Module 3: Semi-supervised learning (PheCAP)

Siyue Yang, Jianhui Gao, and Jesse Gronsbell

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

# 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 environment.
load("environment.RData")
```

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

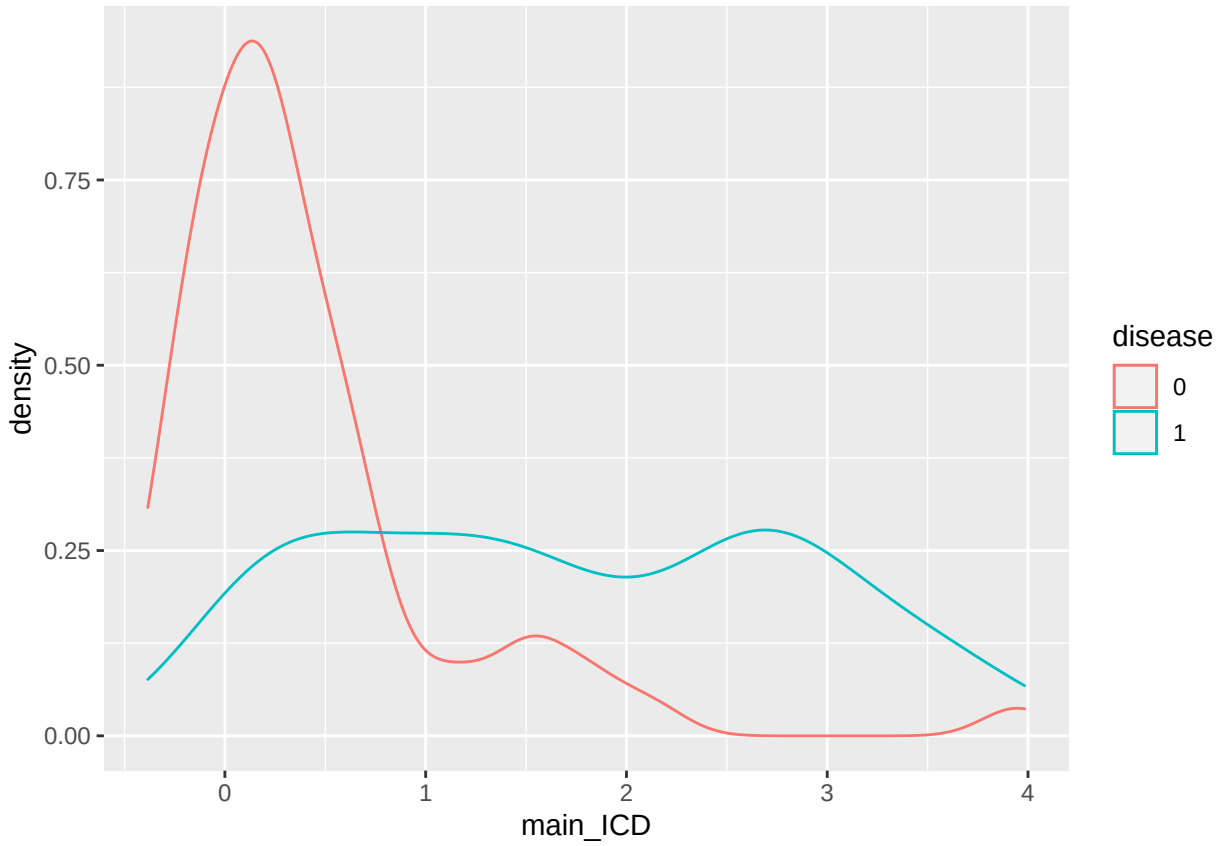
Feature selection

How to select features?

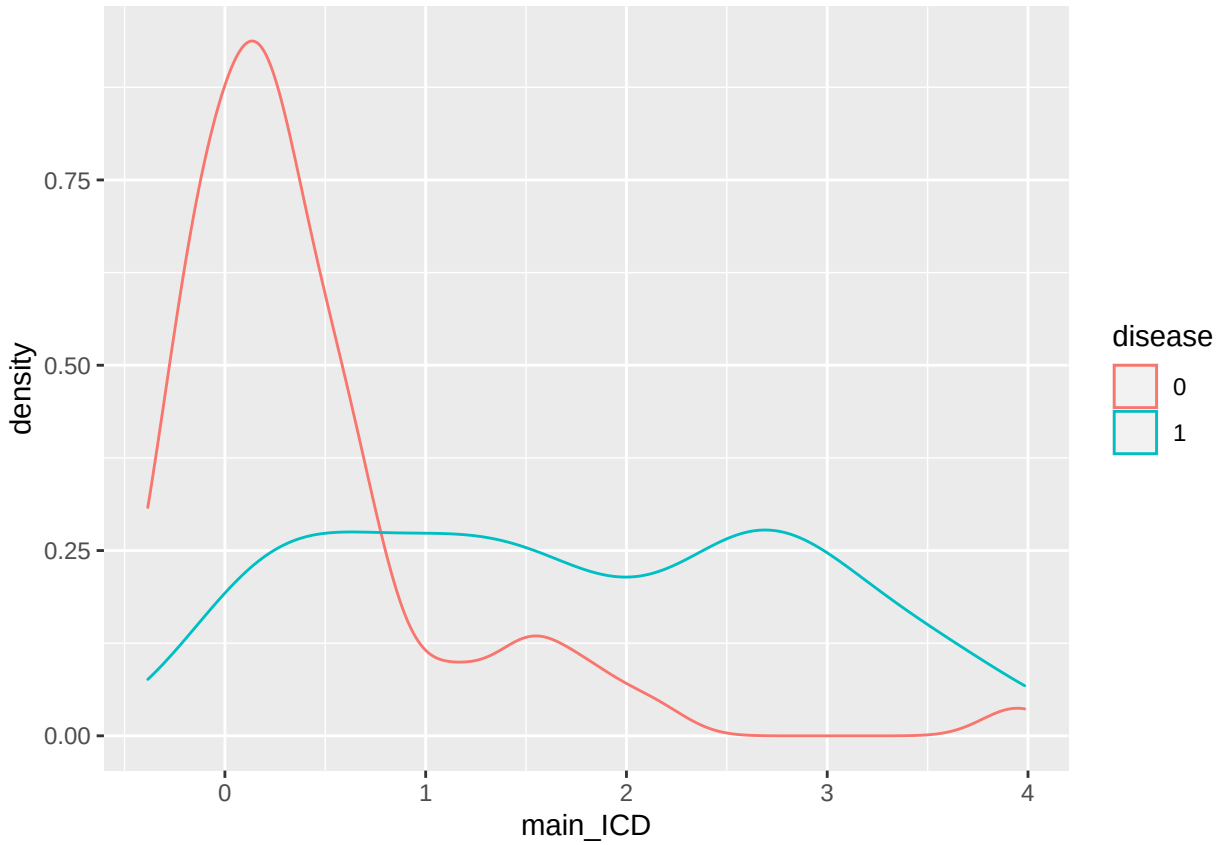
Can leverage some clinical-meaningful features that are related to Y.

e.g. Feature "main_ICD" = the total number of the disease-related billing codes.

