Module 3: Semi-supervised learning (PheCAP)

Siyue Yang, Jianhui Gao, and Jesse Gronsbell

```
# Load the packages.
packages <- c("tidyverse", "PheCAP", "glmnet", "glmpath", "pROC")

# Check if the pacakages are missing or not.
# If missing, install automatically.
# If not missing, load the package.
package.check <- lapply(
      packages,
      FUN = function(x) {
        if (!require(x, character.only = TRUE)) {
          install.packages(x, dependencies = TRUE)
          library(x, character.only = TRUE)
        }
      }
    )
```

```
# Load helper functions.
source("../Rscripts/helper_function.R")
source("../Rscripts/ex2_helper_function.R")
```

```
load('../data/CAD_norm_pub.rda')
```

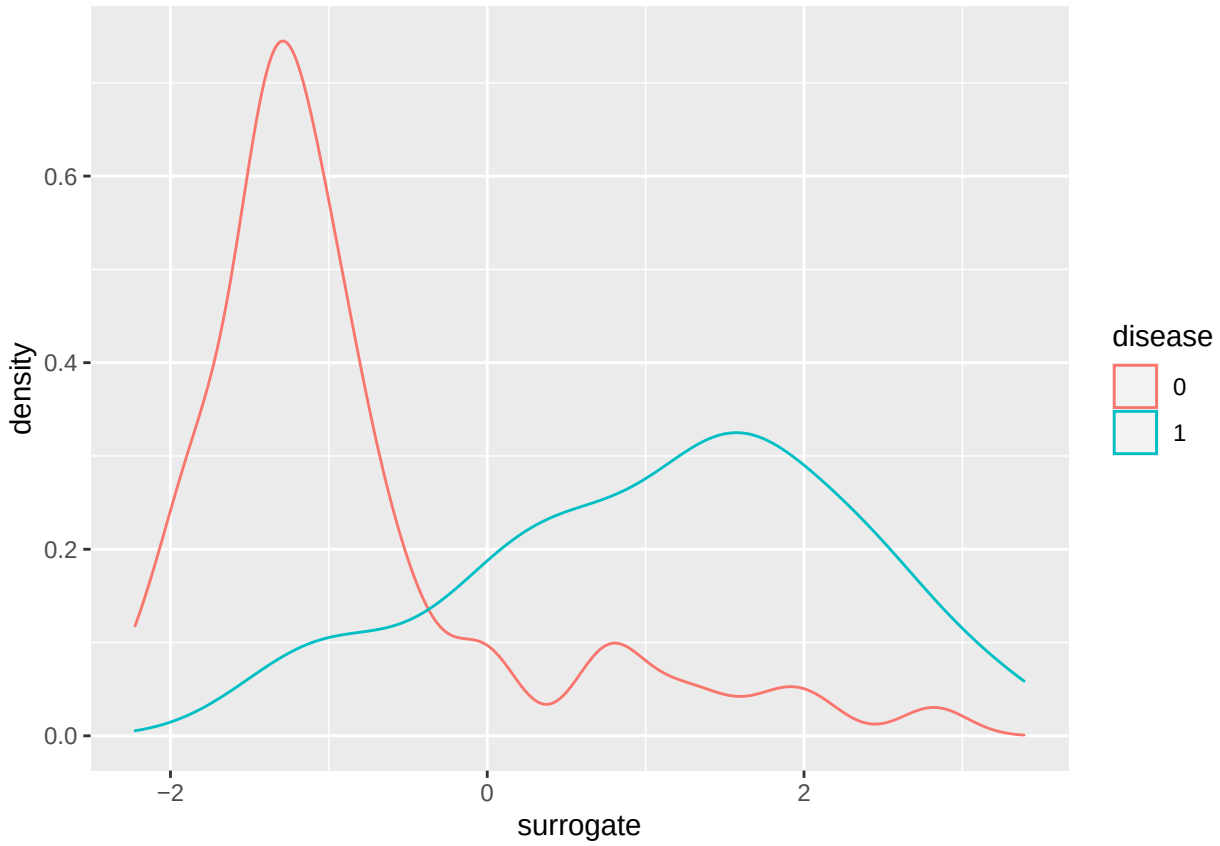
Feature selection

How to select features?

