Pertusis challenge 2.1

Slim Fourati

2024-01-12

```
suppressPackageStartupMessages(library(package = "knitr"))
suppressPackageStartupMessages(library(package = "glmnet"))
suppressPackageStartupMessages(library(package = "edgeR"))
suppressPackageStartupMessages(library(package = "biomaRt"))
suppressPackageStartupMessages(library(package = "GSVA"))
suppressPackageStartupMessages(library(package = "agua"))
suppressPackageStartupMessages(library(package = "tidyverse"))
```

```
workDir <- "/Users/iew5629/Desktop/Projects/20231013_Pertussis/Workspace"
options(readr.show_col_types = FALSE)
agua::h2o_start()
```

```
# 2020
pts2020DF <- read_tsv(file           = file.path(workDir, "documents/2020LD_subject.tsv"))
sample2020DF <- read_tsv(file           = file.path(workDir, "documents/2020LD_specimen.tsv"))
fcm2020DF <- read_tsv(file = file.path(workDir, "documents/2020LD_pbmc_cell_frequency.tsv"))
ab2020DF <- read_tsv(file = file.path(workDir, "documents/2020LD_plasma_ab_titer.tsv"))
olink2020DF <- read_tsv(file = file.path(workDir, "documents/2020LD_plasma_cytokine_concentration.tsv"))
ge2020DF <- read_tsv(file = file.path(workDir, "documents/2020LD_pbmc_gene_expression.tsv"))
```

```
# 2021
pts2021DF <- read_tsv(file = file.path(workDir, "documents/2021LD_subject.tsv"))
sample2021DF <- read_tsv(file = file.path(workDir, "documents/2021LD_specimen.tsv"))
fcm2021DF <- read_tsv(file = file.path(workDir, "documents/2021LD_pbmc_cell_frequency.tsv"))
ab2021DF <- read_tsv(file = file.path(workDir, "documents/2021LD_plasma_ab_titer.tsv"))
olink2021DF <- read_tsv(file = file.path(workDir, "documents/2021LD_plasma_cytokine_concentration.tsv"))
ge2021DF <- read_tsv(file = file.path(workDir, "documents/2021LD_pbmc_gene_expression.tsv"))
```

```
# 2022
pts2022DF <- read_tsv(file = file.path(workDir, "documents/2022BD_subject.tsv"))
sample2022DF <- read_tsv(file = file.path(workDir, "documents/2022BD_specimen.tsv"))
fcm2022DF <- read_tsv(file = file.path(workDir, "documents/2022BD_pbmc_cell_frequency.tsv"))
```

Challenge2.1

```
yDF <- fcm2020DF %>%
  filter(cell_type_name %in% "Monocytes") %>%
  merge(y = sample2020DF, by = "specimen_id", all.x = TRUE) %>%
  merge(y = pts2020DF, by = "subject_id", all.x = TRUE) %>%
  filter(planned_day_relative_to_boost %in% 1) %>%
  select(subject_id, percent_live_cell)
y2DF <- fcm2021DF %>%
```

```
  filter(cell_type_name %in% "Monocytes") %>%
  merge(y = sample2021DF, by = "specimen_id", all.x = TRUE) %>%
  merge(y = pts2021DF, by = "subject_id", all.x = TRUE) %>%
  filter(planned_day_relative_to_boost %in% 1) %>%
  select(subject_id, percent_live_cell)
```

pts info

```
xDF <- pts2020DF %>%
  dplyr::select(-dataset) %>%
  column_to_rownames(var = "subject_id") %>%
  mutate_if(is.character, as.factor) %>%
  mutate_if(is.Date, as.factor) %>%
  slice(match(yDF$subject_id, table = rownames(.)))

xDF <- sapply(xDF, as.numeric) %>%
  scale() %>%
  as.matrix()
fit <- cv.glmnet(x = xDF,
                 y = yDF$percent_live_cell,
                 nfolds = nrow(xDF),
                 grouped = FALSE)
plot(fit)
coef(fit, s = fit$lambda.min) %>%
  as.matrix() %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  filter(s1 != 0) %>%
  kable()

x2DF <- pts2021DF %>%
  dplyr::select(-dataset) %>%
  column_to_rownames(var = "subject_id") %>%
  mutate_if(is.character, as.factor) %>%
  mutate_if(is.Date, as.factor) %>%
  slice(match(y2DF$subject_id, table = rownames(.)))

x2DF <- sapply(x2DF, as.numeric) %>%
  scale() %>%
  as.matrix()

yhat <- predict(fit, newx = as.matrix(x2DF), s = fit$lambda.min, type = "response")

plot(rank(-1 * yhat), rank(-1 * y2DF$percent_live_cell))
cor.test(rank(-1 * yhat), rank(-1 * y2DF$percent_live_cell))
```

