

Module 4: Semi and Weakly Supervised Learning

Jianhui Gao, Siyue Yang, and Jessica Gronsbell

01/06/2022

```
# If a package is installed, it will be loaded. If any
## are not, the missing package(s) will be installed
## from CRAN and then loaded.

## First specify the packages of interest
packages <- c(
  "dplyr", "PheCAP", "glmnet", "randomForestSRC", "PheNorm",
  "MAP", "pROC", "mltools", "data.table", "ggplot2", "parallel"
)

## Now load or install&load all
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("../module4/environment.RData")
source("../Rscripts/helper_function.R")
```

Semi-supervised Learning

(i) Regress the surrogate on the features with penalized least square to get the direction of beta.

```
x <- all_x %>% select(starts_with("COD") | starts_with("NLP")) # COD + NLP
S <- ehr_data$main_ICDNLP

# Step 1
beta.step1 <- adaptive_lasso_fit(
  y = S, # surrogate
  x = x, # all X
  family = "gaussian",
  tuning = "cv"
)

# Features selected
names(beta.step1[abs(beta.step1) > 0])[-1]
```

```
## [1] "COD6" "COD8" "COD10" "NLP5" "NLP7" "NLP14" "NLP18" "NLP21"
## [9] "NLP24" "NLP28" "NLP31" "NLP33" "NLP44" "NLP50" "NLP56" "NLP59"
## [17] "NLP61" "NLP62" "NLP66" "NLP68" "NLP70" "NLP73" "NLP74" "NLP76"
## [25] "NLP81" "NLP89" "NLP92" "NLP93" "NLP95" "NLP98" "NLP102" "NLP104"
## [33] "NLP108" "NLP110" "NLP116" "NLP120" "NLP127" "NLP130" "NLP146" "NLP160"
## [41] "NLP161" "NLP172" "NLP176" "NLP178" "NLP179" "NLP183" "NLP189" "NLP190"
## [49] "NLP192" "NLP199" "NLP202" "NLP206" "NLP215" "NLP225" "NLP231" "NLP232"
## [57] "NLP243" "NLP246" "NLP250" "NLP253" "NLP256" "NLP288" "NLP294" "NLP295"
## [65] "NLP299" "NLP302" "NLP304" "NLP306" "NLP309" "NLP321" "NLP326" "NLP336"
## [73] "NLP338" "NLP342" "NLP343" "NLP347" "NLP349" "NLP350" "NLP351" "NLP357"
## [81] "NLP359" "NLP361" "NLP363" "NLP365" "NLP369" "NLP380" "NLP387" "NLP393"
## [89] "NLP395" "NLP403" "NLP405" "NLP407" "NLP417" "NLP431" "NLP434" "NLP437"
## [97] "NLP440" "NLP446" "NLP451" "NLP452" "NLP456" "NLP463" "NLP465" "NLP468"
## [105] "NLP473" "NLP482" "NLP483" "NLP487" "NLP490" "NLP495" "NLP500" "NLP507"
## [113] "NLP522" "NLP529" "NLP534" "NLP536" "NLP539" "NLP541" "NLP544" "NLP554"
## [121] "NLP560" "NLP564" "NLP568" "NLP572"
```

(ii) Regress the outcome on the linear predictor to get the intercept and multiplier for the beta.

```
# linear predictor without intercept
bhatx <- linear_model_predict(beta = beta.step1, x = as.matrix(x))

# Step 2
step2 <- glm(train_y ~ bhatx[train_data$patient_id] + S[train_data$patient_id] +
  health_count[train_data$patient_id])
beta_step2 <- coef(step2)
beta_step2
```

```
##                (Intercept)                bhatx[train_data$patient_id]
##                0.8210200                0.1048654
##                S[train_data$patient_id] health_count[train_data$patient_id]
##                0.1310581                -0.1312036
```