```
ehr_data %>%
  filter(!is.na(label)) %>%
  mutate(disease = factor(label)) %>%
  ggplot(aes(x = main_ICD)) +
  geom_density(aes(color = disease))
```

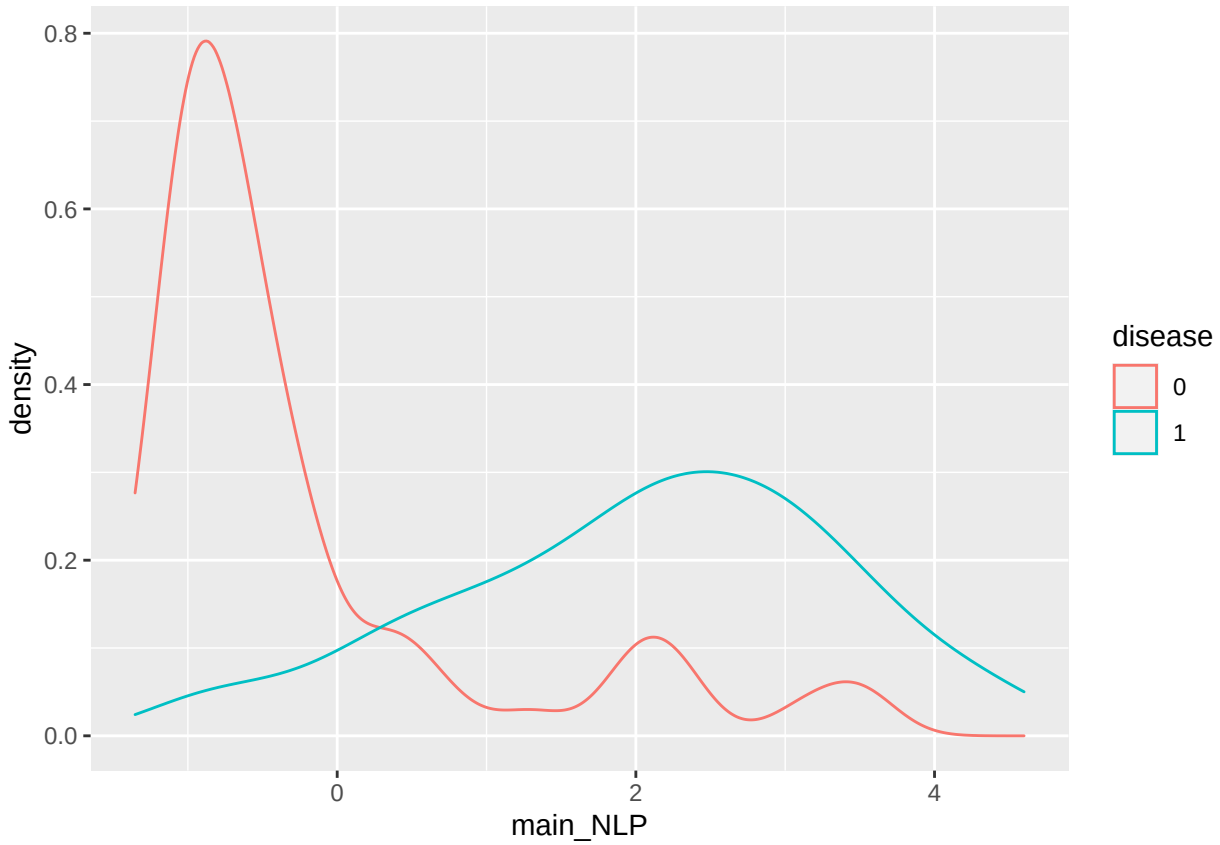



