BEN-seq pipeline: Quantification and Analysis

John M. Ma

8/30/2021

Quantification

A sample usage of `benseq_cli_frontend.py` will be demonstrated in the following cell in bash mode.

```
python //home/jma/nativeApp_BEN_test/NativeApp_BENseq/benseq_cli_frontend.py -a1 /home/data_data
store/Analysis/exp000150_Bulk_Th1_Th2_PBMC/fqlink/Th0_D606_bkADT_S2_L001_R1_001.fastq.gz -a2 /ho
me/data_datastore/Analysis/exp000150_Bulk_Th1_Th2_PBMC/fqlink//Th0_D606_bkADT_S2_L001_R2_001.fas
tq.gz -f /home/jma/nativeApp_BEN_test/exp000150/tags.csv -r1 /home/data_datastore/Analysis/exp00
0150_Bulk_Th1_Th2_PBMC/fqlink//Th0_D606_bkRNA_S2_L001_R1_001.fastq.gz -r2 /home/data_datastore/A
nalysis/exp000150_Bulk_Th1_Th2_PBMC/fqlink//Th0_D606_bkRNA_S2_L001_R2_001.fastq.gz -t /home/data
_datastore/Resource/GRCh38_kallisto/homo_sapiens/transcriptome.idx -o ./Th0_D606/
```

```
## 2021-09-01 16:06:33,723 Protein will be quantified.
## 2021-09-01 16:06:33,724 RNA will be quantified.
## 2021-09-01 16:06:33,724 Weaving RNA files...
## 2021-09-01 16:06:33,724 Weaving ADT files...
## 2021-09-01 16:06:33,724 Quantifing...
## 2021-09-01 16:06:33,724 Quantifying RNA expression with Kallisto quant...
## 2021-09-01 16:13:34,217
## 2021-09-01 16:13:34,218 [quant] fragment length distribution will be estimated from the data
## 2021-09-01 16:13:34,218 [index] k-mer length: 31
## 2021-09-01 16:13:34,218 [index] number of targets: 188,748
## 2021-09-01 16:13:34,218 [index] number of k-mers: 109,543,393
## 2021-09-01 16:13:34,218 [index] number of equivalence classes: 760,741
## 2021-09-01 16:13:34,218 [quant] running in paired-end mode
## 2021-09-01 16:13:34,218 [quant] will process pair 1: /home/data_datastore/Analysis/exp000150_
Bulk_Th1_Th2_PBMC/fqlink//Th0_D606_bkRNA_S2_L001_R1_001.fastq.gz
## 2021-09-01 16:13:34,218                                 /home/data_datastore/Analysis/exp000150_
Bulk_Th1_Th2_PBMC/fqlink//Th0_D606_bkRNA_S2_L001_R2_001.fastq.gz
## 2021-09-01 16:13:34,218 [quant] finding pseudoalignments for the reads ... done
## 2021-09-01 16:13:34,218 [quant] learning parameters for sequence specific bias
## 2021-09-01 16:13:34,218 [quant] processed 108,796,113 reads, 90,613,276 reads pseudoaligned
## 2021-09-01 16:13:34,218 [quant] estimated average fragment length: 155.817
## 2021-09-01 16:13:34,218 [   em] quantifying the abundances ... done
## 2021-09-01 16:13:34,218 [   em] the Expectation-Maximization algorithm ran for 2,672 rounds
## 2021-09-01 16:13:34,219
## 2021-09-01 16:13:34,219 Quantifying ADT expression with Kallisto quant...
## 2021-09-01 16:13:34,219 Generataing ADT search index with Kite and Kallisto...
## 2021-09-01 16:13:38,803
## 2021-09-01 16:13:38,804 [build] loading fasta file ./Th0_D606//ADT_kall_index//kall_idx.fa
## 2021-09-01 16:13:38,804 [build] k-mer length: 15
## 2021-09-01 16:13:38,804 [build] counting k-mers ... done.
## 2021-09-01 16:13:38,804 [build] building target de Bruijn graph ...  done
## 2021-09-01 16:13:38,804 [build] creating equivalence classes ...  done
## 2021-09-01 16:13:38,804 [build] target de Bruijn graph has 279402 contigs and contains 279402
k-mers
## 2021-09-01 16:13:38,805
## 2021-09-01 16:13:52,459
## 2021-09-01 16:13:52,459 [quant] fragment length distribution will be estimated from the data
## 2021-09-01 16:13:52,459 [index] k-mer length: 15
## 2021-09-01 16:13:52,459 [index] number of targets: 596,712
## 2021-09-01 16:13:52,459 [index] number of k-mers: 279,402
## 2021-09-01 16:13:52,459 [index] number of equivalence classes: 876,114
## 2021-09-01 16:13:52,459 [quant] running in paired-end mode
## 2021-09-01 16:13:52,459 [quant] will process pair 1: /home/data_datastore/Analysis/exp000150_
Bulk_Th1_Th2_PBMC/fqlink/Th0_D606_bkADT_S2_L001_R1_001.fastq.gz
## 2021-09-01 16:13:52,459                                 /home/data_datastore/Analysis/exp000150_
Bulk_Th1_Th2_PBMC/fqlink//Th0_D606_bkADT_S2_L001_R2_001.fastq.gz
## 2021-09-01 16:13:52,459 [quant] finding pseudoalignments for the reads ... done
## 2021-09-01 16:13:52,459 [quant] learning parameters for sequence specific bias
## 2021-09-01 16:13:52,459 [quant] processed 8,155,171 reads, 7,620,144 reads pseudoaligned
## 2021-09-01 16:13:52,459 [quant] estimated average fragment length: 0
## 2021-09-01 16:13:52,459 [   em] quantifying the abundances ... done
## 2021-09-01 16:13:52,459 [   em] the Expectation-Maximization algorithm ran for 52 rounds
```

```
## 2021-09-01 16:13:52,459
## 2021-09-01 16:13:52,460 Cleaning temporary files...
## 2021-09-01 16:13:52,480 Finished!
## 2021-09-01 16:13:52,481 ADT quantification results saved to ./Th0_D606//ADT/
## 2021-09-01 16:13:52,481 RNA quantification results saved to ./Th0_D606//RNA/
## 2021-09-01 16:13:52,481 This log is also saved to ./Th0_D606//ben_seq.log
```