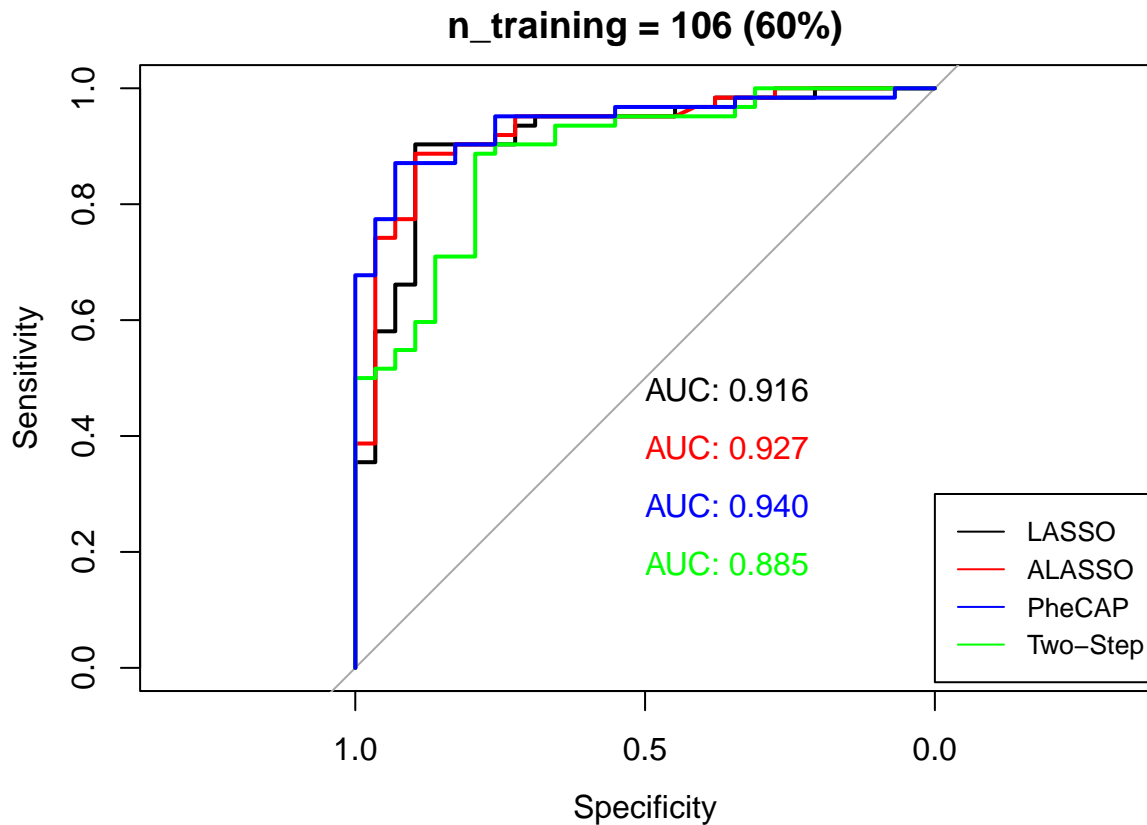
```
# recover beta
beta <- beta_step2[2] * beta.step1
# mu
mu <- beta_step2[1] +
  as.numeric(as.matrix(x[test_data$patient_id, ]) %*% beta[-1]) +
  as.numeric(beta_step2[3] %*% S[test_data$patient_id]) +
  as.numeric(beta_step2[4] %*% health_count[test_data$patient_id])
# expit
y_hat.ss <- plogis(mu)
```

```
plot(roc(test_y, y_hat.lasso),
  print.auc = TRUE, main = "n_training = 106 (60%)")
)
plot(roc(test_y, y_hat.alasso),
  print.auc = TRUE, col = "red", add = TRUE, print.auc.y = 0.4
)
plot(roc(test_y, y_hat.ss),
  print.auc = TRUE, col = "green", add = TRUE, print.auc.y = 0.2
)
plot(roc(test_y, y_hat.phecap),
  print.auc = TRUE, col = "blue", add = TRUE, print.auc.y = 0.3
)
legend(0, 0.3,
```

```

legend = c("LASSO", "ALASSO", "PheCAP", "Two-Step"),
col = c("black", "red", "blue", "green"),
lty = 1, cex = 0.8
)