```
# With log transformation.
ehr_data %>%
  filter(!is.na(label)) %>%
  mutate(disease = factor(label)) %>%
  ggplot(aes(x = main_ICD)) +
  geom_density(aes(color = disease))
```



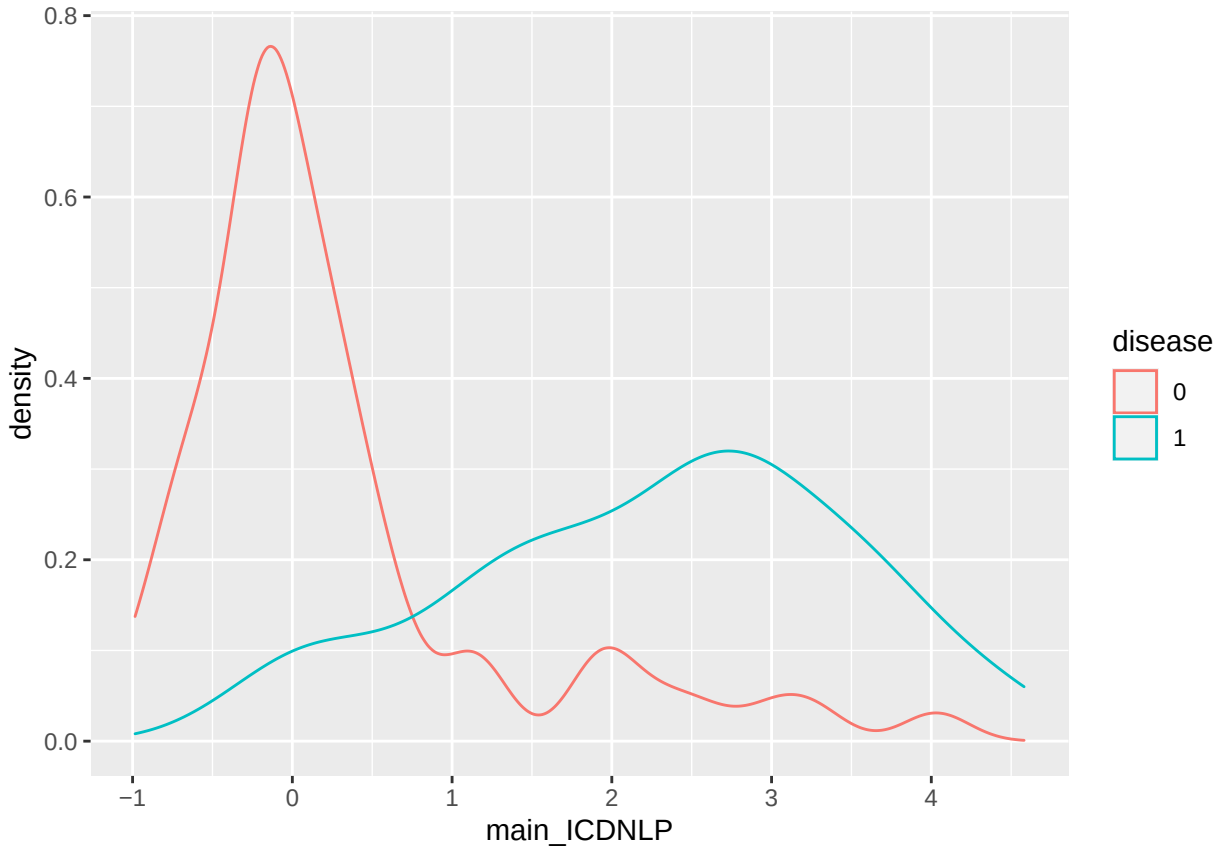
The more the disease-related codes, the more **likely** the patient has the disease.

```
ehr_data %>%
  filter(!is.na(label)) %>%
  mutate(disease = factor(label)) %>%
  ggplot(aes(x = main_NLP)) +
  geom_density(aes(color = disease))
```



Other options? - Combine them.

```
ehr_data %>%
  filter(!is.na(label)) %>%
  mutate(disease = factor(label)) %>%
  ggplot(aes(x = main_ICDNLP)) +
  geom_density(aes(color = disease))
```




```
nonmissing_index <- which(!is.na(ehr_data$label))
y <- ehr_data$label[nonmissing_index]
sicd <- ehr_data$main_ICD
snlp <- ehr_data$main_NLP
sicdnlp <- ehr_data$main_ICDNLP