After running the cell above, A new directory called `Th0_D606` is created under the working directory. The following is its content, generated using the bash command `tree`:

```
## ./Th0_D606/
## ├── ADT
## │   ├── abundance.h5
## │   ├── abundance.tsv
## │   └── run_info.json
## ├── ben_seq.log
## └── RNA
##     ├── abundance.h5
##     ├── abundance.tsv
##     └── run_info.json
##
## 2 directories, 7 files
```

The `ADT` and `RNA` directories hold the results of running `kallisto quant` on the respective libraries, using appropriate indices. Both directories contain `abundance.h5`, a h5 file containing the quantification results, `abundance.tsv`, the tab-separated-text representation of the same data, and `run_info.json`, a JSON file containing quantification metrics. `ben_seq.log` is the log generated by `benseq_cli_frontend.py`

From this point on, all content will be in R.

Quantification

Experimental setup

In this experiment, Naive T Cells (Th0) from three donors (D092, D606 and D837) were stimulated into two types of THelper Cells,Th1 andTh2. Here, we're looking at RNA and Protein changes in these three kinds of cells.

Loading R modules

This analysis requires the following R modules: `DESeq2` for differential expression profiling, `tximport` for importing Kallisto results, `tidyverse` for data wrangling, `ComplexHeatmap` to generate heatmaps, `scico` to provide colour scales, `circlize` to generate color scales for the use of `ComplexHeatmap`, and `rhdf5` for `h5df` file I/O.

```
library(tidyverse)
```

```
## Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'
## had status 1
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✓ ggplot2 3.3.4     ✓ purrr   0.3.4
## ✓ tibble  3.1.2     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.3     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(EnhancedVolcano)
```

```
## Loading required package: ggrepel
```

```
library(DESeq2)
```

