Aidan-Script

Aidan Frazier

2025-11-05

```
library(tidyverse)
library(infer)
library(randomForest)
library(tidymodels)
library(modelr)
library(yardstick)
library(pROC)

set.seed(133233)

clean_rdata <- "data/biomarker-clean.RData"
results_dir <- "results"
dir.create(results_dir, showWarnings = FALSE, recursive = TRUE)

top_k <- 10
learn_threshold <- TRUE

# Loading our data
load(clean_rdata)
biomarker_clean <- biomarker_clean %>%
  mutate(group = factor(group, levels = c("ASD","TD")))

# Stratified train/test split ----
split_obj <- initial_split(biomarker_clean, prop = 0.8, strata = group)
train_df <- training(split_obj)
test_df <- testing(split_obj)

# Training

# t-tests
test_fn <- function(.df){
  t_test(.df,
         formula = level ~ group,
         order = c("ASD","TD"),
         alternative = "two-sided",
         var.equal = FALSE)
}

ttests_out_train <- train_df %>%
  select(-ados) %>%
  pivot_longer(-group, names_to = "protein", values_to = "level") %>%
  nest(data = c(level, group)) %>%
```

```
  mutate(ttest = purrr::map(data, test_fn)) %>%
  unnest(ttest) %>%
  arrange(p_value) %>%
  mutate(
    m = n(),
    hm = log(m) + 1/(2*m) - digamma(1),
    rank = row_number(),
    p.adj = m*hm*p_value/rank
  )

proteins_s1_train <- ttests_out_train %>%
  slice_min(p.adj, n = top_k) %>%
  pull(protein)

write_csv(ttests_out_train %>% select(protein, statistic, p_value, p.adj),
          file.path(results_dir, "train_ttests_all.csv"))

# Random Forest
predictors_train <- train_df %>%
  select(-c(group, ados))
response_train   <- train_df %>%
  pull(group) %>% factor()

rf_out_train <- randomForest(
  x = predictors_train,
  y = response_train,
  ntree = 1000,
  importance = TRUE
)

rf_imp_train <- rf_out_train$importance %>%
  as_tibble() %>%
  mutate(protein = rownames(rf_out_train$importance)) %>%
  arrange(desc(MeanDecreaseGini))

proteins_s2_train <- rf_imp_train %>%
  slice_head(n = top_k) %>%
  pull(protein)

write_csv(rf_imp_train, file.path(results_dir, "train_rf_importance_all.csv"))

# Intersection information setup
proteins_panel <- intersect(proteins_s1_train, proteins_s2_train)
if (length(proteins_panel) < 2) {
  message("Intersection < 2; using union as a fallback.")
  proteins_panel <- union(proteins_s1_train, proteins_s2_train) %>%
    unique()
}

tibble(source = c(rep("t_test_topk", length(proteins_s1_train)),
                  rep("rf_topk", length(proteins_s2_train))),
       protein = c(proteins_s1_train, proteins_s2_train)) %>%
  mutate(in_intersection = protein %in% proteins_panel) %>%
```

```r
  write_csv(file.path(results_dir, "train_selected_proteins.csv"))

# Fitting our models with training data

train_panel <- train_df %>%
  select(group, any_of(proteins_panel)) %>%
  mutate(class = factor(group == "ASD", levels = c(FALSE, TRUE))) %>%
  select(-group)

logit_fit <- glm(class ~ ., data = train_panel, family = binomial())

# Learn threshold for our training data
if (learn_threshold) {
  train_probs <- predict(logit_fit, type = "response")
  roc_obj <- pROC::roc(response = train_panel$class, predictor = as.numeric(train_probs))
  best_coords <- pROC::coords(roc_obj, x = "best", best.method = "youden",
                              ret = c("threshold","sensitivity","specificity","auc"))
  thr <- as.numeric(best_coords["threshold"])
} else {
  thr <- 0.5
}

# Results for testing splits

test_panel <- test_df %>%
  select(group, any_of(proteins_panel)) %>%
  mutate(class = factor(group == "ASD", levels = c(FALSE, TRUE))) %>%
  select(-group)

test_probs <- predict(logit_fit, newdata = test_panel, type = "response")
test_pred <- factor(ifelse(test_probs > thr, TRUE, FALSE), levels = c(FALSE, TRUE))

# Metrics
test_metrics <- bind_rows(
  tibble(.metric = "accuracy", .est = accuracy_vec(truth = test_panel$class, estimate = test_pred)),
