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")
```

```
load('environment_pass.RData')
```

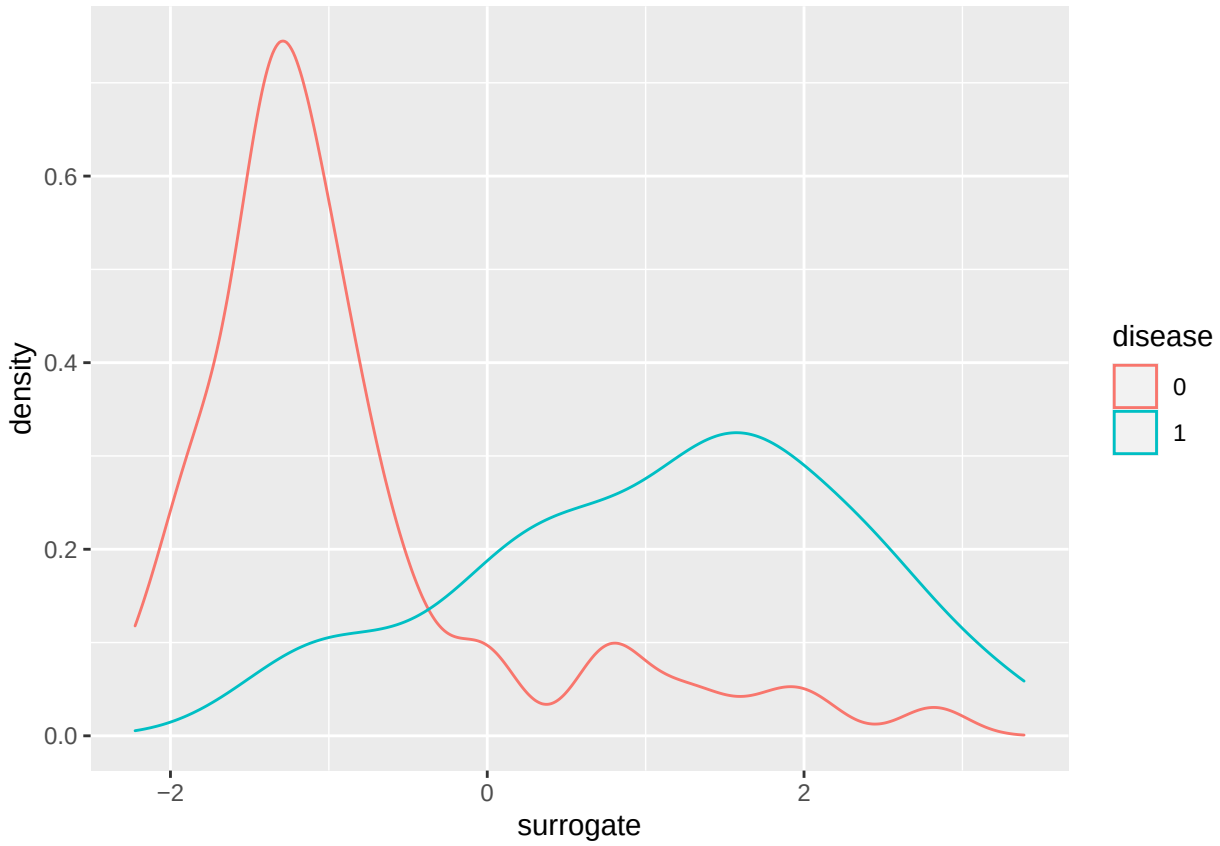
Feature selection