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

```
## [1] 0.8394551
```

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

```
## [1] 0.8841149
```

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

```
## [1] 0.8875034
```

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

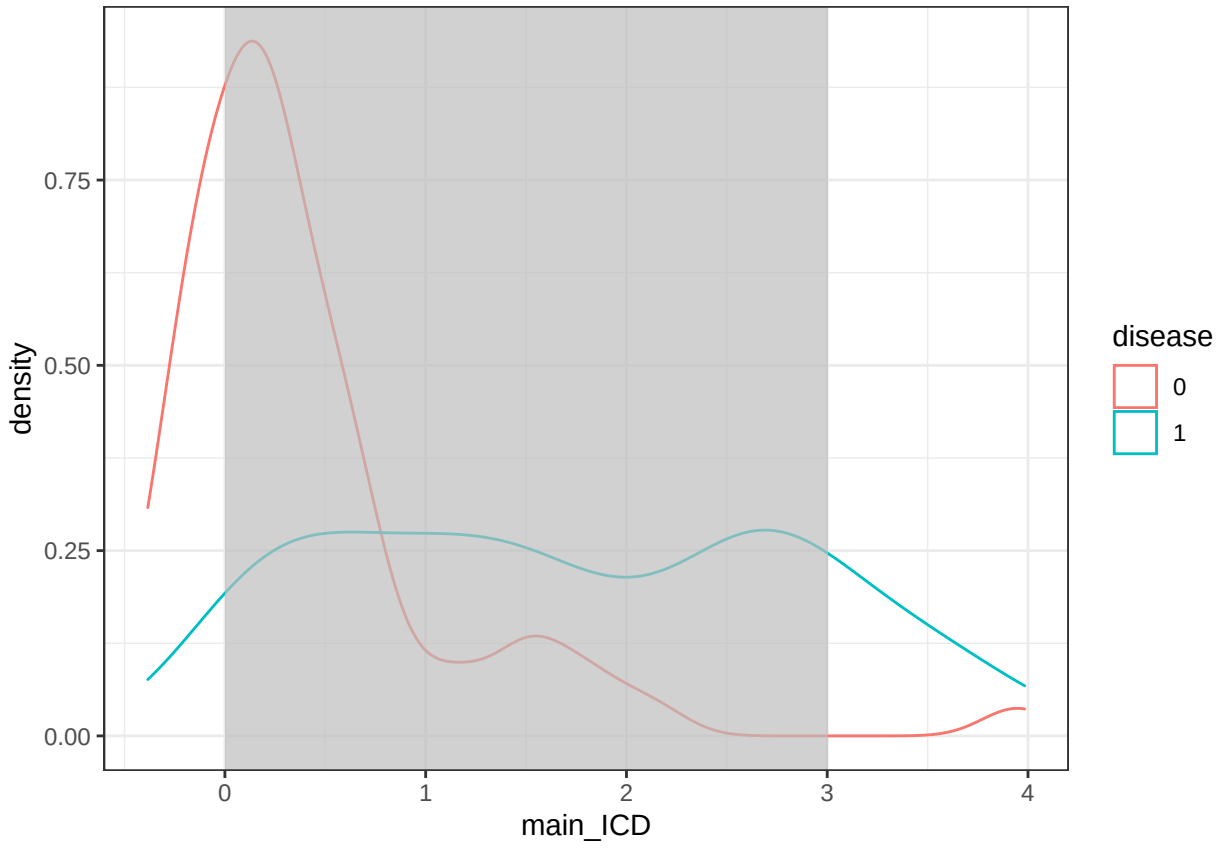
Opportunity 2 and 3 will be covered in the next module!

Feature selection method

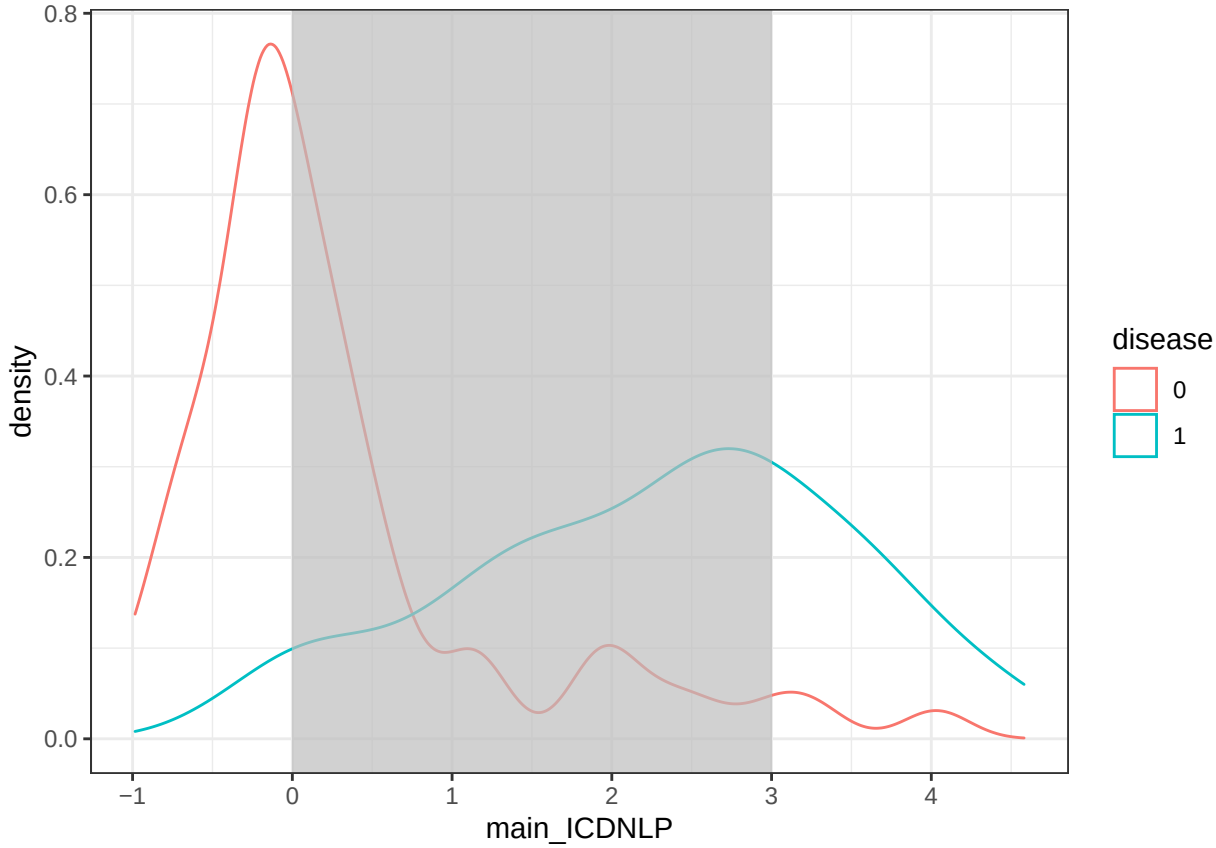
Motivation (Extreme assumption):

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

```
ehr_data %>%
  filter(!is.na(label)) %>%
  mutate(disease = factor(label)) %>%
  ggplot(aes(x = main_ICD)) +
  geom_density(aes(color = disease)) +
  theme_bw() +
  annotate("rect", fill = "grey", alpha = 0.7, xmin = 0, xmax = 3,
           ymin = -Inf, ymax = Inf)
```



```r
ehr_data %>%
  filter(!is.na(label)) %>%
  mutate(disease = factor(label)) %>%
  ggplot(aes(x = main_ICDNLP)) +
  geom_density(aes(color = disease)) +
  theme_bw() +
  annotate("rect", fill = "grey", alpha = 0.7, xmin = 0, xmax = 3,
           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!