```



```

ss.roc.full <- get_roc(test_y, y_hat.ss)
head(ss.roc.full, 10)

```

##	cutoff	pos.rate	FPR	TPR	PPV	NPV	F1
## [1,]	0.7809459	0.005494505	0.00000000	0.2421875	1.0000000	0.3816574	0.3899371
## [2,]	0.7353110	0.120879121	0.00000000	0.3917188	1.0000000	0.4346950	0.5629280
## [3,]	0.7018829	0.351648352	0.03448276	0.5012903	0.9688279	0.4752206	0.6607143
## [4,]	0.7004976	0.351648352	0.03448276	0.5059677	0.9691072	0.4775712	0.6648299
## [5,]	0.6984198	0.362637363	0.03448276	0.5106452	0.9693815	0.4799451	0.6689203
## [6,]	0.6957793	0.362637363	0.03448276	0.5153226	0.9696510	0.4823428	0.6729858
## [7,]	0.6943992	0.373626374	0.06896552	0.5238710	0.9419954	0.4777070	0.6733002
## [8,]	0.6931379	0.384615385	0.06896552	0.5332258	0.9429549	0.4826600	0.6812281
## [9,]	0.6906298	0.384615385	0.06896552	0.5425806	0.9438833	0.4877168	0.6890619
## [10,]	0.6885717	0.406593407	0.10344828	0.5537097	0.9196357	0.4844420	0.6912312

Model Evaluation

```

start <- Sys.time()
auc_twostep <- validate_ss(
  dat = labeled_data, nsim = 600,
  n.train = c(50, 70, 90),
  beta = beta.step1,

```

```

    S = S,
    x = x
  )
end <- Sys.time()
end - start

## Time difference of 38.21492 secs

# median AUC
apply(auc_twostep, 2, median)

## n=50,Two-Step n=70,Two-Step n=90,Two-Step
##    0.9002924    0.9005848    0.9062500

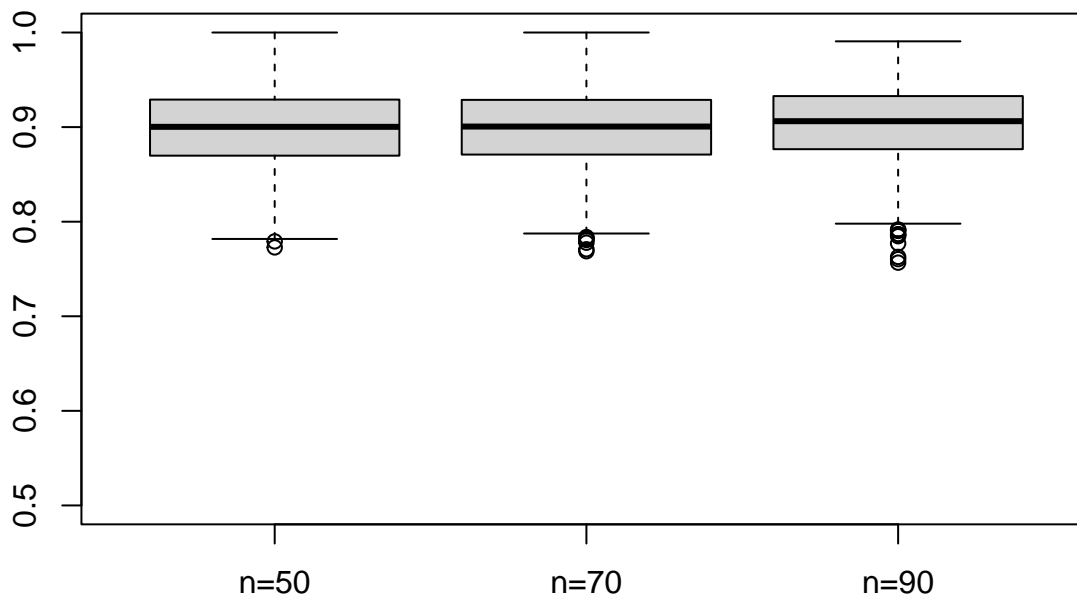
# se
apply(auc_twostep, 2, sd)

## n=50,Two-Step n=70,Two-Step n=90,Two-Step
##    0.04440317    0.04352198    0.04283354

boxplot(auc_twostep,
  ylim = c(0.5, 1), names = c("n=50", "n=70", "n=90"),
  main = "Two-Step Semi-Supervised"
)

