```r
## Cross-validation with model selection

# Dataset from paper "Where does Haydn end and Mozart begin?
# Composer classification of string quartets"
# (Journal of New Music Research, vol 49, 457-476)
HM <- read.csv("haydn-mozart.csv")
HM[, 1] <- NULL # first col is just name of quartet, remove it
dim(HM) # 285 observations, 1116 columns

# Let's treat "number of notes in violin part" as the response
# That's variable name "count_pitch_1" and column 683 of data matrix
# So we have 1115 possible predictors

# More model selection, for clarity start with one train/validation split
N <- nrow(HM)
set.seed(12345678)
trainInd <- sample(1:N, round(N * 0.8), replace = F)
trainSet <- HM[trainInd, ]
validSet <- HM[-trainInd, ]

library(MASS)
# Full model and empty model with just intercept
full <- lm(count_pitch_1 ~ ., data = trainSet)
empty <- lm(count_pitch_1 ~ 1, data = trainSet)

# Stepwise forward with BIC
stepAIC(
  object = empty,
  scope = list(upper = full, lower = empty),
  direction = "forward",
  k = log(nrow(trainSet))
)
m1 <-
  lm(
    formula = count_pitch_1 ~ Prop_m3_num_0_8.1 + count_pitch_3 +
      Prop_m3_num_0_8.3 + Prop_m3_mean_8.1 + count_pitch_4 + Dev_count_8_thresh4.393.1 +
      Prop_m3_mean_8.3 + mean_time_3 + Dev_count_t_14_thresh0.216.3 +
      voicepair_int_dist_6_1.2 + Prop_m3_sd_8.3 + Prop_m3_sd_8.1 +
      Prop_m3_num_.6_8.1 + simult_rest_perc + Dev_perc_t_14.1 +
      count_pitch_2 + Prop_m3_num_0_8.2 + Prop_m3_mean_8.2 + Dev_count_8_thresh4.024.2 +
      Dev_count_18_thresh3.899.2 + Expo_t_count_14.thresh0.7.1 +
      Prop_m3_num_0_12.1 + Expo_acc_8.1 + Expo_perc_8.2 + Prop_m3_num_0_12.2 +
      Dev_count_t_8_thresh0.247.1 + Expo_t_count_8.thresh0.7.1 +
      voicepair_int_dist_1_1.3 + Expo_perc_12.3 + Pairwise_voice_int_mean.1.3 +
      Expo_t_count_18.thresh0.7.3 + Prop_m3_q3_16.2 + Dev_perc_t_14.4 +
      Prop_m3_q3_14.4 + Dev_count_t_14_thresh0.187.3,
    data = trainSet
  )

# we can use the AIC function with our own k for the L0 penalty
AIC(m1, k = log(nrow(trainSet)))
BIC(m1) # in this case matches BIC as we expect (1977.6)
```

```
pred1 <- predict(m1, newdata = validSet)
sqrt(mean((validSet$count_pitch_1 - pred1) ^ 2)) # RMSE on validation
sqrt(mean(m1$residuals ^ 2)) # RMSE on train

# Try ICM to search for a model with a potentially better BIC
# than the one found with stepwise
pen <- log(nrow(trainSet)) #
varlist = c()
varnames = names(trainSet)
n = nrow(trainSet)
varorder <- sample(1:ncol(trainSet)) # random order of variables
minCrit = Inf
noChange = F
while (!noChange) {
  noChange = T
  for (i in varorder) {
    if (i == 683)
      next

    if (i %in% varlist & length(varlist) > 1) {
      index = c(683, varlist[varlist != i])
      trainVars = trainSet[, index]

      fit = lm(count_pitch_1 ~ ., data = trainVars)

      if (AIC(fit, k = pen) < minCrit) {
        minCrit = AIC(fit, k = pen)
        varlist = varlist[varlist != i]
        print(paste0(
          "Criterion: ",
          round(minCrit, 1),
          ", variables: ",
          paste0(varnames[varlist], collapse = " ")
        ))
        best.model = fit
        noChange = F
      }

    } else if (!i %in% varlist) {
      index = c(683, varlist, i)
      trainVars = trainSet[, index]

      fit = lm(count_pitch_1 ~ ., data = trainVars)

      if (AIC(fit, k = pen) < minCrit) {
        minCrit = AIC(fit, k = pen)
        varlist = c(varlist, i)
        print(paste0(
          "Criterion: ",
          round(minCrit, 1),
          ", variables: ",
          paste0(varnames[varlist], collapse = " ")
        ))
```

```
        best.model = fit
        noChange = F
      }
    }
  }
}

summary(best.model)
predICM <- predict(best.model, newdata = validSet)