Can leverage some clinical-meaningful features that are related to Y.

e.g. Feature "surrogate" = the total number of the disease-related billing codes + disease-specific NLP mentions.

```
cbind(label = y, x) %>%
  filter(!is.na(label)) %>%
  mutate(disease = factor(label)) %>%
  ggplot(aes(x = surrogate)) +
  geom_density(aes(color = disease))
```



The more the disease-related codes, the more **likely** the patient has the disease.

```
nonmissing_index <- which(!is.na(y))
surrogate <- x$surrogate

get_auc(y[nonmissing_index], surrogate[nonmissing_index])
```

```
## [1] 0.8877745
```

We call these highly predictive features of the true disease status "surrogates".

Opportunities of using surrogate features

1. Feature selection to reduce p
2. Algorithm development with limited Y
3. Algorithm validation with limited Y

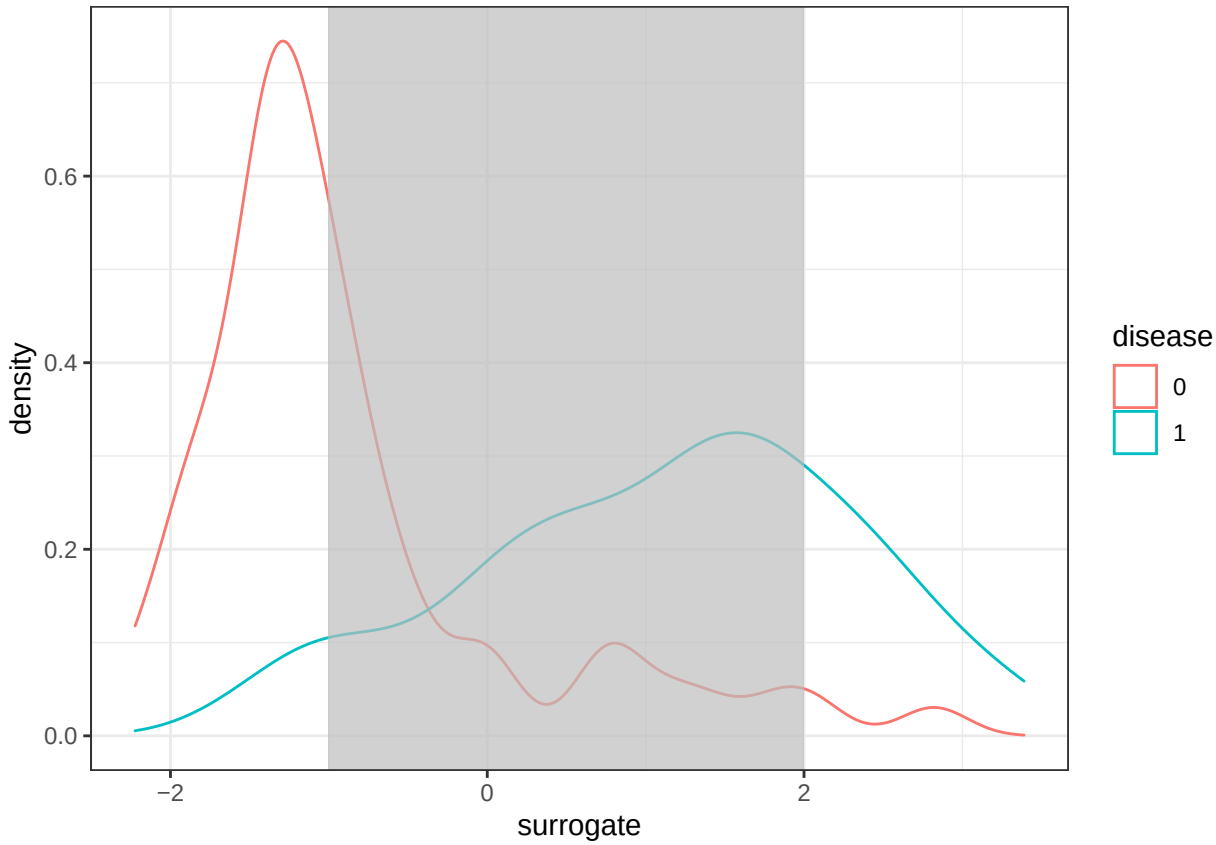
Opportunity 2 and 3 will be covered in the next module!

Feature selection method

Motivation (Extreme assumption):

- Patients with **high** main ICD or NLP mentions generally have the phenotype.
- Patients with **extremely** low counts are unlikely to have the phenotype.

```
cbind(label = y, x) %>%
  filter(!is.na(label)) %>%
  mutate(disease = factor(label)) %>%
  ggplot(aes(x = surrogate)) +
  geom_density(aes(color = disease)) +
  theme_bw() +
  annotate("rect", fill = "grey", alpha = 0.7, xmin = -1, xmax = 2,
           ymin = -Inf, ymax = Inf)
```



- Left white rect: patients not having the disease.
- Right white rect: patients having the disease.