```r
# Prepare 3 surrogates.
sicd <- ehr_data$main_ICD
snlp <- ehr_data$main_NLP
sicdnlp <- ehr_data$main_ICDNLP

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

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

```
# Truncated using 3 and 1.
SAFE_icd <- extreme_method(sicd, x, u_bound = 3, l_bound = 0)
SAFE_nlp <- extreme_method(snlp, x, u_bound = 3, l_bound = 0)
SAFE_both <- extreme_method(sicdnlp, x, u_bound = 3, l_bound = 0)

# Majority voting.
beta <- rbind(SAFE_icd$beta_all, SAFE_nlp$beta_all, SAFE_both$beta_all)
SAFE_select <- which(colMeans(beta, na.rm = T) >= 0.5)
SAFE_feature <- colnames(x)[SAFE_select]
SAFE_feature
```

```
##  [1] "NLP6"   "NLP56"  "NLP93"  "NLP160" "NLP161" "NLP231" "NLP306" "NLP309" "NLP321" "NLP349"
## [11] "NLP403" "NLP434" "NLP446" "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.

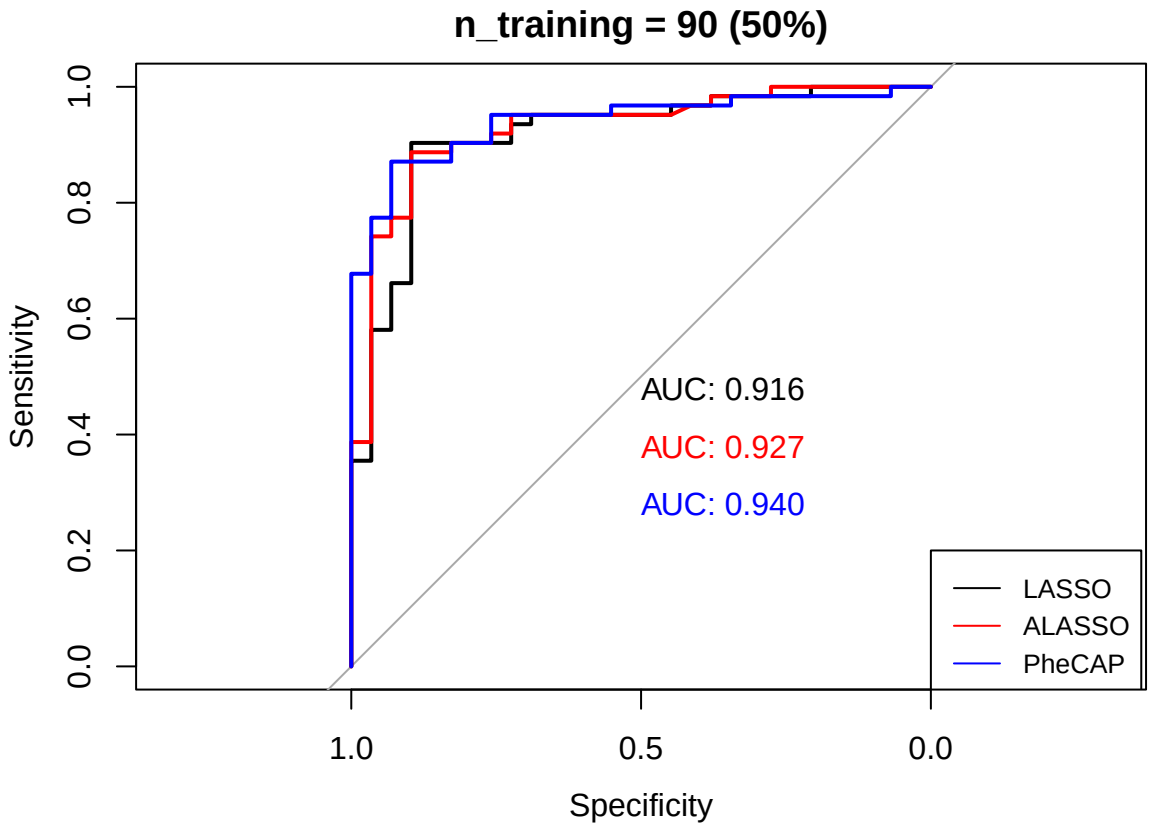
```
selected_features <- c("main_ICD", "main_NLP", "main_ICDNLP", "healthcare_utilization", SAFE_feature)

# Fit PheCAP.
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.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", "blue"),
       lty = 1, cex = 0.8)
```




