Final Project STAT117

Kent Codding

2025-04-17

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.4.2
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 4.4.2
```

```
library(rjags)
```

```
## Warning: package 'rjags' was built under R version 4.4.2
```

```
## Loading required package: coda
```

```
## Warning: package 'coda' was built under R version 4.4.2
```

```
## Linked to JAGS 4.3.1
```

```
## Loaded modules: basemod,bugs
```

```
library(coda)
library(jagsUI)
```

```
## Warning: package 'jagsUI' was built under R version 4.4.2
```

```
##
## Attaching package: 'jagsUI'
```

```
## The following object is masked from 'package:coda':
##
##     traceplot
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.4.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.4.2
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.3
```

```
library(ggridges)
```

```
## Warning: package 'ggridges' was built under R version 4.4.3
```

```
load("C:/Users/khcod/Downloads/pCR_binary.RData")
```

```
baseline_m = glm(pCR[[6]] ~ pam50_pCR[[6]], family=binomial(link="logit"))
baseline_AUC = performance(prediction(fitted(baseline_m), baseline_m$y), "auc")@y.values[[1]]
cat('baseline binary AUC threshold is: ', baseline_AUC)
```

```
## baseline binary AUC threshold is:  0.7537775
```

Study-wise EDA

perform gene-selection using mas-o-menos

```
masomenos.train = function(X,    # pxn data frame of predictors
                           Y,    # nx1 vectors of labels or responses
                           P,    # number of predictors
                           training.criterion="AUC",
                           filtering.fraction=.5)
{
  # eliminate unclassified samples
  # X <- matTraining
  # Y <- YY
  YY = Y[!is.na(Y)]
  XX = X[,!is.na(Y)]

  Nvar = nrow(XX) # Number of genes
  crite = rep(NA,Nvar) # These are to store the criterion for each gene.
  if (training.criterion=="AUC"){
    for (pp in 1:Nvar){
      crite[pp] <- as.numeric( wilcox.test(XX[pp,]~YY)$statistic / (sum(YY==0)*sum(YY==1)) )
      #XX[pp,] gives data values, YY gives levels of the groups to be compared.
    }
  }
  # P = 12
  cutoff = sort(abs(crite- 0.5) + 0.5,decreasing = TRUE)[P]
  # cutoff = sort(abs(crite),decreasing = TRUE)[P]
  # sort the absolute value of the criterion from largest to smallest, then look at the corresponding c
  cutoff
  variables = (1:Nvar)[abs(crite- 0.5) + 0.5 >= cutoff]
  # retrieve the variable name/number of the top P.
  variables
```

```
  variables.signs = ( 2 * ( crite > 0.5 ) - 1 ) [variables]
  scores = apply ( XX[ variables, ] * variables.signs, 2, mean )
  # this is known as the risk score, or the mean expression value for the top P genes for each patient.

  if (training.criterion=="AUC"){
    crite.mom = as.numeric( wilcox.test(scores~YY)$statistic / (sum(YY==0)*sum(YY==1)) )
  }

  MoM = list(XX=XX,YY=YY,cutoff=cutoff,
             training.criterion=training.criterion,
             variables = variables,
             variables.signs=variables.signs,
             variables.criterion=crite[variables],
             scores=scores,
             criterion.mom=crite.mom)
  return(MoM)
}

masomenos.test = function(X,   # pxn data frame of predictors
                          Y,   # nx1 vectors of labels or responses
                          MoM.out # output form masomenos.train
){
  # eliminate unclassified samples
  # Y = 1*as.vector(testingGroup=="Good")
  # X = matTesting
  # X = matTesting[genename,]
  # MoM.out = MoM.train
  YY = Y[!is.na(Y)]
  XX = X[,!is.na(Y)]

  Nvar = nrow(XX)
  scores = apply ( XX[ MoM.out$variables, ] * MoM.out$variables.signs, 2, mean )

  if (MoM.out$training.criterion=="AUC"){
    crite.mom = as.numeric( wilcox.test(scores~YY)$statistic / (sum(YY==0)*sum(YY==1)) )
  }

  MoM = list(XX=XX,YY=YY,
             scores=scores,
             criterion.mom=crite.mom)
  return(MoM)
}
```

First, try on study 1

```
YY_pcr <- pCR[[1]]

# Use the masomenos.train function to select the top 10 discriminatory genes.
mom_pcr <- masomenos.train(X = XX_pCR[[1]], Y = YY_pcr, P = 10, training.criterion = "AUC")

selected_genes <- mom_pcr$variables

cat("The top 10 most discriminatory genes are (by row index):\n")
```

```
## The top 10 most discriminatory genes are (by row index):
```

```
print(selected_genes)
```

```
##  [1] 2003 2042 2103 4135 5363 5539 6481 8511 9642 9671
```

```
if (!is.null(rownames(XX_pCR[[1]]))) {
  selected_gene_names_study1 <- rownames(XX_pCR[[1]])[selected_genes]
  cat("The top 10 most discriminatory gene names are:\n")
  print(selected_gene_names_study1)
}
```

```
## The top 10 most discriminatory gene names are:
##  [1] "GARS"     "PAPSS1"   "LSM2"     "ARHGEF10" "DSG2"     "EPB41L2"
##  [7] "ITPR3"    "RXRB"     "TUSC3"    "UBE2G2"
```

```
XX_all_pCR <- do.call(cbind, XX_pCR)

# pCR is a list of binary response vectors (one per study).
YY_all_pCR <- unlist(pCR)

# Check dimensions of the merged data
cat("Dimensions of merged expression matrix (genes x total patients): ", dim(XX_all_pCR), "\n")
```

```
## Dimensions of merged expression matrix (genes x total patients):  10115 1309
```

```
cat("Length of merged response vector: ", length(YY_all_pCR), "\n")
```

```
## Length of merged response vector:  1309
```

```
mom_pcr_all <- masomenos.train(X = XX_all_pCR,
                               Y = YY_all_pCR,
                               P = 10,
                               training.criterion = "AUC")

# Retrieve the selected gene indices.
selected_genes <- mom_pcr_all$variables
cat("The top 10 most discriminatory genes (by row indices) are:\n")
```

```
## The top 10 most discriminatory genes (by row indices) are:
```

```
print(selected_genes)
```

```
##  [1] 4223 4371 4641 4659 5103 5316 5876 6340 6961 8864
```

```
# If the matrix XX_all_pCR has row names (gene names), extract them.
if (!is.null(rownames(XX_all_pCR))) {
  selected_gene_names_allstudies <- rownames(XX_all_pCR)[selected_genes]
  cat("The top 10 most discriminatory gene names are:\n")
  print(selected_gene_names_allstudies)
}
```

```
## The top 10 most discriminatory gene names are:
##  [1] "ATP1B3" "BMPR1B" "CCND1"  "CCT4"   "CYB5R1" "DNALI1" "GAMT"   "IDUA"
##  [9] "MDH1"   "SLC7A8"
```

```
any(selected_gene_names_allstudies %in% selected_gene_names_study1)
```

```
## [1] FALSE
```

No genes were consistent from study 1 to all using mas-o-menos to select, so it is worthwhile to check for study heterogeneity for biomarkers.

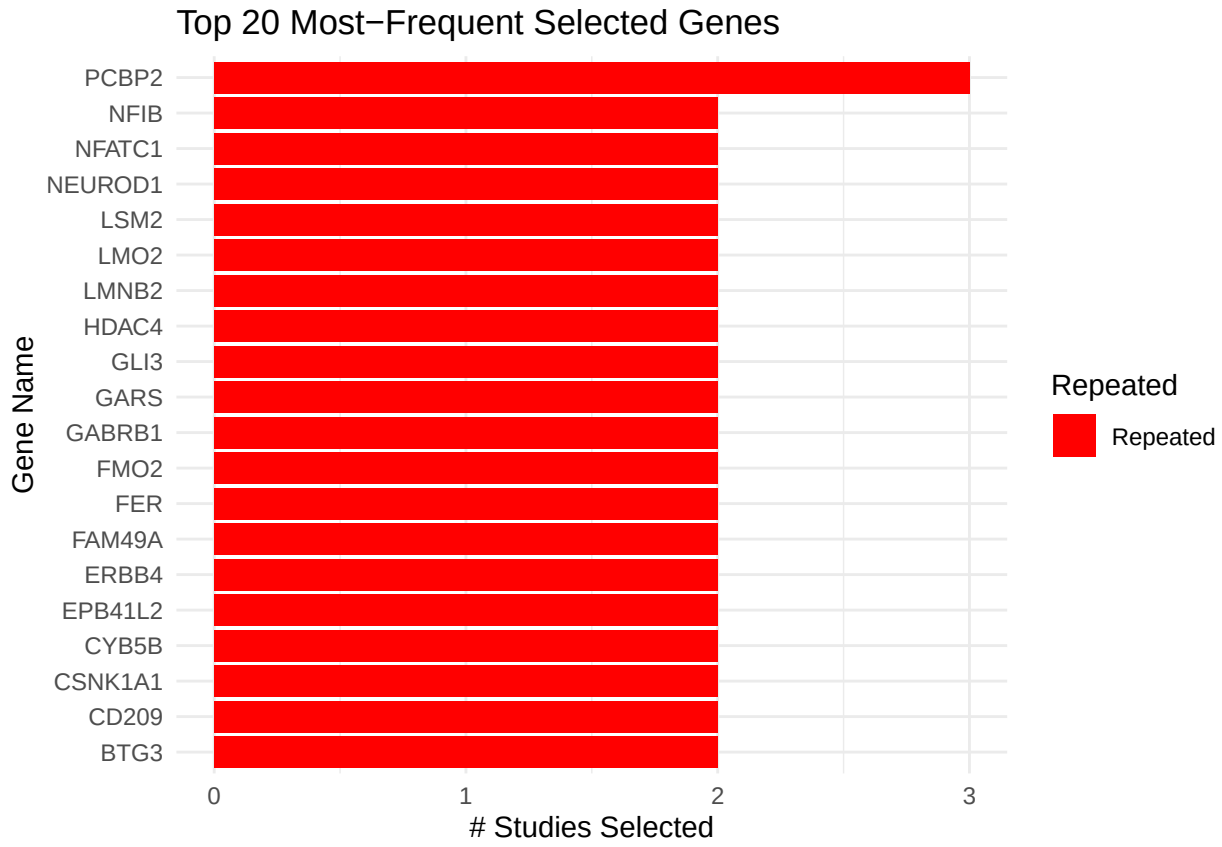
```
selected_genes_list2 <- list()
counter <- 0  # counter for valid studies

for (i in seq_along(XX_pCR)) {
  Y_study <- pCR[[i]]
  if (length(unique(Y_study)) < 2) {
    cat("Study", i, "does not have two outcome levels. Skipping.\n")
    next
  }
  mom_out_i <- masomenos.train(X = XX_pCR[[i]], Y = Y_study, P = 10, training.criterion = "AUC")
  gene_indices <- mom_out_i$variables
  gene_names   <- rownames(XX_pCR[[i]])[gene_indices]
  counter      <- counter + 1
  selected_genes_list2[[counter]] <- gene_names
}
```

