# BIOST 544: Homework 4

Department of Biostatistics @ University of Washington

2024-11-22

## High Throughput Screens and Predictive Models

In the previous homework, we saw that predictive modeling with very large number of covariates and small sample size may not work very well, even with regularization. In this homework, we will revisit the NOAH data to explore a different strategy.

Recall the data can be found on the course website in the following files:

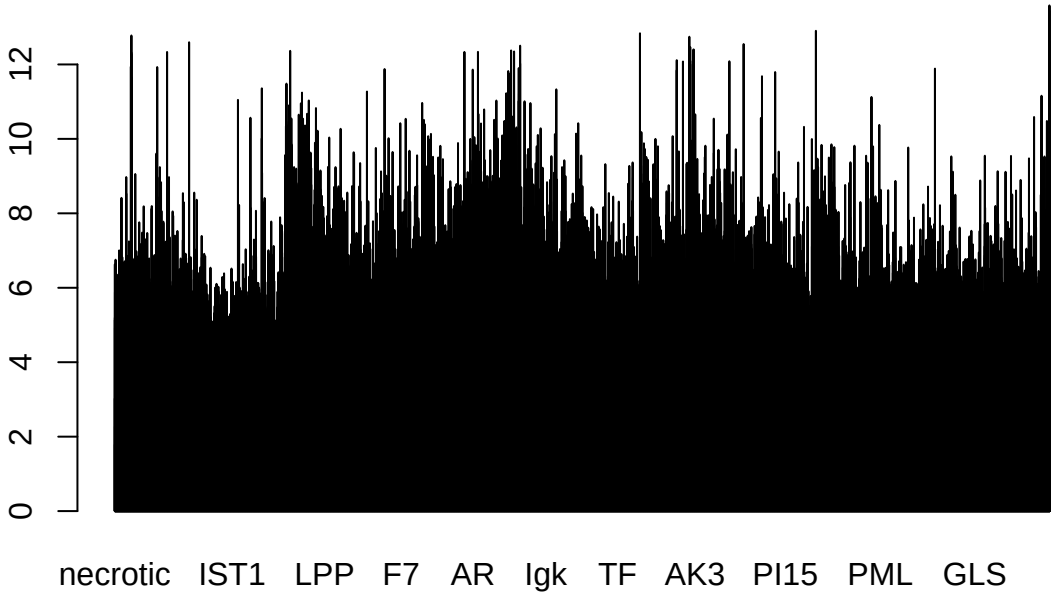
- **clinical_data.csv** contains the clinical/phenotypic information.
- **expression_data_probeID.csv** contains the expression information (by probe set).
- **annotation.csv** contains the gene name identifiers corresponding to each probe set.
- **noah_clean.RData** is a joined table of the others. It contains the information for 152 probe sets across 2 probe set identifiers, 1 outcome of interest, and 54675 gene expression predictors. Genes are labelled by their name.

