Cryoocyte Test Project: Linear Model Selection Part

Ming Chen

Import Libraries

```
library(ggplot2)   # data visualization
library(tidyverse) # a set of module for data manipulation
library(leaps)
library(gridExtra)
set.seed(123)
```

Import Data

```
my_data = read_csv('data/train.csv')
my_test_data = read_csv('data/test.csv')
```

Linear Model selection

The data is being randomly split into training and testing sets in the 70/30 ratio. The forward selection method is implemented to determine the best final model. In the intial round of machine learning analysis, the full (with all 100 variables included) linear model has an R2 score up to **0.999**. In this section, I aim to find a simpler liner model which has an R2 score larger than **0.995** but use less features.

Forward selection

Backward selection generated similar results. Therefore, only forward selection analysis is presented here.

```
n = 20 # 20x cross valiation: train/test = 70/30

#-------first run-------
index = sample(1:nrow(my_data), size = round(0.7*nrow(my_data)))
train = my_data[index, ]
test = my_data[-index, ]
regfit.fwd = regsubsets(y~., data = train, nvmax = 1000, method = "forward")
regfit.fwd.summary = summary(regfit.fwd)

# save train and test errors
best_num_of_variables = head(which(regfit.fwd.summary$adjr2 > 0.995), 1)
best_variables = paste0(names(coef(regfit.fwd, best_num_of_variables))[-1], collapse=',')
best_models = data.frame(best_num_of_variables, best_variables, stringsAsFactors = FALSE)
selected_columns = c(names(coef(regfit.fwd, best_num_of_variables)[-1]), 'y')
lm.fit = lm(y~., data = train[selected_columns])
train_pred = predict(lm.fit, newdata = train[selected_columns])
test_pred = predict(lm.fit, newdata = test[selected_columns])
train_error = (train$y - train_pred)^2
test_error = (test$y - test_pred)^2
errors_df = data.frame(run_01=c(train_error, test_error))
```

```
# save performance metrics
RSS_df = data.frame(run_01 = regfit.fwd.summary$rss)
RSq_df = data.frame(run_01 = regfit.fwd.summary$adjr2)
Cp_df = data.frame(run_01 = regfit.fwd.summary$cp)
BIC_df = data.frame(run_01 = regfit.fwd.summary$bic)

#-------second to n runs------
for (i in 2:n) {
  index = sample(1:nrow(my_data), size = round(0.7*nrow(my_data)))
  train = my_data[index, ]
  test = my_data[-index, ]
  regfit.fwd = regsubsets(y~., data = train, nvmax = 1000, method = "forward")
  regfit.fwd.summary = summary(regfit.fwd)
  if (i < 10) {
    col_name = paste0('run_0', i)
  } else {
    col_name = paste0('run_', i)
  }

  # save train and test errors
  best_num_of_variables = head(which(regfit.fwd.summary$adjr2 > 0.995), 1)
  best_variables = paste0(names(coef(regfit.fwd, best_num_of_variables))[-1], collapse=',')
  best_models[i, ] = c(best_num_of_variables, best_variables)
  selected_columns = c(names(coef(regfit.fwd, best_num_of_variables)[-1]), 'y')
  lm.fit = lm(y~., data = train[selected_columns])
  train_pred = predict(lm.fit, newdata = train[selected_columns])
  test_pred = predict(lm.fit, newdata = test[selected_columns])
  train_error = (train$y - train_pred)^2
  test_error = (test$y - test_pred)^2
  errors_df[col_name] = data.frame(run_1=c(train_error, test_error))

  # save performance metrics
  RSS_df[col_name] = regfit.fwd.summary$rss
  RSq_df[col_name] = regfit.fwd.summary$adjr2
  Cp_df[col_name] = regfit.fwd.summary$cp
  BIC_df[col_name] = regfit.fwd.summary$bic
}
```

Determine which variables should be kept in the final linear model

After randomly spliting the data and implementing the forward selection 20 times, we can determine what's the best number of variables to keep and which variables they are if we want to have an R2 score up to 0.995.

