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 environment from example 1
load("environment.RData")
```

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

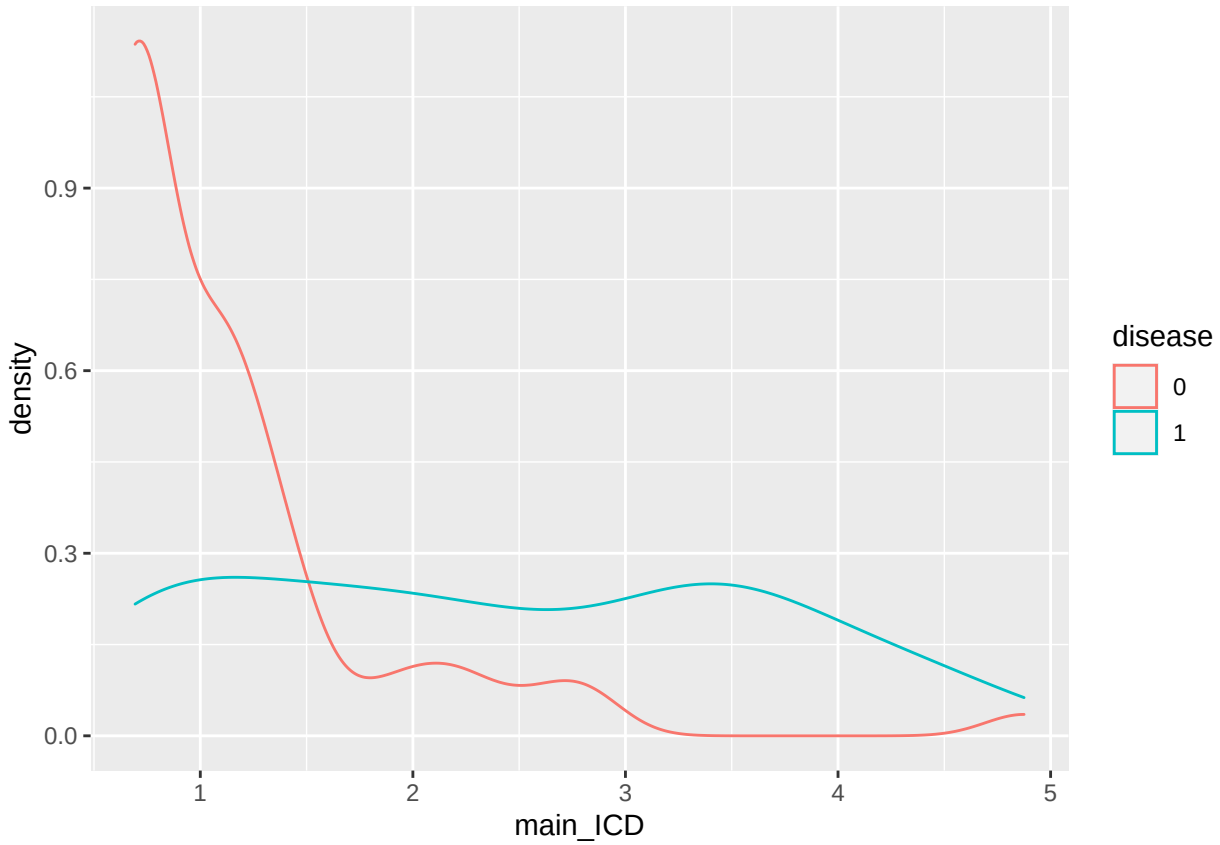