```

Two-Step Semi-Supervised

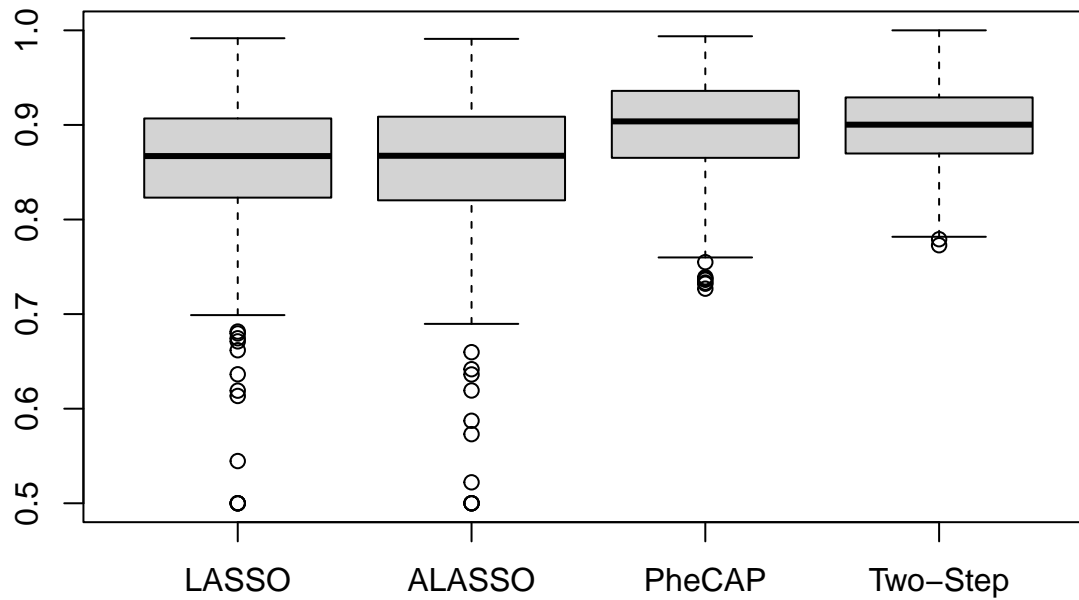


```

# Compare with Previous method
boxplot(cbind(auc_supervised, auc_phecap, auc_twostep)
  %>% select(starts_with("n=50")),
  ylim = c(0.5, 1), names = c("LASSO", "ALASSO", "PheCAP", "Two-Step")
, main = "n=50"
)

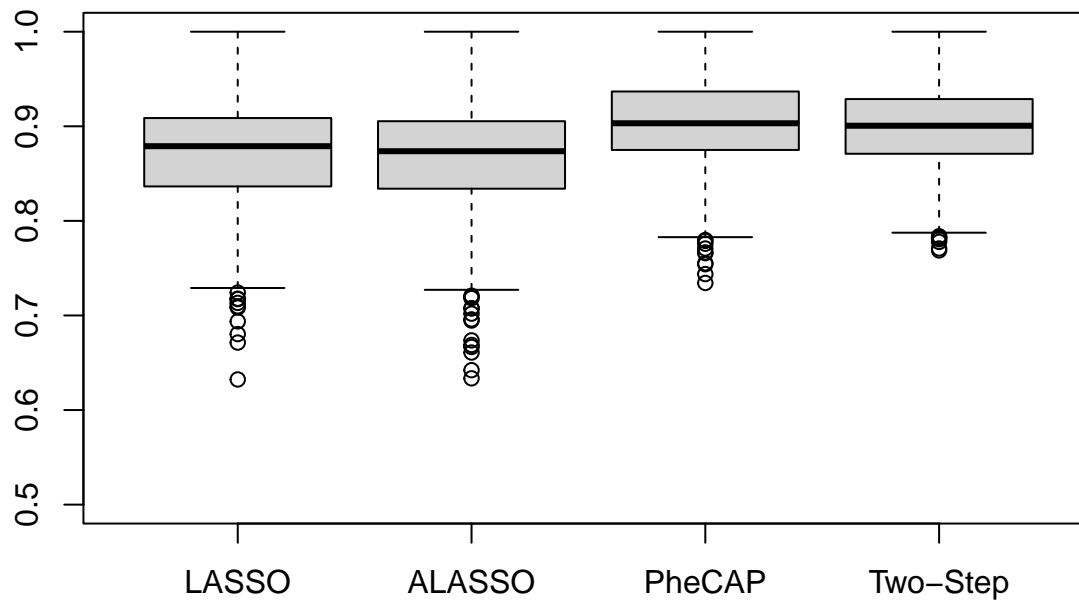
```