Prepare data for feature selection

Prepare surrogates

Surrogates are available for all the patients!

```
# Prepare 3 surrogates.
surrogate <- x$surrogate

# Prepare features to be selected.
features <- data.matrix(x %>% select(starts_with("COD") | starts_with("NLP")))
```

Run surrogate-assisted feature extraction (SAFE) and show result.

```
# Truncated at 2 and -1.
SAFE <- extreme_method(surrogate, features, u_bound = 2, l_bound = -1)
SAFE_feature <- colnames(features)[SAFE$beta_select]
SAFE_feature
```

```
##  [1] "NLP56"  "NLP93"  "NLP160" "NLP161" "NLP176" "NLP231" "NLP304" "NLP306"
##  [9] "NLP309" "NLP321" "NLP349" "NLP403" "NLP434" "NLP446" "NLP456" "NLP495"
```

We select features that occur 50% among the three different surrogate-selected feature sets. This is the idea of *majority voting*.

Train phenotyping model and show the AUC on the testing set.

- Split data into training and testing set
- Training 60% (n = 106), Testing 40% (n = 75)

```
# Split index.
set.seed(1234)
training_set <- sample(nonmissing_index, size = 106, replace = FALSE)
testing_set <- setdiff(nonmissing_index, training_set)

# Training set.
train_x <- as.matrix(x[training_set, ])
train_y <- y[training_set]

# Testing set.
test_x <- as.matrix(x[testing_set, ])
test_y <- y[testing_set]
```

