An Introduction to RNA-sequencing

Differential Expression Analysis

W. Evan Johnson, Ph.D.
Professor, Division of Infectious Disease
Director, Center for Data Science
Co-Direcor, Center for Biomedical Informatics and Health AI
Rutgers University – New Jersey Medical School

2025-07-30

Installing R Packages:

Install the following tools: `Rsubread`, `Rsamtools`, and `SummarizedExperiment`. We will also need help from the `tidyverse`.

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install(c("Rsubread","Rsamtools",
              "tidyverse","SummarizedExperiment"))
```

Load Packages for Today

We will be using the following packages for today's lecture: We will be using the following packages for our RNA-seq lecture:

```
library(tidyverse) ## tools for data wranging
library(Rsubread) ## alignment and feature counts
library(Rsamtools) ## managing .sam and .bam files
library(SummarizedExperiment) ## managing counts data
library(edgeR) ## differential expression
library(DESeq2) ## differential expression
library(ComplexHeatmap) ## Heatmap visualization
library(TBSignatureProfiler) ## TB signature analysis
library(umap) ## dimenstion reduction and plotting data
```

Using Rsubread to do Alignment

The following userguide will be helpful for you:

http://bioinf.wehi.edu.au/subread-package/SubreadUsersGuide.pdf

Indexing your genome

Note that you will rarely do this for human alignment. You will usually download an existing index given to you by others who have already done this work. You will do this often if you are aligning microbial reads, e.g. MTB or some other organism for which others have not already made your index for you.

```
buildindex(basename="../lecture_1/example_data/genome/ucsc.hg19.chr1_120-
 reference="../lecture_1/example_data/genome/ucsc.hg19.chr1_120-150M.fasta
```

Took me ~0.2 minutes!

Aligning your reads:

Note that this outputs results in a .bam file and not a .sam file

```
align(index="rna_seq/genome/ucsc.hg19.chr1_120-150M",
      readfile1="rna_seq/reads/R01_10_short500K.fq.gz",
      output_file="rna_seq/alignments/R01_10_short.bam",
      nthreads=4)
```

My old laptop was an Apple M2, with 8 cores (used 4 cores), 24GB RAM:

- Took 15.7 minutes to align ~60M reads to the 30M bases
- Took 0.7 minutes to align ~6.5M reads to the 30M bases
- Took 0.3 minutes to align ~500K reads to the 30M bases

Algning the reads using STAR

This afternoon we will use the STAR aligner on Amarel to align a set of RNA-seq fastq files!

Aligned Sequencing Data Formats (SAM and BAM)

Note that Rsubread outputs a .bam file (bam = binary alignment map) and not a .sam file (sam = sequence alignment map). Here is some information about a .sam file: https://en.wikipedia.org/wiki/SAM_(file_format)

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,255}`	Query template NAME
2	FLAG	Int	$[0,2^{16}-1]$	bitwise FLAG
3	RNAME	String	`\*|[!-()+-<>-~][!-~]*`	Reference sequence NAME
4	POS	Int	$[0,2^{29}-1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0,2^{8}-1]$	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[!-()+-<>-~][!-~]*`	Ref. name of the mate/next segment
8	PNEXT	Int	$[0,2^{29}-1]$	Position of the mate/next segment
9	TLEN	Int	$[-2^{29}+1,2^{29}-1]$	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	segment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+33

Aligned Sequencing Data Formats (SAM and BAM)

To convert .sam to .bam or vice versa, a package called Rsamtools. Using Rsamtools, you can convert bam to sam as follows:

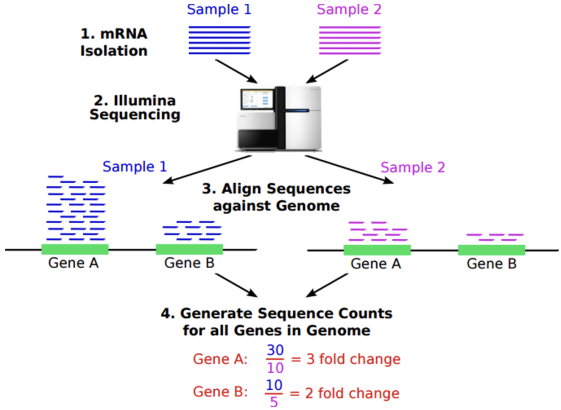
```
asSam("rna_seq/alignments/R01_10_short.bam",
      overwrite=T)