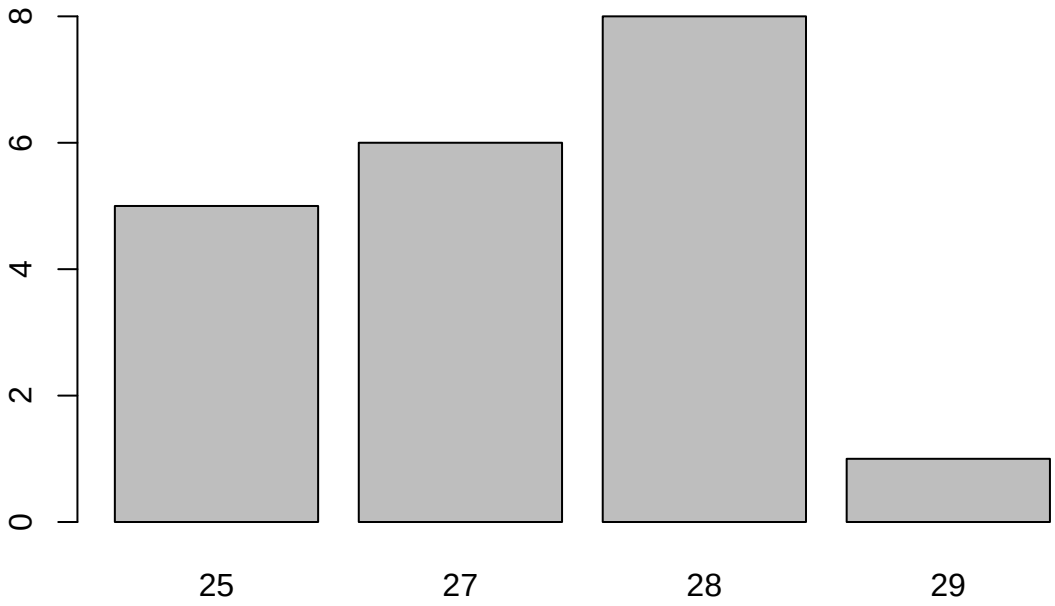
The best of number of features to keep

```
# frequency of best number of variables
table(best_models$best_num_of_variables)
```

```
##
## 25 27 28 29
```

```
##  5  6  8  1
```

```
barplot(table(best_models$best_num_of_variables))
```