```
## Study 9 does not have two outcome levels. Skipping.
## Study 11 does not have two outcome levels. Skipping.
```

```
if (length(selected_genes_list2) > 0) {
  # collapse to one long vector and tabulate
  all_selected_genes <- unlist(selected_genes_list2)
  gene_freq          <- table(all_selected_genes)

  # take top 20 by frequency
  top20            <- sort(gene_freq, decreasing = TRUE)[1:20]
  freq_df_top20    <- data.frame(
    Gene      = names(top20),
    Frequency = as.integer(top20),
    row.names = NULL
  )
  freq_df_top20$Repeated <- ifelse(freq_df_top20$Frequency > 1, "Repeated", "Unique")

  # print just those 20
  print(freq_df_top20)

  # plot only the top 20
  ggplot(freq_df_top20, aes(x = reorder(Gene, Frequency), y = Frequency, fill = Repeated)) +
    geom_col() +
    coord_flip() +
    labs(title = "Top 20 Most-Frequent Selected Genes",
         x = "Gene Name", y = "# Studies Selected") +
    scale_fill_manual(values = c("Repeated" = "red", "Unique" = "grey70")) +
    theme_minimal()
}
```

```
##       Gene Frequency Repeated
## 1    PCBP2         3 Repeated
## 2     BTG3         2 Repeated
## 3    CD209         2 Repeated
## 4  CSNK1A1         2 Repeated
## 5    CYB5B         2 Repeated
## 6  EPB41L2         2 Repeated
```

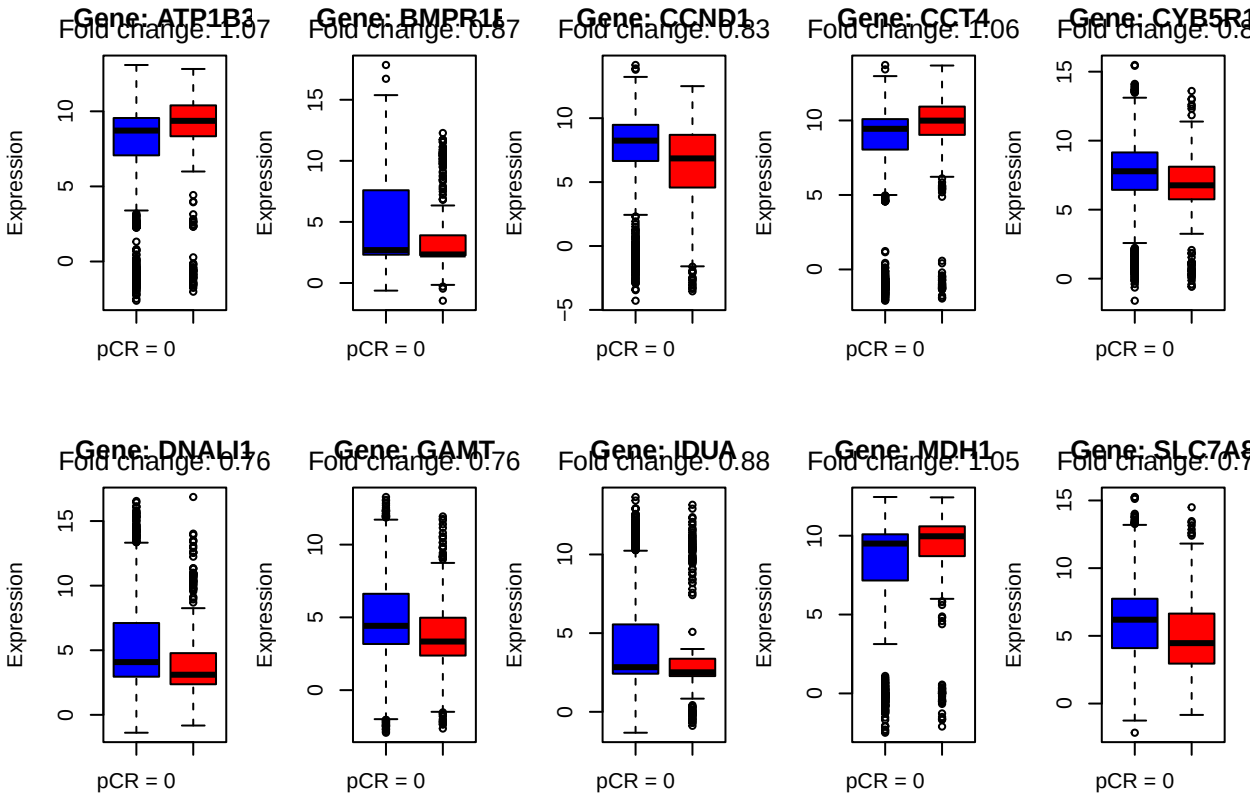
```
## 7    ERBB4         2 Repeated
## 8   FAM49A         2 Repeated
## 9      FER         2 Repeated
## 10    FMO2         2 Repeated
## 11  GABRB1         2 Repeated
## 12    GARS         2 Repeated
## 13    GLI3         2 Repeated
## 14   HDAC4         2 Repeated
## 15   LMNB2         2 Repeated
## 16    LMO2         2 Repeated
## 17    LSM2         2 Repeated
## 18 NEUROD1         2 Repeated
## 19  NFATC1         2 Repeated
## 20    NFIB         2 Repeated
```



Visualize 10 selected genes

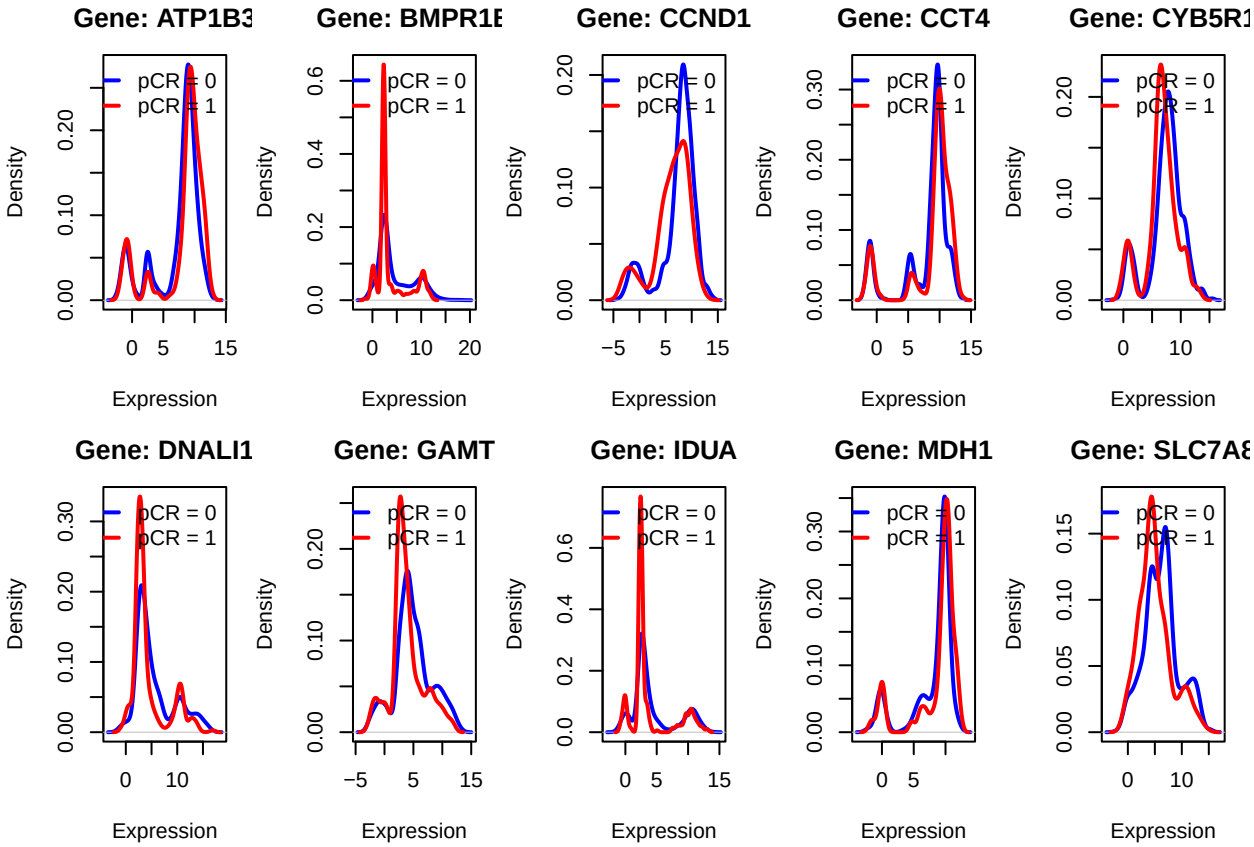
```r
par(mfrow = c(2, 5), mar = c(4, 4, 3, 1))

# Loop over each gene and produce a boxplot comparing pCR=0 and pCR=1
for(gene in selected_gene_names_allstudies) {

  # Extract the gene expression values for the gene.
  gene_expr <- as.numeric(XX_all_pCR[gene, ])

  # Split the values by pCR outcome and remove missing values.
```

```
  expr_0 <- gene_expr[YY_all_pCR == 0]
  expr_1 <- gene_expr[YY_all_pCR == 1]
  expr_0 <- expr_0[!is.na(expr_0)]
  expr_1 <- expr_1[!is.na(expr_1)]

  # Create a list where each element corresponds to an outcome group
  expr_by_group <- list("pCR = 0" = expr_0, "pCR = 1" = expr_1)

  # Create the boxplot
  boxplot(expr_by_group,
          main = paste("Gene:", gene),
          ylab = "Expression",
          col = c("blue", "red"))

