

Tuberculosis Bulk RNA-seq Analysis

Table of contents

1	Overview	2
2	The Data	3
2.1	Download data from GSE	3
2.2	Inspection and storage	4
3	Analysis 1 - Baseline Cross-Section	6
3.1	Preprocessing	7
3.2	Principal Components Analysis	9
3.3	Fitting linear models	11
3.4	Differentially expressed genes	13
3.5	PCA on DE genes subset	16
3.6	Biotypes of DE genes	18
4	WIP	20

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

The first step is to load some initial R packages to make managing data objects easier (e.g. tidyverse, readxl) and to specifically handle data from the Gene Expression Omnibus (GEO; e.g. GEOquery, Biobase). The code chunk below also establishes a directory to store the data.

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

dir.create(file.path("data_raw"), showWarnings = FALSE)
```

2.1 Download data from GSE

Accessing GEO can be time-intensive, so the first section of this code chunk checks if the necessary files are already downloaded before attempting to access GEO. The following code then converts the Excel file into data frame.

```
data_dir <- file.path("data_raw", "GSE107994")

# Accessing GEO can be time-intensive ...
# ... so this checks if the directory and files are present first.
if (!dir.exists(data_dir) || length(list.files(data_dir)) == 0) {
  getGEOSuppFiles("GSE107994", baseDir = "data_raw")
}

counts_file <- list.files(
  path = data_dir,
  pattern = "Raw_counts.*Leicester.*\\.xlsx$",
  full.names = TRUE
)

counts <- readxl::read_xlsx(counts_file)
```

This section accesses GEO again to retrieve the series matrix, which contains the metadata (e.g. patient info, phenotypes) for the RNA-seq profiles.

```
# Grab the series matrix (sample metadata) from GEO ...
gse <- getGEO("GSE107994", GSEMatrix = TRUE)

# ... and format as a usable table.
phenotypes <- Biobase::pData(gse[[1]]) |> as_tibble()
```

2.2 Inspection and storage

The counts object contains raw expression counts for 58,051 genes from 175 whole-blood profiles. Also listed are the gene names and general category (e.g. protein-coding, sRNA, rRNA, non-coding).

```
# 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>, ...
```

The phenotypes object contains all the information regarding the RNA-seq profiles, including patient info and RNA sequencing steps.

```
# 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  Public~ Dec 12 2017   May 15 2019   SRA
  2 Leicester_with_p~ GSM2886275  Public~ Dec 12 2017   May 15 2019   SRA
  3 Leicester_with_p~ GSM2886276  Public~ Dec 12 2017   May 15 2019   SRA
  4 Leicester_with_p~ GSM2886277  Public~ Dec 12 2017   May 15 2019   SRA
  5 Leicester_with_p~ GSM2886278  Public~ Dec 12 2017   May 15 2019   SRA
  6 Leicester_with_p~ GSM2886279  Public~ Dec 12 2017   May 15 2019   SRA
  7 Leicester_with_p~ GSM2886280  Public~ Dec 12 2017   May 15 2019   SRA
  8 Leicester_with_p~ GSM2886281  Public~ Dec 12 2017   May 15 2019   SRA
  9 Leicester_with_p~ GSM2886282  Public~ Dec 12 2017   May 15 2019   SRA
 10 Leicester_with_p~ GSM2886283  Public~ 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>, ...
```

Before moving on to analysis, the data are saved to an intermediate folder for easy referencing later.

```
dir.create("data_processed", showWarnings = FALSE)

write_tsv(phenotypes, file.path("data_processed", "GSE107994_phenotypes.tsv"))
write_tsv(counts, file.path("data_processed", "GSE107994_counts_raw.tsv"))
```

3 Analysis 1 - Baseline Cross-Section

The first analysis will be focused on differences in expression profiles between active TB cases and recent contacts, and on recovering the type I IFN signatures determined by Singhania et al. The data are subset to baseline profiles and outliers previously determined by the authors are removed ahead of analysis.

The code below selects columns of interest from the metadata and renames them for easier calling. Then, it establishes some factor levels to be used in subsequent analyses and visualizations before filtering down to the desired samples. The counts object is also reformatted into a simple matrix for compatibility with various modeling functions.