BL ab

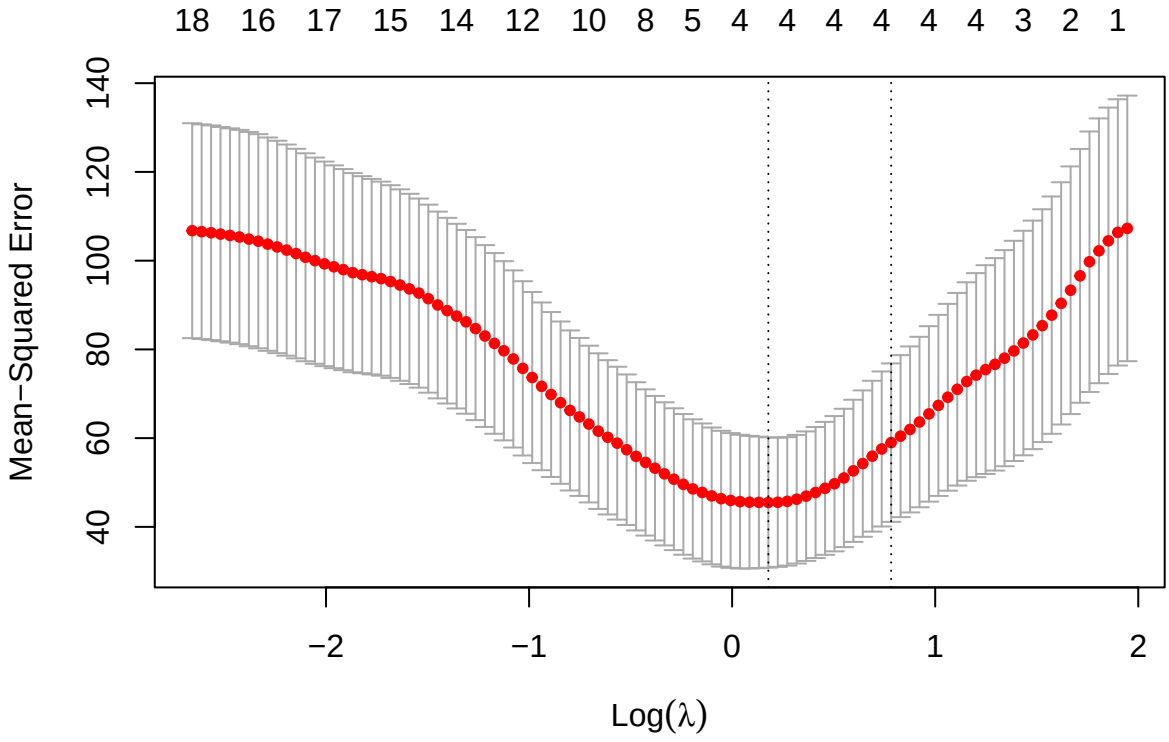
```
xDF <- ab2020DF %>%
  merge(y = sample2020DF, by = "specimen_id", all.x = TRUE) %>%
  merge(y = pts2020DF, by = "subject_id", all.x = TRUE) %>%
```

```
  filter(planned_day_relative_to_boost %in% 0 &
           grepl(pattern = "IgG", isotype)) %>%
  mutate(cname = paste0(isotype, "_", antigen),
         cname = make.names(cname)) %>%
  dplyr::select(subject_id, cname, MFI_normalised) %>%
  distinct() %>%
  pivot_wider(names_from = cname, values_from = MFI_normalised) %>%
  column_to_rownames(var = "subject_id") %>%
  slice(match(yDF$subject_id, table = rownames(.)))

x2DF <- ab2021DF %>%
  merge(y = sample2021DF, by = "specimen_id", all.x = TRUE) %>%
  merge(y = pts2021DF, by = "subject_id", all.x = TRUE) %>%
  filter(planned_day_relative_to_boost %in% 0) %>%
  mutate(cname = paste0(isotype, "_", antigen),
         cname = make.names(cname)) %>%
  dplyr::select(subject_id, cname, MFI_normalised) %>%
  distinct() %>%
  pivot_wider(names_from = cname, values_from = MFI_normalised) %>%
  column_to_rownames(var = "subject_id") %>%
  slice(match(y2DF$subject_id, table = rownames(.)))

xDF <- xDF[, intersect(colnames(xDF), colnames(x2DF))]
x2DF <- x2DF[, colnames(xDF)]

xDF <- scale(xDF)
x2DF <- scale(x2DF)
fit <- cv.glmnet(x = as.matrix(xDF),
                 y = yDF$percent_live_cell,
                 nfolds = nrow(xDF),
                 grouped = FALSE)
plot(fit)
coef(fit, s = fit$lambda.min) %>%
  as.matrix() %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  filter(s1 != 0) %>%
  kable()

yhat <- predict(fit, newx = as.matrix(x2DF), s = fit$lambda.min, type = "response")

plot(yhat[, "s1"], y2DF$percent_live_cell)
cor.test(yhat[, "s1"], y2DF$percent_live_cell)
```

FCM

```
xDF <- fcm2020DF %>%
  mutate(cell_type_name = make.names(cell_type_name)) %>%
  merge(y = sample2020DF, by = "specimen_id", all.x = TRUE) %>%
  merge(y = pts2020DF, by = "subject_id", all.x = TRUE) %>%
  filter(planned_day_relative_to_boost %in% 0) %>%
  dplyr::select(subject_id, cell_type_name, percent_live_cell) %>%
```

```
  pivot_wider(names_from = cell_type_name, values_from = percent_live_cell) %>%
  column_to_rownames(var = "subject_id") %>%
  slice(match(yDF$subject_id, table = rownames(.)))
xDF[is.na(xDF)] <- 0

x2DF <- fcm2021DF %>%
  merge(y = sample2021DF, by = "specimen_id", all.x = TRUE) %>%
  merge(y = pts2021DF, by = "subject_id", all.x = TRUE) %>%
  filter(planned_day_relative_to_boost %in% 0) %>%
  dplyr::select(subject_id, cell_type_name, percent_live_cell) %>%
  pivot_wider(names_from = cell_type_name, values_from = percent_live_cell) %>%
  column_to_rownames(var = "subject_id") %>%
  slice(match(y2DF$subject_id, table = rownames(.))) %>%
  setNames(nm = make.names(names(.))) %>%
  setNames(nm = make.unique(names(.)))
x2DF[is.na(x2DF)] <- 0

xDF <- xDF[, intersect(colnames(xDF), colnames(x2DF))]
x2DF <- x2DF[, colnames(xDF)]

xDF <- scale(xDF)
x2DF <- scale(x2DF)
fit <- cv.glmnet(x = as.matrix(xDF),
                 y = yDF$percent_live_cell,
                 nfolds = nrow(xDF),
                 grouped = FALSE)
plot(fit)
```