Feature selection

How to select features?

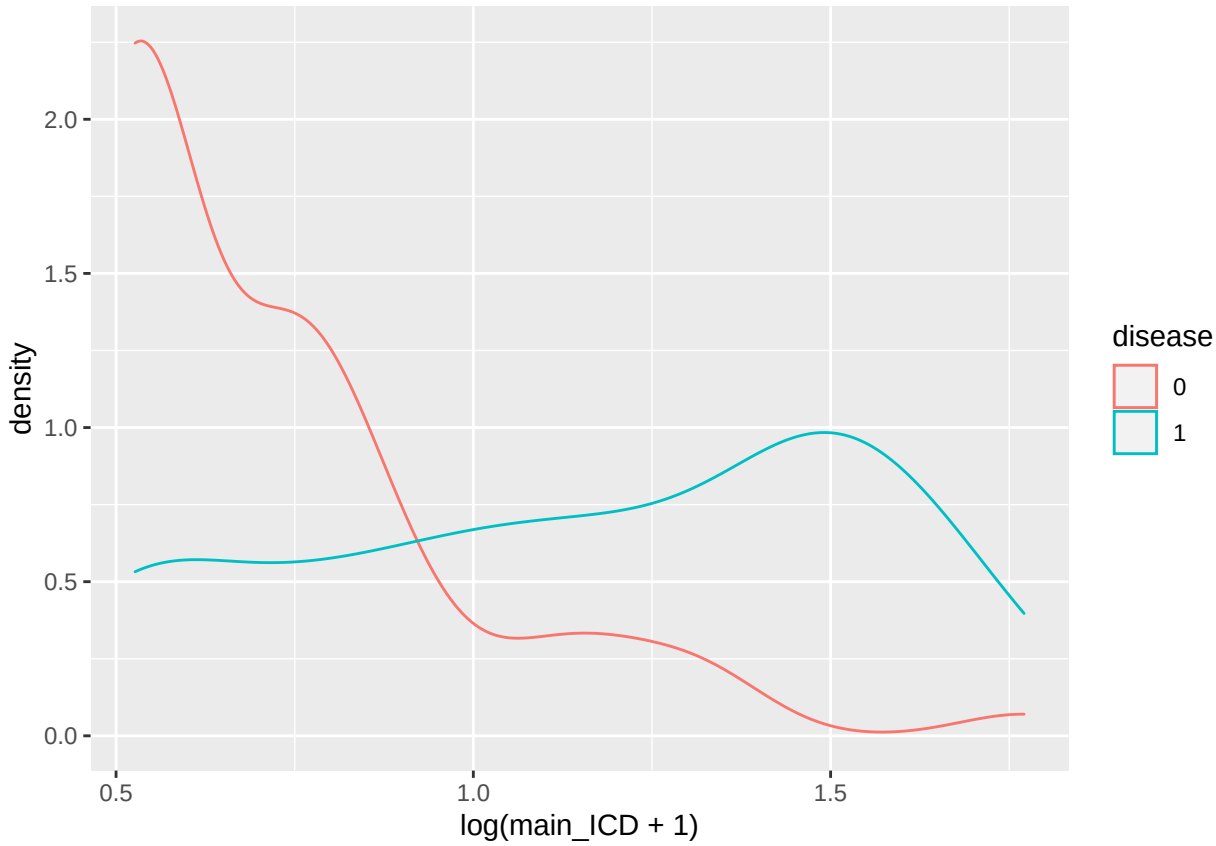
Can leverage some clinical-meaningful features that are related to Y.

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))
```