```
phenotypes_baseline <- phenotypes |>
  select( # Subset phenotypes to the columns of interest.
    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 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",
    timepoint_months == "Baseline"
  )
  # Filter the data to remove predetermined outliers ...
  # ... and focus on the baseline visit profiles.

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

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

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

# Select only cases that match the baseline subset.
counts_baseline <- counts_mat[, phenotypes_baseline$geo_accession]
```

3.1 Preprocessing

The edgeR package is loaded for some of its functions used to prepare the data for modeling, and it calls the limma package as a dependency. One such function, `filterByExpr()`, marks genes for retention that have a minimum count of reads in a worthwhile number of samples. These cutoffs are determined based on each profile's library size, and the process is based on a filtering strategy outlined in Chen et al. (2016).

Chen Y, Lun ATL, and Smyth GK (2016). From reads to genes to pathways: differential expression analysis of RNA-Seq experiments using Rsubread and the edgeR quasi-likelihood pipeline. *F1000Research* 5, 1438. doi:10.12688/f1000research.8987.2 <https://doi.org/10.12688/f1000research.8987.2>

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

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

# Determine which genes have sufficient counts to be retained in a statistical analysis.
# Then, retain only the genes determined to have sufficiently large counts.
keep_genes <- filterByExpr(dge, group = phenotypes_baseline$group)

dge <- dge[keep_genes, ]

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

The DGE object contains the expression values, sample information, and calculated normalization factors. These factors are not applied to the data yet, but are stored in the object for later use.

	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

Importantly, only 18,994 genes were retained by `filterByExpr()` of the initial 58,051. The analyses below are only taking into account 32.72% of the measured genes, just those determined to have large enough counts to potentially exhibit a signal against background noise.

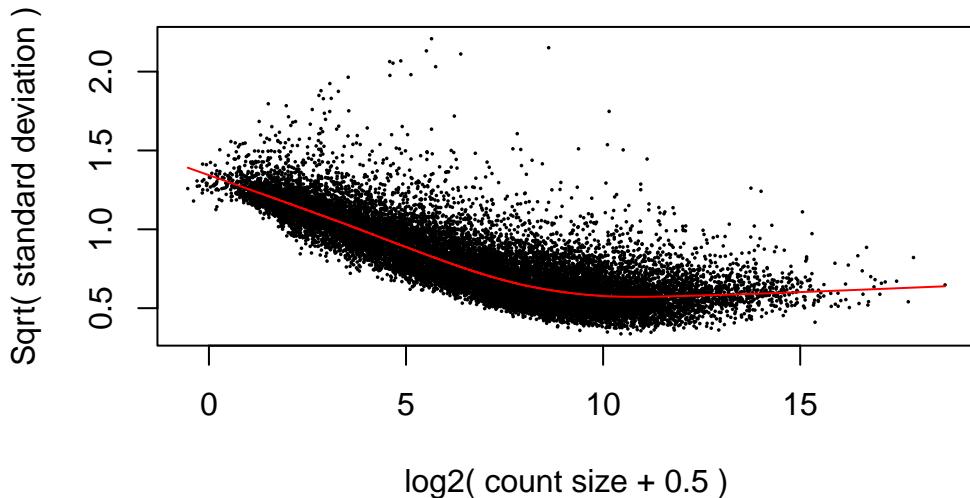
Next, a design matrix is prepared from the levels in \$group to guide the modeling. The voom() function from limma transforms the raw counts into log2 counts-per-million (log2CPM) so that the data are usable to limma's workflow.

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

colnames(design_mat) <- levels(phenotypes_baseline$group)

# Transform the counts data into log2 counts-per-million (log2CPM).
# This step incorporates the calculated normalization factors.
voom_res <- voom(counts = dge, design = design_mat, plot = TRUE)
```

voom: Mean–variance trend



Specifically, this is what voom is doing:

1. Counts are transformed to log2 counts per million reads (CPM), based on the normalization factors calculated earlier.
2. A linear model is fitted to each gene, and the residuals are calculated.
3. A smoothed curve is fitted to the $\text{sqrt}(\text{residual standard deviation})$ by average expression.
4. Weights for each gene and sample are calculated from the smoothed curve and stored for use in limma along with the log2 CPMs.

3.2 Principal Components Analysis

With the preprocessing complete, Principal Components Analysis (PCA) can be used to provide an overview of the sequencing profiles and general variance in the data.

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

# Save the PCA loadings.
pca$rotation |>
  as.data.frame() |>
  rownames_to_column("gene") |>
  write_csv(file = file.path("data_processed", "analysis_1_pca_loadings.csv"))

# Prepare the principal components.
pca_df <- pca$x |>
  as.data.frame() |>
  rownames_to_column("case")

write_csv(pca_df, file = file.path("data_processed", "analysis_1_pca_scores.csv"))

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

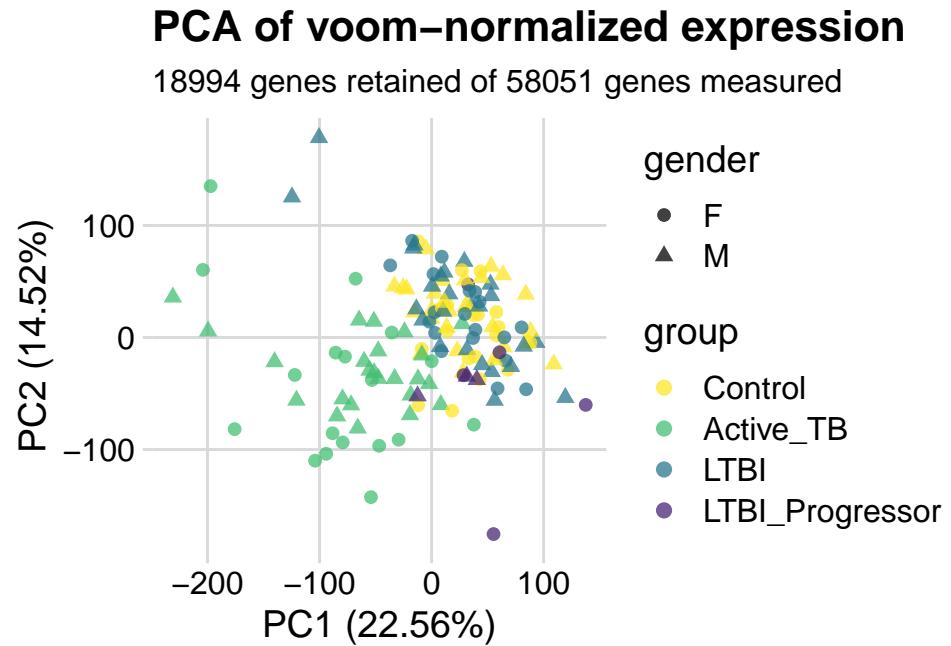
dir.create(file.path("output", "analysis_1", "plots"), showWarnings = FALSE, recursive = TRUE)

library(cowplot)

# Plot the principal components, coloring by group.
p_pca <- 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", "analysis_1", "plots", "pca.png"),
  plot = p_pca,
  bg = "white"
)
```

While the PCA does not cleanly discriminate all groups, PC1 generally shows a contrast between the active TB cases and the recent contact pool of cases. No apparent trend is captured by PC2, and notably, the profiles do not demonstrate any gender-dependent clustering.



3.3 Fitting linear models

The next sensible step is to look into which features drive the differences observed in the PCA and to explore other contrasts of interest. In the following code chunk, the voom-normalized expression values and design matrix are submitted to limma's lmFit() function. This function fits a linear model for each gene in a series.

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

	Control	Active_TB	LTBI	LTBI_Progressor
ENSG000000000003	-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

The coefficients of the fitted models describe the differences between the RNA sources hybridized to the arrays. To actually test the differences between groups, first, the group comparisons are specified in a contrasts matrix. This produces coefficients indicating whether a given gene is enriched in one group versus another.