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which, which.max, which.min
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, rename
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

```
## The following object is masked from 'package:base':
##
##     expand.grid
```

```
## Loading required package: IRanges
```

```
##
## Attaching package: 'IRanges'
```

```
## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice
```

```
## The following object is masked from 'package:purrr':
##
##     reduce
```

```
## Loading required package: GenomicRanges
```

```
## Loading required package: GenomeInfoDb
```

```
## Loading required package: SummarizedExperiment
```

```
## Loading required package: Biobase
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
## Loading required package: DelayedArray
```

```
## Loading required package: matrixStats
```

```
##
## Attaching package: 'matrixStats'
```

```
## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians
```

```
## The following object is masked from 'package:dplyr':
##
##     count
```

```
## Loading required package: BiocParallel
```

```
##
## Attaching package: 'DelayedArray'
```

```
## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges
```

```
## The following object is masked from 'package:purrr':
##
##     simplify
```

```
## The following objects are masked from 'package:base':
##
##     aperm, apply, rowsum
```

```
library(tximport)
library(ComplexHeatmap)
```

```
## Loading required package: grid
```

```
## ========================================
## ComplexHeatmap version 2.2.0
## Bioconductor page: http://bioconductor.org/packages/ComplexHeatmap/
## Github page: https://github.com/jokergoo/ComplexHeatmap
## Documentation: http://jokergoo.github.io/ComplexHeatmap-reference
##
## If you use it in published research, please cite:
## Gu, Z. Complex heatmaps reveal patterns and correlations in multidimensional
##   genomic data. Bioinformatics 2016.
## ========================================
```

```
library(circlize)
```

```
## ========================================
## circlize version 0.4.13
## CRAN page: https://cran.r-project.org/package=circlize
## Github page: https://github.com/jokergoo/circlize
## Documentation: https://jokergoo.github.io/circlize_book/book/
##
## If you use it in published research, please cite:
## Gu, Z. circlize implements and enhances circular visualization
##   in R. Bioinformatics 2014.
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(circlize))
## ========================================
```

```
library(rhdf5)
library(scico)
```

Data Preparation

First, let us make a table of sample properties. This is required for the two-factor analysis. Note the `Phase` and `Donor` fields are modified to the `factor` type with alphabetically ascending levels; this is to facilitate the generation of experimental contrasts during DESeq2.

```
sample_table <- expand_grid(
  Phase=c('Th0', 'Th1'), Donor=c('D092', 'D606')) %>%
  unite(col='sample_name', Phase, Donor, remove=FALSE) %>%
  mutate(Phase=factor(Phase), Donor=factor(Donor))
print(sample_table)
```

```
## # A tibble: 4 x 3
##   sample_name Phase Donor
##   <chr>       <fct> <fct>
## 1 Th0_D092    Th0   D092
## 2 Th0_D606    Th0   D606
## 3 Th1_D092    Th1   D092
## 4 Th1_D606    Th1   D606
```

Generating vectors of the `abundance.h5` files with sample names of the items' names.

```
adt.input.vector <- paste0("./", sample_table$sample_name, "/ADT/abundance.h5")
names(adt.input.vector) <- sample_table$sample_name

rna.input.vector <- paste0("./", sample_table$sample_name, "/RNA/abundance.h5")
names(rna.input.vector) <- sample_table$sample_name

print(adt.input.vector)
```

```
##                       Th0_D092                       Th0_D606
## "./Th0_D092/ADT/abundance.h5" "./Th0_D606/ADT/abundance.h5"
##                       Th1_D092                       Th1_D606
## "./Th1_D092/ADT/abundance.h5" "./Th1_D606/ADT/abundance.h5"
```

ADT Differential Expression Analyses

Reading data in with `tximport`

The package `tximport` from BioConductor is a powerful tool to read in the results of major gene quantification software, including Kallisto. However, it needs a `tx2gene` data frame for isoform-gene conversion, but we can obtain this data from the `.h5` data on the fly, since we know that for ADT, the "transcript" name is just the ADT IDs tagged with where the mismatches were generated during `kite2hd`.

```
h5.data <- h5read(adt.input.vector[1], 'aux')
adt_t2g <- tibble(transcript = h5.data$ids) %>%
  separate(transcript, sep='-', into=c("gene"), remove=FALSE, extra='drop')
print(head(adt_t2g))
```

```
## # A tibble: 6 x 2
##   transcript        gene
##   <chr>             <chr>
## 1 ADT_A1018         ADT_A1018
## 2 ADT_A1018-*-*-0-1 ADT_A1018
## 3 ADT_A1018-*-*-0-2 ADT_A1018
## 4 ADT_A1018-*-*-0-3 ADT_A1018
## 5 ADT_A1018-*-*-1-1 ADT_A1018
## 6 ADT_A1018-*-*-1-2 ADT_A1018
```