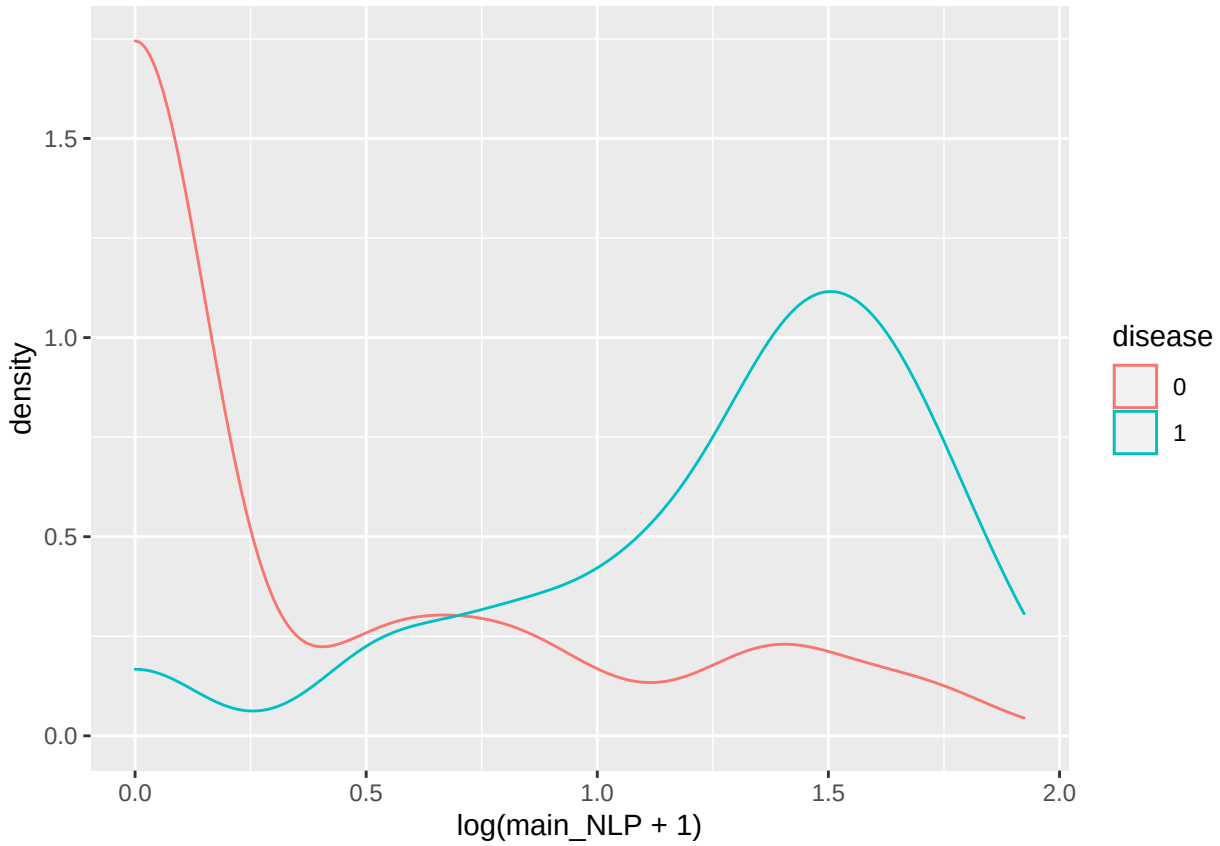



```
# With log transformation.
ehr_data %>%
  filter(!is.na(label)) %>%
  mutate(disease = factor(label)) %>%
  ggplot(aes(x = log(main_ICD + 1))) +
  geom_density(aes(color = disease))
```