```
# Specify the group contrasts to be tested.  
contrasts_mat <- makeContrasts(  
  ATB_vs_allContacts = Active_TB - (Control + LTBI + LTBI_Progressor)/3,  
  ATB_vs_Controls = Active_TB - Control,  
  ATB_vs_LTBI = Active_TB - LTBI,  
  Progressors_vs_LTBI = LTBI_Progressor - LTBI,  
  levels = design_mat  
)  
  
# Compute the coefficients and standard errors ...  
# ... then apply empirical Bayes moderation of the standard errors towards a global value.  
fit2 <- contrasts.fit(fit, contrasts_mat) |>  
  eBayes()  
  
print(head(coef(fit2)))
```

	ATB_vs_allContacts	ATB_vs_Controls	ATB_vs_LTBI
ENSG000000000003	-1.62294081	-1.32466319	-1.41719599
ENSG00000000419	-0.29158719	-0.16761968	-0.19306118
ENSG00000000457	0.07674126	0.09259326	0.15633992
ENSG00000000460	-0.02020818	0.04146429	0.04272468
ENSG00000000938	0.53863351	0.52379306	0.63004309
ENSG00000000971	-0.05825919	0.13110521	-0.12897321

	Progressors_vs_LTBI
--	---------------------

ENSG000000000003	0.70976725
ENSG00000000419	0.32101953
ENSG00000000457	0.17504932
ENSG00000000460	0.18753820
ENSG00000000938	0.16797869
ENSG00000000971	0.04793635

3.4 Differentially expressed genes

The for loop below cycles through each contrast to extract the differentially abundant genes from the linear model fit. The p values are adjusted for multiple comparisons using the Benjamini-Hochberg method.

```
dir.create(file.path("output", "analysis_1", "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", "analysis_1", "tables",
        paste0("limma_", colnames(contrasts_mat)[i], ".csv")
      )
    )

