R Programming

RNAseq
with EdgeR and Limma

2024

RNAseq tutorial

- Based on
  - http://combine-australia.github.io/RNAseq-R/
- Overview
  - Read in table of counts
  - Filter low-expression genes
  - Quality control
  - Normalisation
  - Differential expression analysis
  - Testing
  - Visualisation

Packages needed

- limma
- edgeR
- gplots
- Org.Mm.eg.db
- RColorBrewer
- Glimma

```
install.packages("BiocManager")
BiocManager::install("packageName")
```

Preparation

- The typical experiment will start from NGS data obtained with Illumina sequencing
- Reads will then be aligned to the reference genome
- Then the number of reads mapped will be counted
  - Resulting in a table of counts
- The table of counts is analyzed statistically

- You can do all from within R (but we won't)

Load packages

```
install.packages("BiocManager")

BiocManager::install(c("edgeR",
            "limma", "Glimma", "gplots",
            "org.Mm.eg.db", "RColorBrewer"),
            force=T)   # avoid this if possible

library(edgeR)
library(limma)
library(Glimma)
library(gplots)
library(org.Mm.eg.db)
library(RColorBrewer)
```

Mouse mammary gland dataset

- Available from the Gene Expression Omnibus Database (GEO) as GSE60450

  http://www.ncbi.nlm.nih.gov/pubmed/25730472

  and in

  http://mantra.cnb.csic.es/courses/2023-CNB-R/examples/edgeR

- Basically, we have two replicates each of
  - Basal stem-cell enriched cells (B)
  - Committed luminal cells (L)
- In mice that are
  - Virgin
  - Pregnant
  - Lactating

Prepare your project

- Create two folders
  - 'R' (where you will store your script)
  - 'data' (save the data files here):
    - **`data/GSE60450_Lactation-GenewiseCounts.txt`**
    - **`data/SampleInfo.txt`**
    - **`data/SampleInfo_Corrected.txt`**

Aligning sequences

- Make a list with the reads in directory 'fastq'

```
R1.fastq.files <- list.files(path='fastq', pattern='R1',
    full.names=TRUE)
R2.fastq.files <- list.files(path='fastq, pattern='R2',
    full.names=TRUE)
```

- Build an index for the reference genome in *aln.dir*

```
setwd(aln.dir)   # where aln.dir has been defined before
buildindex(basename='reference',reference='reference.fna')
```

Align

```
align(index='reference', readfile1=R1.fastq.files,
     readfile2=R2.fastq.files)
```

Compute feature counts

```
bam.files <- list.files(path=aln.dir, pattern='.BAM$',
                        full.names = TRUE)
setwd('..')
fc <- featureCounts(files=bam.files,
            annot.ext='reference.gtf',
            isGTFAnnotationFile=T,
            isPairedEnd=T,
            requireBothEndsMapped=T,
            primaryOnly=T,
            ignoreDup=T,
            useMetaFeatures=T)
```

- <u>This has already been done for you</u>

Read the data

```
# Read the data into R
seqdata <- read.delim(
        "data/GSE60450_Lactation-GenewiseCounts.txt",
        stringsAsFactors = FALSE)
# Read the sample information into R
sampleinfo <- read.delim("data/SampleInfo.txt")
head(seqdata)
dim(seqdata)
sampleinfo
```

- We got one gene per row, with its ID, length and numbers of reads in each sample.

Format the data

- We need a matrix containing only the counts
  - We'll save the gene names (1st column) as rownames

```
# Remove first two columns from seqdata
countdata <- seqdata[,-(1:2)]
# Look at the output
head(countdata)
# Store EntrezGeneID as rownames
rownames(countdata) <- seqdata[,1]
head(countdata)
colnames(countdata)
```

Take it easy

- There are column names that are too long
  - Use ‘substr()’ to simplify them

```
# using substr, you extract the characters starting at
# position 1 and stopping at position 7 of each name
colnames(countdata) <-
        substr(colnames(countdata),start=1,stop=7)
head(countdata)
# check that the order is OK
table(colnames(countdata)==sampleinfo$SampleName)
```

Convert to CPM

- We'll use filtering on minimum 0.5 counts per million (CPMs) present in at least two samples

```
# Obtain CPMs
myCPM <- cpm(countdata)
# Have a look at the output
head(myCPM)
```

- Note that, by converting, we normalize for the different sequencing depths of each sample
  - we now work with proportions which are independent of the sample size (number of reads in each experiment)

Filter by CPM

```
# Which values in myCPM are greater than 0.5?
thresh <- myCPM > 0.5
# This produces a logical matrix with TRUEs and FALSEs
head(thresh)
# Summary of how many TRUEs there are in each row
# There are 11433 genes that have TRUEs in all 12 samples.
table(rowSums(thresh))
# we profit here of TRUE==1 and FALSE==0
```

- A CPM of 0.5 corresponds to a count of 10-15 for the library sizes in this dataset: we consider smaller counts as unusable.

