A reproducible re-analysis of the RNA-seq study of Jaffe *et al.* (2014)

Nima Hejazi
nh@nimahejazi.org

November 2016

1 Introduction

The present project concerns the re-analysis of transcriptomic data from the study described in the paper "Developmental regulation of human cortex transcription and its clinical relevance at base resolution", Jaffe *et al.*, *Nature Neuroscience*. This document details the full bioinformatical and statistical pipeline used to pre-process and analyze data from the aforementioned study. *As the full codebase produced for this re-analysis cannot be adequately contained in a single document, please consult the GitHub repository for this project at* `https://github.com/nhejazi/neurodevstat` *to examine all scripts in detail.*

2 Pseudo-alignment of RNA-seq reads

In order to quantify RNA-Seq reads, alignment against a reference transcriptome must be performed. Here, we take advantage of **pseudo-alignment** to probabilistically align reads using the **Kallisto** software pacakge. Below, we describe pseudo-alignment and the results of its application to the Jaffe *et al.* data. Pseudo-alignment of reads constituted the primary pre-processing step necessary for re-analysis of the data; for all scripts in the pre-processing pipeline, see this GitHub link: `https://github.com/nhejazi/neurodevstat/tree/master/preprocess`.

2.1 Summary of pseudo-alignment procedure

Pseudo-alignment is a novel process for quantifying a set of samples of RNA-Seq reads by performing partial matching against a reference transcriptome. The novel pseudo-alignment process, implemented in the command line tool **kallisto**, takes into account all of the information contained in a set of reads while reducing the computational burden imposed by more traditional alignment techniques. For a complete description of pseudo-alignment, consult the paper "Near-optimal probabilistic RNA-seq quantification", Bray *et al.*, *Nature Biotechnology*, 2016.

2.2 Results of pseudo-alignment procedure

Using a publicly available trascriptome assembled from the **GRCh38** *Homo sapiens* genome, sets of paired-end RNA-Seq reads for each of the 12 subjects involved in the study were pseudo-aligned, resulting in count tables mapping each set of reads to **173,259** transcriptomic objects. *Please note that while similar quantification tools (e.g., Cufflinks) produce estimates of mappings of isoforms, kallisto provides estimates of transcript abundance.* Tables of pseudocounts are produced for each set of paired-end RNA-Seq reads for each subject in a tab-separated file format; these files are suitable for concatenation into a single count table containing quantification results for all subjects, which, after summarizing transcripts at the gene level, can be used as input for statistical analysis.

2.3 Sample code for pseudo-alignment with Kallisto

The Kallisto software package was used to perform pseudo-alignment of paired-end RNA-seq reads. After installation, Kallisto is available as a command-line tool. In order to invoke the Kallisto pseudo-aligner on each pair of RNA-seq reads, a wrapper script was written (in Python) to iteratively pass calls to the shell to invoke the pseudo-aligner. The wrapper script is available on GitHub at `https://github.com/nhejazi/neurodevstat/blob/master/preprocess/04_pseudoAlign.py`. **Excerpted code is displayed below:**

```
import os
import sys
import subprocess
import numpy as np

dir_data = os.path.abspath(os.getcwd() + "/data/" + str(data_dir))
dir_fastq = dir_data + "/" + fastq_dir
dir_out = dir_data + "/" + out_dir

samples = [s[:10] for s in os.listdir(dir_fastq)]
samples = list(np.unique(samples))

for i in samples:
    pseudoalign = ("kallisto quant -i" + " " +
                   "./data/Homo_sapiens.GRCh38.rel79.idx -o" + " " +
                   str(dir_data) + "/" + str(out_dir) + "/" + str(i) + " " +
                   "-b 100" + " " + dir_fastq + "/" + str(i) + "_1.fastq.gz"
                   + " " + dir_fastq + "/" + str(i) + "_2.fastq.gz")
    subprocess.call(pseudoalign, shell = True)
```