n=50

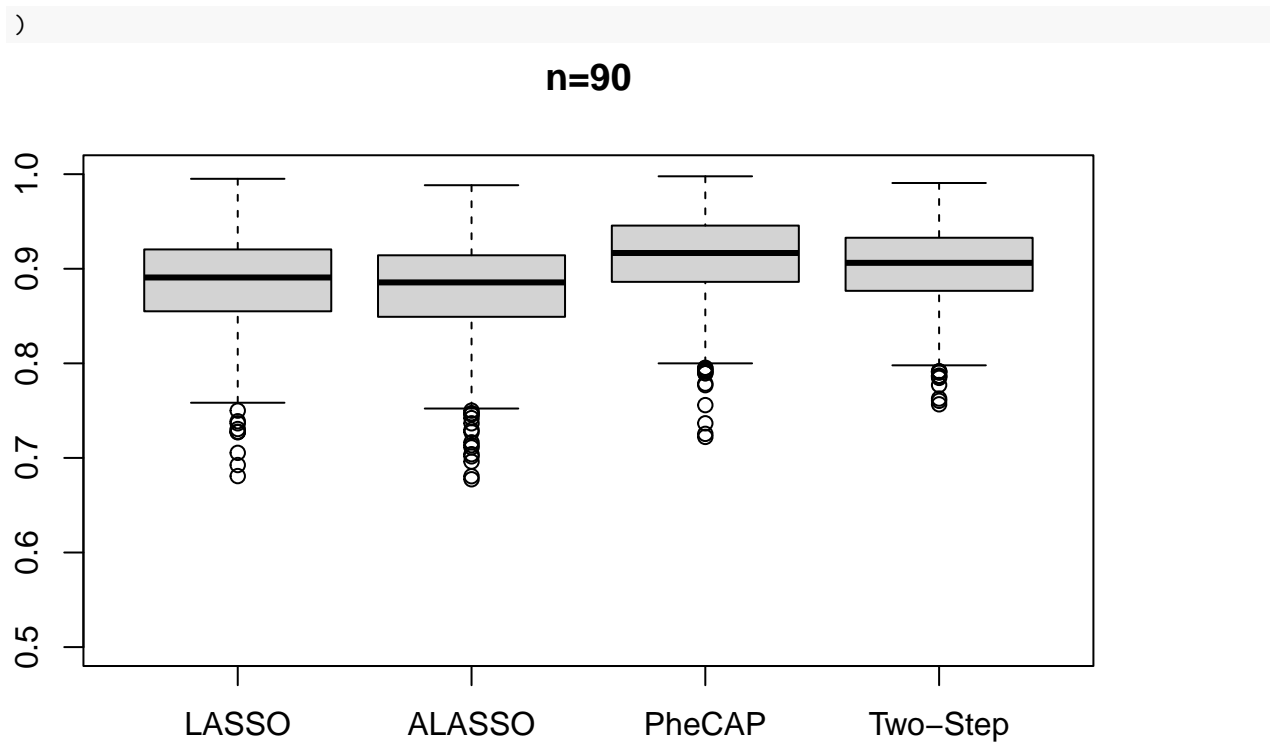


```
boxplot(cbind(auc_supervised, auc_phecap, auc_twostep)
%>% select(starts_with("n=70")),
ylim = c(0.5, 1), names = c("LASSO", "ALASSO", "PheCAP", "Two-Step"),
main = "n=70"
)
```

