Tuberculosis Bulk RNA-seq Analysis

Table of contents

1 Overview

Singhania et al. profiled whole-blood gene expression from humans with active tuberculosis, latent TB infection (LTBI), and uninfected controls across multiple cohorts, using microarrays and RNA-seq. Their main goal was to identify an optimal reduced gene set to serve as a diagnostic biomarker for active TB that did not pick up other diseases.

The Leicester RNA-seq cohort (GSE107994) used here includes longitudinal whole-blood samples from a low-incidence setting, and consists of active TB patients (n = 53) and recent close contacts (n = 108). The close contacts were categorized as either Interferon-Gamma Release Assay (IGRA) negative (n = 50) or IGRA positive (n = 49), indicating the absence or presence of *Mycobacterium tuberculosis*, respectively. A subset of the initial close contact subjects were identified as having active TB during the longitudinal assessment (n = 9) based on microbiological confirmation.

The authors used a modular analysis framework to group co-expressed genes into immune "modules" (e.g. type I interferon–inducible, B-cell, and T-cell modules). They showed that active TB is characterized by overabundance of type I interferon-inducible transcripts and underabundance of IFNG/TBX21 and B-/T-cell–associated genes, and that some individuals with LTBI display "outlier" active-TB-like signatures, suggesting phenotypic heterogeneity and possible higher progression risk.

For this project intended to familiarize myself with bulk RNA-seq data, I focus first on a cross-sectional baseline comparison between active TB, LTBI, and controls to reproduce key aspects of the transcriptional signature (e.g., interferon-related changes), and then extend to longitudinal dynamics and heterogeneity within the LTBI and LTBI-progressor groups.

Singhania A, Verma R, Graham CM, Lee J et al. A modular transcriptional signature identifies phenotypic heterogeneity of human tuberculosis infection. Nat Commun 2018 Jun 19;9(1):2308. PMID: 29921861

2 The Data

(download data from GSE, establish needed local directories, format data objects)

```
library(tidyverse) # dplyr, tidyr, tibble, readr, ggplot2
library(readxl)
library(Biobase)
library(GEOquery)

dir.create(file.path("data_raw"), showWarnings = FALSE) # establish directory to raw data downloads

data_dir <- file.path("data_raw", "GSE107994") # establish path to specific study directory
```

(describe procedure for downloading data from GSE)

```
if (!dir.exists(data_dir) || length(list.files(data_dir)) == 0) { # checks for supplemental files fro
  getGEOSuppFiles("GSE107994", baseDir = "data_raw") # accesses GEO for supplemental files (gene coun
}

counts_file <- list.files( # grabs the specific file name for the raw gene counts data
  path = data_dir,
  pattern = "Raw_counts.*Leicester.*\\.xlsx$",
  full.names = TRUE
)

counts <- readxl::read_xlsx(counts_file) # function to import .xlsx file as data table
```

(get the phenotypic metadata)

```
gse <- getGEO("GSE107994", GSEMatrix = TRUE) # grabs the series matrix from GEO
phenotypes <- Biobase::pData(gse[[1]]) |> as_tibble() # formats GSE download as tabular metadata
```

(describe following code; create storage directory for reformatted data)

```
dir.create("data_processed", showWarnings = FALSE) # create data_processed directory folder if it doe

write_tsv(phenotypes, file.path("data_processed", "GSE107994_phenotypes.tsv")) # save phenotypes loca
write_tsv(counts, file.path("data_processed", "GSE107994_counts_raw.tsv")) # save counts locally
```