3 Gene-level summarization of transcripts

To prepare for statistical analysis, the R package **tximport** was used to summarize the transcripts quantified by the **Kallisto** pseudo-aligner at the level of known genes. This step in the pre-processing pipeline is necessary in order to justify the downstream use of popular software packages for statistical modeling. R scripts for performing this stage of bioinformatical and statistical pre-processing are available in the *munge* subdirectory of the GitHub repository for this project (see this link `https://github.com/nhejazi/neurodevstat/tree/master/munge`). **Excerpted code for gene-level summarization is displayed below:**

```
# summarize data from transcript to genes for modeling and inference
txdf <- transcripts(EnsDb.Hsapiens.v79, columns = c("tx_id", "gene_name"),
                    return.type = "DataFrame")
tx2gene <- as.data.frame(txdf)
txi <- tximport(filenames, type = "kallisto", tx2gene = tx2gene,
                reader = read_tsv) #, countsFromAbundance = "scaledTPM")

pseudocounts_genes <- as.data.frame(txi$counts)
colnames(pseudocounts_genes) <- sapply(strsplit(filenames, split = "/"),
                                       function(x) x[9])
pseudocounts_genes$geneID <- rownames(pseudocounts_genes)
```

4 Statistical analysis at the level of genes

4.1 Pre-processing of (pseudo)counts with Limma "voom"

In order to analyze differential expression after summarizing transcripts at the gene level, the linear modeling method of the popular R package **Limma** was employed, including the "voom" transformation for analyzing RNA-seq digital sequencing data in the form of counts (or, in this case pseudocounts). In order to adequately use this transformation, we first filter out all genes for which there were less than 10 mapped reads across subjects *on average*. After this filtering step, the "voom" transformation, including quality weights for samples, was performed. Sample code and the resultant plot are displayed below:

```
v_simple <- voomWithQualityWeights(pseudocounts_filtered, design_simple,
                                   normalization = "scale", plot = TRUE)
```