  # add a note about fold change by comparing medians.
  median0 <- median(expr_0, na.rm = TRUE)
  median1 <- median(expr_1, na.rm = TRUE)
  fold_change <- median1 / median0
  mtext(paste("Fold change:", round(fold_change, 2)), side = 3, line = 0.5, cex = 0.8)
}
```




```
# Set up a 2 x 5 plotting space for the 10 genes.
par(mfrow = c(2, 5), mar = c(4, 4, 3, 1))

# Loop over each of the 10 selected genes and produce density plots.
for(gene in selected_gene_names_allstudies) {
```

```
  # Extract the gene expression values for the current gene.
  gene_expr <- as.numeric(XX_all_pCR[gene, ])

  # For each outcome, remove NAs before computing the density.
  expr_0 <- gene_expr[YY_all_pCR == 0]
  expr_0 <- expr_0[!is.na(expr_0)]

  expr_1 <- gene_expr[YY_all_pCR == 1]
  expr_1 <- expr_1[!is.na(expr_1)]

  # Check that there are enough observations to calculate a density.
  if(length(expr_0) < 2 || length(expr_1) < 2) {
    cat("Gene", gene, "does not have enough non-missing values for both outcomes. Skipping plot.\n")
    next
  }

  # Calculate density estimates for each outcome.
  density_0 <- density(expr_0)
  density_1 <- density(expr_1)

  # Determine the common x and y limits for the plot.
  xlim_range <- range(c(density_0$x, density_1$x))
  ylim_range <- range(c(density_0$y, density_1$y))

  # Plot the density for pCR = 0 in blue.
  plot(density_0, xlim = xlim_range, ylim = ylim_range,
       main = paste("Gene:", gene),
       xlab = "Expression", ylab = "Density",
       col = "blue", lwd = 2)

  # Overlay the density for pCR = 1 in red.
  lines(density_1, col = "red", lwd = 2)

  # Add a legend to the plot.
  legend("topright", legend = c("pCR = 0", "pCR = 1"),
         col = c("blue", "red"), lwd = 2, bty = "n")
}
```



The multimodality of each of these density plots suggest that different studies may be responsible for different peaks in gene expression for both outcomes. Further, this suggests that expression for many of these genes are on a different scale, providing motivation for a hierarchical unpooled model for intercept and slope. The mas-o-menos gene selection for 10 genes produces an AUC of

```
mom_pcr_all$criterion.mom
```

```
## [1] 0.6957664
```

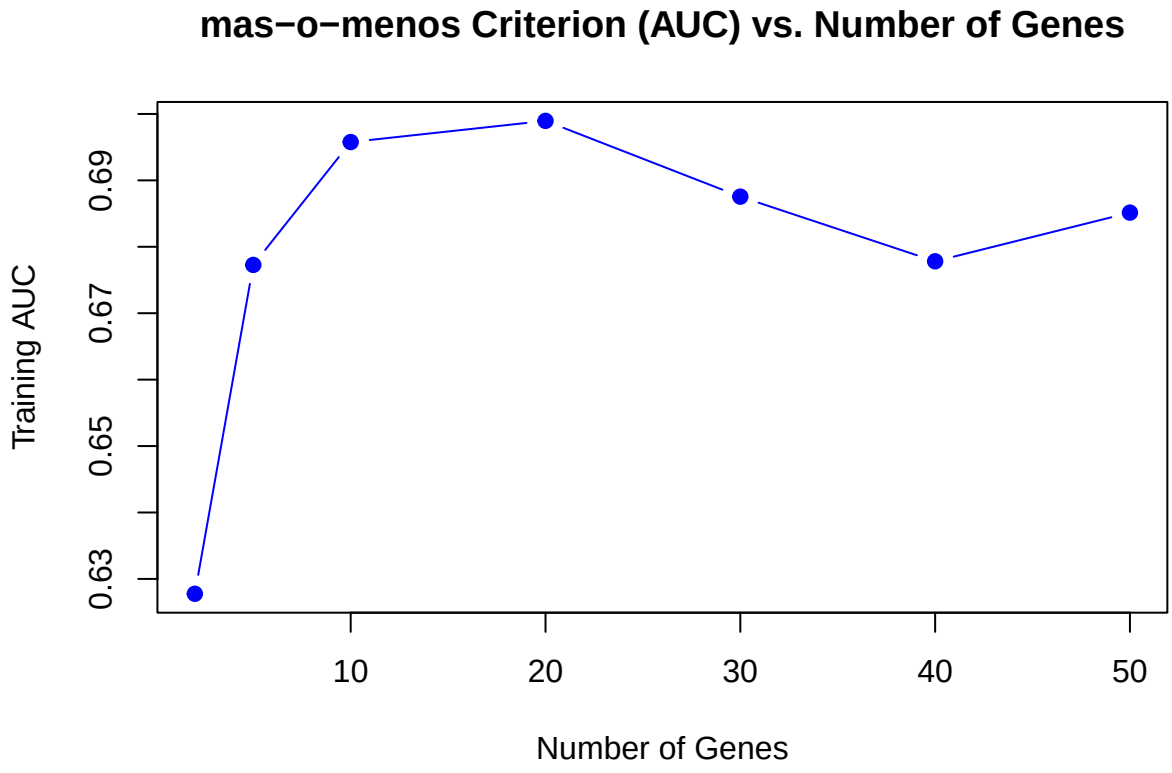
Plot mas-o-menos for 2,5,10,20,30,40,50 genes

```
# Define the vector of gene counts to try.
genes_to_try <- c(2, 5, 10, 20, 30, 40, 50)

# Initialize a vector to store the AUC (criterion.mom) for each gene count.
auc_values <- numeric(length(genes_to_try))

# Loop over the gene counts.
for (i in seq_along(genes_to_try)) {
  P <- genes_to_try[i]

  # Run the mas-o-menos training procedure.
  # The function calculates a Wilcoxon AUC for the risk scores generated using the top P genes.
  mom_out <- masomenos.train(X = XX_all_pCR, Y = YY_all_pCR, P = P, training.criterion = "AUC")

  auc_values[i] <- mom_out$criterion.mom
```

```
  cat("For", P, "genes, training AUC:", round(auc_values[i], 3), "\n")
}
```

```
## For 2 genes, training AUC: 0.628
## For 5 genes, training AUC: 0.677
## For 10 genes, training AUC: 0.696
## For 20 genes, training AUC: 0.699
## For 30 genes, training AUC: 0.688
## For 40 genes, training AUC: 0.678
## For 50 genes, training AUC: 0.685
```

```
# Plot the AUC versus the number of genes.
plot(genes_to_try, auc_values, type = "b",
     xlab = "Number of Genes",
     ylab = "Training AUC",
     main = "mas-o-menos Criterion (AUC) vs. Number of Genes",
     pch = 19, col = "blue")
```



This plot shows the expected plateau after more features, genes in this case, increase noise that dilutes the signal of the most discriminative genes, suggesting that the optimal number of genes to maximize AUC is likely between 10 and 20.

Pooled Logistic Regression Model

```
# Subset the matrix to the 10 genes
X_data <- t(XX_all_pCR[selected_gene_names_allstudies, ])
```

```r
# Create a data frame and add the outcome
data_lr <- data.frame(X_data)
data_lr$pCR <- factor(YY_all_pCR)  # factors with levels "0" and "1"

# Fit a logistic regression model using all 10 gene expressions as predictors.
model_logistic <- glm(pCR ~ ., data = data_lr, family = binomial, na.action = na.exclude)

summary(model_logistic)
```

```
##
## Call:
## glm(formula = pCR ~ ., family = binomial, data = data_lr, na.action = na.exclude)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.43289    0.24051  -5.958 2.56e-09 ***
## ATP1B3       0.04559    0.06708   0.680 0.496763
## BMPR1B       0.01568    0.05571   0.281 0.778416
## CCND1       -0.25084    0.04188  -5.990 2.10e-09 ***
## CCT4         0.09347    0.07399   1.263 0.206480
## CYB5R1      -0.04397    0.06387  -0.688 0.491165
## DNALI1       0.01095    0.05788   0.189 0.849940
## GAMT        -0.26680    0.06579  -4.055 5.00e-05 ***
## IDUA         0.25816    0.06998   3.689 0.000225 ***
## MDH1         0.22659    0.06727   3.368 0.000757 ***
## SLC7A8      -0.15097    0.05433  -2.779 0.005456 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1371.4  on 1302  degrees of freedom
## Residual deviance: 1195.8  on 1292  degrees of freedom
##   (6 observations deleted due to missingness)
## AIC: 1217.8
##
## Number of Fisher Scoring iterations: 4
```

```r
pred_probs <- predict(model_logistic, type = "response")

roc_obj <- roc(response = data_lr$pCR, predictor = pred_probs)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
auc_value <- auc(roc_obj)

cat("The AUC for the logistic regression model using the 10 selected genes is:", round(auc_value, 3), "
```

```
## The AUC for the logistic regression model using the 10 selected genes is: 0.742
```

Leave-one-study-out

```
study_indicator <- unlist(lapply(seq_along(XX_pCR), function(j) rep(j, ncol(XX_pCR[[j]]))))


X_mat <- t(XX_all_pCR[selected_gene_names_allstudies, ])

# Build a data frame with the outcome, gene expression predictors, and study indicator.
data_hier <- data.frame(
  Y = YY_all_pCR,                # binary outcome (0/1)
  X_mat,
  ZZ = study_indicator
)

# Remove any rows with missing values.
data_hier <- data_hier[complete.cases(data_hier), ]
```

```
library(pROC)

data_df <- data_hier
study_numbers <- unique(data_df$ZZ)
auc_values_glm <- numeric(length(study_numbers))
auc_values_glm[] <- NA  # preallocate as NA

for (i in seq_along(study_numbers)) {
  test_study <- study_numbers[i]

  # Partition data: training is all studies except the current one; testing is the left-out study.
  train_data <- subset(data_df, ZZ != test_study)
  test_data  <- subset(data_df, ZZ == test_study)

  # Fit logistic regression on the training set.
  model_glm <- glm(Y ~ ., data = train_data[, c(selected_gene_names_allstudies, "Y")],
                   family = binomial)

  # Predict on the test set.
  preds <- predict(model_glm, newdata = test_data, type = "response")

  # Check if test_data has both outcome classes.
  if(length(unique(test_data$Y)) < 2){
    cat("Left-out Study", test_study, "does not have two outcome classes. Skipping AUC calculation.\n")
    next  # Skip this iteration.
  }