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

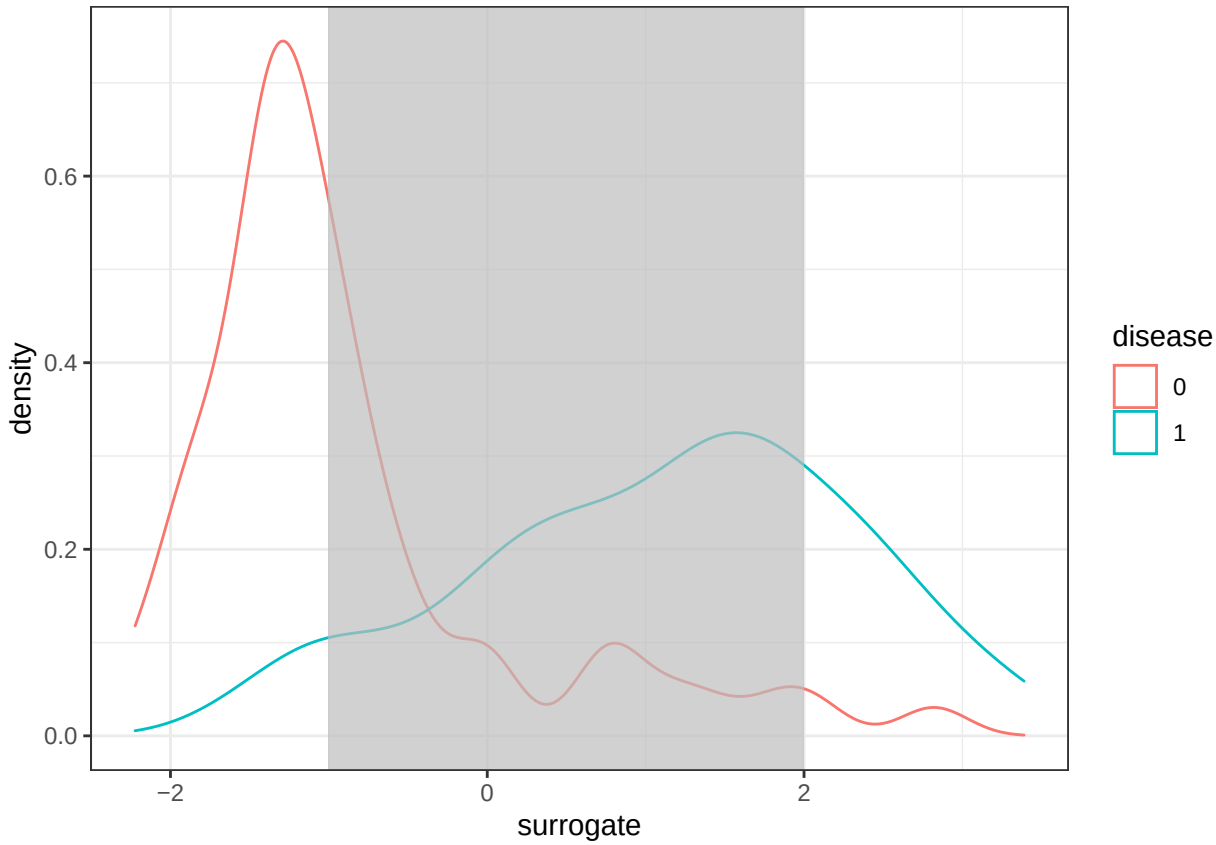
Opportunity 2 and 3 will be covered in the next module!