n=70



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



Weakly supervised learning

```
model_phenorm <- PheNorm.Prob(
  nm.logS.ori = "main_ICDNLP", # name of surrogates
  nm.utl = "healthcare_utilization", # name of HU
  nm.X = colnames(ehr_data)[-1:-6], # Other predictors X
  dat = ehr_data %>% select(-patient_id, -main_ICD, -main_NLP),
  train.size = nrow(ehr_data)
)

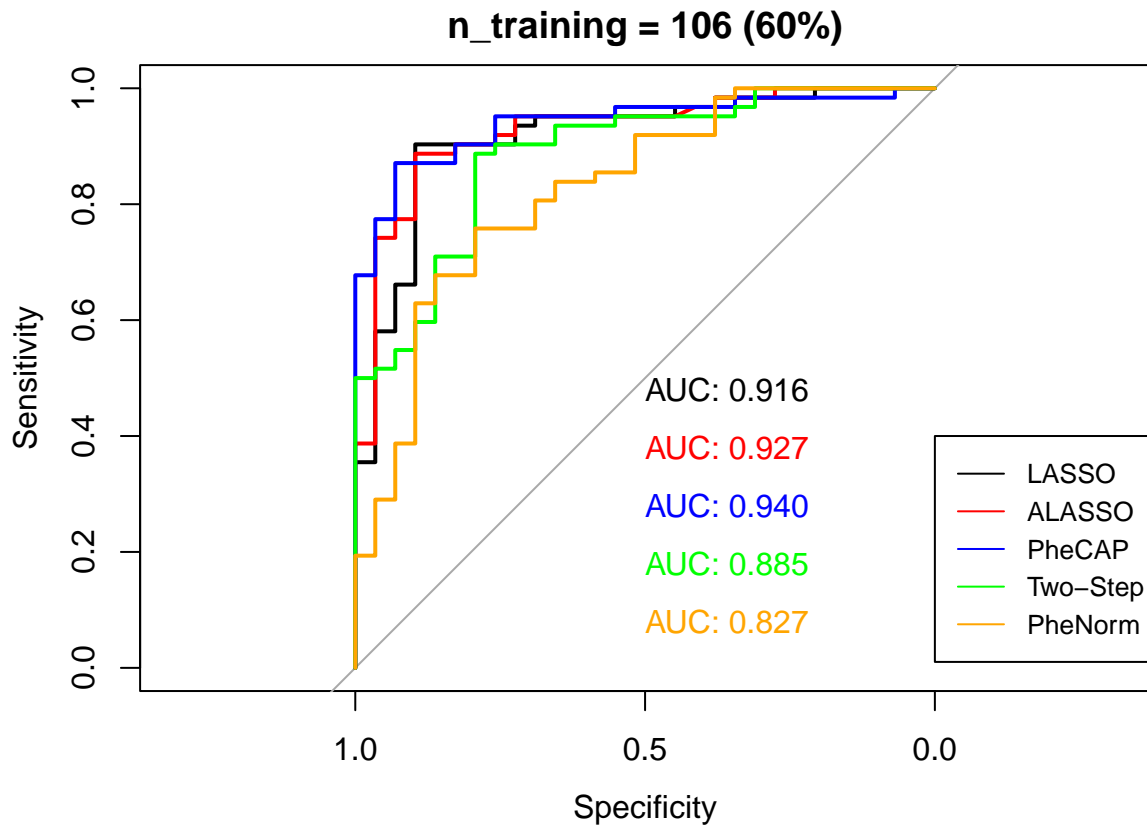
y_hat.phenorm <- model_phenorm$probs[data$validation_set]
plot(roc(test_y, y_hat.lasso),
  print.auc = TRUE, main = "n_training = 106 (60%)"
)
plot(roc(test_y, y_hat.lasso),
  print.auc = TRUE, col = "red", add = TRUE, print.auc.y = 0.4
)
plot(roc(test_y, y_hat.phecap),
  print.auc = TRUE, col = "blue", add = TRUE, print.auc.y = 0.3
)
plot(roc(test_y, y_hat.ss),
  print.auc = TRUE, col = "green", add = TRUE, print.auc.y = 0.2
)
plot(roc(test_y, y_hat.phenorm),
  print.auc = TRUE, col = "orange", add = TRUE, print.auc.y = 0.1
)

legend(0, 0.4,
```

```

legend = c("LASSO", "ALASSO", "PheCAP", "Two-Step", "PheNorm"),
col = c("black", "red", "blue", "green", "orange"),
lty = 1, cex = 0.8
)