  # Gather the results in a list.
  DE_genes[[colnames(contrasts_mat)[i]]] <- top_table
}

}
```

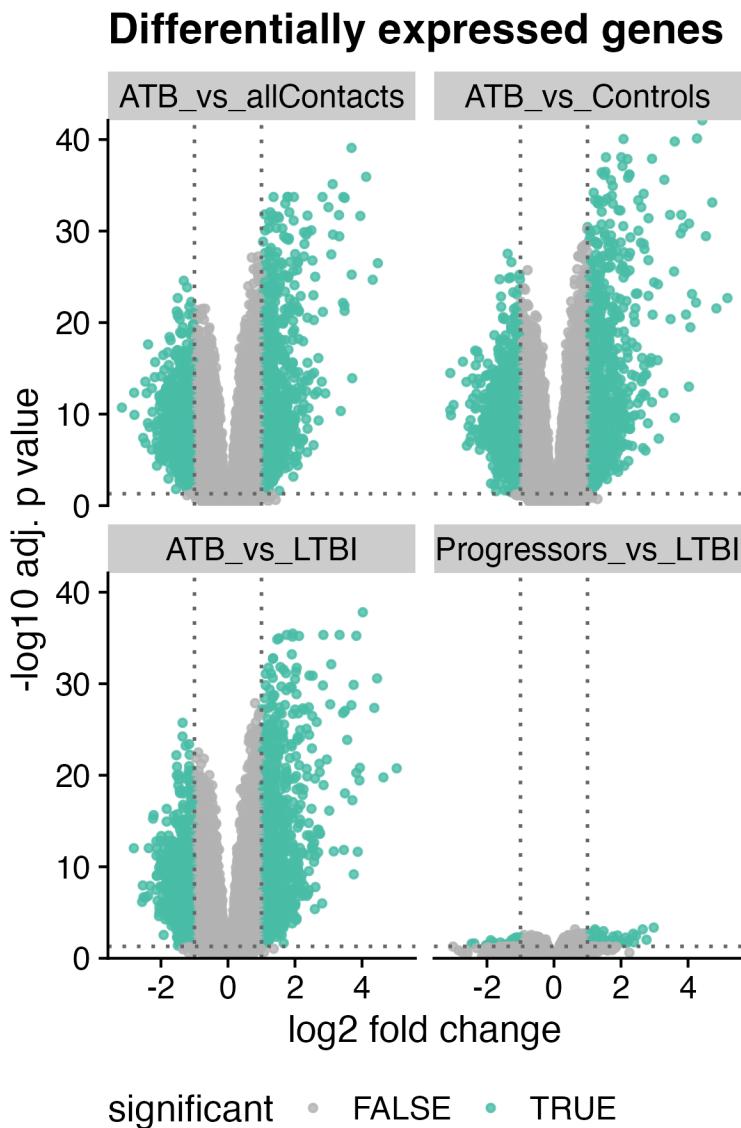
This next section collapses the list of differentially-expressed genes into a dataframe for plotting each contrast in a facet of the figure.

```
# Collapse results list to dataframe.
DE_genes_df <- bind_rows(DE_genes, .id = "contrast")

# Create volcano plots of gene log2 fold changes by contrast.
p_DE_genes <- 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) +
  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", "analysis_1", "plots", "limma_contrasts.png"),
  plot = p_DE_genes,
  bg = "white",
  base_height = 6,
  base_width = 4
)
```

These volcano plots corroborate the group differences observed in the PCA, where active TB cases show distinct expression profiles compared to the recent contact cases.



For these active TB contrasts, over half of the retained features were found to be differentially expressed. The LTBI_Progressor vs LTBI profiles still exhibited over 2,000 differentially expressed genes.

```
DE_genes_df |> filter(adj.P.Val < .05) |> count(contrast)
```

	contrast	n
1	ATB_vs_Controls	12203
2	ATB_vs_LTBI	11886
3	ATB_vs_allContacts	12269
4	Progressors_vs_LTBI	2270

3.5 PCA on DE genes subset

To check the disease “signal” captured by these DE genes, another PCA is performed on the voom-normalized expression data subset to contain only the 12,269 DE genes from the active TB versus all recent contacts contrast.

```
DE_activeTB <- DE_genes_df |>
  filter(contrast == "ATB_vs_allContacts" & adj.P.Val < .05) |>
  pull(gene) |>
  unique()

expr_DE_activeTB <- voom_res$E[DE_activeTB, ]

pca_DE_activeTB <- prcomp(
  x = t(expr_DE_activeTB),
  center = TRUE,
  scale. = TRUE
)