The most important features

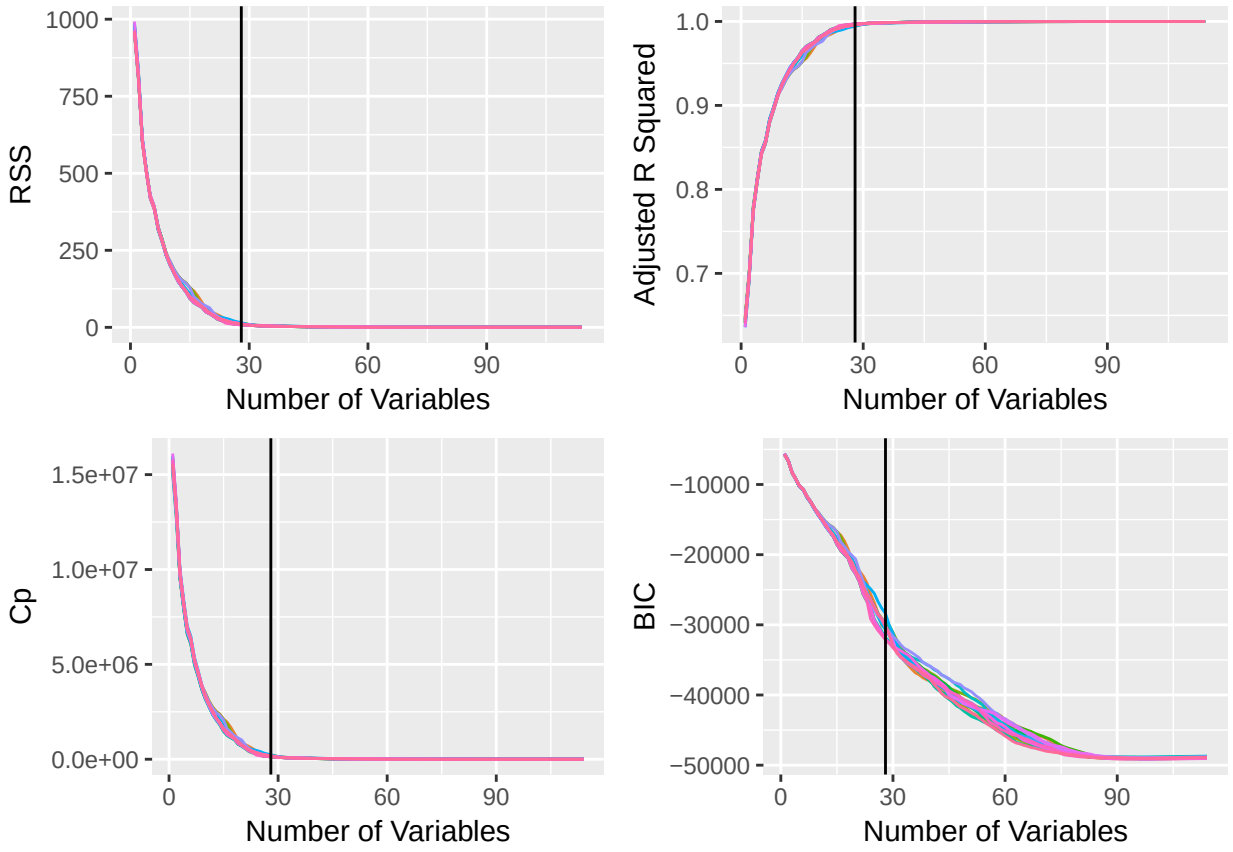
```
# most important variables
highest_best_num = as.integer(names(which.max(table(best_models$best_num_of_variables))))
most_important_features = sort(table(
  strsplit(paste0(best_models$best_variables, collapse=','), ',')[[1]]),
     decreasing = TRUE)[1:highest_best_num]
write(paste0(names(most_important_features), collapse = ','),
      "results/most_important_features.csv")
most_important_features
```

```
##
##  x1 x15 x19  x2 x25 x29 x31 x42  x5 x59 x62 x72 x74 x76 x84 x87 x98 x99
##  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20
## x28 x78  x3 x55  x7 x53 x14 x69 x79 x80
##  18  17  16  16  15  14  11  11  11  11
```

The relationship between the number of variables and the metrics of model performance

```
# RSS
gather_RSS_df = gather(RSS_df)
gather_RSS_df['x'] = rep(1:nrow(RSS_df), n)
p1 = ggplot(data = gather_RSS_df, aes(x=x, y=value, color=key)) +
  geom_line() +
  geom_vline(xintercept = highest_best_num) +
  xlab("Number of Variables") +
```