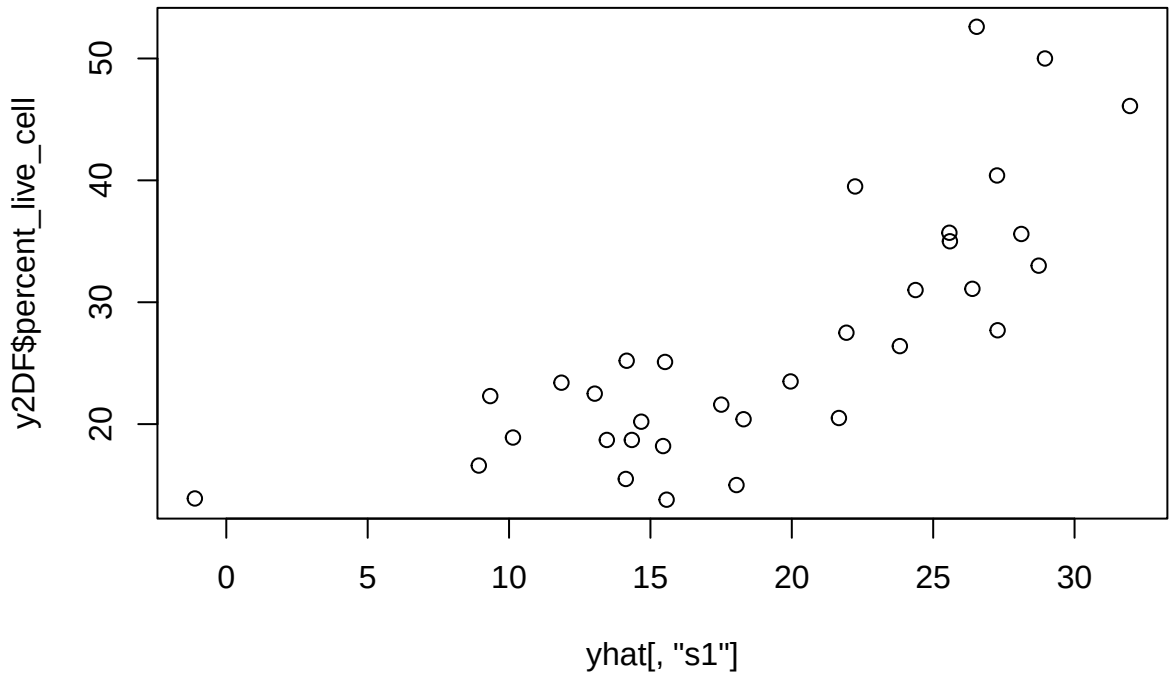



```
coef(fit, s = fit$lambda.min) %>%
  as.matrix() %>%
```

```
  as.data.frame() %>%
  rownames_to_column() %>%
  filter(s1 != 0) %>%
  kable()
```

rowname	s1
(Intercept)	19.202793
NaiveCD8	-3.722675
TcmCD8	-2.608348
pDC	1.218087
Classical_Monocytes	3.567372

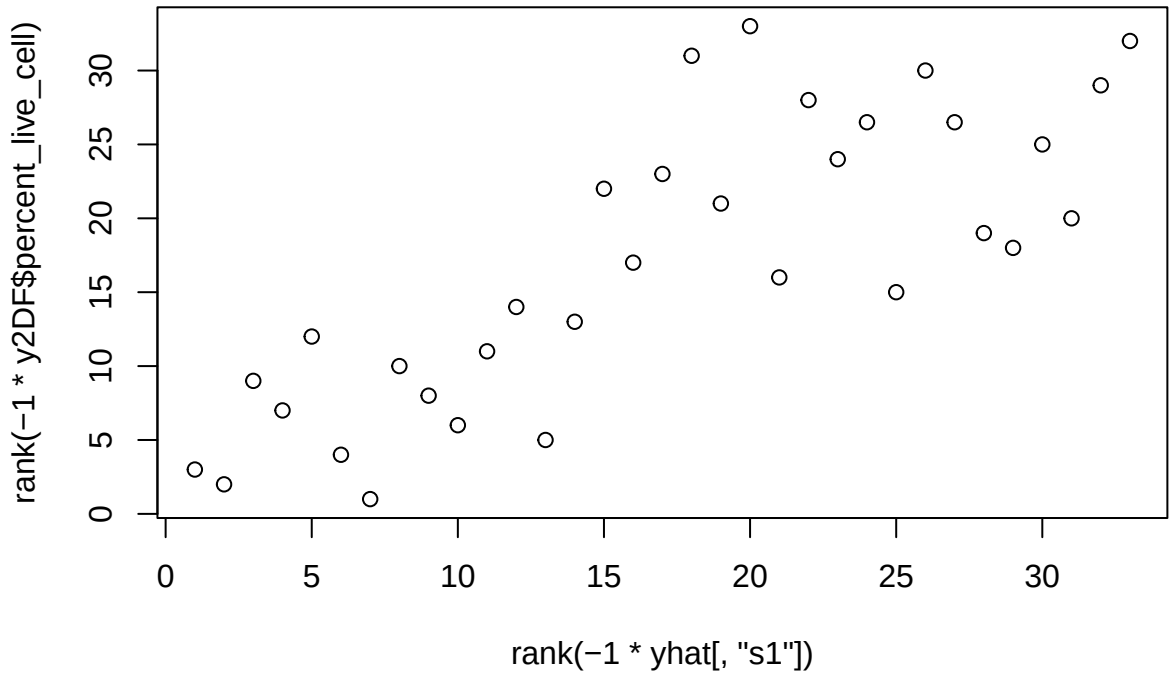
```
yhat <- predict(fit, newx = as.matrix(x2DF), s = fit$lambda.min, type = "response")

plot(yhat[, "s1"], y2DF$percent_live_cell)
```




```
cor.test(yhat[, "s1"], y2DF$percent_live_cell)
```

```
##
##  Pearson's product-moment correlation
##
## data:  yhat[, "s1"] and y2DF$percent_live_cell
## t = 6.9843, df = 31, p-value = 7.731e-08
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5995945 0.8871095
## sample estimates:
##       cor
## 0.7819434
```