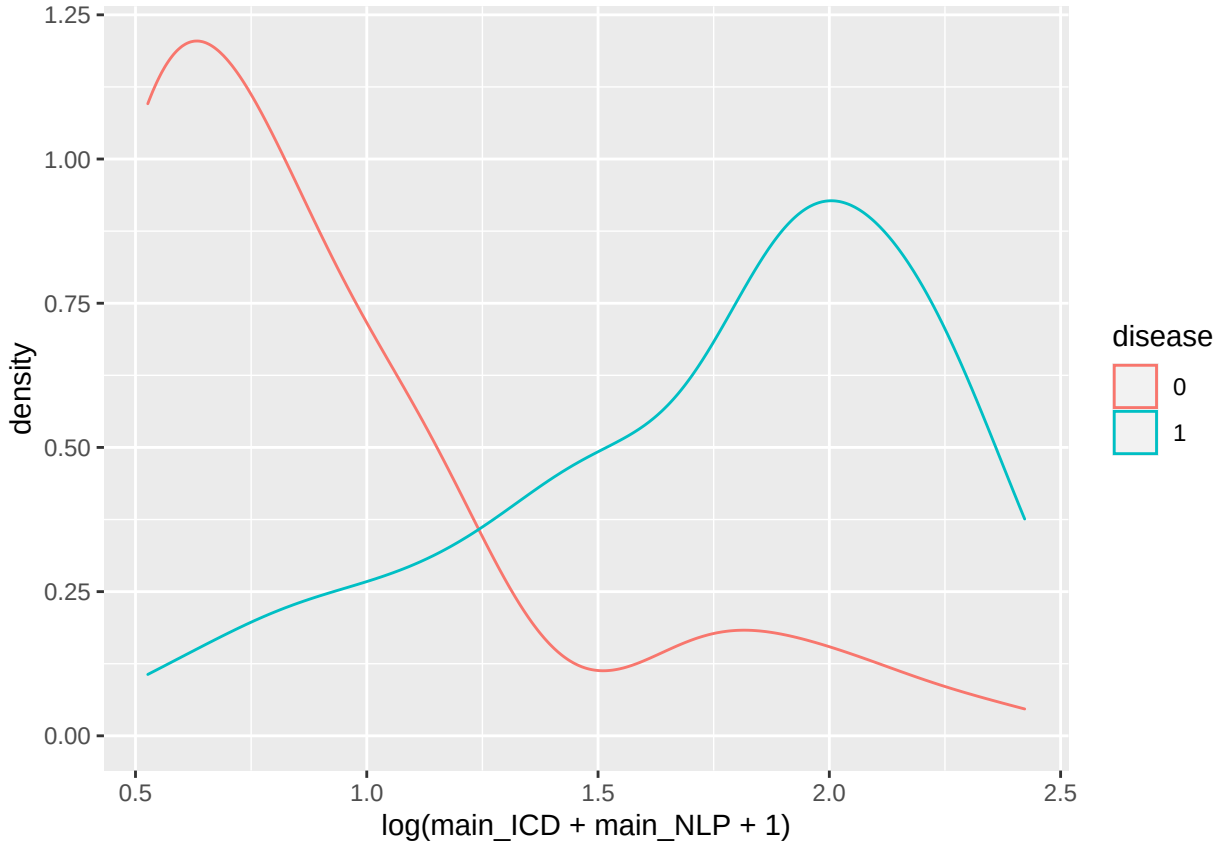
The more the disease-related codes, the more **likely** the patient has the disease.

```
ehr_data %>%
  filter(!is.na(label)) %>%
  mutate(disease = factor(label)) %>%
  ggplot(aes(x = log(main_NLP + 1))) +
  geom_density(aes(color = disease))
```