  # Calculate the AUC using the pROC package.
  roc_obj <- roc(response = test_data$Y, predictor = preds)
  auc_values_glm[i] <- auc(roc_obj)

  cat("Left-out Study", test_study, "AUC:", round(auc_values_glm[i], 3), "\n")
}
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

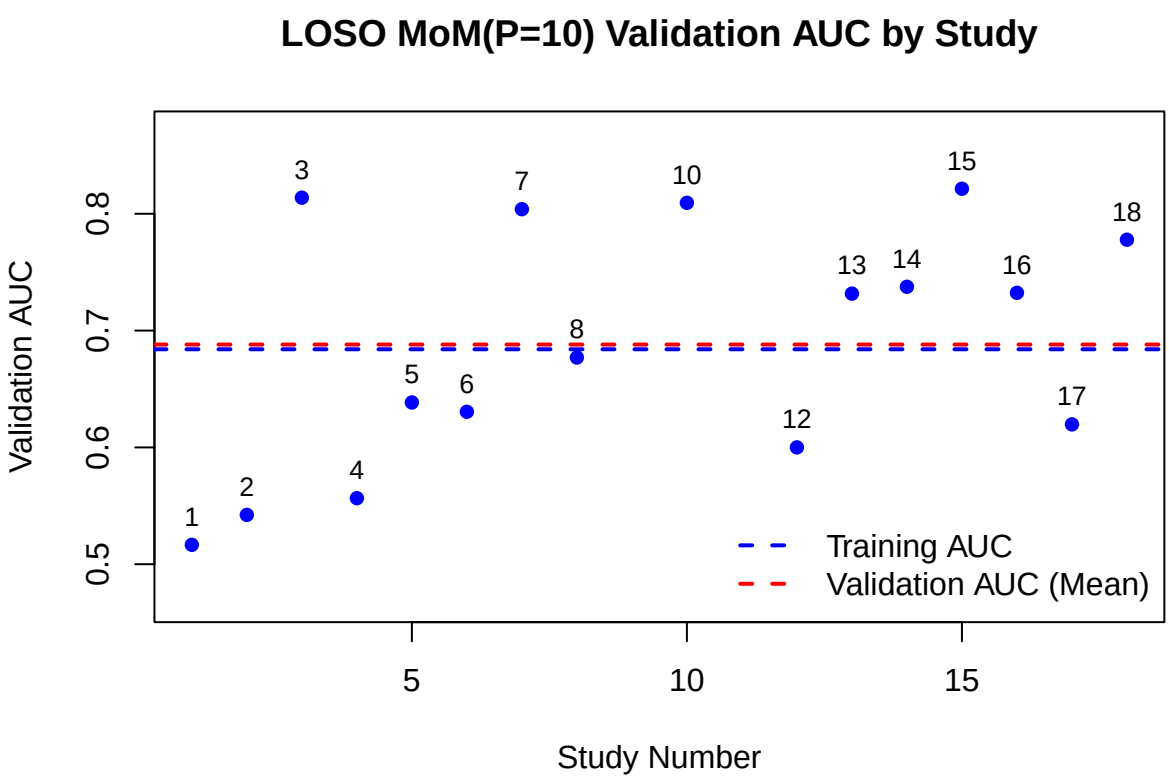
```
## Left-out Study 1 AUC: 0.517

## Setting levels: control = 0, case = 1
## Setting direction: controls > cases

## Left-out Study 2 AUC: 0.542

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Left-out Study 3 AUC: 0.814

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 4 AUC: 0.557

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 5 AUC: 0.638

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 6 AUC: 0.63

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 7 AUC: 0.804

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 8 AUC: 0.677
## Left-out Study 9 does not have two outcome classes. Skipping AUC calculation.

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 10 AUC: 0.809
## Left-out Study 11 does not have two outcome classes. Skipping AUC calculation.

## Setting levels: control = 0, case = 1

## Setting direction: controls > cases

## Left-out Study 12 AUC: 0.6

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Left-out Study 13 AUC: 0.732

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 14 AUC: 0.738

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 15 AUC: 0.821
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 16 AUC: 0.732

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 17 AUC: 0.62

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 18 AUC: 0.778
```

```
# Compute mean AUC across studies that produced a valid AUC
mean_auc_glm <- mean(auc_values_glm, na.rm = TRUE)
cat("Mean LOSO AUC (glm):", round(mean_auc_glm, 3), "\n")
```

```
## Mean LOSO AUC (glm): 0.688
```

```
# Remove studies with NA AUC values.
valid_indices <- !is.na(auc_values_glm)
valid_studies <- study_numbers[valid_indices]
valid_auc <- auc_values_glm[valid_indices]

# Create the scatter plot.
plot(valid_studies, valid_auc,
     pch = 16, col = "blue",
     xlab = "Study Number",
     ylab = "Validation AUC",
     main = "LOSO MoM(P=10) Validation AUC by Study",
     ylim = c(min(valid_auc) - 0.05, max(valid_auc) + 0.05))

# Add horizontal lines:
abline(h = mean_auc_glm, col = "red", lwd = 2, lty = 2)   # Mean Validation AUC
abline(h = 0.684, col = "blue", lwd = 2, lty = 2)          # Training AUC (0.684)

# add text labels for each study.
text(valid_studies, valid_auc, labels = valid_studies, pos = 3, cex = 0.8)

# Add a legend that labels the two horizontal lines.
legend("bottomright",
       legend = c("Training AUC", "Validation AUC (Mean)"),
       col = c("blue", "red"),
       lwd = 2, lty = 2,
       bty = "n")
```



This could mean study heterogeneity... motivating hierarchical unpooled modeling

IQR Filtering

```
XX_all_PCR <- do.call(cbind, XX_pCR)

data_all <- data.frame(t(XX_all_PCR))
data_all$ZZ <- study_indicator

# Compute IQR for each gene in XX_all_PCR
gene_iqr <- apply(XX_all_PCR, 1, IQR, na.rm = TRUE)

# Rank the genes by IQR (highest first) and select the top 50 indices
top50_indices <- order(gene_iqr, decreasing = TRUE)[1:50]

# Get the gene names corresponding to these top 50 genes
filtered_genes <- rownames(XX_all_PCR)[top50_indices]

# Create a filtered expression matrix with only these 50 genes
XX_filtered <- XX_all_PCR[top50_indices, ]

# Transpose so that each row is a patient. Then add the ZZ column.
data_filtered <- data.frame(t(XX_filtered))
data_filtered$ZZ <- study_indicator
```

Retry gene selection within ranked IQR subset

```
YY_all_PCR <- unlist(pCR)

filtered_genes <- rownames(XX_all_PCR)[top50_indices]

data_filtered <- data.frame(t(XX_filtered))
colnames(data_filtered) <- filtered_genes  # ensure gene columns are correctly named
data_filtered$ZZ <- study_indicator
data_filtered$Y  <- YY_all_PCR
```

Mas-o-menos

```
study_ids <- sort(unique(data_filtered$ZZ))
auc_mom <- numeric(length(study_ids))

for (s in study_ids) {
  # Partition data: training = all studies except s; test = study s.
  train_data <- subset(data_filtered, ZZ != s)
  test_data  <- subset(data_filtered, ZZ == s)

  # Convert outcomes to numeric if necessary.
  Y_train <- as.numeric(as.character(train_data$Y))
  Y_test  <- as.numeric(as.character(test_data$Y))

  # For mas-o-menos, prepare predictor matrices (genes are rows).
  X_train <- t(as.matrix(train_data[, filtered_genes]))
  X_test  <- t(as.matrix(test_data[, filtered_genes]))

  # Check if test data has both outcome classes.
  if(length(unique(Y_test)) < 2){
    cat("LOSO Study", s, "test data does not have two outcome classes. Skipping mas-o-menos AUC calcula
    auc_mom[s] <- NA
    next  # Move to the next study.
  }

  # Train mas-o-menos on training data using P = 10.
  mom_model <- masomenos.train(X = X_train, Y = Y_train, P = 10, training.criterion = "AUC")
  # Test the model on the left-out study.
  mom_test <- masomenos.test(X = X_test, Y = Y_test, MoM.out = mom_model)

  # Store the computed criterion/ AUC value.
  auc_mom[s] <- mom_test$criterion.mom

  cat("LOSO Study", s, "mas-o-menos AUC:", round(auc_mom[s], 3), "\n")
}
```

```
## LOSO Study 1 mas-o-menos AUC: 0.429
## LOSO Study 2 mas-o-menos AUC: 0.424
## LOSO Study 3 mas-o-menos AUC: 0.576
## LOSO Study 4 mas-o-menos AUC: 0.67
## LOSO Study 5 mas-o-menos AUC: 0.808
## LOSO Study 6 mas-o-menos AUC: 0.679
## LOSO Study 7 mas-o-menos AUC: 0.882
```

```
## LOSO Study 8 mas-o-menos AUC: 0.631
## LOSO Study 9 test data does not have two outcome classes. Skipping mas-o-menos AUC calculation.
## LOSO Study 10 mas-o-menos AUC: 0.744
## LOSO Study 11 test data does not have two outcome classes. Skipping mas-o-menos AUC calculation.
## LOSO Study 12 mas-o-menos AUC: 0.617
## LOSO Study 13 mas-o-menos AUC: 0.797
## LOSO Study 14 mas-o-menos AUC: 0.7
## LOSO Study 15 mas-o-menos AUC: 0.571
## LOSO Study 16 mas-o-menos AUC: 0.662
## LOSO Study 17 mas-o-menos AUC: 0.504
## LOSO Study 18 mas-o-menos AUC: 0.889
```

```
# Report mean LOSO mas-o-menos AUC (for studies with valid AUCs)
mean_auc_mom <- mean(auc_mom, na.rm = TRUE)
cat("Mean LOSO AUC for mas-o-menos:", round(mean_auc_mom, 3), "\n\n")
```