# To convert to bam:
#asBam("rna_seq/alignments/R01_10_short.bam")
```

Feature counts

Now we can count reads hitting genes. Approaches/software:

- HT-Seq
- STAR
- Cufflinks
- RPKM FPKM or CPM
- RSEM
- edgeR
- findOverlaps (GenomicRanges)
- featureCounts (Rsubread)

Feature counts



Feature counts

```
fCountsList = featureCounts(
  "rna_seq/alignments/R01_10_short.bam",
  annot.ext="rna_seq_files/genome/genes.chr1_120-150M.gtf",
  isGTFAnnotationFile=TRUE)

featureCounts = cbind(fCountsList$annotation[,1],
                      fCountsList$counts)

write.table(featureCounts,
    "rna_seq/alignments/R01_10_short.features.txt",
    sep="\t", col.names=FALSE, row.names=FALSE, quote=FALSE)
```

Data Structures

A data structure is a particular way of organizing data in a computer so that it can be used effectively. The idea is to reduce the space and time complexities of different tasks.

Data Structures

Data structures in R programming are tools for holding multiple values, variables, and sometimes functions

Please think very carefully about the way you manage and store your data! This can make your life much easier and make your code and data cleaner and more portable!

Data Frames

A large proportion of data analysis challenges start with data stored in a data frame. For example, we stored the data for our motivating example in a data frame. You can access this dataset by loading `TBNanostring.rds` object in R:

```
TBnanostring <- readRDS("example_data/TBnanostring.rds")
```

Data Frames

In RStudio we can view the data with the `View` function:

```
View(TBnanostring)
```

Data Frames

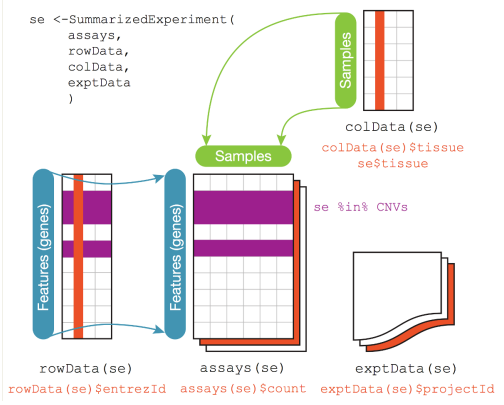
Or in RMarkdown you can use the `datatable` function from the `DT` package:

```
datatable(TBnanostring)
```

You will notice that the TB status is found in the first column of the data frame, followed by the genes in the subsequent columns. The rows represent each individual patient.

Advanced Data Structures

There are advanced R data structures that can facilitate object orientated programming. One useful example is the **SummarizedExperiment** object.



Make a SummarizedExperiment

Using an example dataset from: Verma, et al., 2018

```
## read in data
counts <- read.table(
  "example_data/features_combined.txt",
  sep="\t", header=T, row.names=1)
meta_data <- read.table(
  "example_data/meta_data.txt",
  sep="\t", header=T, row.names=1)
group <- meta_data$Disease
```

Make a SummarizedExperiment

```
## Make SummarizedExperiment
se_hivtb <- SummarizedExperiment(assays=list(counts=counts),
                      colData = meta_data)

## Make log counts, counts per million (cpm), logcpm
se_hivtb <- mkAssay(se_hivtb, log = TRUE,
                      counts_to_CPM = TRUE)
assays(se_hivtb)
```

```
## List of length 4
## names(4): counts log_counts counts_cpm log_counts_cpm
```

EdgeR Example

Implements statistical methods for DE analysis based on the negative binomial model:

```
#Gene Filtering
counts<-counts[which(rowSums(counts)>100),]
#Computes library size
dge <- DGEList(counts=counts, group=group)
#TMM normalization
dge <- calcNormFactors(dge)
# Design matrix
design<-model.matrix(~Disease, data=meta_data)
#Estimates common, trended and tagwise dispersion
dge<-estimateDisp(counts,design)
```

EdgeR Example

In negative binomial models, each gene is given a dispersion parameter. Dispersions control the variances of the gene counts and underestimation will lead to false discovery and overestimation may lead to a lower rate of true discovery.

EdgeR Example

```
# Neg Bin GLM with the dispersion estimates
fit<-glmFit(counts,design,
            dispersion=dge$tagwise.dispersion)
# Performs likelihood ratio test
# Compares full versus reduced model
lrt<-glmLRT(fit, coef=2)
```

EdgeR Example