prop_var <- pca_DE_activeTB$sdev^2 / sum(pca_DE_activeTB$sdev^2) * 100

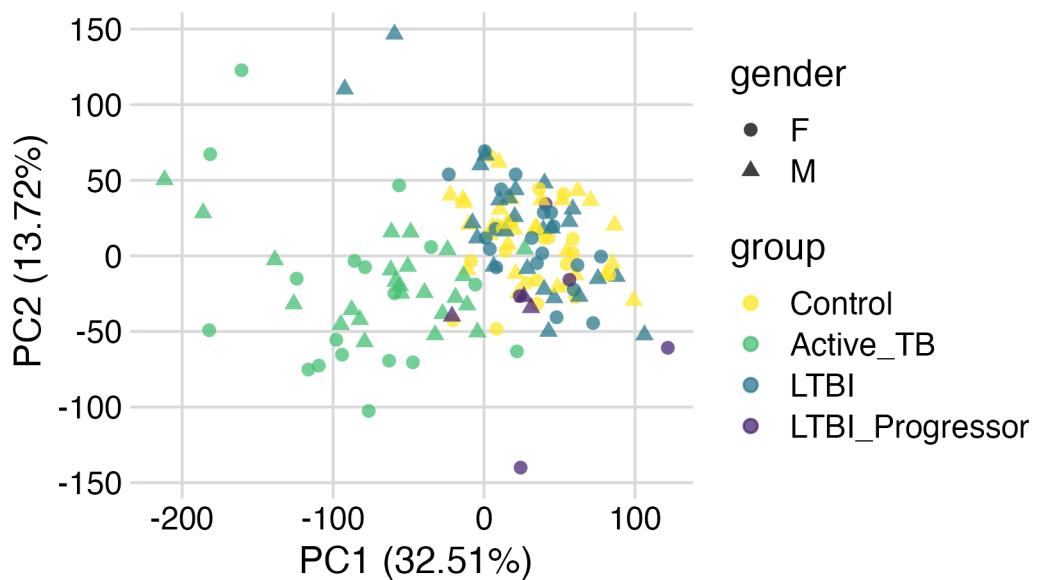
p_pca_DE_activeTB <- pca_DE_activeTB$x |>
  as.data.frame() |>
  rownames_to_column("case") |>
  left_join(y = phenotypes_baseline, by = c("case" = "geo_accession")) |>
  ggplot(aes(x = PC1, y = PC2, color = group, shape = gender)) +
  geom_point(alpha = 0.75, size = 2) +
  theme_minimal_grid() +
  coord_fixed(ratio = 1) +
  scale_color_viridis_d(direction = -1, begin = 0.1, end = 1) +
  labs(
    title = "PCA of voom-normalized expression",
    subtitle = paste0(
      "On ", length(DE_activeTB),
      " DE genes for active TB (n = 53) vs contacts (n = 108)"
    ),
    x = paste0("PC1 (", round(prop_var[1], 2), "%)"),
    y = paste0("PC2 (", round(prop_var[2], 2), "%)")
  )

cowplot::save_plot(
  filename = file.path("output", "analysis_1", "plots", "pca_DE_activeTB.png"),
  plot = p_pca_DE_activeTB,
  bg = "white"
)
```

This DE-subset PCA shows the same structure and distribution as the PCA on the global data. Notably, more variance is captured in the first PC, while slightly less variance is captured by PC2, compared to the global PCA. This is an expected result considering over half of the genes retained by initial library size filters were found to be differentially expressed between these groups.

PCA of voom-normalized expression

On 12269 DE genes for active TB (n = 53) vs contacts (n = 108)



3.6 Biotypes of DE genes

To quickly gauge the kinds of genes contributing to these differences, the biotype information provided by the initial counts file can be easily counted both globally and per contrast from limma.

```
global_biotypes <- counts |>
  count(Gene_biotype) |>
  mutate(contrast = "Global")