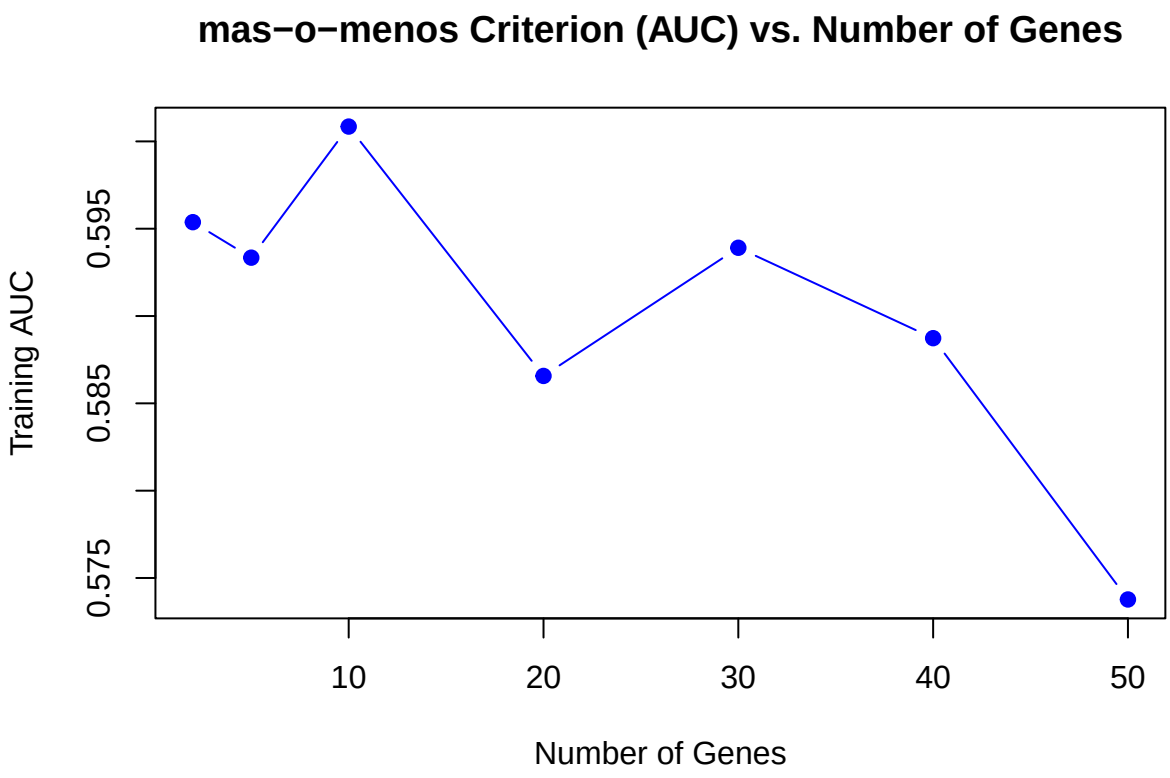
```
## Mean LOSO AUC for mas-o-menos: 0.661
```

```
# Define the vector of gene counts to try.
genes_to_try <- c(2, 5, 10, 20, 30, 40, 50)

auc_values <- numeric(length(genes_to_try))

# Loop over the gene counts.
for (i in seq_along(genes_to_try)) {
  P <- genes_to_try[i]

  # Prepare the expression matrix from data_filtered using the filtered genes.
  # Note: We transpose so that each row corresponds to a gene and each column to a patient.
  X_mat <- t(as.matrix(data_filtered[, filtered_genes]))

  # Use the outcome from data_filtered$Y.
  # (If Y is a factor, convert it to numeric as needed.)
  Y_vec <- as.numeric(as.character(data_filtered$Y))

  # Run the mas-o-menos training procedure using the filtered data.
  # P here sets the number of genes (from the filtered 50) to use.
  mom_out <- masomenos.train(X = X_mat, Y = Y_vec, P = P, training.criterion = "AUC")

  # Store the computed AUC in the vector.
  auc_values[i] <- mom_out$criterion.mom

  # Print progress.
  cat("For", P, "genes, training AUC:", round(auc_values[i], 3), "\n")
}
```

```
## For 2 genes, training AUC: 0.595
## For 5 genes, training AUC: 0.593
## For 10 genes, training AUC: 0.601
## For 20 genes, training AUC: 0.587
## For 30 genes, training AUC: 0.594
## For 40 genes, training AUC: 0.589
## For 50 genes, training AUC: 0.574
```

```
# Plot the AUC versus the number of genes.
plot(genes_to_try, auc_values, type = "b",
```

```
     xlab = "Number of Genes",
     ylab = "Training AUC",
     main = "mas-o-menos Criterion (AUC) vs. Number of Genes",
     pch = 19, col = "blue")
```




```
mom_pcr_all <- masomenos.train(X = t(as.matrix(data_filtered[, filtered_genes])),
                               Y = as.numeric(as.character(data_filtered$Y)),
                               P = 10,
                               training.criterion = "AUC")

# Retrieve the selected gene indices (relative to the set of filtered genes).
selected_genes <- mom_pcr_all$variables
cat("The top 10 most discriminatory genes (by row indices) are:\n")
```

```
## The top 10 most discriminatory genes (by row indices) are:
```

```
print(selected_genes)
```

```
##  [1]  4  6 16 19 21 27 31 33 35 37
```

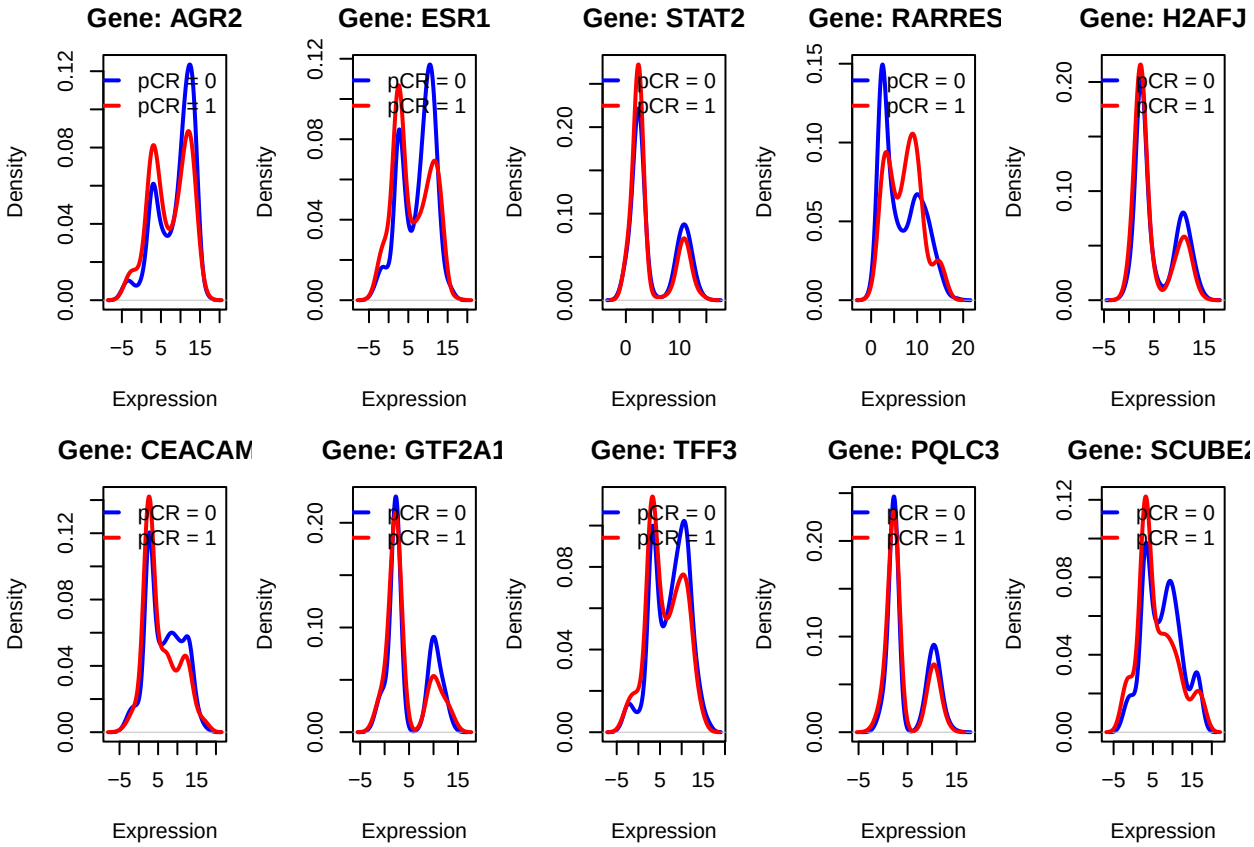
```
# select the top genes using the indices from masomenos.train.
selected_gene_names_allstudies <- filtered_genes[selected_genes]
cat("The top 10 most discriminatory gene names are:\n")
```

```
## The top 10 most discriminatory gene names are:
```

```
print(selected_gene_names_allstudies)
```

```
##  [1] "AGR2"    "ESR1"    "STAT2"   "RARRES1" "H2AFJ"   "CEACAM6" "GTF2A1"
##  [8] "TFF3"    "PQLC3"   "SCUBE2"
```

```r
# Set up a 2 x 5 plotting area.
par(mfrow = c(2, 5), mar = c(4, 4, 3, 1))

# Loop over each selected gene and produce a density plot.
for (gene in selected_gene_names_allstudies) {

  # Extract the gene expression values from data_filtered.
  # data_filtered contains columns with gene expression (names = filtered_genes).
  gene_expr <- as.numeric(data_filtered[[gene]])

  # Split the expression values by outcome (assumed coded as 0/1).
  expr_0 <- gene_expr[ data_filtered$Y == 0 ]
  expr_1 <- gene_expr[ data_filtered$Y == 1 ]

  # Remove missing values.
  expr_0 <- expr_0[ !is.na(expr_0) ]
  expr_1 <- expr_1[ !is.na(expr_1) ]

  # Check that each group has at least two observations.
  if (length(expr_0) < 2 || length(expr_1) < 2) {
    cat("Gene", gene, "does not have enough non-missing values for both outcomes. Skipping plot.\n")
    next
  }

  # Compute density estimates for each outcome group.
  density_0 <- density(expr_0)
  density_1 <- density(expr_1)

  # Determine common x and y limits for the plot.
  xlim_range <- range(c(density_0$x, density_1$x))
  ylim_range <- range(c(density_0$y, density_1$y))

  # Plot the density for pCR = 0 (blue).
  plot(density_0, xlim = xlim_range, ylim = ylim_range,
       main = paste("Gene:", gene),
       xlab = "Expression", ylab = "Density",
       col = "blue", lwd = 2)
  # Overlay the density for pCR = 1 (red).
  lines(density_1, col = "red", lwd = 2)

  # Add a legend to the plot.
  legend("topright", legend = c("pCR = 0", "pCR = 1"),
         col = c("blue", "red"), lwd = 2, bty = "n")
}
```