Other options? - Combine them.

```
ehr_data %>%
  filter(!is.na(label)) %>%
  mutate(disease = factor(label)) %>%
  ggplot(aes(x = log(main_ICD+main_NLP+1))) +
  geom_density(aes(color = disease))
```




```
nonmissing_index <- which(!is.na(ehr_data$label))
y <- ehr_data$label[nonmissing_index]
surrogate_icd <- ehr_data$main_ICD[nonmissing_index]
surrogate_nlp <- ehr_data$main_NLP[nonmissing_index]

get_auc(y, surrogate_icd)
```

```
## [1] 0.8046218
```

```
get_auc(y, surrogate_nlp)
```

```
## [1] 0.8712388
```

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

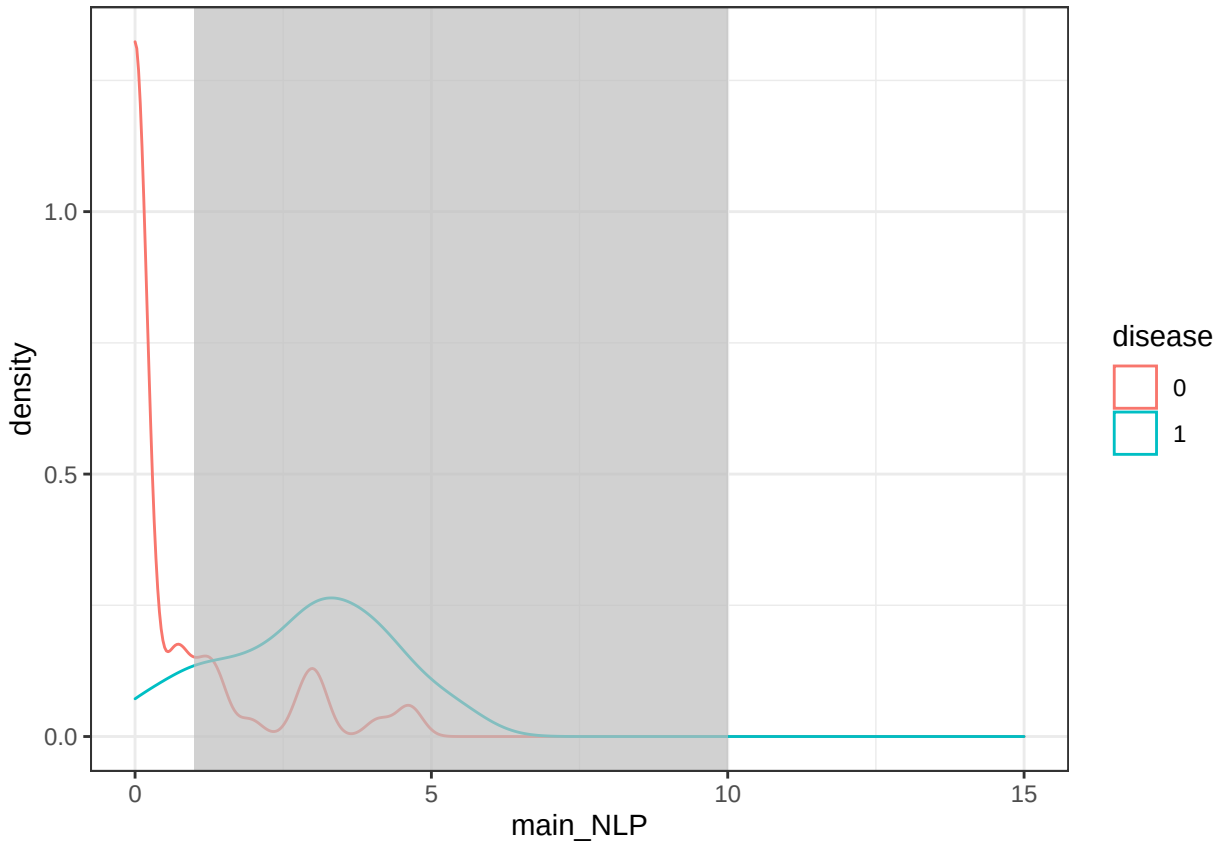
Opportunity 2 and 3 will be covered in the next module!

Feature selection method - Tail method (Yu et al. 2017)

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_NLP)) +
  geom_density(aes(color = disease)) +
  theme_bw() +
  annotate("rect", fill = "grey", alpha = 0.7, xmin = 1, xmax = 10, ymin = -Inf, ymax = Inf) +
  xlim(c(0,15))
```



- 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!

```
surrogates <- list(
  PhecapSurrogate(
    variable_names = "main_ICD",
    lower_cutoff = 1, upper_cutoff = 10),
  PhecapSurrogate(
    variable_names = "main_NLP",
    lower_cutoff = 1, upper_cutoff = 10),
  PhecapSurrogate(
    variable_names = c("main_ICD", "main_NLP"),
    lower_cutoff = 1, upper_cutoff = 10))
```

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

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