```r
# Prints the top results
topTags(lrt)
```

```
## Coefficient:  Diseasetb_hiv
##             logFC   logCPM       LR       PValue          FDR
## IL1R2    4.334138 8.208316 93.61044 3.841569e-22 6.150736e-18
## AP3B2    5.751207 2.952926 64.01082 1.237377e-15 6.755923e-12
## FCGR1C   2.818442 4.536519 63.96598 1.265865e-15 6.755923e-12
## VNN1     3.150042 8.072140 60.78606 6.362689e-15 2.546825e-11
## CYP1B1   3.135490 6.873383 59.54714 1.193995e-14 3.823412e-11
## IL18R1   2.726093 6.487874 58.88977 1.667570e-14 4.449912e-11
## SOCS3    2.665152 6.476286 55.84118 7.856849e-14 1.797086e-10
## SLC29A1 -4.135726 3.970313 54.67430 1.422541e-13 2.699678e-10
## CACNG8  -4.134739 3.190544 54.35946 1.669722e-13 2.699678e-10
## ZAK      2.601721 6.127149 54.34023 1.686139e-13 2.699678e-10
```

EdgeR Example

```
## Use the a quasi-likelihood F-test
fit<-glmQLFit(counts, design,
        dispersion=dge$tagwise.dispersion)
## use for small datasets
qlf<-glmQLFTest(fit, coef=2)
```

EdgeR Example

```r
# Prints the top results
topTags(qlf)
```

```
## Coefficient:  Diseasetb_hiv
##             logFC   logCPM        F       PValue          FDR
## IL1R2    4.334144 8.208316 96.16041 1.411235e-22 2.259528e-18
## FCGR1C   2.818462 4.536519 65.27824 7.434147e-16 4.644916e-12
## AP3B2    5.751790 2.952926 64.96483 8.703233e-16 4.644916e-12
## VNN1     3.150045 8.072140 62.45782 3.077241e-15 1.231742e-11
## CYP1B1   3.135482 6.873383 61.22274 5.734734e-15 1.836376e-11
## IL18R1   2.726091 6.487874 60.59190 7.882229e-15 2.103373e-11
## SOCS3    2.665156 6.476286 57.45227 3.842609e-14 8.789144e-11
## ZAK      2.601720 6.127149 55.89059 8.455732e-14 1.621038e-10
## FKBP5    2.426321 7.528467 55.66226 9.489672e-14 1.621038e-10
## SLC29A1 -4.135688 3.970313 55.53406 1.012453e-13 1.621038e-10
```

EdgeR Example

```
#For visualization, heatmaps/PCA
Logcpm<-cpm(counts,log=TRUE)
```

DESeq2 Example

```
#colData is a data frame of demographic/phenotypic data
dds <- DESeqDataSetFromMatrix(countData = counts,
                              colData=meta_data,
                              design=~Disease)
#Gene Filtering
dds<-dds[rowSums(counts(dds))>100,]
```

DESeq2 Example

```
#Performs estimation of size factors,
#dispersion, and negative binomial GLM fitting
dds<-DESeq(dds)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## -- replacing outliers and refitting for 138 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)
```

DESeq2 Example

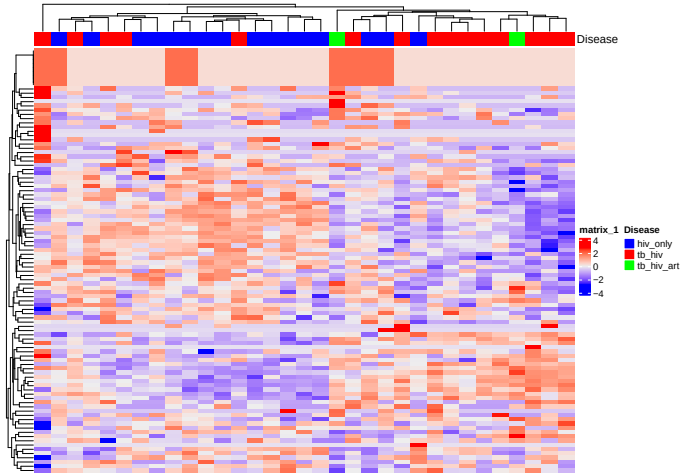
```
res <- results(dds)[order(results(dds)[,6]),]
res[1:25,]
```