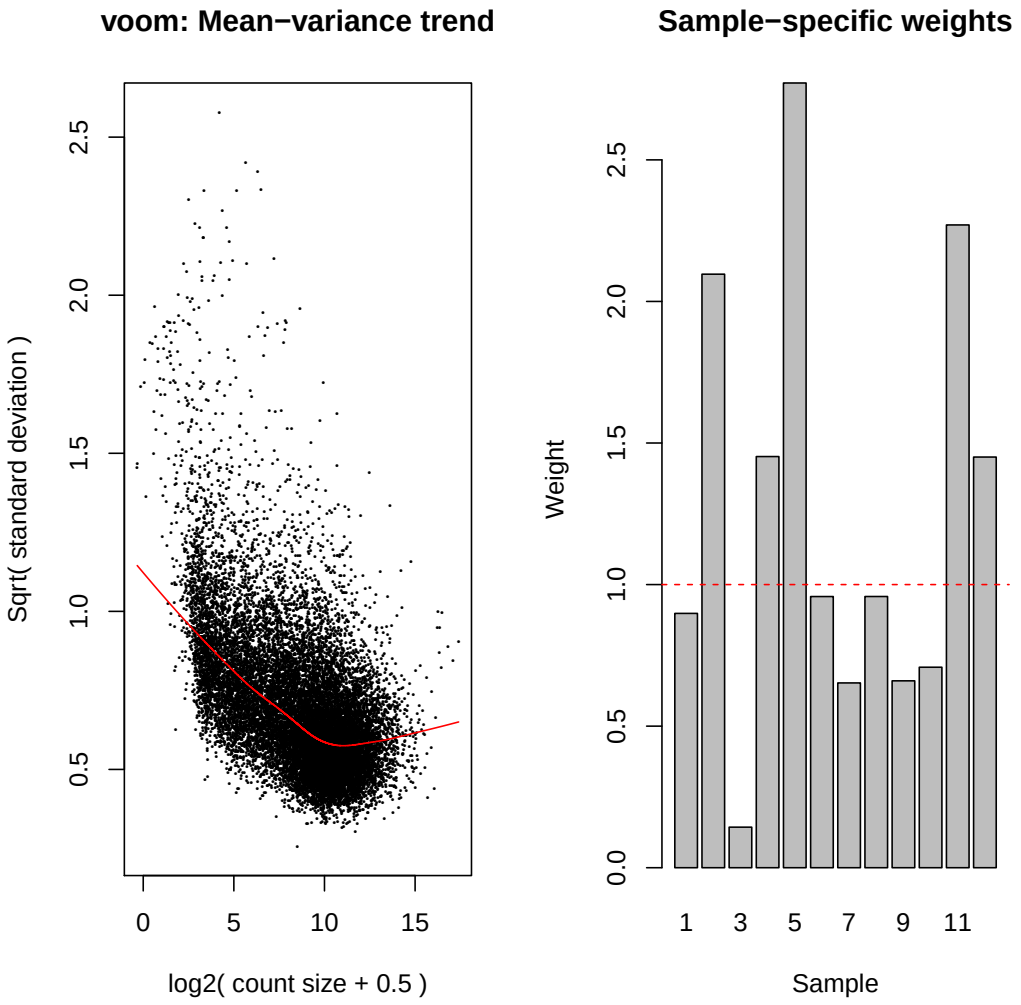

Figure 1: Linear mean-variance trend and sample-specific weighting.

4.2 Statistical analysis via linear modeling with Limma

After summarizing transcripts at the gene level and generating subject-specific weights via the "voom" transformation, the standard linear modeling method of **Limma** was employed to analyze differential expression. Using a simple design matrix, including terms only for an intercept and sample status (adult vs.

fetal), linear models were fit to each gene, and a moderated t-statistic is computed based on the difference of means and a shrunken variance estimate, the latter computed from performing empirical Bayes shrinkage across all genes. The resulting table of differential expression is cleaned using the **dplyr** R package, with the table being used downstream for data visualization. For scripts used in the statistical analysis, see the *src* subdirectory in the project repo: `https://github.com/nhejazi/neurodevstat/tree/master/src`. **Sample code for linear modeling with Limma is given below:**

```
vfit_simple <- limma::lmFit(v_simple)
vfit_simple <- limma::eBayes(vfit_simple)
tt1 <- limma::topTable(vfit_simple,
                       coef = which(colnames(design_simple) == "type"),
                       adjust.method = "BH", number = Inf,
                       sort.by = "none", confint = TRUE)

# clean up topTable output to generate results tables
tt_out1 <- tt1 %>%
  dplyr::mutate(
    geneID = geneIDs,
    lowerCI = exp(CI.L),
    FoldChange = exp(logFC),
    upperCI = exp(CI.R),
    pvalue = I(P.Value),
    fdrBH = I(adj.P.Val)
  ) %>%
  dplyr::select(which(colnames(.) %ni% colnames(tt1)))
```

4.3 Data Visualization and Results

After assessing differential expression via the suite of statistical tests available in the linear modeling R package **Limma**, genes were ranked by the computed Benjamini-Hochberg FDR and a volcano plot (using **ggplot2**) of the results was generated, with labels for the *the top 25 genes*. Just as in the case of statistical analyses performed, all code for data visualization can be found in the *src* subdirectory of the project GitHub repository (available at this link `https://github.com/nhejazi/neurodevstat/tree/master/src`). **Sample code for and the resultant volcano plot are given below:**

```
p3 <- ggplot(tt_out1_gg, aes(x = logFC, y = logPval)) +
  geom_point(aes(colour = color)) +
  geom_text(aes(label = ifelse(top != 0, as.character(geneID), '')),
            hjust = 0, vjust = 0, check_overlap = TRUE) +
  xlab("log2(Fold Change)") + ylab("-log10(raw p-value)") +
  ggtitle("Volcano Plot \n (from simple model)") +
  scale_colour_manual(values = pal2[1:3], guide = FALSE)
print(p3)
```