```
system.time(feature_selected <- phecap_run_feature_extraction(data, surrogates))
```

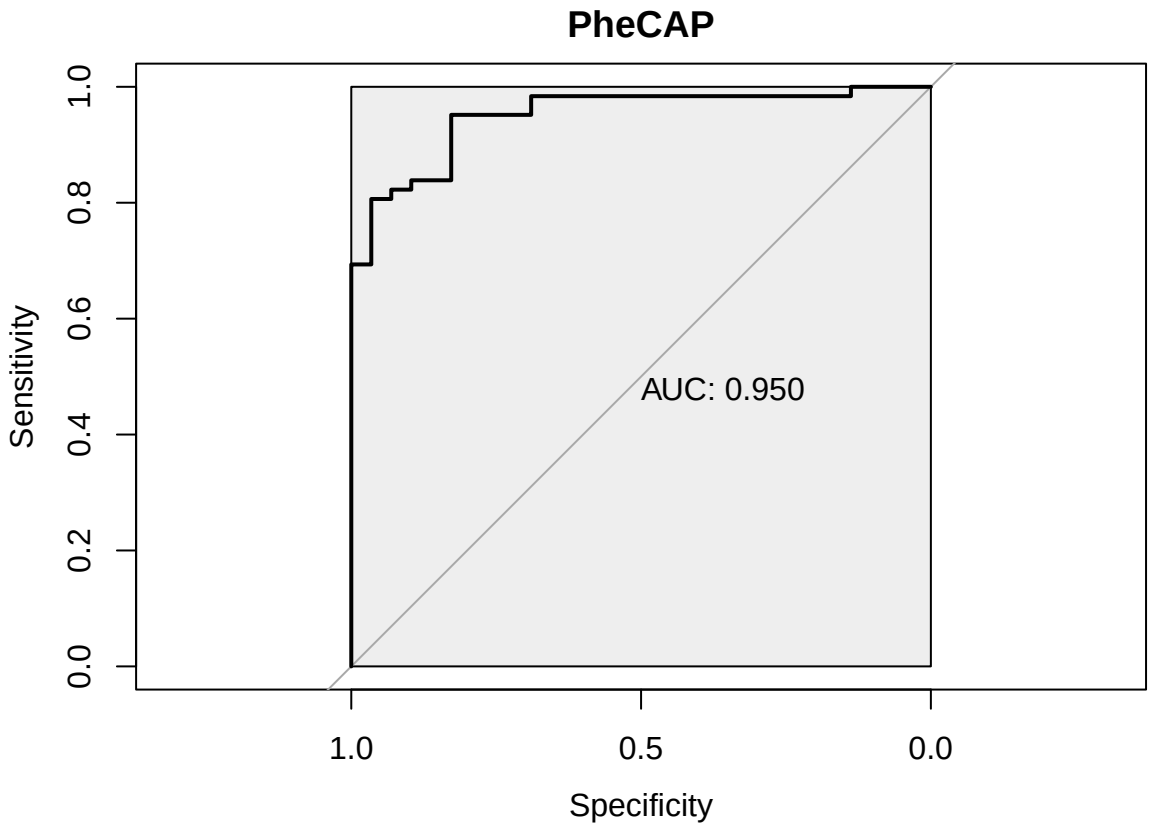
```
##    user  system elapsed
##  56.651   0.238  57.100
```

```
feature_selected
```

```
## Feature(s) selected by surrogate-assisted feature extraction (SAFE)
## [1] "main_ICD" "main_NLP" "NLP56"    "NLP93"    "NLP160"   "NLP161"   "NLP306"
## [8] "NLP403"   "NLP536"
```

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

```
modelssl_lr <- fit_alasso_bic(
  y = train_y,
  x = train_x[, c("healthcare_utilization", feature_selected[[1]])]
)

# Prediction on testing set.
y_hat <- expit(cbind(1, test_x[, c("healthcare_utilization", feature_selected[[1]])])
%*% modelssl_lr$beta_hat) # Inverse Logit

