC) / (1 - hatvalues(forward.BIC))) ^ 2)
back_BIC.cv = mean((residuals(backward.BIC) / (1 - hatvalues(backward.BIC))) ^ 2)
which.min(c(for_AIC.cv, back_AIC.cv, for_BIC.cv, back_BIC.cv))
```

```
backward.AIC$model
forward.AIC$model
```

```
# Add interaction terms by F-test
```