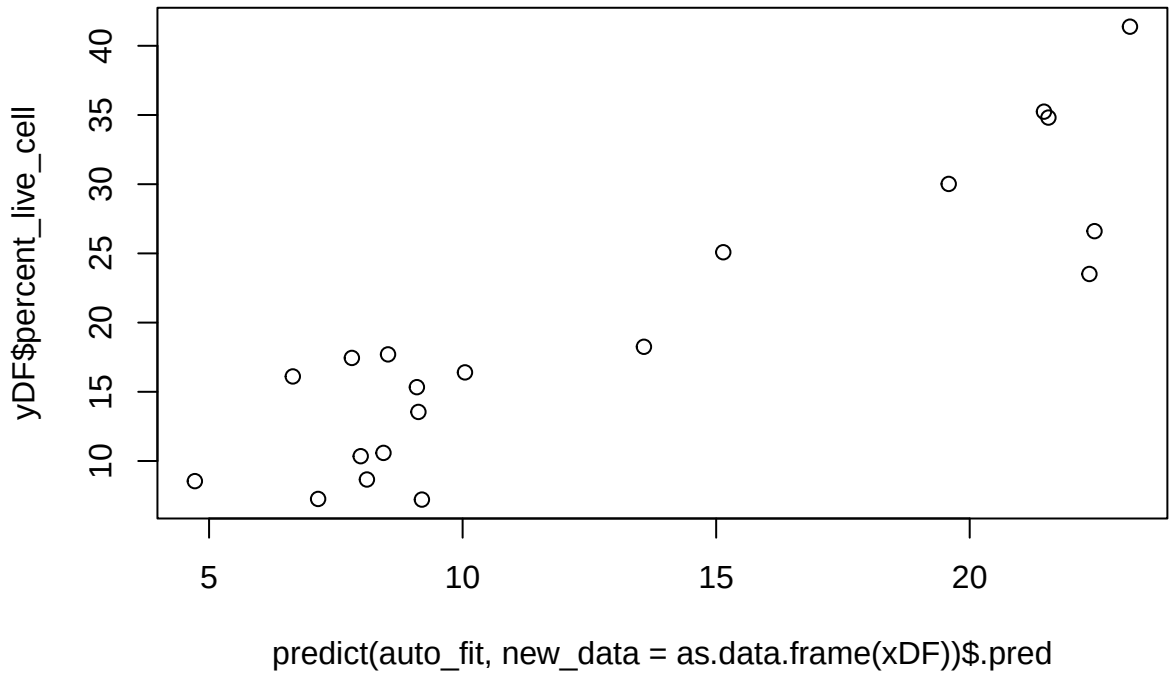
```r
plot(rank(-1 * yhat[, "s1"]), rank(-1 * y2DF$percent_live_cell))
```