And then we can use `tximport()` to read in the quantified expression data and combine it to gene-level expressions.

```
adt.tximport <- tximport(files=adt.input.vector, type='kallisto', varReduce=TRUE, tx2gene=adt_t2g)
```

```
## 1 2 3 4
## summarizing abundance
## summarizing counts
## summarizing length
```

Importing and normalizing with `DESeq2`

The following cell converts the `tximport` object cerated above into a `DESeq2` object in preparation of expression analysis. As mentioned above, we're planning on perform a 2-factor analyis, with expression being a factor of both `Phase` and `Donor`.

```
adt.deseq2 <- DESeqDataSetFromTximport(adt.tximport, colData=sample_table, design= ~Phase+Donor)
```

```
## using counts and average transcript lengths from tximport
```

```
print(adt.deseq2)
```

```
## class: DESeqDataSet
## dim: 282 4
## metadata(1): version
## assays(2): counts avgTxLength
## rownames(282): ADT_A0005 ADT_A0006 ... ADT_A0948 ADT_A1018
## rowData names(0):
## colnames(4): Th0_D092 Th0_D606 Th1_D092 Th1_D606
## colData names(3): sample_name Phase Donor
```

The following step perform normalization. According to our experience, the dispersions of ADT data should be fitted using the `local` method (which uses `locfit`).

```
adt.deseq2 <- DESeq(adt.deseq2, fitType='local')
```

```
## estimating size factors
```

```
## using 'avgTxLength' from assays(dds), correcting for library size
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

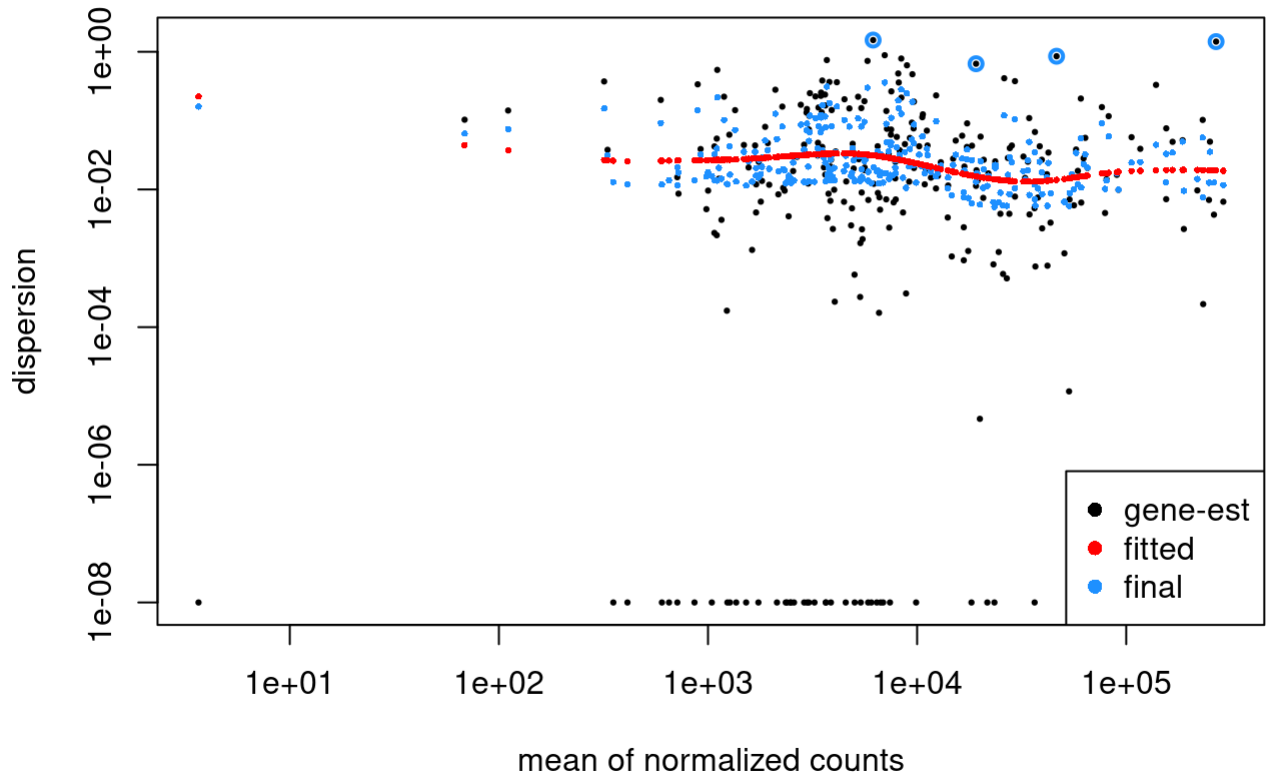
```
## final dispersion estimates
```