sqrt(mean((validSet$count_pitch_1 - predICM) ^ 2)) # RMSE on validation
sqrt(mean(best.model$residuals ^ 2)) # RMSE on train


# Try stepwise again, with a larger L0 penalty (e.g., twice the usual BIC penalty)
stepAIC(
  object = empty,
  scope = list(upper = full, lower = empty),
  direction = "forward",
  k = 2 * log(nrow(trainSet))
)
m2 <-
  lm(
    formula = count_pitch_1 ~ Prop_m3_num_0_8.1 + count_pitch_3 +
      Prop_m3_num_0_8.3 + Prop_m3_mean_8.1 + count_pitch_4 + Dev_count_8_thresh4.393.1 +
      Prop_m3_mean_8.3 + mean_time_3 + Dev_count_t_14_thresh0.216.3,
    data = trainSet
  )
AIC(m2, k = 2 * log(nrow(trainSet)))
# calculate the value of criterion based on this larger L0 penalty
pred2 <- predict(m2, newdata = validSet)
sqrt(mean((validSet$count_pitch_1 - pred2) ^ 2)) # RMSE on validation
sqrt(mean(m2$residuals ^ 2)) # RMSE on train

# Try ICM as well with this penalty
pen <- 2 * log(nrow(trainSet))
varlist = c()
varnames = names(trainSet)
n = nrow(trainSet)
varorder <- sample(1:ncol(trainSet))
minCrit = Inf
noChange = F
while (!noChange) {
  noChange = T
  for (i in varorder) {
    if (i == 683)
      next

    if (i %in% varlist & length(varlist) > 1) {
      index = c(683, varlist[varlist != i])
      trainVars = trainSet[, index]

      fit = lm(count_pitch_1 ~ ., data = trainVars)
```

```
      if (AIC(fit, k = pen) < minCrit) {
        minCrit = AIC(fit, k = pen)
        varlist = varlist[varlist != i]
        print(paste0(
          "Criterion: ",
          round(minCrit, 1),
          ", variables: ",
          paste0(varnames[varlist], collapse = " ")
        ))
        best.model = fit
        noChange = F
      }

    } else if (!i %in% varlist) {
      index = c(683, varlist, i)
      trainVars = trainSet[, index]

      fit = lm(count_pitch_1 ~ ., data = trainVars)

      if (AIC(fit, k = pen) < minCrit) {
        minCrit = AIC(fit, k = pen)
        varlist = c(varlist, i)
        print(paste0(
          "Criterion: ",
          round(minCrit, 1),
          ", variables: ",
          paste0(varnames[varlist], collapse = " ")
        ))
        best.model = fit
        noChange = F
      }
    }
  }
}

predICM <- predict(best.model, newdata = validSet)
sqrt(mean((validSet$count_pitch_1 - predICM) ^ 2)) # RMSE on validation
sqrt(mean(best.model$residuals ^ 2)) # RMSE on train


# K fold cross validation to choose model selection method
K <- 5
validSetSplits <- sample((1:N) %% K + 1)
RMSE1 <- c()
RMSE2 <- c()
for (k in 1:K) {
  validSet <- HM[validSetSplits == k, ]
  trainSet <- HM[validSetSplits != k, ]

  full <- lm(count_pitch_1 ~ ., data = trainSet)
  empty <- lm(count_pitch_1 ~ 1, data = trainSet)

  m1 <-
```

```
    stepAIC(
      object = empty,
      scope = list(upper = full, lower = empty),
      direction = "forward",
      k = log(nrow(trainSet))
    )
  pred1 <- predict(m1, newdata = validSet)
  RMSE1[k] <- sqrt(mean((validSet$count_pitch_1 - pred1) ^ 2))

  m2 <-
    stepAIC(
      object = empty,
      scope = list(upper = full, lower = empty),
      direction = "forward",
      k = 2 * log(nrow(trainSet))
    )
  pred2 <- predict(m2, newdata = validSet)
  RMSE2[k] <- sqrt(mean((validSet$count_pitch_1 - pred2) ^ 2))

}

RMSE1
RMSE2
mean(RMSE1)
mean(RMSE2)

# turns out m2 is indeed the better procedure among these two based on CV prediction error
# if we decide on procedure m2, we can apply procedure m2 to the entire 285 observations
# to get a final model for future prediction
# e.g.,
full <- lm(count_pitch_1 ~ ., data = HM)
empty <- lm(count_pitch_1 ~ 1, data = HM)

mfinal <-
  stepAIC(
    object = empty,
    scope = list(upper = full, lower = empty),
    direction = "forward",
    k = 2 * log(nrow(trainSet))
  )
```