```
FPR = 0.05
sens_spec <- rbind(coords(roc = roc(test_y, y_hat.lasso), x = FPR, input = "fpr")[-1],
                   coords(roc = roc(test_y, y_hat.alasso), x = FPR, input = "fpr")[-1],
                   coords(roc = roc(test_y, y_hat.phecap), x = FPR, input = "fpr")[-1])

rownames(sens_spec) <- c("LASSO", "ALASSO", "PheCAP")
sens_spec
```

```
FPR = 0.1
sens_spec <- rbind(coords(roc = roc(test_y, y_hat.lasso), x = FPR, input = "fpr")[-1],
                   coords(roc = roc(test_y, y_hat.alasso), x = FPR, input = "fpr")[-1],
                   coords(roc = roc(test_y, y_hat.phecap), x = FPR, input = "fpr")[-1])

rownames(sens_spec) <- c("LASSO", "ALASSO", "PheCAP")
sens_spec
```

Different training size

- randomly sample traning size = 50, 70, 90
- rest as testing set
- repeat 500 times

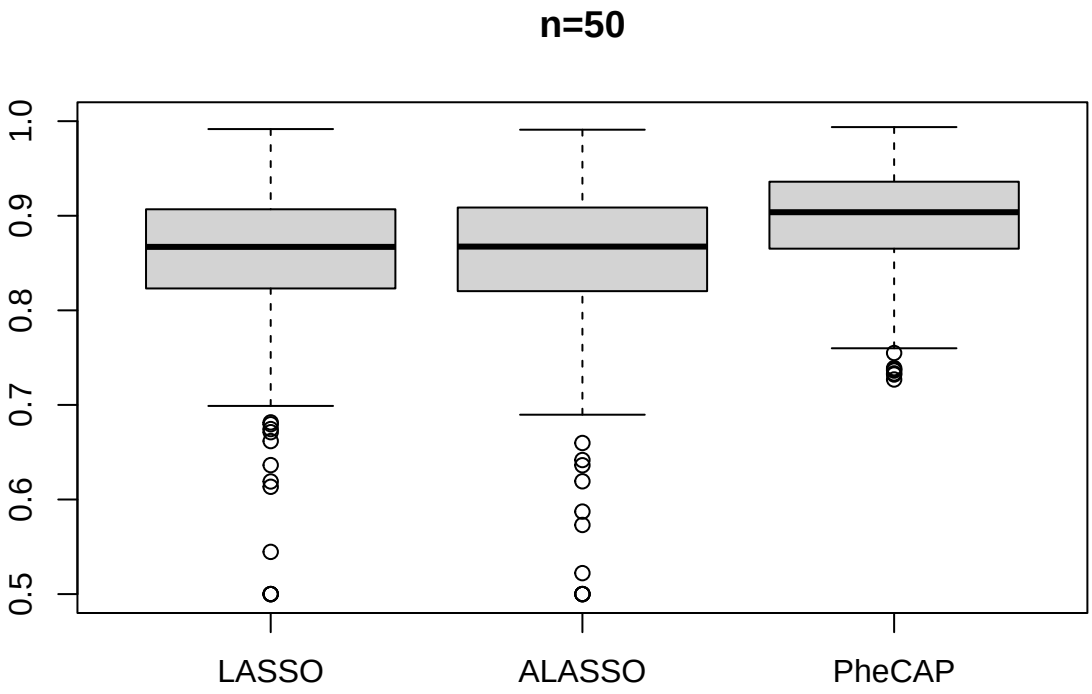
```
selected_index <- which(colnames(ehr_data) %in% selected_features == TRUE)
```

```
start<- Sys.time()
auc_phecap <- validate_phecap(dat = labeled_data, nsim = 600,
                              n.train = c(50, 70, 90),
                              selected_features = selected_index)
end <- Sys.time()
end - start
```

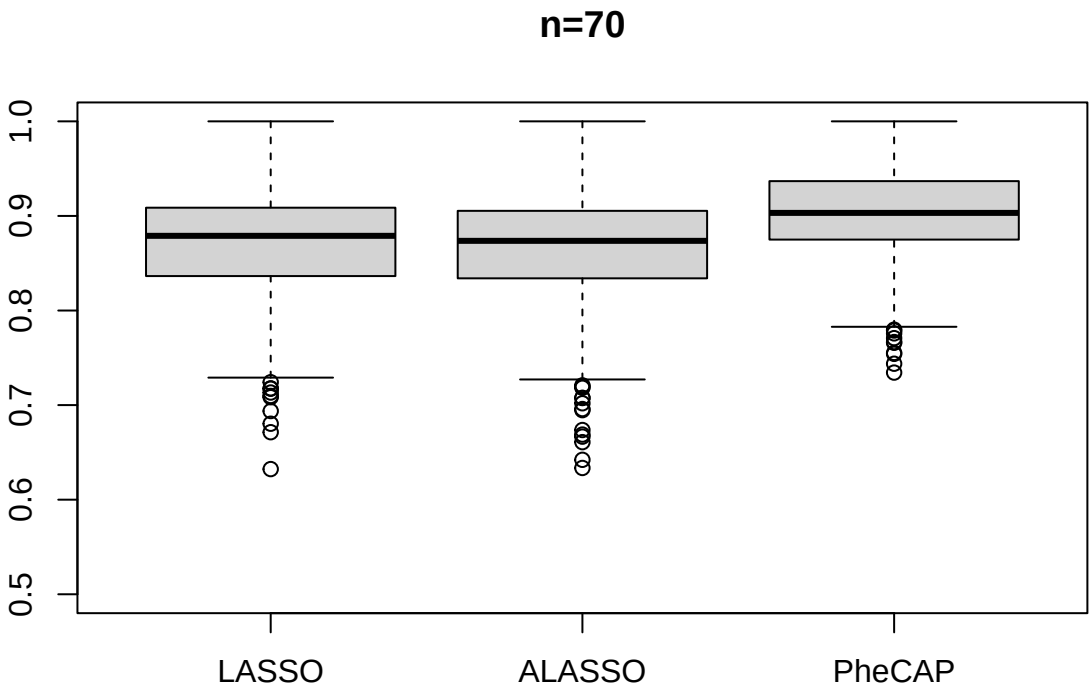
```
## Time difference of 2.093254 mins
```