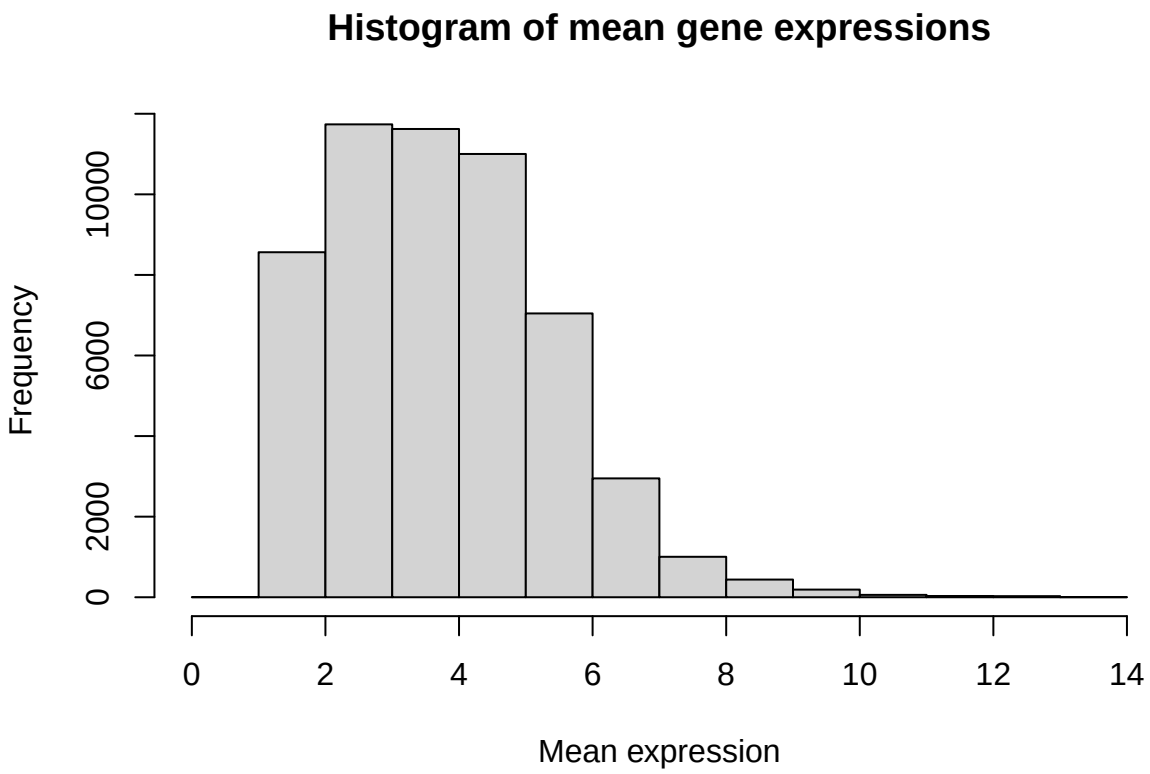
To get more information on the probesets, feel free to look into the affymetrix human 133 plus 2.0 array annotation (on the affymetrix site). In particular, the variable `necrotic_cells.pct` (the percentage of necrotic tissue in a tumor found by pathology) may be useful.

### Problem

In the previous homework, we used predictive models to identify the genes associated with quantity of necrotic tumor tissue. In this homework, we will explore two screening-based approaches:

i) In the first strategy, we will identify the genes whose expression levels are associated with `necrotic_cells.pct`. For this, you need to use a measure of association between continuous variables (as opposed to the difference of means that we used with a binary variable, in class). Please be sure to properly evaluate over-optimism and/our account for multiplicity in your analyses.