(snapshot of counts and phenotypes tables, discuss the cases)

```
print(counts)
```

```
# A tibble: 58,051 x 178
   Genes    Gene_name Gene_biotype Leicester_with_progr~1 Leicester_with_progr~2
   <chr>    <chr>     <chr>                         <dbl>                   <dbl>
 1 ENSG000~ TSPAN6    protein_cod~                      1                      16
 2 ENSG000~ TNMD      protein_cod~                      0                       0
 3 ENSG000~ DPM1      protein_cod~                    215                     263
 4 ENSG000~ SCYL3     protein_cod~                    233                     333
 5 ENSG000~ C1orf112  protein_cod~                     54                      57
 6 ENSG000~ FGR       protein_cod~                  21694                   18130
 7 ENSG000~ CFH       protein_cod~                     46                      44
 8 ENSG000~ FUCA2     protein_cod~                    604                     428
 9 ENSG000~ GCLC      protein_cod~                     69                     153
10 ENSG000~ NFYA      protein_cod~                    310                     420
# i 58,041 more rows
# i abbreviated names: 1: Leicester_with_progressor_longitudinal_Sample1,
#   2: Leicester_with_progressor_longitudinal_Sample2
# i 173 more variables: Leicester_with_progressor_longitudinal_Sample3 <dbl>,
#   Leicester_with_progressor_longitudinal_Sample4 <dbl>,
#   Leicester_with_progressor_longitudinal_Sample5 <dbl>,
#   Leicester_with_progressor_longitudinal_Sample6 <dbl>, ...
```

```
print(phenotypes)
```

```
# A tibble: 175 x 60
   title             geo_accession status submission_date last_update_date type
   <chr>             <chr>         <chr>  <chr>           <chr>            <chr>
 1 Leicester_with_p~ GSM2886274    Publi~ Dec 12 2017     May 15 2019      SRA
 2 Leicester_with_p~ GSM2886275    Publi~ Dec 12 2017     May 15 2019      SRA
 3 Leicester_with_p~ GSM2886276    Publi~ Dec 12 2017     May 15 2019      SRA
 4 Leicester_with_p~ GSM2886277    Publi~ Dec 12 2017     May 15 2019      SRA
 5 Leicester_with_p~ GSM2886278    Publi~ Dec 12 2017     May 15 2019      SRA
 6 Leicester_with_p~ GSM2886279    Publi~ Dec 12 2017     May 15 2019      SRA
 7 Leicester_with_p~ GSM2886280    Publi~ Dec 12 2017     May 15 2019      SRA
 8 Leicester_with_p~ GSM2886281    Publi~ Dec 12 2017     May 15 2019      SRA
 9 Leicester_with_p~ GSM2886282    Publi~ Dec 12 2017     May 15 2019      SRA
10 Leicester_with_p~ GSM2886283    Publi~ Dec 12 2017     May 15 2019      SRA
# i 165 more rows
# i 54 more variables: channel_count <chr>, source_name_ch1 <chr>,
#   organism_ch1 <chr>, characteristics_ch1 <chr>, characteristics_ch1.1 <chr>,
#   characteristics_ch1.2 <chr>, characteristics_ch1.3 <chr>,
#   characteristics_ch1.4 <chr>, characteristics_ch1.5 <chr>,
#   characteristics_ch1.6 <chr>, characteristics_ch1.7 <chr>,
#   characteristics_ch1.8 <chr>, characteristics_ch1.9 <chr>, ...
```

3 Analysis 1 - Baseline Cross-Section

(Active TB vs inactive. Baseline samples. No outliers. Both sexes) (look for enrichment of type I IFN-inducible genes) (outliers determined by authors from transcriptional analyses)

```
phenotypes_baseline <- phenotypes |> # subset phenotypes to the columns of interest
  select(
    geo_accession,
    patient_id = 'patient_id:ch1',
    group = 'group:ch1',
    tb_disease_type = 'tb_disease_type:ch1',
    smear_result = 'smear_result:ch1',
    outlier = 'outlier:ch1',
    gender = 'gender:ch1',
    ethnicity = 'ethnicity:ch1',
    birthplace = 'birth_place:ch1',
    age_baseline = 'age_at_baseline_visit:ch1',
    timepoint_months = 'timepoint_months:ch1',
    visit_date = 'visit_date:ch1',
    title
) |>
  mutate( # apply some factor levels to multiple variables of interest
    group = factor(group, levels = c("Control", "Active_TB", "LTBI", "LTBI_Progressor")),
    tb_disease_type = factor(tb_disease_type),
    smear_result = factor(smear_result, levels = c("Negative", "Positive")),
    gender = factor(gender)
  ) |>
  dplyr::filter(
    outlier == "No", # filter the data to remove predetermined outliers ...
    timepoint_months == "Baseline" # ... and focus on those who had active/inactive TB at baseline
  )

write_tsv(phenotypes_baseline, file = file.path("data_processed", "phenotypes_baseline.tsv"))

counts_mat <- counts |> # need to convert counts to a simple matrix
  select(-Gene_name, -Gene_biotype) |>
  column_to_rownames("Genes") |>
  as.matrix()

colnames(counts_mat) <- phenotypes$geo_accession[match(colnames(counts_mat), phenotypes$title)] # rep

counts_baseline <- counts_mat[, phenotypes_baseline$geo_accession] # select only baseline samples
```