Pooled Logistic Regression Model:IQR subset

```
X_data <- t(XX_all_pCR[filtered_genes, ])

data_lr <- data.frame(X_data)
data_lr$pCR <- factor(YY_all_pCR)  # factors with levels "0" and "1"

model_logistic_IQR <- glm(pCR ~ ., data = data_lr, family = binomial, na.action = na.exclude)

summary(model_logistic_IQR)
```

```
##
## Call:
## glm(formula = pCR ~ ., family = binomial, data = data_lr, na.action = na.exclude)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.446063   0.353729  -4.088 4.35e-05 ***
## RPL23        0.166148   0.042525   3.907 9.34e-05 ***
## RPL28       -0.021596   0.033181  -0.651 0.515133
## SCGB2A2      0.055454   0.051280   1.081 0.279524
## AGR2        -0.097945   0.046897  -2.089 0.036753 *
## GNB2L1       0.021765   0.046590   0.467 0.640392
## ESR1        -0.047386   0.044656  -1.061 0.288634
## GABRP        0.024174   0.041666   0.580 0.561790
## COL11A1      0.086477   0.053886   1.605 0.108534
```

```
## RPS20       -0.053712   0.042078  -1.276 0.201782
## SULF1       -0.055502   0.066807  -0.831 0.406096
## LTF          0.005140   0.032533   0.158 0.874464
## PROM1        0.065489   0.043470   1.507 0.131930
## TFF1         0.029973   0.046500   0.645 0.519204
## ASPN        -0.220815   0.068387  -3.229 0.001243 **
## EIF4A1      -0.190889   0.049374  -3.866 0.000111 ***
## STAT2       -0.197720   0.087547  -2.258 0.023918 *
## DDX6         0.022719   0.126135   0.180 0.857060
## LUM         -0.095607   0.062391  -1.532 0.125430
## RARRES1      0.143391   0.041737   3.436 0.000591 ***
## VCAM1        0.104842   0.066245   1.583 0.113504
## H2AFJ       -0.016270   0.085424  -0.190 0.848945
## IFI44L      -0.005090   0.049874  -0.102 0.918707
## GAS1         0.039949   0.078199   0.511 0.609445
## CXCL9        0.033743   0.047310   0.713 0.475708
## DPT          0.162253   0.066605   2.436 0.014848 *
## MFAP5        0.031784   0.068634   0.463 0.643300
## CEACAM6     -0.009543   0.033402  -0.286 0.775103
## SCGB1D2     -0.064426   0.062651  -1.028 0.303795
## POSTN        0.013730   0.056555   0.243 0.808183
## COL15A1     -0.039667   0.071690  -0.553 0.580046
## GTF2A1       0.074622   0.137724   0.542 0.587941
## HLA.DQA1    -0.005849   0.033088  -0.177 0.859696
## TFF3         0.051605   0.054897   0.940 0.347201
## TFAP2B       0.056751   0.036107   1.572 0.116006
## PQLC3       -0.198219   0.136972  -1.447 0.147857
## SLC35F5      0.040472   0.113477   0.357 0.721350
## SCUBE2      -0.102863   0.046208  -2.226 0.026009 *
## ITGB8       -0.168379   0.094969  -1.773 0.076232 .
## CILP        -0.115753   0.068437  -1.691 0.090761 .
## ZBTB33       0.051852   0.122986   0.422 0.673310
## PHF14        0.026382   0.120163   0.220 0.826221
## KRT6B       -0.072768   0.037259  -1.953 0.050819 .
## PIP          0.053532   0.031719   1.688 0.091471 .
## CXCL13       0.047843   0.040596   1.178 0.238599
## CPB1         0.041374   0.032819   1.261 0.207424
## CCNK         0.224679   0.130073   1.727 0.084110 .
## SNAI2       -0.128778   0.084941  -1.516 0.129497
## FBN1         0.285041   0.080127   3.557 0.000375 ***
## SCGB2A1     -0.037760   0.046640  -0.810 0.418167
## GZMB        -0.026978   0.067074  -0.402 0.687526
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1371.4  on 1302  degrees of freedom
## Residual deviance: 1149.9  on 1252  degrees of freedom
##   (6 observations deleted due to missingness)
## AIC: 1251.9
##
## Number of Fisher Scoring iterations: 5
```

```r
# Calculate predicted probabilities from the logistic regression model.
pred_probs <- predict(model_logistic_IQR, type = "response")

# Compute the AUC using the pROC package.
roc_obj <- roc(response = data_lr$pCR, predictor = pred_probs)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
auc_value <- auc(roc_obj)

# Print the AUC value
cat("The AUC for the logistic regression model using the 50 selected genes is:", round(auc_value, 3), "
```

```
## The AUC for the logistic regression model using the 50 selected genes is: 0.781
```

```r
library(pROC)

study_numbers <- unique(data_filtered$ZZ)
auc_values_glm <- numeric(length(study_numbers))
auc_values_glm[] <- NA  # preallocate as NA

for (i in seq_along(study_numbers)) {
  test_study <- study_numbers[i]

  # Partition data: training is all studies except the current one; testing is the left-out study.
  train_data <- subset(data_filtered, ZZ != test_study)
  test_data  <- subset(data_filtered, ZZ == test_study)

  # Fit logistic regression on the training set.
  model_glm <- glm(Y ~ ., data = train_data[, c(filtered_genes, "Y")],
                   family = binomial)

  # Predict on the test set.
  preds <- predict(model_glm, newdata = test_data, type = "response")

  # Check if test_data has both outcome classes.
  if(length(unique(test_data$Y)) < 2){
    cat("Left-out Study", test_study, "does not have two outcome classes. Skipping AUC calculation.\n")
    next  # Skip this iteration.
  }

  # Calculate the AUC using the pROC package.
  roc_obj <- roc(response = test_data$Y, predictor = preds)
  auc_values_glm[i] <- auc(roc_obj)

  cat("Left-out Study", test_study, "AUC:", round(auc_values_glm[i], 3), "\n")
}
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
## Left-out Study 1 AUC: 0.619
```

```
## Setting levels: control = 0, case = 1
```

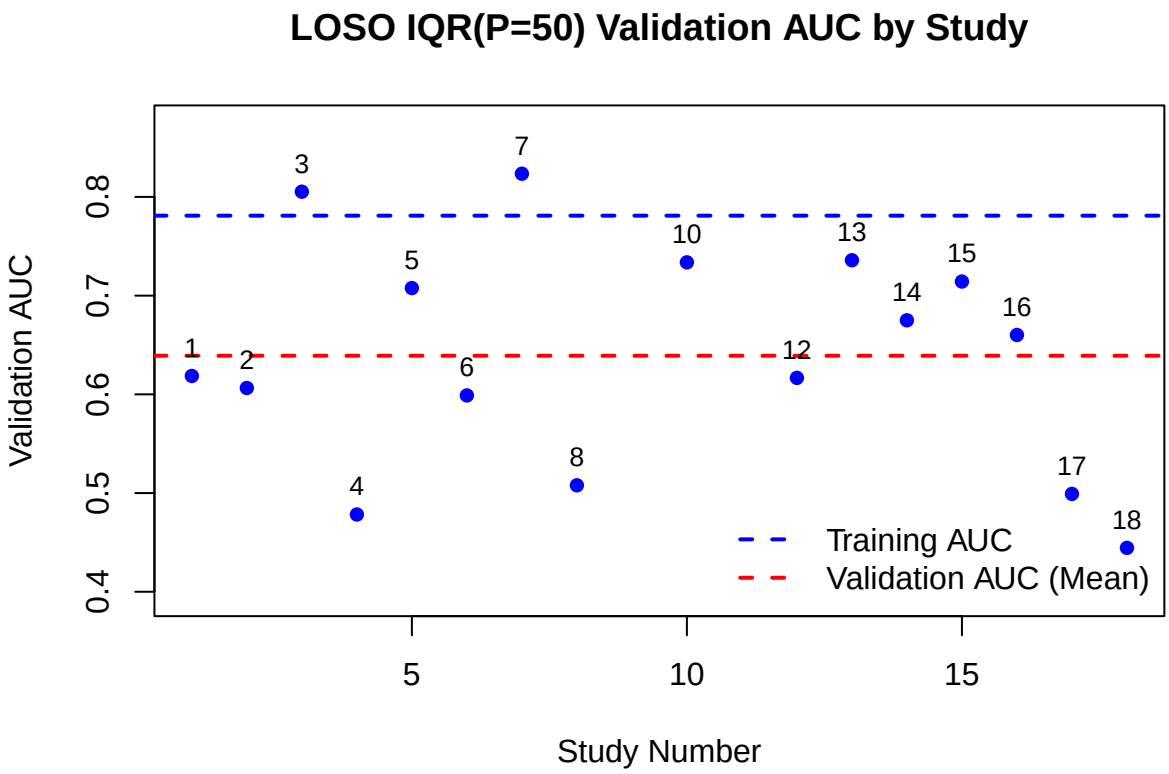
```
## Setting direction: controls > cases

## Left-out Study 2 AUC: 0.606

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Left-out Study 3 AUC: 0.805

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 4 AUC: 0.478

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 5 AUC: 0.708

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 6 AUC: 0.599

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 7 AUC: 0.824

## Setting levels: control = 0, case = 1

## Setting direction: controls > cases

## Left-out Study 8 AUC: 0.508
## Left-out Study 9 does not have two outcome classes. Skipping AUC calculation.

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Left-out Study 10 AUC: 0.734
## Left-out Study 11 does not have two outcome classes. Skipping AUC calculation.

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 12 AUC: 0.617

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 13 AUC: 0.736

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 14 AUC: 0.675

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 15 AUC: 0.714

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Left-out Study 16 AUC: 0.66
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls > cases

## Left-out Study 17 AUC: 0.499

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Left-out Study 18 AUC: 0.444
```

```
# Optionally, compute mean AUC across studies that produced a valid AUC
mean_auc_glm <- mean(auc_values_glm, na.rm = TRUE)
cat("Mean LOSO AUC (glm):", round(mean_auc_glm, 3), "\n")
```

```
## Mean LOSO AUC (glm): 0.639
```

```
valid_indices <- !is.na(auc_values_glm)
valid_studies <- study_numbers[valid_indices]
valid_auc <- auc_values_glm[valid_indices]

plot(valid_studies, valid_auc,
     pch = 16, col = "blue",
     xlab = "Study Number",
     ylab = "Validation AUC",
     main = "LOSO IQR(P=50) Validation AUC by Study",
     ylim = c(min(valid_auc) - 0.05, max(valid_auc) + 0.05))

# Add horizontal lines:
abline(h = mean_auc_glm, col = "red", lwd = 2, lty = 2)    # Mean Validation AUC
abline(h = 0.781, col = "blue", lwd = 2, lty = 2)           # Training AUC (0.684)

text(valid_studies, valid_auc, labels = valid_studies, pos = 3, cex = 0.8)

legend("bottomright",
       legend = c("Training AUC", "Validation AUC (Mean)"),
       col = c("blue", "red"),
       lwd = 2, lty = 2,
       bty = "n")
```