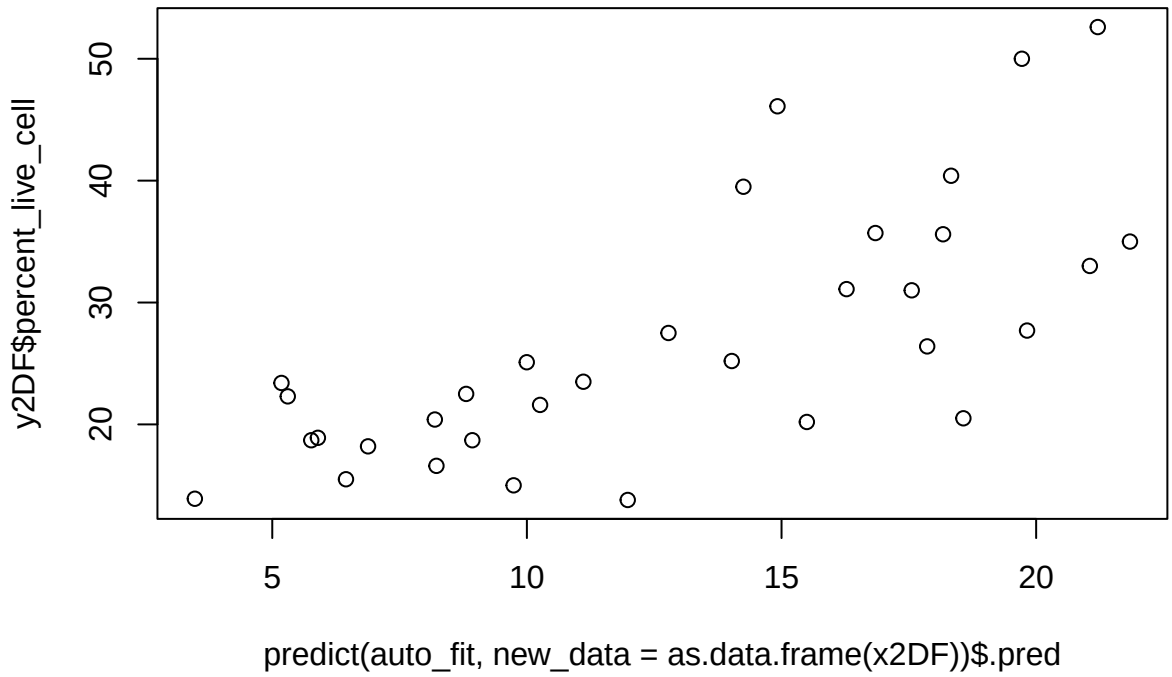

```r
cor.test(rank(-1 * yhat[, "s1"]), rank(-1 * y2DF$percent_live_cell))
```

```
##
##  Pearson's product-moment correlation
##
## data:  rank(-1 * yhat[, "s1"]) and rank(-1 * y2DF$percent_live_cell)
## t = 7.4159, df = 31, p-value = 2.368e-08
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6291079 0.8967978
## sample estimates:
##       cor
## 0.7996992
```

```r
set.seed(seed= 3)
trainDF <- xDF %>%
  as.data.frame() %>%
  mutate(percent_live_cell = yDF$percent_live_cell)

auto_fit <- auto_ml() %>%
     set_engine("h2o", max_runtime_secs = 5) %>%
     set_mode("regression") %>%
     fit(percent_live_cell ~ ., data = trainDF)

plot(predict(auto_fit, new_data = as.data.frame(xDF))$.pred,
     yDF$percent_live_cell)
```




```
plot(predict(auto_fit, new_data = as.data.frame(x2DF))$.pred,
     y2DF$percent_live_cell)
```




```
cor.test(predict(auto_fit, new_data = as.data.frame(x2DF))$.pred,
     y2DF$percent_live_cell)
```

```
##
##  Pearson's product-moment correlation
##
## data:  predict(auto_fit, new_data = as.data.frame(x2DF))$.pred and y2DF$percent_live_cell
## t = 5.76, df = 31, p-value = 2.433e-06
```

```
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4988209 0.8520023
## sample estimates:
##       cor
## 0.7190012
```

```
cor.test(rank(-1 * predict(auto_fit, new_data = as.data.frame(x2DF))$.pred),
     rank(-1 * y2DF$percent_live_cell))
```

```
##
##  Pearson's product-moment correlation
##
## data:  rank(-1 * predict(auto_fit, new_data = as.data.frame(x2DF))$.pred) and rank(-1 * y2DF$percent
## t = 6.0267, df = 31, p-value = 1.137e-06
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5231263 0.8607707
## sample estimates:
##       cor
## 0.7345199
```

Olink

```
xDF <- olink2020DF %>%
  merge(y = sample2020DF, by = "specimen_id", all.x = TRUE) %>%
  merge(y = pts2020DF, by = "subject_id", all.x = TRUE) %>%
  filter(planned_day_relative_to_boost %in% 0) %>%
  dplyr::select(subject_id, protein_id, protein_expression) %>%
  pivot_wider(names_from = protein_id, values_from = protein_expression) %>%
  column_to_rownames(var = "subject_id") %>%
  slice(match(yDF$subject_id, table = rownames(.)))

x2DF <- olink2021DF %>%
  merge(y = sample2021DF, by = "specimen_id", all.x = TRUE) %>%
  merge(y = pts2021DF, by = "subject_id", all.x = TRUE) %>%
  filter(planned_day_relative_to_boost %in% 0) %>%
  dplyr::select(subject_id, protein_id, protein_expression) %>%
  pivot_wider(names_from = protein_id, values_from = protein_expression) %>%
  column_to_rownames(var = "subject_id") %>%
  slice(match(y2DF$subject_id, table = rownames(.)))
x2DF[is.na(x2DF)] <- 0

xDF <- xDF[, intersect(colnames(xDF), colnames(x2DF))]
x2DF <- x2DF[, colnames(xDF)]
xDF <- scale(xDF)
x2DF <- scale(x2DF)
fit <- cv.glmnet(x = as.matrix(xDF),
                 y = yDF[match(rownames(xDF), table = yDF$subject_id), "percent_live_cell"],
                 nfolds = nrow(xDF),
                 grouped = FALSE)
plot(fit)
coef(fit, s = fit$lambda.min) %>%
  as.matrix() %>%
```

```
  as.data.frame() %>%
  rownames_to_column() %>%
  filter(s1 != 0) %>%
  kable()

yhat <- predict(fit, newx = as.matrix(x2DF), s = fit$lambda.min, type = "response")

plot(yhat[, "s1"], y2DF$percent_live_cell)
cor.test(yhat[, "s1"], y2DF$percent_live_cell)

plot(rank(-1 *yhat[, "s1"]), rank(-1 * y2DF$percent_live_cell))
cor.test(rank(-1* yhat[, "s1"]), rank(-1* y2DF$percent_live_cell))
```

```
set.seed(seed= 3)
trainDF <- xDF %>%
  as.data.frame() %>%
  mutate(percent_live_cell = yDF[match(rownames(xDF), table = yDF$subject_id), "percent_live_cell"])

auto_fit <- auto_ml() %>%
     set_engine("h2o", max_runtime_secs = 5) %>%
     set_mode("regression") %>%
     fit(percent_live_cell ~ ., data = trainDF)

plot(predict(auto_fit, new_data = as.data.frame(xDF))$.pred,
     yDF[match(rownames(xDF), table = yDF$subject_id), "percent_live_cell"])


plot(predict(auto_fit, new_data = as.data.frame(x2DF))$.pred,
     y2DF$percent_live_cell)

cor.test(predict(auto_fit, new_data = as.data.frame(x2DF))$.pred,
     y2DF$percent_live_cell)

cor.test(rank(-1 * predict(auto_fit, new_data = as.data.frame(x2DF))$.pred),
     rank(-1 * y2DF$percent_live_cell))
```