```



```

# Use untransformed data; MAP requires sparse matrix
# Create sparse matrix for surrogates
data_fit <- sparsify(PheCAP::ehr_data %>%
  select(main_ICD, main_NLP) %>%
  rename(ICD = main_ICD) %>% data.table())

# Create sparse matrix for HU
note <- Matrix(PheCAP::ehr_data$healthcare_utilization, ncol = 1, sparse = TRUE)
model_map <- MAP(mat = data_fit, note = note, full.output = TRUE)

```

```

## #####
## MAP only considers patients who have note count data and
##       at least one nonmissing variable!
## #####
## Here is a summary of the input data:
## Total number of patients: 10000
##   ICD main_NLP note   Freq
## 1 YES      YES   YES 10000
## #####

```

```

y_hat.map <- model_map$scores[data$validation_set]

```

```

plot(roc(test_y, y_hat.lasso),

```

```

    print.auc = TRUE, main = "n_training = 106 (60%)")
)
plot(roc(test_y, y_hat.lasso),
     print.auc = TRUE, col = "red", add = TRUE, print.auc.y = 0.4
)
plot(roc(test_y, y_hat.phecap),
     print.auc = TRUE, col = "blue", add = TRUE, print.auc.y = 0.3
)
plot(roc(test_y, y_hat.ss),
     print.auc = TRUE, col = "green", add = TRUE, print.auc.y = 0.2
)
plot(roc(test_y, y_hat.phenorm),
     print.auc = TRUE, col = "orange", add = TRUE, print.auc.y = 0.1
)
plot(roc(test_y, y_hat.map),
     print.auc = TRUE, col = "orchid", add = TRUE, print.auc.y = 0
)

legend(0, 0.4,
      legend = c("LASSO", "ALASSO", "PheCAP", "Two-Step", "PheNorm", "MAP"),
      col = c("black", "red", "blue", "green", "orange", "orchid"),
      lty = 1, cex = 0.8
)

```