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)

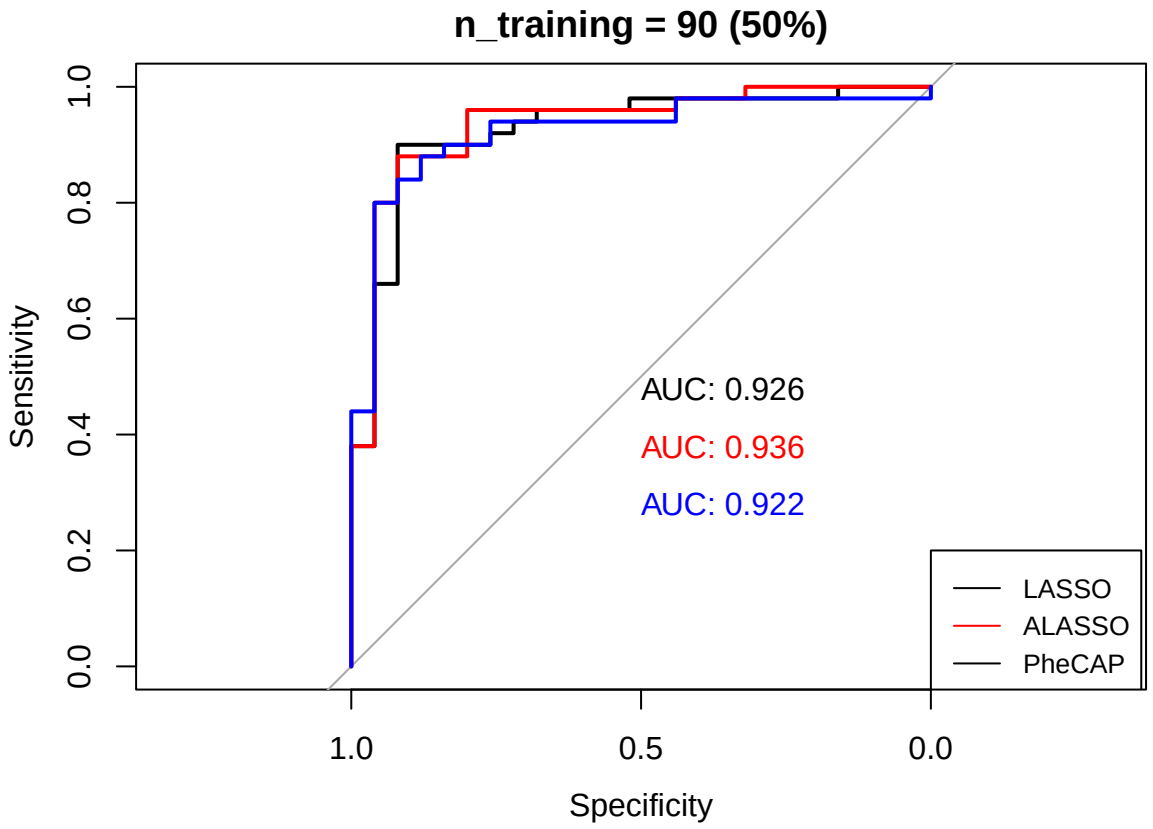
```
selected_features <- c("surrogate", "healthcare_utilization", SAFE_feature)