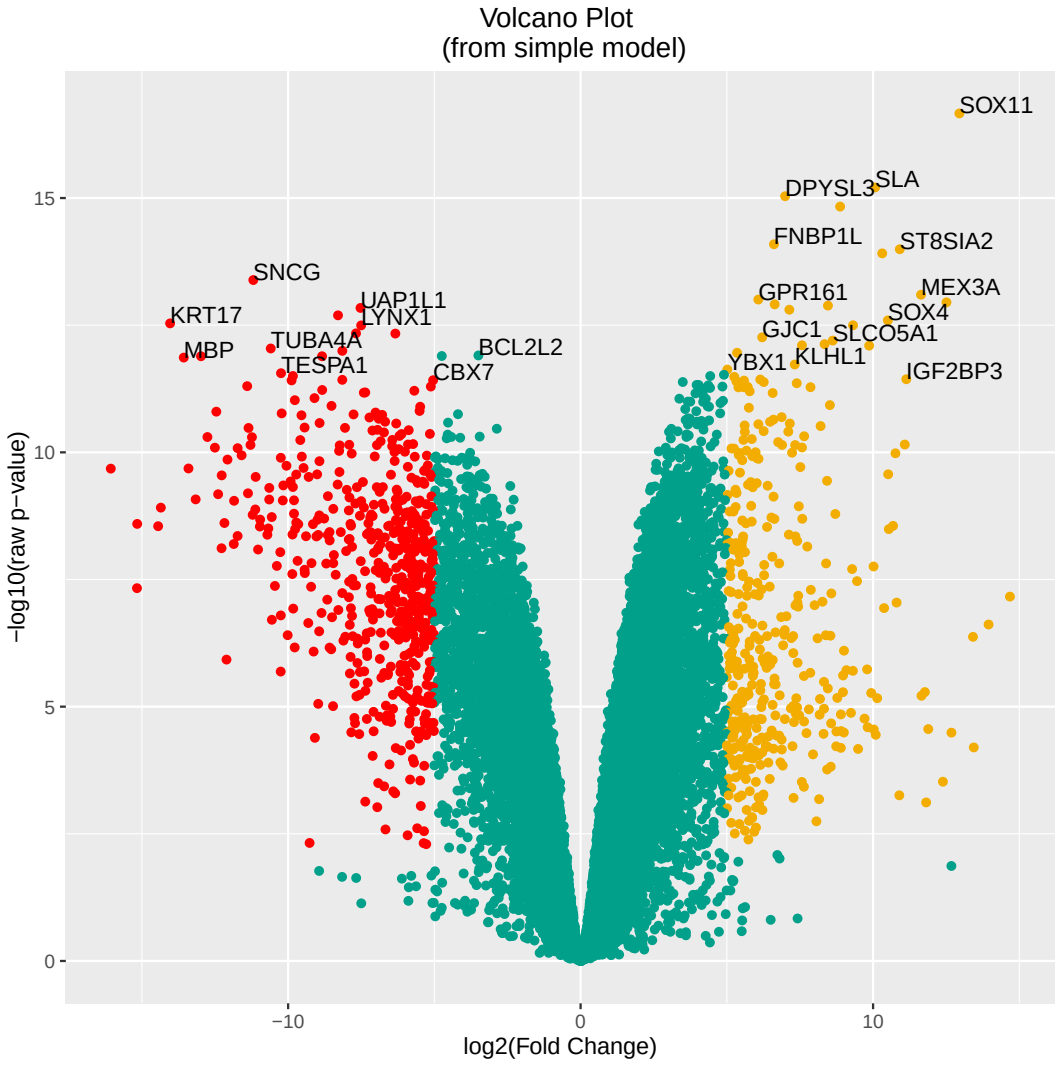

Figure 2: Volcano plot of **Limma** results using **ggplot2**.

5 Reproducibility notice

In the spirit of computationally reproducible research, the material used producing all of the analyses reported on in this project is publicly available on GitHub at `https://github.com/nhejazi/neurodevstat`. Minimally sufficient documentation is provided so that all reported results can be reproduced with relative ease. With any concerns, contact the author at nh@nimahejazi.org. *A table of the core software packages and the versions used for this analysis is given below:*

Software	***Version***
R (language)	3.3.2
Python (language)	3.5.2
Kallisto (pseudoalignment tool)	0.42.5
EnsDb.Hsapiens.v79 (R package)	1.1.0
tximport (R package)	1.2.0
limma (R package)	3.30.0
ggplot2 (R package)	2.1.0
dplyr (R package)	0.5.0