```
## fitting model and testing
```

Diagnoistic Figures

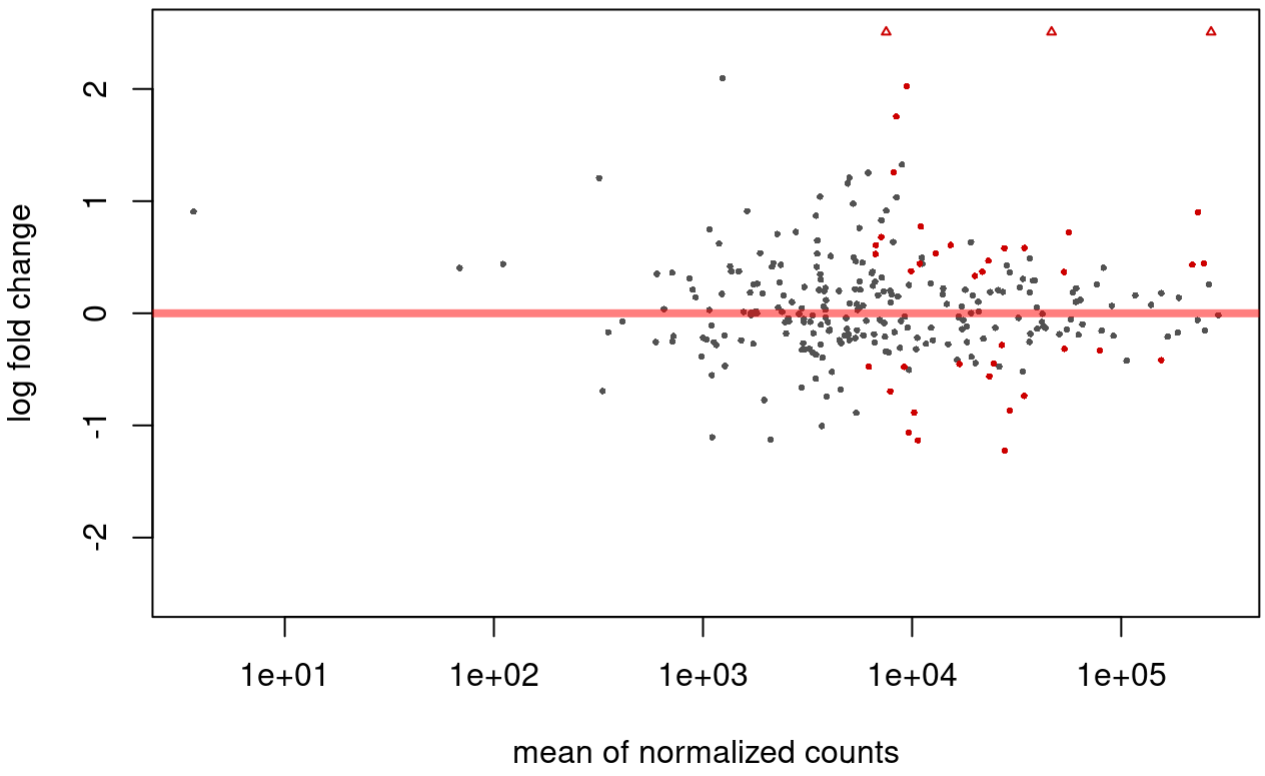
`DESeq2` provides several ways to visualize the data for diagnose the normalization. ##### Dispersion estimates

```
plotDispEsts(adt.deseq2)
```



MA-plot

```
plotMA(adt.deseq2)
```



Statistical Analyses of Differential Expression

While historically `DESeq2` uses `results()` to calculate the p-values of differential expression in a data set, `lfcShrink()` is now the preferred method. `lfcShrink()` estimates the variance (specifically a posterior Standard Deviation) of the log2-scaled fold changes(LFC), and (using the default arguments) calculates the p-value of the sign of LFC being false for a particular feature.