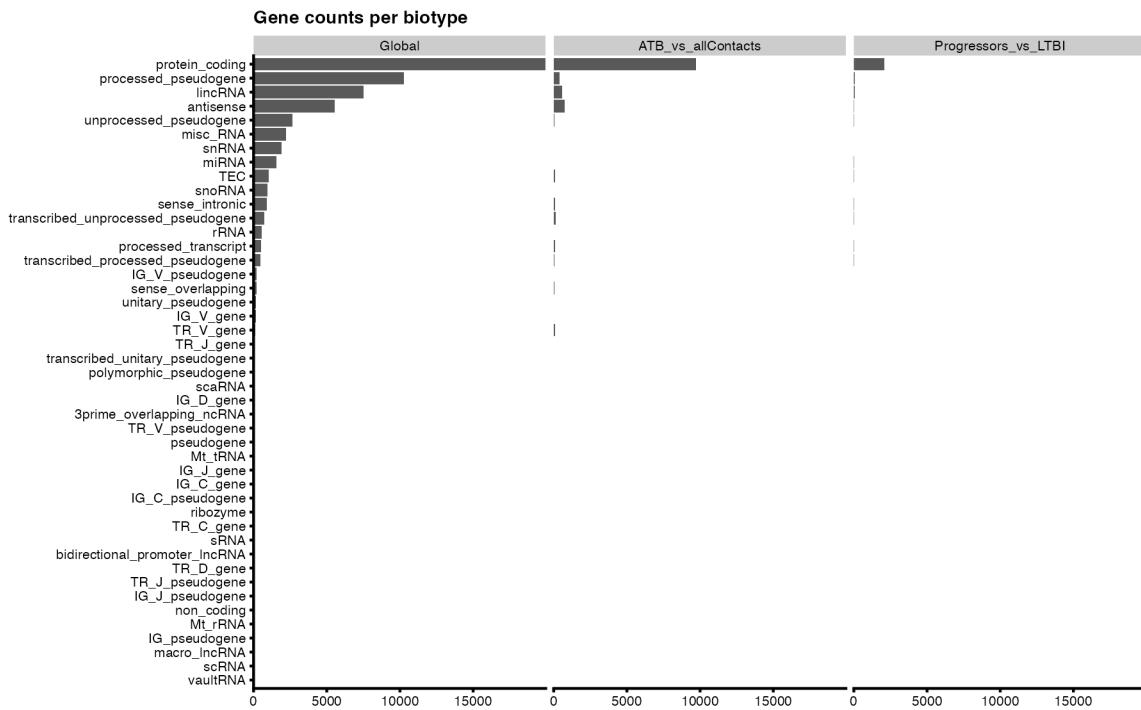
DE_genes_info <- left_join(
  x = DE_genes_df,
  y = select(counts, Genes, Gene_name, Gene_biotype),
  by = c("gene" = "Genes")
)

DE_biotype_counts <- DE_genes_info |>
  filter(adj.P.Val < .05) |> # Retain only significant features.
  group_by(contrast) |>
  count(Gene_biotype) |> # Counts of genes belonging to each biotype, per group.
  ungroup() |> # Ungroup to add new cases.
  bind_rows(global_biotypes) # Add global biotype counts.

p_biotypes <- DE_biotype_counts |>
  filter(!contrast %in% c("ATB_vs_Controls", "ATB_vs_LTBI")) |> # Remove uninformative contrasts.
  mutate(contrast = factor(contrast,
    levels = c("Global", "ATB_vs_allContacts", "Progressors_vs_LTBI"))
)) |>
  group_by(contrast) |>
  mutate(Gene_biotype =forcats::fct_reorder(Gene_biotype, n, .desc = TRUE)) |>
  ungroup() |>
  ggplot(aes(x = Gene_biotype, y = n)) +
  geom_col() +
  scale_x_discrete(limits = rev) +
  scale_y_continuous(limits = c(0, NA), expand = c(0,0)) +
  coord_flip() +
  facet_wrap(~ contrast, drop = FALSE) +
  theme_half_open(6) +
  labs(
    title = "Gene counts per biotype",
    x = NULL,
    y = NULL
  )

cowplot::save_plot(
  filename = file.path("output", "analysis_1", "plots", "gene_biotypes_counts.png"),
  plot = p_biotypes,
  bg = "white"
)
```

The majority of genes accounted for in the data are, as would be expected, protein-coding (around 19,000). The majority of differentially expressed genes are similarly of the protein-coding variety. Many other biotypes have counts below 100 or even below 10, and thus are practically invisible on the plot.



4 WIP

- check gene biotypes and identities
- compare DE genes to authors' findings
- Analysis 2 — longitudinal view within LTBI