3.1 Preprocessing

(describe intro to limma workflow)

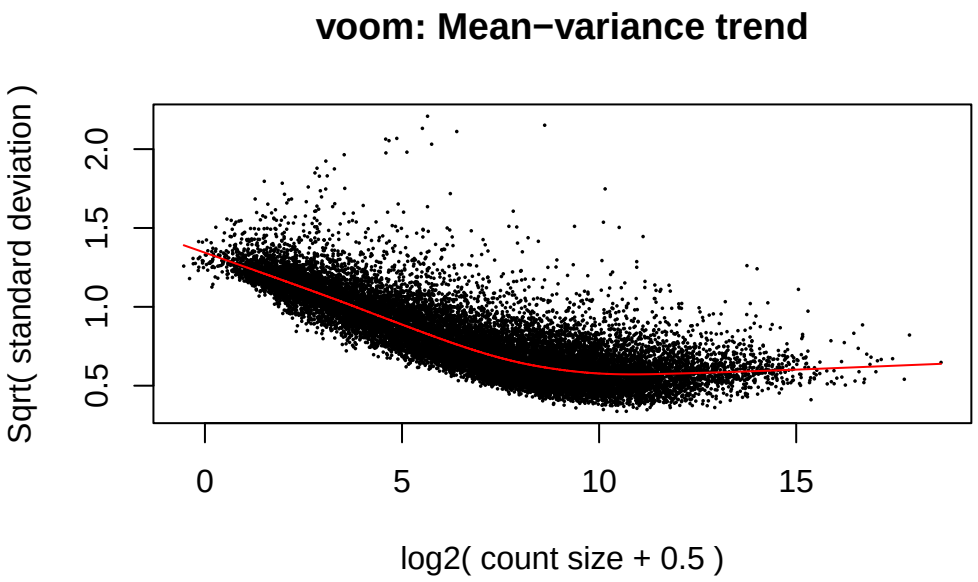
```
library(edgeR) # limma loaded as dependency

dge <- DGEList(counts = counts_baseline) # create DGEList object from raw counts matrix

keep_genes <- filterByExpr(dge, group = phenotypes_baseline$group) # Determine which genes have suffi

dge <- dge[keep_genes, ] # retains only the genes determined to have sufficiently large counts

dge <- calcNormFactors( # Calculate scaling factors to convert the raw library sizes into normalized
  dge,
  method = "TMM" # trimmed mean of M-values
) # the data are not normalized yet. The factors are just stored in dge$samples$norm.factors for late
```

(discuss DGE object and norm factors, not yet applied)

```
           group lib.size norm.factors
GSM2886274     1 18956705    0.8145288
GSM2886275     1 21023702    1.1360703
GSM2886279     1 21229317    0.9785857
GSM2886280     1 14304650    0.8690486
GSM2886281     1 12560609    0.8428912
GSM2886282     1 20234496    0.9660870
```

(code commentary)

```
design_mat <- model.matrix(~ 0 + group, data = phenotypes_baseline) # prepare the design

colnames(design_mat) <- levels(phenotypes_baseline$group) # adjust names

voom_res <- voom(counts = dge, design = design_mat, plot = TRUE) # transform count data to log2 count
```



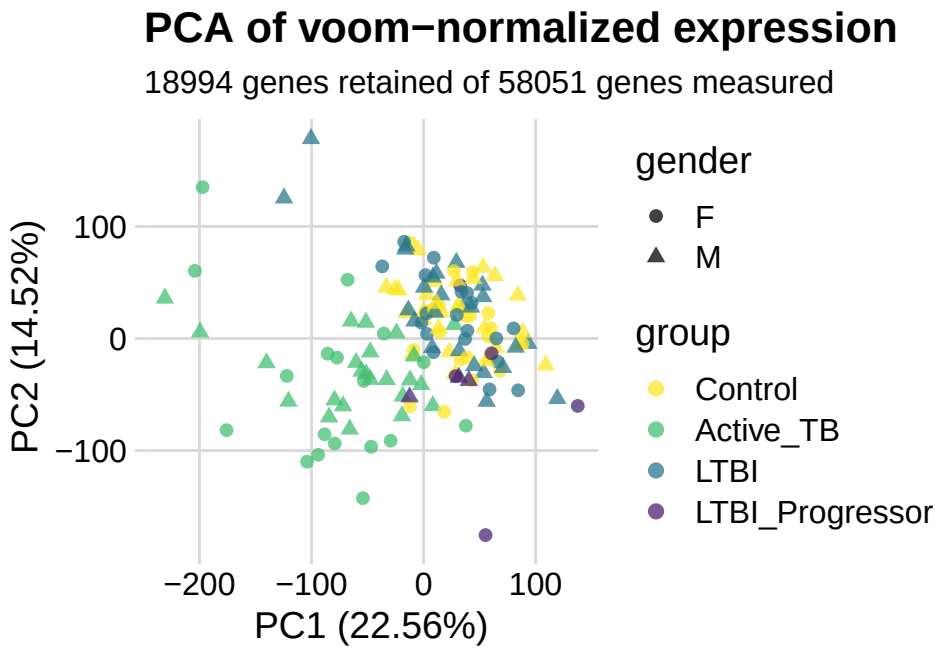

```
  # incorporates the calc norm factors
```