Pooled Logistic Regression Model:IQR subset, masomenos top 10

```
X_data <- t(XX_all_pCR[selected_gene_names_allstudies, ])

data_lr <- data.frame(X_data)
data_lr$pCR <- factor(YY_all_pCR)  # factors with levels "0" and "1"

# Fit a logistic regression model using all 10 gene expressions as predictors.
model_logistic_IQR_mom <- glm(pCR ~ ., data = data_lr, family = binomial, na.action = na.exclude)

summary(model_logistic_IQR_mom)
```

```
##
## Call:
## glm(formula = pCR ~ ., family = binomial, data = data_lr, na.action = na.exclude)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.28963    0.19963  -6.460 1.05e-10 ***
## AGR2        -0.06330    0.03302  -1.917  0.05524 .
## ESR1         0.02087    0.03546   0.589  0.55619
## STAT2       -0.20065    0.07385  -2.717  0.00659 **
## RARRES1      0.17765    0.02818   6.304 2.90e-10 ***
## H2AFJ       -0.10753    0.07139  -1.506  0.13201
## CEACAM6      0.01218    0.02766   0.440  0.65972
## GTF2A1       0.38858    0.08968   4.333 1.47e-05 ***
```

```
## TFF3         0.04080    0.03865   1.056  0.29112
## PQLC3       -0.25808    0.10537  -2.449  0.01432 *
## SCUBE2      -0.04367    0.03314  -1.318  0.18758
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1371.4  on 1302  degrees of freedom
## Residual deviance: 1282.1  on 1292  degrees of freedom
##   (6 observations deleted due to missingness)
## AIC: 1304.1
##
## Number of Fisher Scoring iterations: 4
```

```
pred_probs <- predict(model_logistic_IQR_mom, type = "response")

roc_obj <- roc(response = data_lr$pCR, predictor = pred_probs)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
auc_value <- auc(roc_obj)

cat("The AUC for the logistic regression model using the 10 selected genes is:", round(auc_value, 3), "
```

```
## The AUC for the logistic regression model using the 10 selected genes is: 0.684
```

Unpooled logistic model

$$
\begin{aligned}
i &= 1, \ldots, N \quad \text{(patients)} \\
j &= 1, \ldots, J \quad \text{(studies)} \\
k &= 1, \ldots, K \quad \text{(predictors)}
\end{aligned}
$$

$$
\mathbf{X}_i = \begin{pmatrix} X_{i,1} \\ X_{i,2} \\ \vdots \\ X_{i,K} \end{pmatrix}
$$

$$
Y_i \sim \text{Bernoulli}(p_i)
$$

$$
\text{logit}(p_i) = \alpha_{s_i} + \sum_{k=1}^{K} \beta_{s_i,k} X_{i,k}
$$

$$
\text{logit}(p_i) = \alpha_{s_i} + \mathbf{X}_i^\top \boldsymbol{\beta}_{s_i}
$$

$$
\alpha_j \sim \mathcal{N}\left(\mu_\alpha, \tau_\alpha^{-1}\right), \quad j = 1, \ldots, J
$$

$$
\beta_{j,k} \sim \mathcal{N}\left(\mu_{\beta,k}, \tau_{\beta,k}^{-1}\right), \quad j = 1, \ldots, J, \quad k = 1, \ldots, K
$$

$$
\mu_\alpha \sim \mathcal{N}(0,\, 0.01), \quad \tau_\alpha \sim \text{Gamma}(0.01,\, 0.01)
$$

$$\mu_{\beta,k} \sim \mathcal{N}(0,\, 0.01), \quad k = 1, \ldots, K$$

$$\tau_{\beta,k} = \left(\tau_{\text{int},k} \times 0.01\right)^2, \quad \tau_{\text{int},k} \sim \text{Discrete}(\mathbf{p})$$

```
N <- nrow(data_hier)                    # number of patients
K <- 10                                 # number of genes/predictors
N_studies <- length(unique(data_hier$ZZ))  # number of studies

model_string <- "
model {
  for (i in 1:N) {
    Y[i] ~ dbern(p[i])
    logit(p[i]) <- alpha[ZZ[i]] + inprod(beta[ZZ[i], 1:K], X[i,])
  }

  for (j in 1:N_studies) {
    alpha[j] ~ dnorm(mu_alpha, tau_alpha)
    for (k in 1:K) {
      beta[j,k] ~ dnorm(mu_beta[k], precision_tau[k])
    }
  }

  # Hyperpriors for the intercepts.
  mu_alpha ~ dnorm(0.0, 0.01)
  tau_alpha ~ dgamma(0.01, 0.01)

  # Hyperpriors for the slopes.
  for (k in 1:K) {
    mu_beta[k] ~ dnorm(0.0, 0.01)
    # Discrete prior for tau: tau_int_prior is provided as data; for example, a vector of length 25.
    tau_int[k] ~ dcat(tau_int_prior[])
    tau[k] <- tau_int[k] * 0.01
    precision_tau[k] <- 1 / (tau[k] * tau[k])
  }
}
"

data_jags <- list(
  N = N,
  K = K,
  N_studies = N_studies,
  Y = as.numeric(as.character(data_hier$Y)),   # ensure Y is numeric 0/1
  X = as.matrix(X_data),  # predictors matrix (N x K)
  ZZ = data_hier$ZZ,                             # study indicator for each patient
  tau_int_prior = rep(0.03, 25)                   # discrete prior vector for tau.int (length 25)
)

# Initialize 3 chains, run a burn-in, then sample posterior draws.
model <- jags.model(textConnection(model_string), data = data_jags, n.chains = 3, n.adapt = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
```

```
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1303
##    Unobserved stochastic nodes: 220
##    Total graph size: 21208
##
## Initializing model
```

```
update(model, 1000)  # burn-in period
samples <- coda.samples(model,
                        variable.names = c("alpha", "beta", "mu_beta", "tau", "precision_tau"),
                        n.iter = 5000)

# Print the summary of posterior samples.
#print(summary(samples))
```

Compute AUC

```
# take posterior mean for each param
post_summary <- summary(samples)$statistics
# alpha[1], …, alpha[J] are rows; beta[j,1], …, beta[j,K] are rows
alpha_hat <- post_summary[paste0("alpha[", 1:N_studies, "]"), "Mean"]
beta_hat  <- matrix(
  post_summary[grep("^beta\\[", rownames(post_summary)), "Mean"],
  nrow = N_studies, byrow = TRUE
)

linear_preds <- numeric(N)
for(i in 1:N){
  j <- data_hier$ZZ[i]
  x_i <- X_data[i, ]
  linear_preds[i] <- alpha_hat[j] + sum(beta_hat[j, ] * x_i)
}
p_hat <- plogis(linear_preds)

roc_train <- roc(response = data_hier$Y, predictor = p_hat)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
auc_train <- auc(roc_train)
cat("Train AUC =", round(auc_train,3), "\n")
```

```
## Train AUC = 0.553
```

Attempt 2: pooled intercept

```
model_string_pooled_int <- "
model {
  for (i in 1:N) {
    Y[i] ~ dbern(p[i])
    logit(p[i]) <- alpha + inprod(beta[ZZ[i],], X[i,])
  }
```

```
  # One global intercept
  alpha ~ dnorm(0.0, 0.01)

  # Study-specific slopes
  for (j in 1:N_studies) {
    for (k in 1:K) {
      beta[j,k] ~ dnorm(mu_beta[k], precision_tau[k])
    }
  }

  # Hyperpriors for slopes
  for (k in 1:K) {
    mu_beta[k] ~ dnorm(0.0, 0.01)
    tau_int[k] ~ dcat(tau_int_prior[])
    tau[k] <- tau_int[k] * 0.01
    precision_tau[k] <- 1 / (tau[k]^2)
  }
}
"

# 3. Prepare data for JAGS
data_jags <- list(
  N = N,
  K = K,
  N_studies = N_studies,
  Y = as.numeric(as.character(data_hier$Y)),   # ensure Y is numeric 0/1
  X = as.matrix(X_data),  # predictors matrix (N x K)
  ZZ = data_hier$ZZ,                            # study indicator for each patient
  tau_int_prior = rep(0.03, 25)                  # discrete prior vector for tau.int (length 25)
)

# 4. Initialize and run the JAGS model
# Initialize 3 chains, run a burn-in, then sample posterior draws.
model <- jags.model(textConnection(model_string_pooled_int), data = data_jags, n.chains = 3, n.adapt =
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1303
##    Unobserved stochastic nodes: 201
##    Total graph size: 21190
##
## Initializing model
```

```
update(model, 1000)  # burn-in period
samples <- coda.samples(model,
                        variable.names = c("alpha", "beta", "mu_beta", "tau", "precision_tau"),
                        n.iter = 5000)

# Print the summary of posterior samples.
#print(summary(samples))
```

Compute AUC

```
# take posterior mean for each param
post_summary <- summary(samples)$statistics
beta_hat  <- matrix(
  post_summary[grep("^beta\\[", rownames(post_summary)), "Mean"],
  nrow = N_studies, byrow = TRUE
)

alpha_hat <- post_summary["alpha", "Mean"]

linear_preds <- numeric(N)
for(i in 1:N){
  j <- data_hier$ZZ[i]
  x_i <- X_data[i, ]
  linear_preds[i] <- alpha_hat + sum(beta_hat[j, ] * x_i)
}
p_hat <- plogis(linear_preds)

roc_train <- roc(response = data_hier$Y, predictor = p_hat)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
auc_train <- auc(roc_train)
cat("Train AUC =", round(auc_train,3), "\n")
```

```
## Train AUC = 0.544
```