```
colnames(auc_phecap) <- paste(paste0("n=", c(50, 70, 90)), c("PheCAP"), sep = ",")
```

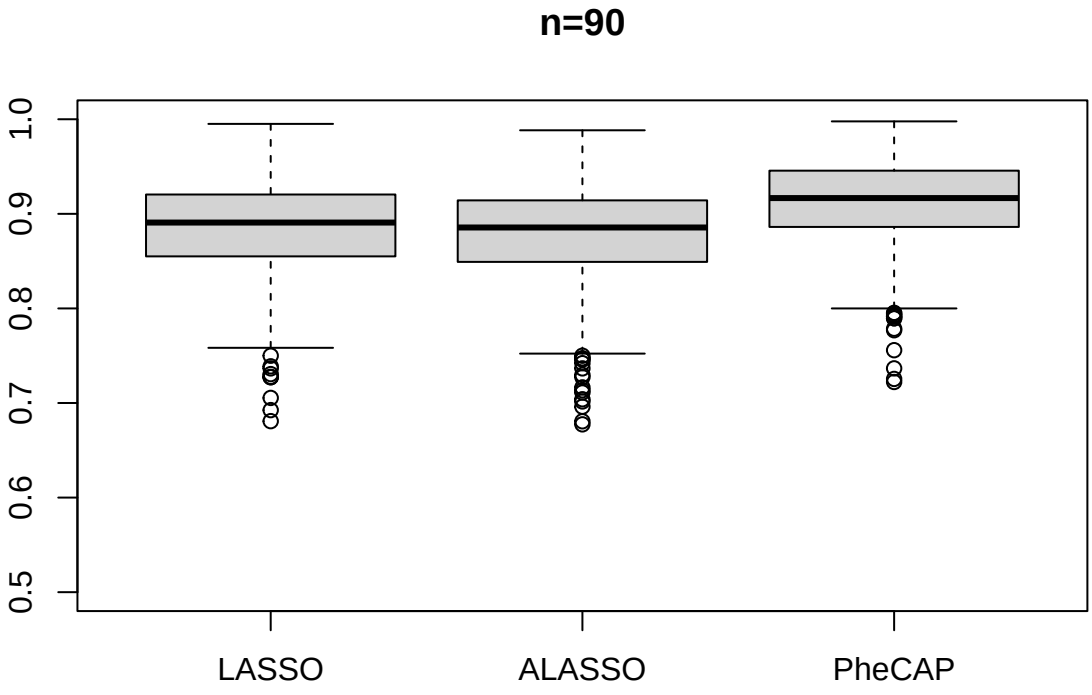
```
boxplot(cbind(auc_supervised, auc_phecap) %>% select(starts_with("n=50")),
        ylim = c(0.5, 1), names = c("LASSO", "ALASSO", "PheCAP"), main = "n=50")
```



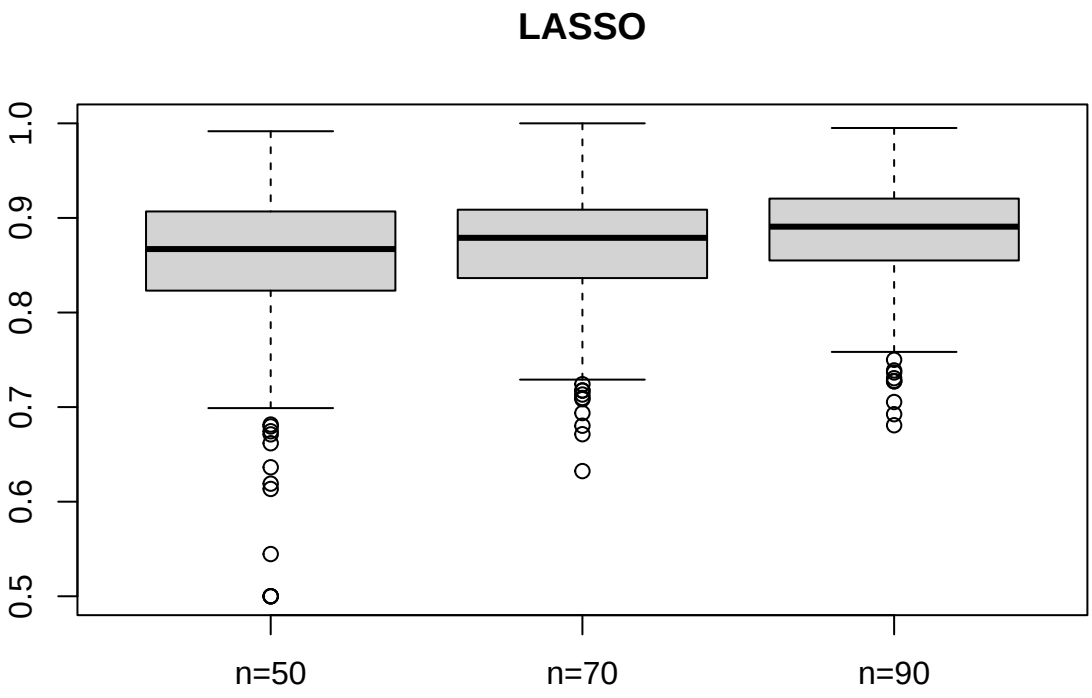