GE

```
# extract d0 GE
d0RawMat <- ge2020DF %>%
  merge(y = sample2020DF, by = "specimen_id", all.x = TRUE) %>%
  merge(y = pts2020DF, by = "subject_id", all.x = TRUE) %>%
  filter(planned_day_relative_to_boost %in% 0) %>%
  dplyr::select(subject_id, versioned_ensembl_gene_id, raw_count) %>%
  pivot_wider(names_from = subject_id, values_from = raw_count) %>%
  column_to_rownames(var = "versioned_ensembl_gene_id")

# TMM normalization
dge <- DGEList(counts       = d0RawMat,
               remove.zeros = TRUE)
dge <- calcNormFactors(object = dge, method = "TMM")
normalizedCounts <- cpm(dge, normalized.lib.sizes = TRUE, log = TRUE)
```

```r
# convert ensembl_gene_id to gene_symbol
human <- useMart("ensembl", dataset = "hsapiens_gene_ensembl")
id2symbol <- getBM(attributes  = c("ensembl_gene_id_version", "hgnc_symbol"),
                   filters     = "ensembl_gene_id_version",
                   values      = rownames(normalizedCounts),
                   mart        = human)
geneD0Mat <- normalizedCounts %>%
  as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id_version") %>%
  merge(y = id2symbol, by = "ensembl_gene_id_version") %>%
  select(-ensembl_gene_id_version) %>%
  slice(order(apply(select(., -hgnc_symbol), MARGIN = 1, FUN = var), decreasing = TRUE)) %>%
  filter(!duplicated(hgnc_symbol) & hgnc_symbol != "") %>%
  column_to_rownames(var = "hgnc_symbol") %>%
  as.matrix()

# GSVA with BTM
load(file = "/Users/iew5629/Desktop/Projects/20231013_Pertussis/Workspace/documents/GeneSets.rda")
BTM.geneSets <- BTM.geneSets[!grepl(pattern = "TBA", names(BTM.geneSets))]
names(BTM.geneSets) <- make.names(names(BTM.geneSets))

flag <- BTM.geneSets %>%
  stack() %>%
  filter(values %in% rownames(geneD0Mat)) %>%
  group_by(ind) %>%
  summarize(n = n()) %>%
  arrange(desc(n)) %>%
  filter(n >= 3)

gsD0Mat <- gsva(expr = geneD0Mat, gset.idx.list = BTM.geneSets[flag$ind], verbose = FALSE)

# prediction
xDF <- t(gsD0Mat)
xDF <- xDF[intersect(yDF$subject_id, rownames(xDF)), ]
fit <- cv.glmnet(x = as.matrix(xDF),
                 y = yDF[match(rownames(xDF), table = yDF$subject_id), "percent_live_cell"],
                 nfolds = nrow(xDF),
                 grouped = FALSE)
plot(fit)
coef(fit, s = fit$lambda.min) %>%
  as.matrix() %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  filter(s1 != 0) %>%
  kable()

# extract d0 GE of 2021
d0RawMat <- ge2021DF %>%
  merge(y = sample2021DF, by = "specimen_id", all.x = TRUE) %>%
  merge(y = pts2021DF, by = "subject_id", all.x = TRUE) %>%
  filter(planned_day_relative_to_boost %in% 0) %>%
  dplyr::select(subject_id, versioned_ensembl_gene_id, raw_count) %>%
  pivot_wider(names_from = subject_id, values_from = raw_count) %>%
```

```
  column_to_rownames(var = "versioned_ensembl_gene_id")

# TMM normalization
dge <- DGEList(counts       = d0RawMat,
               remove.zeros = TRUE)
dge <- calcNormFactors(object = dge, method = "TMM")
normalizedCounts <- cpm(dge, normalized.lib.sizes = TRUE, log = TRUE)

# convert ensembl_gene_id to gene_symbol
id2symbol <- getBM(attributes  = c("ensembl_gene_id_version", "hgnc_symbol"),
                   filters     = "ensembl_gene_id_version",
                   values      = rownames(normalizedCounts),
                   mart        = human)
geneD0Mat <- normalizedCounts %>%
  as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id_version") %>%
  merge(y = id2symbol, by = "ensembl_gene_id_version") %>%
  select(-ensembl_gene_id_version) %>%
  slice(order(apply(select(., -hgnc_symbol), MARGIN = 1, FUN = var), decreasing = TRUE)) %>%
  filter(!duplicated(hgnc_symbol) & hgnc_symbol != "") %>%
  column_to_rownames(var = "hgnc_symbol") %>%
  as.matrix()

# GSVA with BTM
gsD0Mat <- gsva(expr = geneD0Mat, gset.idx.list = BTM.geneSets[flag$ind], verbose = FALSE)

# prediction
x2DF <- t(gsD0Mat)
yhat <- predict(fit, newx = as.matrix(x2DF), s = fit$lambda.min, type = "response")

plot(yhat[as.character(y2DF$subject_id), "s1"], y2DF$percent_live_cell)
cor.test(yhat[as.character(y2DF$subject_id), "s1"], y2DF$percent_live_cel)

plot(rank(-1 * yhat[as.character(y2DF$subject_id), "s1"]), rank(-1 * y2DF$percent_live_cell))
cor.test(rank(-1 * yhat[as.character(y2DF$subject_id), "s1"]), rank(-1 * y2DF$percent_live_cell))
```

```
set.seed(seed= 3)
trainDF <- xDF %>%
  as.data.frame() %>%
  mutate(percent_live_cell = yDF[match(rownames(xDF), table = yDF$subject_id), "percent_live_cell"])

auto_fit <- auto_ml() %>%
     set_engine("h2o", max_runtime_secs = 5) %>%
     set_mode("regression") %>%
     fit(percent_live_cell ~ ., data = trainDF)

plot(predict(auto_fit, new_data = as.data.frame(xDF))$.pred,
     yDF[match(rownames(xDF), table = yDF$subject_id), "percent_live_cell"])


plot(predict(auto_fit, new_data = as.data.frame(x2DF))$.pred,
     y2DF[match(rownames(x2DF), table = y2DF$subject_id), "percent_live_cell"])

cor.test(predict(auto_fit, new_data = as.data.frame(x2DF))$.pred,
```

```
     y2DF[match(rownames(x2DF), table = y2DF$subject_id), "percent_live_cell"])

cor.test(rank(-1 * predict(auto_fit, new_data = as.data.frame(x2DF))$.pred),
     rank(-1 * y2DF[match(rownames(x2DF), table = y2DF$subject_id), "percent_live_cell"]))
```