```
## log2 fold change (MLE): Disease tb hiv art vs hiv only
## Wald test p-value: Disease tb hiv art vs hiv only
## DataFrame with 25 rows and 6 columns
##           baseMean log2FoldChange     lfcSE      stat      pvalue        padj
##          <numeric>      <numeric> <numeric> <numeric>   <numeric>   <numeric>
## ZEB2     1702.8164        1.75543  0.217589   8.06764 7.16683e-16 1.14748e-11
## DCUN1D3    48.7029        2.52499  0.362090   6.97337 3.09430e-12 2.47714e-08
## TIPARP    724.9429        2.22978  0.333852   6.67894 2.40672e-11 1.28447e-07
## ITPKC     196.8723        3.04144  0.465478   6.53401 6.40298e-11 2.56295e-07
## LAIR1    1528.6852        3.01392  0.471809   6.38800 1.68065e-10 4.25413e-07
## ...            ...            ...       ...       ...         ...         ...
## ATP6V1C1  1006.405        2.50826  0.420512   5.96477 2.44975e-09 1.84789e-06
## TAB3       546.645        1.52153  0.254844   5.97044 2.36615e-09 1.84789e-06
## IL18R1    1777.342        4.19796  0.706351   5.94317 2.79570e-09 1.94617e-06
## NR3C1     2177.200        1.46964  0.247644   5.93451 2.94720e-09 1.96615e-06
## IRAK3     4822.486        4.00584  0.678075   5.90765 3.47012e-09 2.13939e-06
```

Heatmap of DEGs

```
# Make a Heatmap of DEGs
mat = as.matrix(assay(se_hivtb,"counts")
                )[order(results(dds)[,6])[1:100],]
                # Using first 100 genes to simplify
mat = t(scale(t(mat))) ## scale gene expression
df=data.frame(Disease=colData(se_hivtb)$Disease)
ha = HeatmapAnnotation(df = df,
                       col = list(Disease=c(
                         "tb_hiv"="Red",
                         "hiv_only"="Blue",
                         "tb_hiv_art"="Green")))
Heatmap(mat,show_row_names=F, show_column_names = F,
        top_annotation = ha)
```

Heatmap of DEGs



Limma Example

- Most similar to microarray data flow
- Reads counts are converted to log2 counts per million (logCPM) and the mean-variance relationship is modeled with precision weights (voom transform)

Limma Example

```r
#From edgeR, Computes library size
dge <- DGEList(counts=counts, group=group)
#Gene Filtering
counts<-counts[which(rowSums(cpm(counts))>1),]
dge <- calcNormFactors(dge) #TMM normalization
```

Limma Example

```
design<-model.matrix(~group)
#voom transform to calculate weights to
#eliminate mean-variance relationship
v<-voom(dge, design)
#use usual limma pipelines
fit<-lmFit(v,design)
fit<-eBayes(fit)
```

Limma Example

```
topTable(fit, coef=ncol(design))
```

```
##                logFC    AveExpr         t      P.Value    adj.P.Val         B
## DCUN1D3     2.377176  1.1217001 7.735826 4.125950e-09 3.709568e-05 10.608653
## ZEB2        1.648629  6.4212073 7.535723 7.411363e-09 3.709568e-05 10.235835
## FAM151B     3.664559  1.2897952 7.433182 1.002186e-08 3.709568e-05  9.914427
## C7orf61     2.563702  2.7351541 7.384072 1.158443e-08 3.709568e-05  9.837465
## LINC01093   5.450634  0.8713944 7.395686 1.119395e-08 3.709568e-05  9.782461
## CYP19A1     6.857346 -0.9436098 7.049550 3.127199e-08 6.958151e-05  8.596031
## PGF         5.100817 -3.0391377 7.014062 3.476685e-08 6.958151e-05  7.894619
## IGF2BP3     3.299973  3.6443142 6.650327 1.035781e-07 1.658390e-04  7.749589
## COL4A2-AS1 5.437991 -3.3138293 7.102621 2.669553e-08 6.958151e-05  7.700178
## TIPARP      2.088081  5.0819834 6.585151 1.260852e-07 1.835228e-04  7.561353
```

Session info

```
sessionInfo()
```

```
## R version 4.5.1 (2025-06-13)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sequoia 15.5
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;  LAPACK version 3.12.1
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/New_York
## tzcode source: internal
##
## attached base packages:
## [1] grid      stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] umap_0.2.10.0               TBSignatureProfiler_1.20.0
##  [3] ComplexHeatmap_2.24.1       DESeq2_1.48.1
##  [5] edgeR_4.6.3                 limma_3.64.1
##  [7] SummarizedExperiment_1.38.1 Biobase_2.68.0
##  [9] MatrixGenerics_1.20.0       matrixStats_1.5.0
## [11] Rcpptools_2.24.0            Biostrings_2.76.0
```