0
2
4
6
8
10
12
necrotic
IST1
LPP
F7
AR
Igk
TF
AK3
PI15
PML
GLS

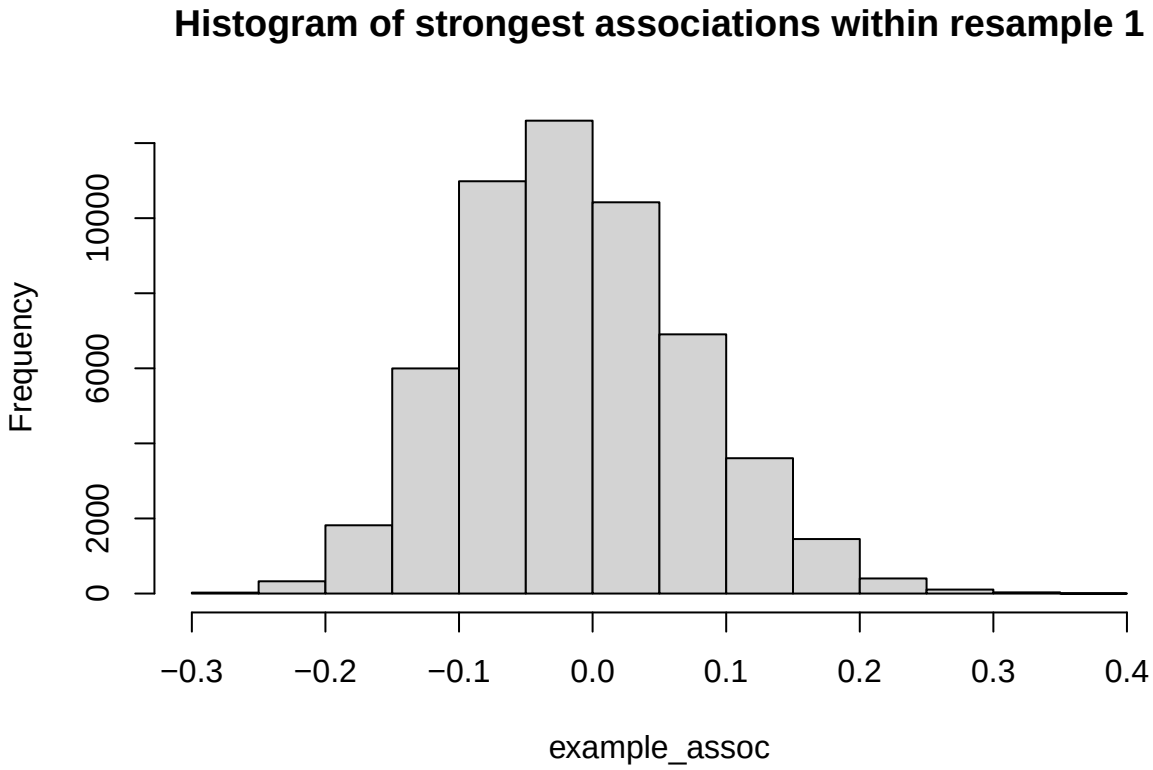



This suggest we may not have to worry about huge differences in the numeric expressions across genes, which helps keep things straightforward.

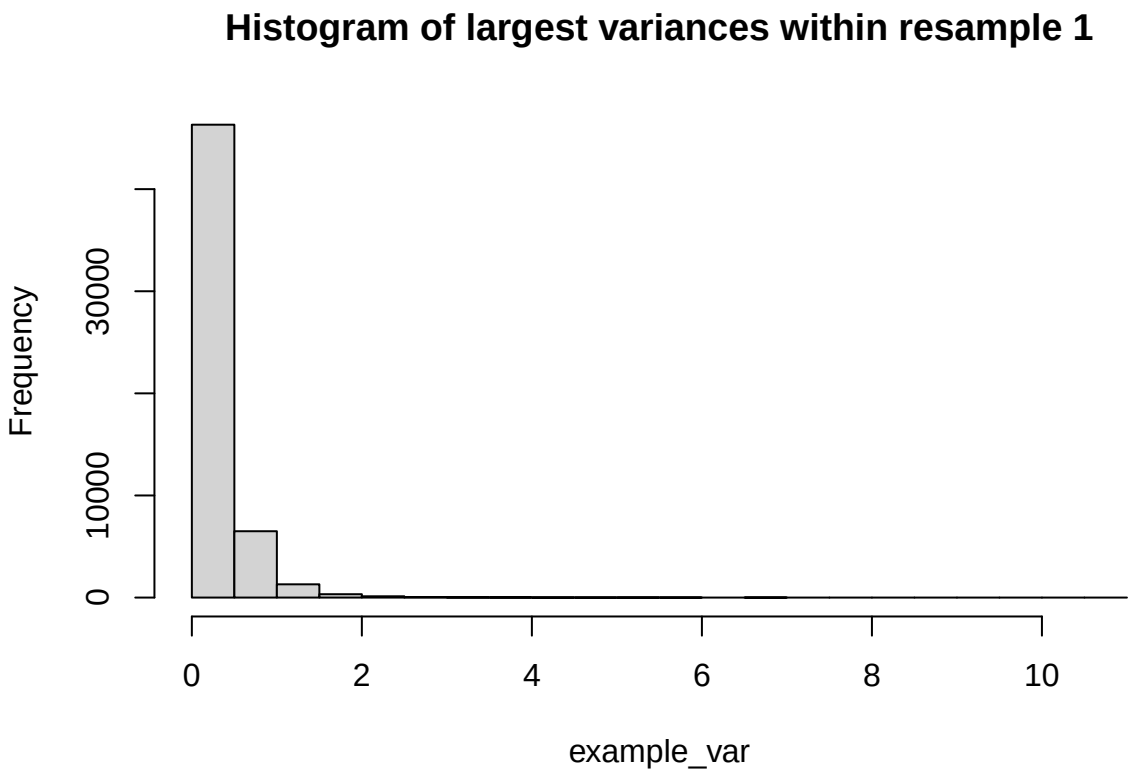
ii) In the second strategy, we will limit our analysis to genes whose expression varies across the samples. Specifically, find the 1000 genes largest variance of expression levels across all samples. Explain why these genes may be interesting and why this strategy can be helpful. As a second step in our analysis, we will build predictive models (similar to the previous homework) but using genes selected using either of the above screening approaches. To this end, use penalized regression to identify the genes that are predictive of `necrotic_cells.pct`, among the genes screened using either (i) or (ii) above. Explain how the interpretation of genes identified using the predictive models relates to the interpretation of genes identified using either screening strategy.