```
adt.lfcshrink <- lfcShrink(adt.deseq2, coef="Phase_Th1_vs_Th0", type='apeglm', svalue=TRUE)
```

```
## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

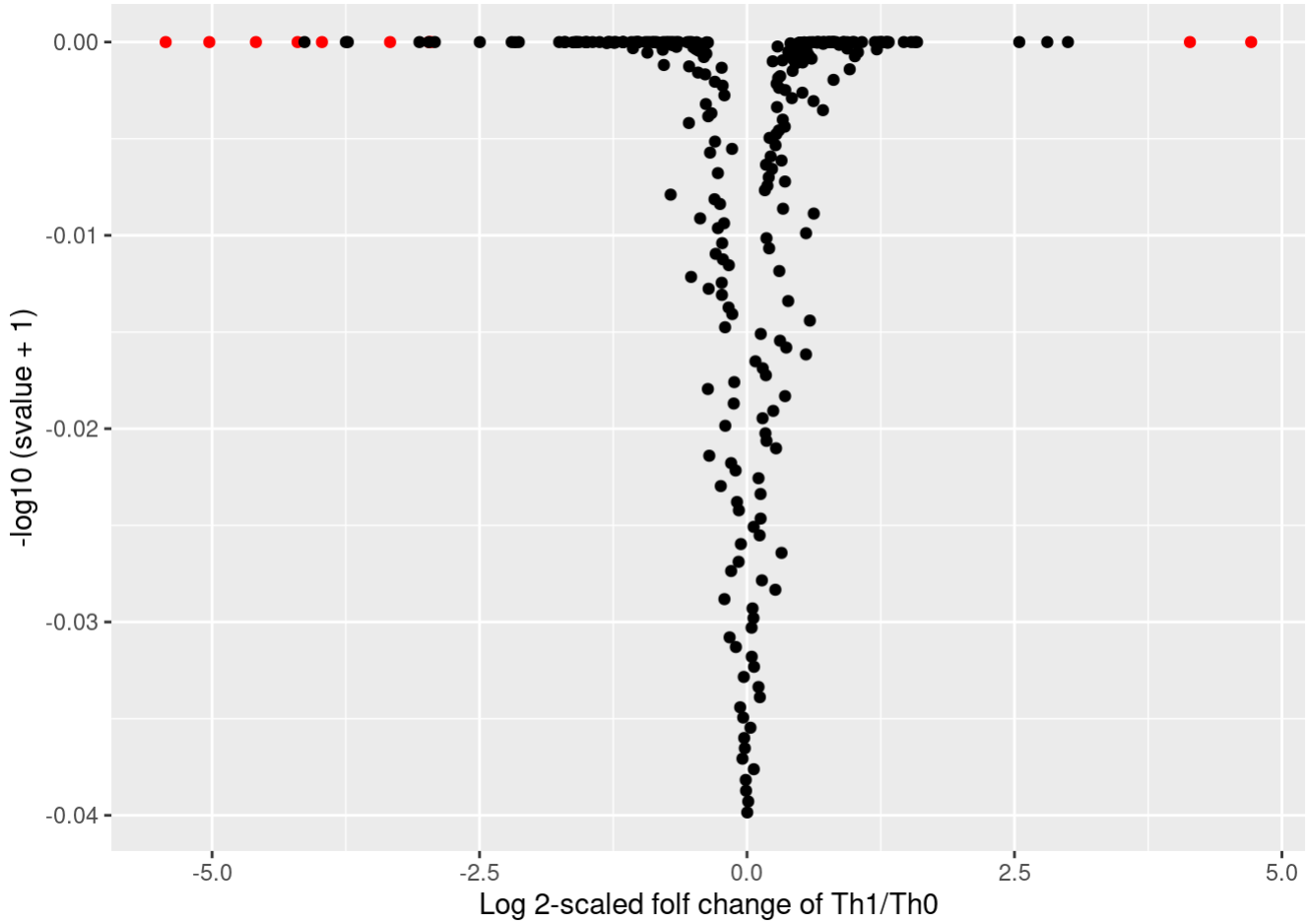
For this demonstration, I will only take the most differentially expressed features, i.e. those with s-values smaller than 1e-50.

```
## log2 fold change (MAP): Phase Th1 vs Th0
##
## DataFrame with 10 rows and 4 columns
##                   baseMean    log2FoldChange             lfcSE
##                  <numeric>         <numeric>         <numeric>
## ADT_A0032 38576.5561414673  4.71263616978008 0.231511192600302
## ADT_A0072 79214.1465518094 -2.97367828661029 0.146236623372163
## ADT_A0083 12307.2014097747 -5.43356551234765 0.185443990515427
## ADT_A0146 218831.449228247  4.14013188026028 0.172315417933249
## ADT_A0147 7720.45466289222 -4.20066914124049 0.189123434215693
## ADT_A0161 7114.42806338414  -2.9525296250367  0.19182567746705
## ADT_A0216 27167.3396608771 -4.59019885021454 0.186549719533268
## ADT_A0218 23171.5176693187 -5.02593825066074 0.120279422358447
## ADT_A0853 10873.4365456707 -3.33612433579473 0.190329451035539
## ADT_A0866 11021.7461478795  -3.9729351002628 0.210385191329904
##                          svalue
##                       <numeric>
## ADT_A0032  3.42490065132678e-93
## ADT_A0072  7.46580824081927e-93
## ADT_A0083    2.56971817308e-189
## ADT_A0146 1.83869511912906e-128
## ADT_A0147 2.67354717964993e-110
## ADT_A0161  9.29916785019272e-55
## ADT_A0216 1.82640938069865e-134
## ADT_A0218                     0
## ADT_A0853  4.85314670260448e-70
## ADT_A0866  9.63383301624401e-81
```