  tibble(.metric = "sensitivity", .est = sensitivity_vec(truth = test_panel$class, estimate = test_pred
  tibble(.metric = "specificity", .est = specificity_vec(truth = test_panel$class, estimate = test_pred
  tibble(.metric = "roc_auc", .est = roc_auc_vec(truth = test_panel$class, estimate = as.numeric(test_p
)

write_csv(test_metrics, file.path(results_dir, "test_metrics_panel.csv"))

# Confusion matrix of data
cm <- yardstick::conf_mat(
  tibble(truth = test_panel$class, estimate = test_pred),
  truth = truth, estimate = estimate
)
cm_tbl <- as_tibble(cm$table) %>% rename(Actual = Truth, Predicted = Prediction)
write_csv(cm_tbl, file.path(results_dir, "test_confusion_matrix.csv"))

print(test_metrics)
```

```
## # A tibble: 4 x 2
```

```
##   .metric      .est
##   <chr>       <dbl>
## 1 accuracy    0.625
## 2 sensitivity 0.625
## 3 specificity 0.625
## 4 roc_auc     0.602
```

```
print(cm$table)
```

```
##           Truth
## Prediction FALSE TRUE
##      FALSE    10    6
##      TRUE      6   10
```

```
# Putting results into report
report_md <- file.path(results_dir, "bullet1_train_test_first_summary.md")
cat(
  "# Bullet 1: Train/Test-First Analysis\n\n",
  "## Panel (selected on training only)\n\n",
  paste0("- k_ttest = ", top_k, ", k_rf = ", top_k, "\n"),
  paste0("- Intersection size = ", length(proteins_panel), "\n"),
  paste0("- Proteins: ", paste(proteins_panel, collapse = ", "), "\n\n"),
  "## Test Set Metrics\n\n",
  paste0(readr::format_csv(test_metrics, na = "", eol = "\n")),
  "\n\n## Confusion Matrix (Test)\n\n",
  paste0(readr::format_csv(cm_tbl, na = "", eol = "\n")),
  file = report_md
)

# Best protein results

ttop <- ttests_out_train %>%
  arrange(p.adj) %>%
  slice_head(n = top_k) %>%
  transmute(rank_t = row_number(),
            protein,
            statistic, p_value, p.adj)

write_csv(ttop, file.path(results_dir, "proteins_ttest_topk.csv"))

cat("\nTop-k from t-tests (training):\n")
```

```
##
## Top-k from t-tests (training):
```

```
print(ttop %>% select(rank_t, protein))
```

```
## # A tibble: 10 x 2
##    rank_t protein
##     <int> <chr>
##  1      1 DERM
##  2      2 IgD
```

```
##  3      3 CK-MB
##  4      4 MAPK2
##  5      5 aldolase A
##  6      6 MAPK14
##  7      7 C1QR1
##  8      8 M2-PK
##  9      9 Calcineurin
## 10     10 RELT
```

```
# Top-k from RF importance (training)
rf_top <- rf_imp_train %>%
  slice_head(n = top_k) %>%
  transmute(rank_rf = row_number(),
            protein,
            MeanDecreaseGini)

write_csv(rf_top, file.path(results_dir, "proteins_rf_topk.csv"))

cat("\nTop-k from RF importance (training):\n")
```

```
##
## Top-k from RF importance (training):
```

```
print(rf_top %>% select(rank_rf, protein))
```

```
## # A tibble: 10 x 2
##    rank_rf protein
##      <int> <chr>
##  1       1 IgD
##  2       2 eIF-4H
##  3       3 MAPK14
##  4       4 MAPK2
##  5       5 CK-MB
##  6       6 M2-PK
##  7       7 DERM
##  8       8 TSP4
##  9       9 aldolase A
## 10      10 DKK3
```

```
# Intersection panel
panel_tbl <- tibble(protein = proteins_panel) %>%
  left_join(ttop %>% select(protein, rank_t), by = "protein") %>%
  left_join(rf_top %>% select(protein, rank_rf), by = "protein") %>%
  arrange(coalesce(rank_t, Inf), coalesce(rank_rf, Inf))

write_csv(panel_tbl, file.path(results_dir, "proteins_panel_intersection.csv"))

cat("\nFinal panel (intersection) used for the logistic model:\n")
```

```
##
## Final panel (intersection) used for the logistic model:
```

```r
print(panel_tbl)
```

```
## # A tibble: 7 x 3
##   protein    rank_t rank_rf
##   <chr>       <int>   <int>
## 1 DERM            1       7
## 2 IgD             2       1
## 3 CK-MB           3       5
## 4 MAPK2           4       4
## 5 aldolase A      5       9
## 6 MAPK14          6       3
## 7 M2-PK           8       6
```