## Methods

### First Screener



**Second Screener**




## Results

**End of report. Code appendix begins on the next page.**

## Code Appendix

```
# clear environment
rm(list = ls())

# setup options
knitr::opts_chunk$set(echo = FALSE, results = 'hide',
                      warning = FALSE, message = FALSE)
options(knitr.kable.NA = '-', digits = 3)
labs <- setdiff(knitr::all_labels(), c("setup", "allcode"))
set.seed(1)
## Load relevant packages
library(dplyr)         # data manipulation
library(glmnet)        # lasso regularization
library(knitr)         # pretty tables

## Load clean NOAH data set
load("../data/noah_clean.RData")
dim(noah)  # 152 tumors; 2 identifiers, 1 outcome, 54675 predictors
head(noah[1:5])
# Bar plot of mean gene expressions
noah %>% colMeans %>% barplot
# Histogram of mean gene expressions
noah %>% colMeans %>%
  hist(xlab = "Mean expression", main = "Histogram of mean gene expressions")

#### --- FUNCTIONS --- ####

## Compute measure of associations between Y and X's
# data: data frame of Y and X's
# nmax: number of maximum associations to be returned
measure_association <- function (data, nmax = 1000) {
  # Measure a gene expression's association with percent necrosis
  assoc = cor(noah[,1], noah[,-1])
  # Sort the associations
  sorted_abs_assoc = abs(assoc) %>% sort(decreasing = TRUE, index.return = TRUE)

  # Return the genes with the largest variance
  index = sorted_abs_assoc$ix[1:nmax]
  max_assoc = assoc[index]
  name = names(data)[index]

  return(list(all_assoc=assoc, index=index, max_assoc=max_assoc, name=name))
  }


## Compute measure of spread across X's
# data: data frame of Y and X's
# nmax: number of maximum variances to be returned
measure_variance <- function (data, nmax = 1000) {
  # Measure a gene expression's variance
  vars = data[-1] %>%                          # across gene expressions
    lapply(var) %>%                            # calculate variance
```

```
    data.frame(row.names = "Variance") %>%    # name the single row
    t %>% data.frame                          # flip rows and columns
  # Sort and select the largest variances
  sorted_vars = vars$Variance %>% sort(decreasing = TRUE, index.return = TRUE)
  top_sorted_vars <- data.frame(sorted_vars) %>% head(nmax)

  # Return the genes with the largest variance
  all_var = sorted_vars$x
  index  = top_sorted_vars$ix
  max_var = top_sorted_vars$x
  name = names(data)[index]

  return(list(all_var=all_var, index=index, max_var=max_var, name=name))
  }


## Calculate summary statistics from a bootstrap resample
# data: data frame of Y and X's
# nmax: number of maximum summary statistics to be returned
do_one_boot <- function (data, nmax) {
  ix_boot = sample(nrow(data), replace = TRUE)
  resample = data[ix_boot,]
  # Return summary statistics from this resample
  return(list(
    associations = measure_association(resample, nmax),
    variances = measure_variance(resample, nmax)))
  }

#### --- BOOTSTRAPPING --- ####

# Hyper-parameters
nmax <- 100   # Number of maximum associations we will assess
nboot <- 2   # Number of resamples for bootstrapping

# Calculate summary statistics within many resamples of the data
set.seed(1)
boot_res <- replicate(nboot, do_one_boot(noah, nmax))
colnames(boot_res) <- paste("Boot", 1:nboot)


#### --- SCREENING BY ASSOCIATION --- ####

# Extract associations from bootstrap resamples
boot_assoc <- boot_res["associations",]

# View a distribution of strongest associations and largest variances
# from a random resample
randboot <- sample(nboot, 1)
example_assoc <- boot_assoc[[randboot]]$all_assoc
hist(example_assoc,
  main = paste("Histogram of strongest associations within resample", randboot))

## Order strongest associations by their frequency in our resamples
```

```r
assoc_combined <- do.call(c, boot_assoc)
# get the genes with the top measures of association in each resample
top_assoc_ix <- assoc_combined[grepl("index", names(assoc_combined))] %>%
  unlist %>%
  data.frame %>%
  table %>%           # count the times a gene shows up at the top of a resample
  data.frame %>%
  arrange(desc(Freq)) # arrange by frequency
names(top_assoc_ix) <- c("Index", "Freq")
top_assoc_ix

#### --- SCREENING BY VARIATION --- ####

# Extract variances from bootstrap resamples
boot_var <- boot_res["variances",]

# View a distribution of largest variances from a random resample
randboot <- sample(nboot, 1)
example_var <- boot_var[[randboot]]$all_var
hist(example_var,
  main = paste("Histogram of largest variances within resample", randboot))

## Order largest variances by their frequency in our resamples
var_combined <- do.call(c, boot_var)
# get the genes with the top variances in each resample
top_var_ix <- var_combined[grepl("index", names(var_combined))] %>%
  unlist %>%
  data.frame %>%
  table %>%           # count the times a gene shows up at the top of a resample
  data.frame %>%
  arrange(desc(Freq)) # arrange by frequency
names(top_var_ix) <- c("Index", "Freq")
top_var_ix

#### --- MODELING --- ####

# hyper-parameters
nlambda <- 100

## Model percentage of necrotic tissue in a tumor from gene expression values
# set.seed(1)
# lasso_cv_fit <- glmnet::cv.glmnet(y = noah[,1],
#                                   x = as.matrix(noah[,top_var_ix$Index]),
#                                   alpha = 1,  # lasso regularization
#                                   nfolds = nfold, nlambda = nlambda)
#
# ## evaluate model
# # plot MSE across candidate lasso values
# plot(lasso_cv_fit)
# # get nonzero coefficient estimate(s)
# coefs <- coef(lasso_cv_fit, s = lasso_cv_fit$lambda.min)
# nonzero_coefs <- data.frame(Estimate = coefs[as.list(coefs) != 0,])
# if(nrow(nonzero_coefs) == 1) { row.names(nonzero_coefs) <- "(Intercept)"}
```

```
# nonzero_coefs %>% knitr::kable(caption = "Non-zero regression coefficient(s)
#                                from cross-validated, regularized linear model.")
```

**End of document.**