2022 test set

```
set.seed(seed= 3)
# train on 2020 and 2021 baseline ab only
xDF <- fcm2020DF %>%
  mutate(cell_type_name = make.names(cell_type_name)) %>%
  merge(y = sample2020DF, by = "specimen_id", all.x = TRUE) %>%
  merge(y = pts2020DF, by = "subject_id", all.x = TRUE) %>%
  filter(planned_day_relative_to_boost %in% 0) %>%
  dplyr::select(subject_id, cell_type_name, percent_live_cell) %>%
  pivot_wider(names_from = cell_type_name, values_from = percent_live_cell) %>%
  column_to_rownames(var = "subject_id") %>%
  slice(match(yDF$subject_id, table = rownames(.)))
xDF[is.na(xDF)] <- 0

x2DF <- fcm2021DF %>%
  merge(y = sample2021DF, by = "specimen_id", all.x = TRUE) %>%
  merge(y = pts2021DF, by = "subject_id", all.x = TRUE) %>%
  filter(planned_day_relative_to_boost %in% 0) %>%
  dplyr::select(subject_id, cell_type_name, percent_live_cell) %>%
  pivot_wider(names_from = cell_type_name, values_from = percent_live_cell) %>%
  column_to_rownames(var = "subject_id") %>%
  slice(match(y2DF$subject_id, table = rownames(.))) %>%
  setNames(nm = make.names(names(.))) %>%
  setNames(nm = make.unique(names(.)))
x2DF[is.na(x2DF)] <- 0

x3DF <- fcm2022DF %>%
  merge(y = sample2022DF, by = "specimen_id", all.x = TRUE) %>%
  merge(y = pts2022DF, by = "subject_id", all.x = TRUE) %>%
  filter(planned_day_relative_to_boost %in% 0) %>%
  dplyr::select(subject_id, cell_type_name, percent_live_cell) %>%
  pivot_wider(names_from = cell_type_name, values_from = percent_live_cell) %>%
  column_to_rownames(var = "subject_id") %>%
  setNames(nm = make.names(names(.))) %>%
  setNames(nm = make.unique(names(.)))

xDF <- xDF[, intersect(colnames(xDF), colnames(x2DF))]
x2DF <- x2DF[, colnames(xDF)]
x3DF <- x3DF[, colnames(xDF)]

xDF <- scale(xDF)
x2DF <- scale(x2DF)
x3DF <- scale(x3DF)

trainDF <- rbind(xDF, x2DF) %>%
  as.data.frame() %>%
```

```
  mutate(percent_live_cell = c(yDF$percent_live_cell, y2DF$percent_live_cell))

auto_fit <- auto_ml() %>%
     set_engine("h2o", max_runtime_secs = 5) %>%
     set_mode("regression") %>%
     fit(percent_live_cell ~ ., data = trainDF)

yhat <- predict(auto_fit, new_data = x3DF)$.pred
rhat <- rank(-1 * yhat)
print(cbind(rownames(x3DF), rhat))
```

```
##              rhat
##  [1,] "97"  "8"
##  [2,] "98"  "21"
##  [3,] "99"  "4"
##  [4,] "100" "7"
##  [5,] "101" "18"
##  [6,] "102" "5"
##  [7,] "103" "2"
##  [8,] "104" "9"
##  [9,] "105" "6"
## [10,] "106" "16"
## [11,] "107" "13"
## [12,] "108" "14"
## [13,] "109" "10"
## [14,] "110" "3"
## [15,] "111" "1"
## [16,] "112" "15"
## [17,] "114" "11"
## [18,] "115" "20"
## [19,] "116" "12"
## [20,] "117" "17"
## [21,] "118" "19"
```

```
agua::h2o_end()
sessionInfo()
```

```
## R version 4.3.2 (2023-10-31)
## Platform: aarch64-apple-darwin23.0.0 (64-bit)
## Running under: macOS Sonoma 14.2.1
##
## Matrix products: default
## BLAS:   /opt/homebrew/Cellar/openblas/0.3.25/lib/libopenblasp-r0.3.25.dylib
## LAPACK: /opt/homebrew/Cellar/r/4.3.2/lib/R/lib/libRlapack.dylib;  LAPACK version 3.11.0
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Chicago
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
```

```