Visualizing differentially expressed features with volcano plots

We are using simple Tidyverse (`dplyr` / `ggplot2`) functions to make a simple volcano plot with the 10 genes chosen above highlighted.

```
tbl.object <- rownames_to_column(as.data.frame(adt.lfcshrink), "gene_ids") %>%
  select(gene_ids, log2FoldChange, svalue) %>%
  mutate(
    svalue = -log10(svalue + 1),
    group = case_when(gene_ids %in% top.genes ~ 'Most DE', TRUE~'Others'),
    group = factor(group, levels = c("Most DE", "Others"))
  )
ggplot(tbl.object, aes(x=log2FoldChange, y=svalue, colour=group)) +
  geom_point() +
  scale_color_discrete(type=c('red', 'black')) +
  xlab ("Log 2-scaled folf change of Th1/Th0") +
  ylab ("-log10 (svalue + 1)") +
  guides(color='none')
```



Visualizing differentially expressed features with heatmap

Generating a matrix of Variant-Stablizing Transformed counts

The steps above do not, strictly speaking, calculate a "normalized" expression for every cell of the expression matrix; it merely creates mathematical models of the features. However, to generate a heat map, we /do/ need to have a normalized expression expression. This can be generated by `varianceStabilizingTransformation()`, which calculates a variance-stablizing transformation from the models, and then applies it on the raw reads. This function also corrects for size factors--while it's certainly not an issue in ADT analyses, it is performed out of consistency.

The argument `blind` controls whether the expression is to be /blind/ed to the experimental design. In this case, it should not be, since we should be looking at differences after accounting for it.

```
adt.vst <- assay(varianceStabilizingTransformation(adt.deseq2, blind=FALSE))
```

Reading in Feature information file

As you notice, all the gene expression data uses the product ID as an identifier. I'm now using the Feature information file--the same file used during the pipeline--to load the data.

```
adt.feature.data <- as.data.frame(column_to_rownames(read_csv('tags.csv'), "id"))
```

```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   id = col_character(),
##   name = col_character(),
##   read = col_character(),
##   pattern = col_character(),
##   sequence = col_character(),
##   feature_type = col_character()
## )
```

Preparing for ComplexHeatmap

`ComplexHeatmap` is a very powerful module for drawing heatmaps, however the color scale of the plot needs to be identified beforehand.

```
adt.heatmap.mtx <- adt.vst[top.genes, ]
adt.colorRamp <- colorRamp2(seq(min(adt.heatmap.mtx), max(adt.heatmap.mtx), length.out=256),
                            colors=scico(256, begin=0.1, end=0.9, palette='vikO'))
adt.topnames <- adt.feature.data[top.genes, "name"]
```

```
Heatmap(matrix=adt.heatmap.mtx, col=adt.colorRamp,
        name="Phase_Th1_vs_Th0", cluster_rows=FALSE, row_labels=adt.topnames,
        heatmap_legend_param=list(title="VST-normalized\nExpression"))
```