(code commentary)

3.2 Principal Components Analysis

```
pca <- prcomp( # perform PCA
  x = t(voom_res$E), # must transpose data to samples x variables format
  center = TRUE,
  scale. = TRUE
)

pca$rotation |>
  as.data.frame() |>
  rownames_to_column("gene") |>
  write_csv(file = file.path("data_processed", "analysis1_pca_loadings.csv"))

pca_df <- pca$x |>
  as.data.frame() |>
  rownames_to_column("case")

write_csv(pca_df, file = file.path("data_processed", "analysis1_pca_scores.csv"))

prop_var <- pca$sdev^2 / sum(pca$sdev^2) * 100 # calculaute proportion of variance per PC

dir.create(file.path("output", "plots"), showWarnings = FALSE, recursive = TRUE)

library(cowplot)

pca_plot <- pca_df |>
  left_join(y = phenotypes_baseline, by = c("case" = "geo_accession")) |>
  ggplot(aes(x = PC1, y = PC2, color = group, shape = gender)) +
    geom_point(alpha = 0.75, size = 2) +
    theme_minimal_grid() +
    coord_fixed(ratio = 1) +
    scale_color_viridis_d(direction = -1, begin = 0.1, end = 1) +
    labs(
      title = "PCA of voom-normalized expression",
      x = paste0("PC1 (", round(prop_var[1], 2), "%)"),
      y = paste0("PC2 (", round(prop_var[2], 2), "%)"),
      subtitle = paste0(nrow(voom_res), " genes retained of ", nrow(counts), " genes measured")
    )

save_plot(
  filename = file.path("output", "plots", "analysis1_pca.png"),
  plot = pca_plot,
  bg = "white"
)
```

(describe PCA overview) (noticeable contrast between active TB and the recent contact subjects)



3.3 Fitting linear models

(uses original limma function)

```
fit <- lmFit(
  object = voom_res,
  design = design_mat
)

print(head(coef(fit)))
```

```
                  Control  Active_TB       LTBI LTBI_Progressor
ENSG00000000003 -2.4447037 -3.7693669 -2.3521709       -1.642404
ENSG00000000419  3.6259457  3.4583260  3.6513872        3.972407
ENSG00000000457  3.9059472  3.9985405  3.8422006        4.017250
ENSG00000000460  1.4440985  1.4855627  1.4428381        1.630376
ENSG00000000938  9.9293754 10.4531684  9.8231253        9.991104
ENSG00000000971  0.7392169  0.8703221  0.9992953        1.047232
```

(describe limma output)

```
contrasts_mat <- makeContrasts(
  ATB_vs_allContacts = Active_TB - (Control + LTBI + LTBI_Progressor)/3,
  ATB_vs_Controls = Active_TB - Control,
  Progressors_vs_LTBI = LTBI_Progressor - LTBI,
  levels = design_mat
)

fit2 <- contrasts.fit(fit, contrasts_mat) |>
  eBayes()

print(head(coef(fit2)))
```