## other attached packages:
##  [1] lubridate_1.9.3 forcats_1.0.0   stringr_1.5.1   dplyr_1.1.4
##  [5] purrr_1.0.2     readr_2.1.5     tidyr_1.3.0     tibble_3.2.1
##  [9] ggplot2_3.4.4   tidyverse_2.0.0 agua_0.1.3      parsnip_1.1.1
## [13] GSVA_1.50.0     biomaRt_2.58.0  edgeR_4.0.6     limma_3.58.1
## [17] glmnet_4.1-8    Matrix_1.6-5    knitr_1.45
##
## loaded via a namespace (and not attached):
##   [1] jsonlite_1.8.8              rstudioapi_0.15.0
##   [3] shape_1.4.6                 magrittr_2.0.3
##   [5] rmarkdown_2.25              zlibbioc_1.48.0
##   [7] vctrs_0.6.5                 memoise_2.0.1
##   [9] DelayedMatrixStats_1.24.0   RCurl_1.98-1.14
##  [11] htmltools_0.5.7             S4Arrays_1.2.0
##  [13] dials_1.2.0                 progress_1.2.3
##  [15] curl_5.2.0                  Rhdf5lib_1.24.1
##  [17] SparseArray_1.2.3           rhdf5_2.46.1
##  [19] parallelly_1.36.0           cachem_1.0.8
##  [21] lifecycle_1.0.4             iterators_1.0.14
##  [23] pkgconfig_2.0.3             rsvd_1.0.5
##  [25] R6_2.5.1                    fastmap_1.1.1
##  [27] future_1.33.1               GenomeInfoDbData_1.2.11
##  [29] MatrixGenerics_1.14.0       tune_1.1.2
##  [31] digest_0.6.34               colorspace_2.1-0
##  [33] furrr_0.3.1                 AnnotationDbi_1.64.1
##  [35] S4Vectors_0.40.2            irlba_2.3.5.1
##  [37] GenomicRanges_1.54.1        RSQLite_2.3.4
##  [39] beachmat_2.18.0             filelock_1.0.3
##  [41] yardstick_1.2.0             timechange_0.2.0
##  [43] fansi_1.0.6                 httr_1.4.7
##  [45] abind_1.4-5                 compiler_4.3.2
##  [47] bit64_4.0.5                 withr_2.5.2
##  [49] BiocParallel_1.36.0         DBI_1.2.1
##  [51] highr_0.10                  HDF5Array_1.30.0
##  [53] lava_1.7.3                  MASS_7.3-60
##  [55] rappdirs_0.3.3              DelayedArray_0.28.0
##  [57] tools_4.3.2                 future.apply_1.11.1
##  [59] nnet_7.3-19                 glue_1.7.0
##  [61] h2o_3.44.0.3                rhdf5filters_1.14.1
##  [63] grid_4.3.2                  generics_0.1.3
##  [65] recipes_1.0.9               gtable_0.3.4
##  [67] tzdb_0.4.0                  class_7.3-22
##  [69] rsample_1.2.0               data.table_1.14.10
##  [71] hms_1.1.3                   BiocSingular_1.18.0
##  [73] ScaledMatrix_1.10.0         xml2_1.3.6
##  [75] utf8_1.2.4                  XVector_0.42.0
##  [77] BiocGenerics_0.48.1         foreach_1.5.2
##  [79] pillar_1.9.0                vroom_1.6.5
##  [81] splines_4.3.2               lhs_1.1.6
##  [83] BiocFileCache_2.10.1        lattice_0.22-5
##  [85] survival_3.5-7              bit_4.0.5
##  [87] annotate_1.80.0             tidyselect_1.2.0
##  [89] SingleCellExperiment_1.24.0 locfit_1.5-9.8
##  [91] Biostrings_2.70.1           IRanges_2.36.0
```

```
##  [93] SummarizedExperiment_1.32.0 stats4_4.3.2
##  [95] xfun_0.41                   Biobase_2.62.0
##  [97] statmod_1.5.0               hardhat_1.3.0
##  [99] timeDate_4032.109           matrixStats_1.2.0
## [101] stringi_1.8.3               DiceDesign_1.10
## [103] yaml_2.3.8                  workflows_1.1.3
## [105] evaluate_0.23               codetools_0.2-19
## [107] graph_1.80.0                cli_3.6.2
## [109] rpart_4.1.23                xtable_1.8-4
## [111] munsell_0.5.0               Rcpp_1.0.12
## [113] GenomeInfoDb_1.38.5         globals_0.16.2
## [115] dbplyr_2.4.0                png_0.1-8
## [117] XML_3.99-0.16               parallel_4.3.2
## [119] gower_1.0.1                 blob_1.2.4
## [121] prettyunits_1.2.0           sparseMatrixStats_1.14.0
## [123] bitops_1.0-7                listenv_0.9.0
## [125] GPfit_1.0-8                 GSEABase_1.64.0
## [127] ipred_0.9-14                prodlim_2023.08.28
## [129] scales_1.3.0                crayon_1.5.2
## [131] rlang_1.1.3                 KEGGREST_1.42.0
```