Leave-One-Study-Out Validation

```
coef_table <- summary(model_logistic_IQR)$coefficients
sig_genes <- rownames(coef_table)[
  coef_table[, "Pr(>|z|)"] < 0.05 &
  rownames(coef_table) != "(Intercept)"
]
cat("Significant genes (p < 0.05):\n")
```

```
## Significant genes (p < 0.05):
```

```
print(sig_genes)
```

```
## [1] "RPL23"   "AGR2"    "ASPN"    "EIF4A1"  "STAT2"   "RARRES1" "DPT"
## [8] "SCUBE2"  "FBN1"
```

```
data_loso <- data_filtered

study_numbers <- sort(unique(data_loso$ZZ))
auc_sig <- rep(NA_real_, length(study_numbers))

for (i in seq_along(study_numbers)) {
  s <- study_numbers[i]
  train <- subset(data_loso, ZZ != s)
  test  <- subset(data_loso, ZZ == s)

  # skip if the test set has only one class
```

```
  if (length(unique(test$Y)) < 2) {
    message("Skipping study ", s, ": only one outcome class")
    next
  }

  # build formula dynamically
  fml_sig <- as.formula(paste("Y ~", paste(sig_genes, collapse = " + ")))

  # fit on train
  mod_sig <- glm(fml_sig, data = train, family = binomial)

  # predict & AUC on test
  preds_sig <- predict(mod_sig, newdata = test, type = "response")
  roc_obj   <- roc(response = test$Y, predictor = preds_sig)
  auc_sig[i] <- auc(roc_obj)

  cat("Study", s, "LOSO AUC =", round(auc_sig[i], 3), "\n")
}
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 1 LOSO AUC = 0.635
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 2 LOSO AUC = 0.481
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 3 LOSO AUC = 0.797
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 4 LOSO AUC = 0.557
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 5 LOSO AUC = 0.592
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 6 LOSO AUC = 0.713
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 7 LOSO AUC = 1
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 8 LOSO AUC = 0.575
## Skipping study 9: only one outcome class
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
## Study 10 LOSO AUC = 0.769
## Skipping study 11: only one outcome class
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 12 LOSO AUC = 0.833
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 13 LOSO AUC = 0.736
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 14 LOSO AUC = 0.562
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 15 LOSO AUC = 0.821
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 16 LOSO AUC = 0.678
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 17 LOSO AUC = 0.578
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 18 LOSO AUC = 0.889
```

```r
#average over studies with valid AUC
mean_auc_sig <- mean(auc_sig, na.rm = TRUE)
cat("Mean LOSO AUC (significant genes):", round(mean_auc_sig, 3), "\n")
```

```
## Mean LOSO AUC (significant genes): 0.701
```

The validation AUC improved by removing insignificant genes. Could removing less generalizable studies from validation provide better generalizability?

```r
data_loso <- data_filtered

# run LOSO again on the *significant* genes , excluding studies with AUC  0.6
study_numbers <- sort(unique(data_loso$ZZ))
auc_sig <- rep(NA_real_, length(study_numbers))

for (i in seq_along(study_numbers)) {
  s <- study_numbers[i]
  train <- subset(data_loso, ZZ != s)
  test  <- subset(data_loso, ZZ == s)

  # skip if the test set has only one class
  if (length(unique(test$Y)) < 2) {
```

```
    message("Skipping study ", s, ": only one outcome class")
    next
  }

  # fit model on train
  fml_sig <- as.formula(paste("Y ~", paste(sig_genes, collapse = " + ")))
  mod_sig <- glm(fml_sig, data = train, family = binomial)

  # predict & compute AUC on test
  preds_sig <- predict(mod_sig, newdata = test, type = "response")
  roc_obj   <- roc(response = test$Y, predictor = preds_sig)
  auc_val   <- auc(roc_obj)

  # exclude if AUC  0.6
  if (auc_val <= 0.6) {
    message("Excluding study ", s, ": LOSO AUC = ", round(auc_val, 3), "  0.6")
    next
  }

  auc_sig[i] <- auc_val
  cat("Study", s, "LOSO AUC =", round(auc_val, 3), "\n")
}
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 1 LOSO AUC = 0.635
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Excluding study 2: LOSO AUC = 0.481  0.6
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 3 LOSO AUC = 0.797
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Excluding study 4: LOSO AUC = 0.557  0.6
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Excluding study 5: LOSO AUC = 0.592  0.6
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 6 LOSO AUC = 0.713
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Study 7 LOSO AUC = 1
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Excluding study 8: LOSO AUC = 0.575   0.6
```

```
## Skipping study 9: only one outcome class
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
## Study 10 LOSO AUC = 0.769
```

```
## Skipping study 11: only one outcome class
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
## Study 12 LOSO AUC = 0.833
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Study 13 LOSO AUC = 0.736
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Excluding study 14: LOSO AUC = 0.562   0.6
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
## Study 15 LOSO AUC = 0.821
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Study 16 LOSO AUC = 0.678
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Excluding study 17: LOSO AUC = 0.578   0.6
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
## Study 18 LOSO AUC = 0.889
```

```
# average over the retained studies
valid_auc <- na.omit(auc_sig)
mean_auc_sig <- mean(valid_auc)
cat("Mean LOSO AUC (significant genes, excluding IQR P=50 validation AUC   0.6):", round(mean_auc_sig, 3
```

```
## Mean LOSO AUC (significant genes, excluding IQR P=50 validation AUC   0.6): 0.787
```

Motivation Behind Classifier Choice

The original classifier choice was mas-o-menos due to the univariate gene selection method used in homework 4 that in general outperformed k-TSP. In addition, considering only pairwise comparisons of a highly dimensional dataset likely would provide far too many comparisons for k-TSP as an initial gene selection method. After observing other students' ideas in class, Alyssa pointed out that filtering genes by IQR could select

for genes with only variability in X. As follows, mas-o-menos could be used as an additional gene selection method focusing on the response variable and discrimination ability for given genes and the binary Y.

For the model, a linear decision boundary given logistic regression is an appropriate choice for this binary classification problem as shown in Project 2. In addition, a pooled logistic regression model for P=50 and P=10 predictors displayed near-threshold AUC without accounting for study specific effects. Given the EDA and the clear multimodal distribution of gene expression corresponsing to different studies and Michael's PCA analysis showing clear clustering of different studies in reduced space, an unpooled logistic regression with varying slopes and/or means could account for the study-wide heterogeneity.

Motivation behind choice of training and testing studies

For the training and testing studies, I chose to use Leave-One-Study-Out evaluation due to both the study batch effects and the variation in the plots of "LOSO IQR(P=50) Validation AUC by Study". However, LOSO validation led to problems in this dataset. Firstly, the response variable in the validation set had only one class, 1 or 0 in this binary classification problem, which led to studies 9 and 11 being omitted from the validation set list. Additionally, both the total number of samples and the number of samples in each class in the remaining studies in the validation set list is imbalanced, leading to drastic differences in testing AUC based on the validation study of that iteration. To account for this, the final validation did not include studies below a classifier imbalance and an AUC threshold similar to the threshold of 110 samples for Leave-One-In cross validation exhibited in Trippa et al. 2015.

Discussion

The first unpooled logistic regression model clearly overfit likely due to too many parameter estimations causing overfitting on batch-specific effects. As follows, I went against my original hypothesis and tested another jags model which only varied study-wise intercepts instead of intercepts and slopes. Nonetheless, both models struggled to generalize to validation set(s). I am curious whether other students had success with hierarchical models. Specifically, some students like Michael used clustering to reduce the number of dimensions and thus parameter estimates from 18 to 6 or fewer, preserving batch-specific effects by grouping studies with similar gene expression patterns.

Countering my original hypothesis, using mas-o-menos to select genes for subsequent logistic regression reduced AUC. This is likely due to the univariate nature of mas-o-menos. That is, mas-o-menos does not consider interactions between variables and may have selected genes with high collinearity that performed poorly when interacting in logistic regression. Noticing this pattern, I chose to remove genes based on performance within logistic regression instead of a separate mas-o-menos classifier as originally planned based on p-value to account for collinearity concerns.

Some cons of the final pooled modeling approach include dropping validation studies that provided poor discrimination. For instance, if an inference population resembles study 17, which was dropped for poor AUC validation on the IQR filtered dataset with P=50, the model would likely generalize poorly. In addition, the LOSO approach to this dataset had limitations due to studies 9 and 11 having small sample sizes and only 1 class, making it impossible to use studies 9 and 11 as viable validation sets without data imputation to address class imbalance. Additionally, according to Patil & Parmigiani 2018, ensemble methods across meta-analyses may improve results for metadata like these with study heterogeneity, which were not considered in this analysis

Some pros of the final pooled modeling approach include simplicity and interpretability from filtering only by IQR range on X and following significant predictor selection based on p-value with a logistic regression of P=50. The approach leverages both selection of high gene expression variability in X and the likelihood of response discrimination in Y to reduce the subset of predictors. As follows, the pooled model can generalize well to any inference population that was not dropped from the validation set. In addition, using different studies / hospitals as folds often penalizes model performance more and offers an effective approach to quantify model generalizability (Schmid et al., 2021). To conclude, based on the findings from Bernau et

al. 2014, the final pooled logistic regression generalizes well to most studies in the validation set, suggesting that dropped studies had a different and less common patient profile.

Bibliography

Bernau C. et al. (2014). "Cross-study validation for the assessment of prediction algorithms." Bioinformatics 30(12):i105-i112. DOI: 10.1093/bioinformatics/btu279

Lorenzo Trippa. Levi Waldron. Curtis Huttenhower. Giovanni Parmigiani. "Bayesian nonparametric cross-study validation of prediction methods." Ann. Appl. Stat. 9 (1) 402 - 428, March 2015. https://doi.org/10.1214/14-AOAS798

OpenAI. (2025). ChatGPT (model 04-mini-high) [Large language model]. Retrieved from https://chat.openai.com/ Note. Used for code writing, debugging, and rendering issues. LLM's did not assist in writing any text outside of code chunks.

Patil P. & Parmigiani G. (2018). "Training replicable predictors in multiple studies." PNAS 115(11):2578-2583. DOI: 10.1073/pnas.1708283115.

Schmid CH. et al. (2021). "Internal-external cross-validation helped to evaluate the generalizability of prediction models in large clustered datasets." J. Clin. Epidemiol. 137:83-91. DOI: 10.1016/j.jclinepi.2021.03.020.