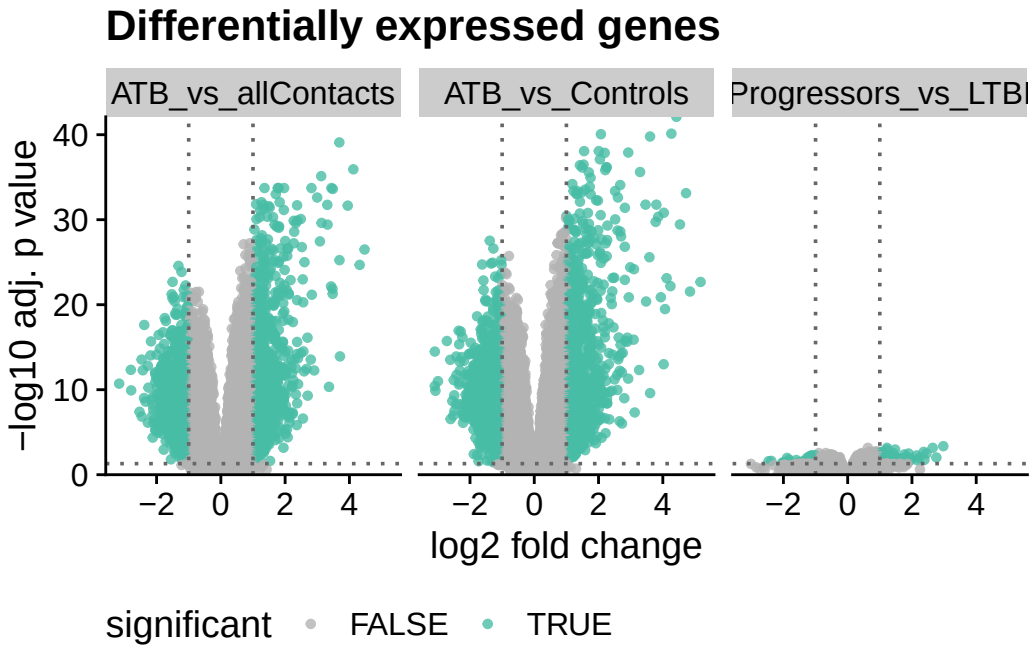
```
                Contrasts
                 ATB_vs_allContacts ATB_vs_Controls Progressors_vs_LTBI
  ENSG00000000003        -1.62294081     -1.32466319          0.70976725
  ENSG00000000419        -0.29158719     -0.16761968          0.32101953
  ENSG00000000457         0.07674126      0.09259326          0.17504932
  ENSG00000000460        -0.02020818      0.04146429          0.18753820
  ENSG00000000938         0.53863351      0.52379306          0.16797869
  ENSG00000000971        -0.05825919      0.13110521          0.04793635
```

(describe additional steps of making contrasts, eBayes)

```r
dir.create(file.path("output", "tables"), showWarnings = FALSE, recursive = TRUE)

DE_genes <- list()

for (i in seq_along(colnames(contrasts_mat))) {

  top_table <- topTable(
    fit = fit2,
    coef = i,
    number = Inf,
    adjust.method = "BH",
    sort.by = "P"
  ) |>
    rownames_to_column("gene")

  top_table |>
    dplyr::filter(adj.P.Val < .05) |> # only export significant results
    write_csv(
      file = file.path(
        "output", "tables",
        paste0("analysis1_limma_", colnames(contrasts_mat)[i], ".csv")
      )
    )

  DE_genes[[colnames(contrasts_mat)[i]]] <- top_table

}
```

3.4 Plotting differentially expressed features

```
DE_genes_df <- bind_rows(DE_genes, .id = "contrast") # collapse list to data frame

DE_genes_plot <- DE_genes_df |>
  mutate(significant = adj.P.Val < .05 & abs(logFC) > 1) |>
  ggplot(aes(x = logFC, y = -log10(adj.P.Val), color = significant)) +
    geom_point(size = 1, alpha = 0.8) +
    facet_wrap(~ contrast, nrow = 1) +
    theme_half_open() +
    theme(legend.position = "bottom") +
    scale_color_manual(values = c("FALSE" = "grey70", "TRUE" = "#48bda6")) +
    scale_y_continuous(expand = c(0,0), limits = c(0,NA)) +
    geom_hline(yintercept = -log10(0.05), linetype = "dotted", color = "grey40") +
    geom_vline(xintercept = 1, linetype = "dotted", color = "grey40") +
    geom_vline(xintercept = -1, linetype = "dotted", color = "grey40") +
    labs(
      x = "log2 fold change",
      y = "-log10 adj. p value",
      title = "Differentially expressed genes"
    )

cowplot::save_plot(
  filename = file.path("output", "plots", "analysis1_limma_contrasts.png"),
  plot = DE_genes_plot,
  bg = "white"
)

DE_genes_plot
```



4 Analysis 2

(look at the longitudinal view within LTBI)