beta.phecap <- adaptive_lasso_fit(x = train_x[, selected_features],
                                  y = train_y,
                                  tuning = "cv", family = "binomial")

y_hat.phecap <- linear_model_predict(beta = beta.phecap,
                                     x = test_x[, selected_features],
                                     probability = TRUE)
```

```
roc.lasso <- roc(test_y, y_hat.lasso)
roc.alasso <- roc(test_y, y_hat.alasso)
roc.phecap <- roc(test_y, y_hat.phecap)

plot(roc.lasso,
  print.auc = TRUE, main = "n_training = 90 (50%)"
)
plot(roc.alasso,
  print.auc = TRUE, col = 'red', add = TRUE, print.auc.y = 0.4
)
plot(roc.phecap,
  print.auc = TRUE, col = 'blue', add = TRUE, print.auc.y = 0.3
)
legend(0, 0.2, legend = c("LASSO", "ALASSO", "PheCAP"), col = c("black","red"),
       lty = 1, cex = 0.8)
```




```
roc_full.lasso <- get_roc(y_true = test_y, y_score = y_hat.lasso)
head(roc_full.lasso,10)
```

```
##          cutoff    pos.rate  FPR     TPR       PPV       NPV        F1
##  [1,] 0.9438312 0.006666667 0.00 0.18050 1.0000000 0.3789314 0.3058026
##  [2,] 0.9033418 0.093333333 0.00 0.28025 1.0000000 0.4099201 0.4378051
##  [3,] 0.8628523 0.260000000 0.02 0.38000 0.9743590 0.4414414 0.5467626
##  [4,] 0.8605276 0.266666667 0.04 0.45000 0.9574468 0.4660194 0.6122449
##  [5,] 0.8582029 0.273333333 0.04 0.52000 0.9629630 0.5000000 0.6753247
##  [6,] 0.8015623 0.400000000 0.04 0.59000 0.9672131 0.5393258 0.7329193
##  [7,] 0.7449217 0.460000000 0.06 0.66000 0.9565217 0.5802469 0.7810651
##  [8,] 0.7430289 0.466666667 0.08 0.72000 0.9473684 0.6216216 0.8181818
##  [9,] 0.7411362 0.473333333 0.08 0.78000 0.9512195 0.6764706 0.8571429
## [10,] 0.6939832 0.573333333 0.08 0.84000 0.9545455 0.7419355 0.8936170
```

```
roc_full.alasso <- get_roc(y_true = test_y, y_score = y_hat.alasso)
head(roc_full.lasso,10)
```

```
##          cutoff    pos.rate  FPR     TPR       PPV       NPV        F1
##  [1,] 0.9438312 0.006666667 0.00 0.18050 1.0000000 0.3789314 0.3058026
##  [2,] 0.9033418 0.093333333 0.00 0.28025 1.0000000 0.4099201 0.4378051
##  [3,] 0.8628523 0.260000000 0.02 0.38000 0.9743590 0.4414414 0.5467626
##  [4,] 0.8605276 0.266666667 0.04 0.45000 0.9574468 0.4660194 0.6122449
##  [5,] 0.8582029 0.273333333 0.04 0.52000 0.9629630 0.5000000 0.6753247
##  [6,] 0.8015623 0.400000000 0.04 0.59000 0.9672131 0.5393258 0.7329193
##  [7,] 0.7449217 0.460000000 0.06 0.66000 0.9565217 0.5802469 0.7810651
```

```
##  [8,] 0.7430289 0.466666667 0.08 0.72000 0.9473684 0.6216216 0.8181818
##  [9,] 0.7411362 0.473333333 0.08 0.78000 0.9512195 0.6764706 0.8571429
## [10,] 0.6939832 0.573333333 0.08 0.84000 0.9545455 0.7419355 0.8936170
```

```
roc_full.phecap <- get_roc(y_true = test_y, y_score = y_hat.phecap)
head(roc_full.phecap,10)
```

```
##          cutoff    pos.rate  FPR       TPR       PPV       NPV        F1
##  [1,] 0.9984874 0.006666667 0.00 0.2104348 1.0000000 0.3877276 0.3477011
##  [2,] 0.9709349 0.173333333 0.00 0.3252174 1.0000000 0.4256107 0.4908136
##  [3,] 0.9433825 0.300000000 0.02 0.4400000 0.9777778 0.4666667 0.6068966
##  [4,] 0.9426084 0.306666667 0.04 0.5300000 0.9636364 0.5052632 0.6838710
##  [5,] 0.9418344 0.313333333 0.04 0.6200000 0.9687500 0.5581395 0.7560976
##  [6,] 0.8900529 0.466666667 0.04 0.7100000 0.9726027 0.6233766 0.8208092
##  [7,] 0.8382714 0.553333333 0.06 0.8000000 0.9638554 0.7014925 0.8743169
##  [8,] 0.8212902 0.560000000 0.08 0.8100000 0.9529412 0.7076923 0.8756757
##  [9,] 0.8043089 0.566666667 0.08 0.8200000 0.9534884 0.7187500 0.8817204
## [10,] 0.8004121 0.573333333 0.08 0.8300000 0.9540230 0.7301587 0.8877005
```

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

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