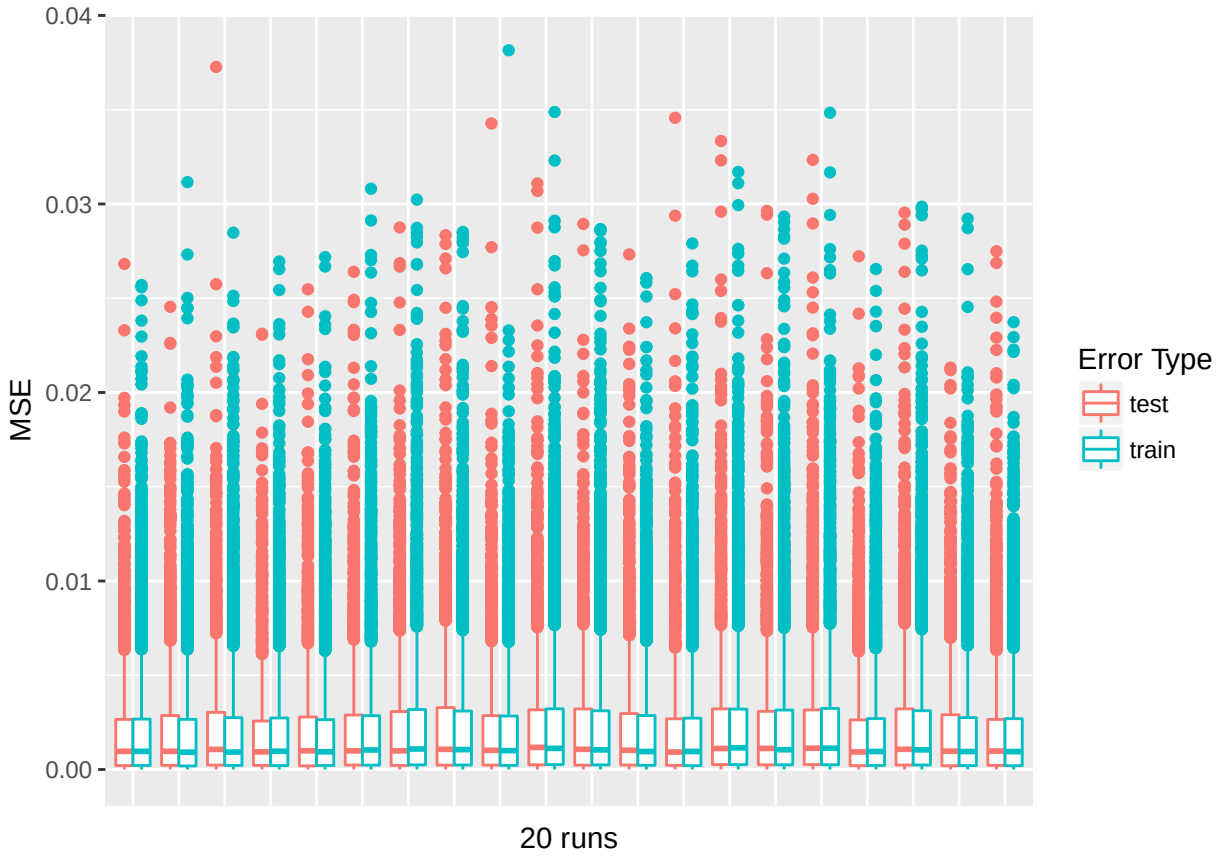
```
  ylab("RSS") +
  theme(legend.position="none")


# RSq
gather_RSq_df = gather(RSq_df)
gather_RSq_df['x'] = rep(1:nrow(RSq_df), n)
p2 = ggplot(data = gather_RSq_df, aes(x=x, y=value, color=key)) +
  geom_line() +
  geom_vline(xintercept = highest_best_num) +
  xlab("Number of Variables") +
  ylab("Adjusted R Squared") +
  theme(legend.position="none")

# grid.arrange(p1, p2, ncol=2)

# Cp
gather_Cp_df = gather(Cp_df)
gather_Cp_df['x'] = rep(1:nrow(Cp_df), n)
p3 = ggplot(data = gather_Cp_df, aes(x=x, y=value, color=key)) +
  geom_line() +
  geom_vline(xintercept = highest_best_num) +
  xlab("Number of Variables") +
  ylab("Cp") +
  theme(legend.position="none")

# BIC
gather_BIC_df = gather(BIC_df)
gather_BIC_df['x'] = rep(1:nrow(BIC_df), n)
p4 = ggplot(data = gather_BIC_df, aes(x=x, y=value, color=key)) +
  geom_line() +
  geom_vline(xintercept = highest_best_num) +
  xlab("Number of Variables") +
  ylab("BIC") +
  theme(legend.position="none")

grid.arrange(p1, p2, p3, p4, nrow=2, ncol=2)
```



Training vs. Test errors

The plot below shows that the training and test errors are very low and pretty close to each other across all 20 runs, which means that model fits both the training and test data sets well.

```
# Erros DataFrame
errors_df$error_type = c(rep('train', nrow(train)), rep('test', nrow(test)))
gather_errors_df = gather(errors_df, key = "key", value = "value",
                          setdiff(names(errors_df), 'error_type'))

ggplot(data = gather_errors_df, aes(x = key, y = value, color=error_type)) +
  geom_boxplot() +
  xlab(paste0(n, ' runs')) +
  ylab('MSE') +
  scale_color_discrete(name = "Error Type") +
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
```


0.04
0.03
0.02
0.01
0.00
MSE
20 runs
Error Type
test
train