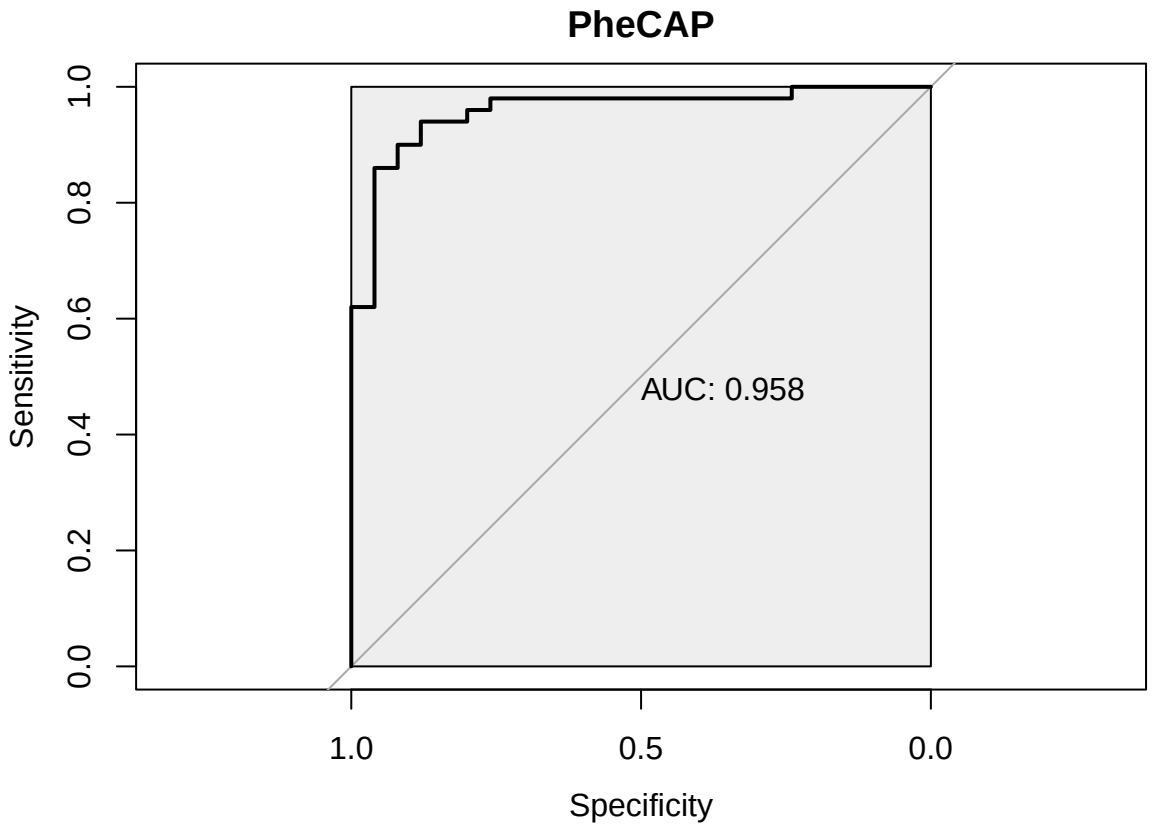
```
selected_features <- c("surrogate", "healthcare_utilization", SAFE_feature)

modelssl_lr <- fit_alasso_bic(
  y = train_y,
  x = train_x[, selected_features]
)

newx <- cbind(1, test_x[, selected_features])

# Prediction on testing set.
y_hat <- expit(newx %*% modelssl_lr$beta_hat) # Inverse Logit

plot(roc(test_y, y_hat),
  print.auc = TRUE,
  max.auc.polygon = TRUE, main = "PheCAP"
)
```




```
head(get_roc(test_y, y_hat))
```

```
##         cutoff    pos.rate  FPR       TPR       PPV       NPV        F1
## [1,] 0.9940930 0.006666667 0.00 0.3003125 1.0000000 0.4167752 0.4619082
## [2,] 0.9526009 0.293333333 0.00 0.4601562 1.0000000 0.4808415 0.6302836
## [3,] 0.9111088 0.420000000 0.02 0.6200000 0.9841270 0.5632184 0.7607362
## [4,] 0.9023323 0.426666667 0.04 0.6800000 0.9714286 0.6000000 0.8000000
## [5,] 0.8935558 0.433333333 0.04 0.7400000 0.9736842 0.6486486 0.8409091
## [6,] 0.8326341 0.493333333 0.04 0.8000000 0.9756098 0.7058824 0.8791209
```

```
test_auc <- c()
for (i in round(seq(0.2, 0.7, 0.1) * length(nonmissing_index))) {
  set.seed(123456)
  train_idx <- sample(nonmissing_index, i)
  test_idx <- setdiff(nonmissing_index, train_idx)
  train_data <- data.frame(label = y[train_idx], x[train_idx, ])
  test_data <- data.frame(label = y[test_idx], x[test_idx, ])

  train_data <- as.matrix(train_data)
  test_data <- as.matrix(test_data)

  # LASSO
  metric <- validate_model(
    train_y = train_data$label,
    test_y = test_data$label,
    test_y_hat = lasso_pred(train_data, test_data),
```

```
    train_y_hat = lasso_pred(train_data, train_data)
  )

  # Formatting
  test_auc <- rbind(test_auc, data.frame(
    n_training = i,
    method = "LASSO",
    median = metric$test_AUC,
    L = as.numeric(sub(".*?(\\d+\\.\\d+).*", "\\1", metric$test_CI)),
    U = as.numeric(sub(".*\\b(\\d+\\.\\d+).*", "\\1", metric$test_CI))
  ))

  # ALASSO
  metric <- validate_model(
    train_y = train_data$label,
    test_y = test_data$label,
    test_y_hat = alasso_pred(train_data, test_data),
    train_y_hat = alasso_pred(train_data, train_data)
  )

  test_auc <- rbind(test_auc, data.frame(
    n_training = i,
    method = "ALASSO",
    median = metric$test_AUC,
    L = as.numeric(sub(".*?(\\d+\\.\\d+).*", "\\1", metric$test_CI)),
    U = as.numeric(sub(".*\\b(\\d+\\.\\d+).*", "\\1", metric$test_CI))
  ))

  # PheCAP
  metric <- validate_model(
    train_y = train_data$label,
    test_y = test_data$label,
    test_y_hat = phe_pred(train_data, test_data),
    train_y_hat = phe_pred(train_data, train_data)
  )

  test_auc <- rbind(test_auc, data.frame(
    n_training = i,
    method = "PheCAP",
    median = metric$test_AUC,
    L = as.numeric(sub(".*?(\\d+\\.\\d+).*", "\\1", metric$test_CI)),
    U = as.numeric(sub(".*\\b(\\d+\\.\\d+).*", "\\1", metric$test_CI))
  ))
}

# Facet Plot
test_auc %>% ggplot(aes(
  x = n_training, y = median,
  group = method, color = method
)) +
  geom_point() +
  geom_line() +
  geom_errorbar(aes(ymin = L, ymax = U)) +
```

```
  facet_grid(. ~ method)
```

Save the data and feature selected for module 4 and model fitting.

```
save(list = ls(), file = "../module4/environment.RData")
```