```
boxplot(cbind(auc_supervised, auc_phecap) %>% select(starts_with("n=70")),
        ylim = c(0.5, 1), names = c("LASSO", "ALASSO", "PheCAP"), main = "n=70")
```



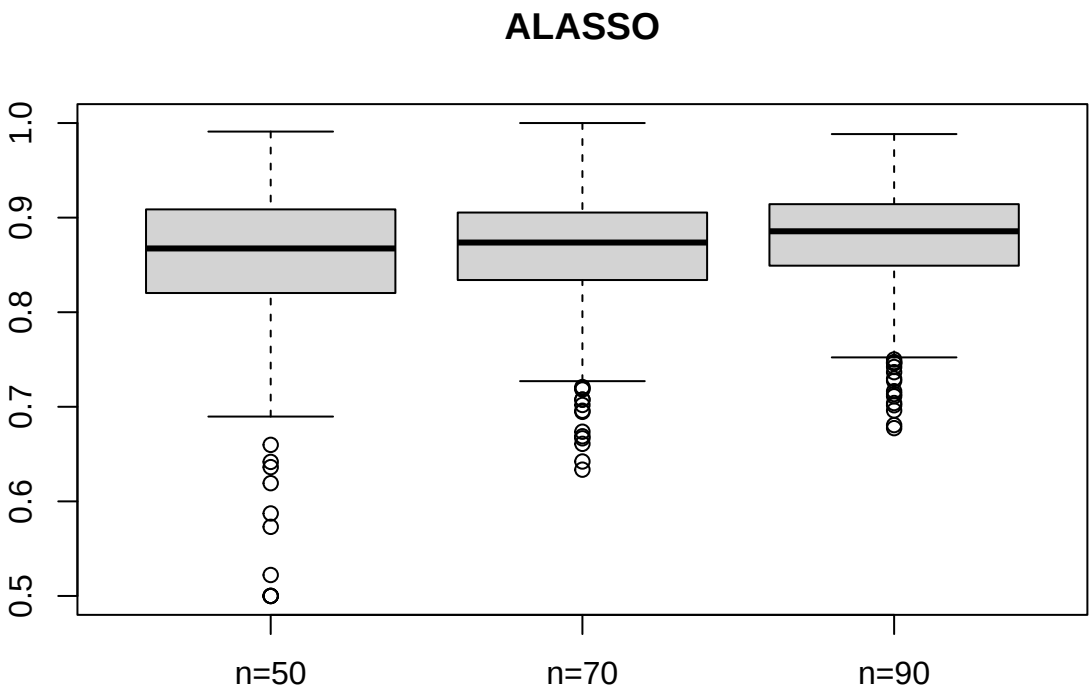

```
boxplot(cbind(auc_supervised, auc_phecap) %>% select(starts_with("n=90")),
        ylim = c(0.5, 1), names = c("LASSO", "ALASSO", "PheCAP"), main = "n=90")
```



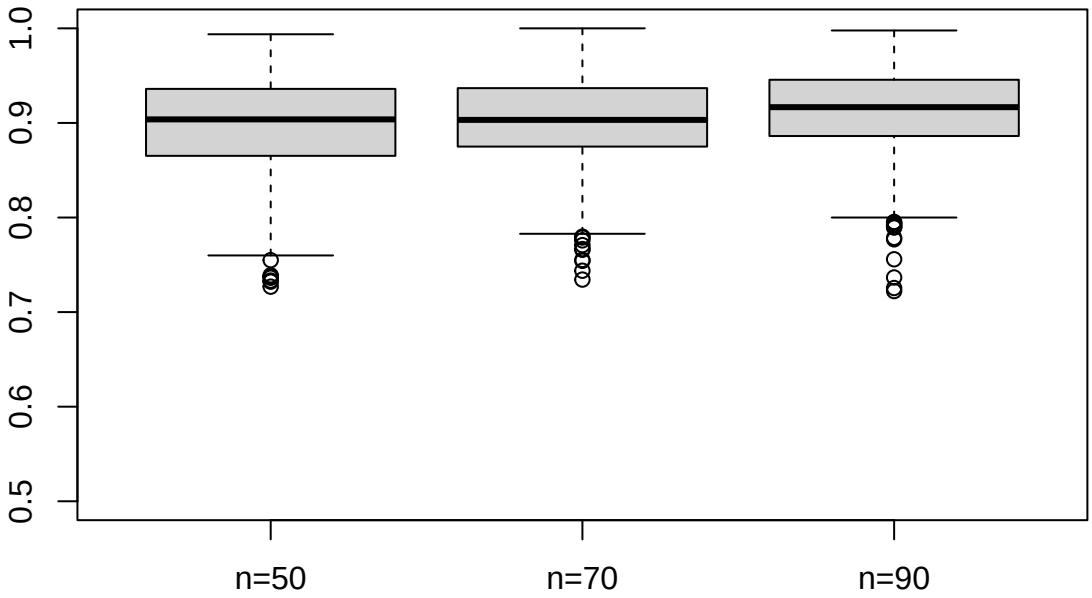

```
boxplot(auc_supervised[,1:3], ylim = c(0.5, 1),
        names = c("n=50", "n=70", "n=90"), main = "LASSO")
```




```
boxplot(auc_supervised[,4:6], ylim = c(0.5, 1),
        names = c("n=50", "n=70", "n=90"), main = "ALASSO")
```




```
boxplot(auc_phecap[,1:3], ylim = c(0.5, 1),
        names = c("n=50", "n=70", "n=90"), main = "PheCAP")
```



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

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