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

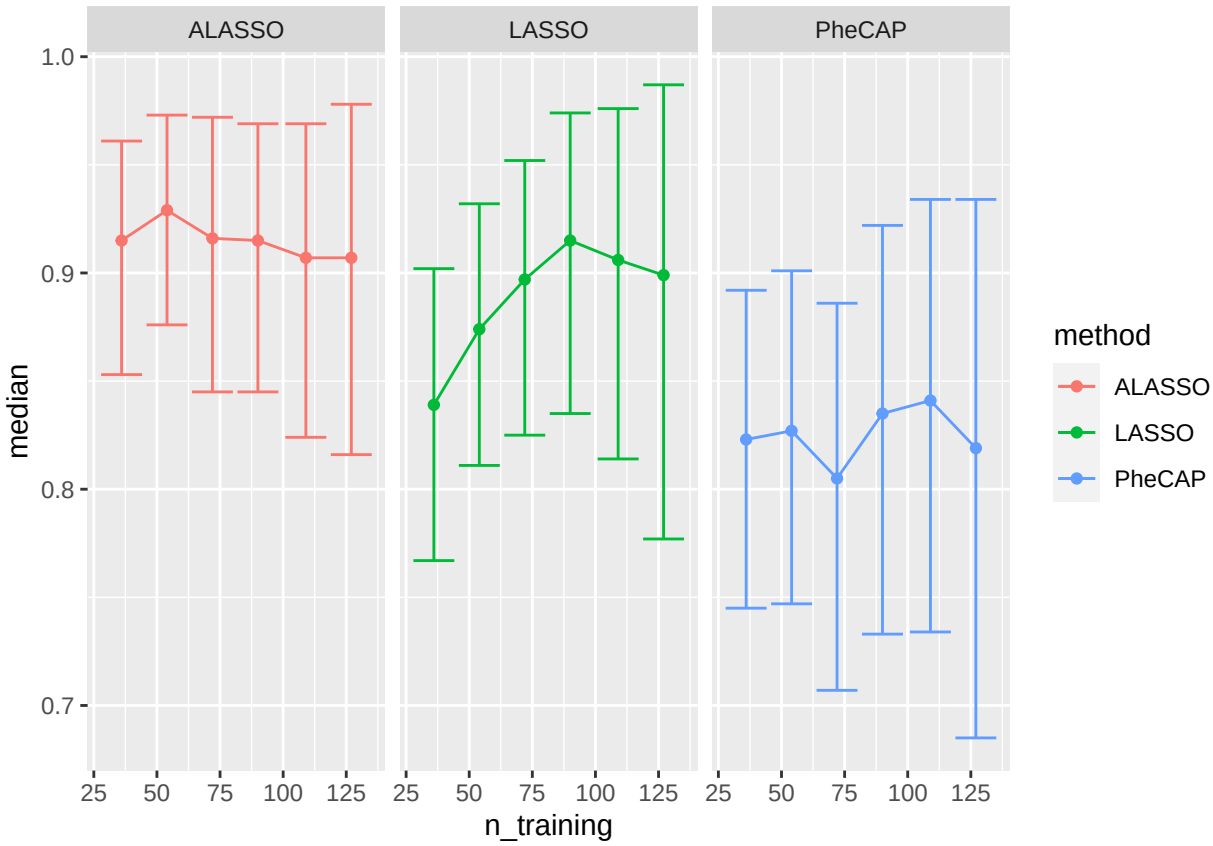



```
head(get_roc(test_y, y_hat))
```

```
##         cutoff    pos.rate        FPR       TPR       PPV       NPV        F1
## [1,] 0.9673776 0.005494505 0.00000000 0.3388930 1.0000000 0.4143530 0.5062286
## [2,] 0.8976103 0.384615385 0.00000000 0.5445931 1.0000000 0.5066810 0.7051606
## [3,] 0.8441824 0.483516484 0.03448276 0.7025806 0.9775583 0.6029285 0.8175676
## [4,] 0.8336456 0.483516484 0.03448276 0.7353226 0.9785362 0.6304886 0.8396722
## [5,] 0.8087518 0.549450549 0.03448276 0.7680645 0.9794323 0.6606890 0.8609655
## [6,] 0.7721929 0.560439560 0.03448276 0.8008065 0.9802567 0.6939281 0.8814913
```

```
labeled_data <- ehr_data %>% dplyr::filter(!is.na(label))

test_auc <- c()
for (i in round(seq(0.2, 0.7, 0.1) * nrow(labeled_data))) {
  set.seed(123456)
  idx <- sample(labeled_data$patient_id, i)
  train_data <- labeled_data %>% filter(patient_id %in% idx)
  test_data <- labeled_data %>% filter(!(patient_id %in% idx))
  idy <- test_data$patient_id

  # 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")
```