Ensure expressed genes are kept

- We have two replicas, we'll keep genes even it they are only expressed in one group

```
# we would like to keep genes that have at least 2 TRUES in
# each row of thresh
keep <- rowSums(thresh) >= 2
summary(keep)
# Subset the rows of countdata to keep the more highly
# expressed genes
counts.keep <- countdata[keep,]
dim(counts.keep)
# as a general rule a good CPM corresponds to a count of 10
# it is better to filter with CPMs than with counts
```

Plots!

```
# Let us see whether our threshold of 0.5 does indeed
# correspond to a count of about 10-15
# We will look at the first sample
plot(myCPM[,1],countdata[,1])

# Let us limit the x and y-axis so we can see
# what is happening to the smaller counts
plot(myCPM[,1],countdata[,1],ylim=c(0,50),xlim=c(0,3))

# Add a vertical line at 0.5 CPM
abline(v=0.5)
```

Convert to DGEList

- A DGEList is used by DGE (differential gene expression) to store count data

```
y <- DGEList(counts.keep)

# have a look at y
y
# See what slots are stored in y
names(y)
# Library size information is stored in the 'samples'
# slot
y$samples
```

Some Quality Control

- Library Sizes and distribution plots

```
y$samples$lib.size
# The names argument tells the barplot to use
# the sample names on the x-axis
# The “las” argument rotates the axis names
barplot(y$samples$lib.size,names=colnames(y),las=2)
# Add a title to the plot
title("Barplot of library sizes")
```

Count data

- We do not expect counts to be normally distributed, so we'll use logarithms

```
# Get log2 counts per million
logcounts <- cpm(y, log=TRUE)
# Check distributions of samples using boxplots
boxplot(logcounts, xlab="", ylab="Log2 counts per million",las=2)
# Let's add a blue horizontal line that corresponds to
# the median logCPM
abline(h=median(logcounts),col="blue")
title("Boxplots of logCPMs (unnormalised)")
# distributions, although not identical are not overly
# different.
```

MultiDimensionalScaling plots

- MDS plots are most important: they are a visualization of PCA and identify the main sources of variation.

```
plotMDS(y)
```

- It would be better if we could color the samples...

MDS plot (II)

```
# We specify the option to let us plot two plots
# side-by-side
par(mfrow=c(1,2))
# Let's set up color schemes for CellType
# How many cell types and in what order are they stored?
levels(sampleinfo$CellType)
# err...
sampleinfo$CellType <- factor(sampleinfo$CellType)
levels(sampleinfo$CellType)
## Let's choose purple for basal and orange for luminal
col.cell <- c("purple","orange")[sampleinfo$CellType]
data.frame(sampleinfo$CellType,col.cell)
# Redo the MDS with cell type colouring
plotMDS(y,col=col.cell)
```

MDSplot (III)

```
# Let's add a legend to the plot so we know which colors correspond
# to which cell type
legend("topleft",fill=c("purple","orange"),
       legend=levels(sampleinfo$CellType))
# Add a title
title("Cell type")
# Similarly for status
levels(sampleinfo$Status)
sampleinfo$Status <- factor(sampleinfo$Status)
levels(sampleinfo$Status)
col.status <- c("blue","red","dark green")[sampleinfo$Status]
col.status
plotMDS(y,col=col.status)
legend("topleft",fill=c("blue","red","dark green"),
       legend=levels(sampleinfo$Status),cex=0.8)
title("Status")
```

What???

- **We all make mistakes**. No reason to shy.
- Did you see any group that looks misplaced?
- We have another ‘sampleinfo’ file for you to try, this time a corrected one

```
# There is a sample info corrected file in your data directory
# Old sampleinfo
sampleinfo
# We are going to write over the ‘sampleinfo’ object with
# the corrected sample info
sampleinfo <- read.delim("data/SampleInfo_Corrected.txt")
sampleinfo
sampleinfo$CellType <- factor(sampleinfo$CellType)
sampleinfo$Status <- factor(sampleinfo$Status)
```

Play it again, Sam

```
# Redo the MDSplot with corrected information
par(mfrow=c(1,2))
col.cell <- c("purple","orange")[sampleinfo$CellType]
col.status <- c("blue","red","dark green")[sampleinfo$Status]
plotMDS(y,col=col.cell)
legend("topleft",fill=c("purple","orange"),legend=levels(
        sampleinfo$CellType))
title("Cell type")
plotMDS(y,col=col.status)
legend("topleft",fill=c("blue","red","dark
       green"),legend=levels(sampleinfo$Status),cex=0.8)
title("Status")
```

Into the 3rd dimension

```
# Dimension 3 appears to separate pregnant samples
# from the rest. Dim4?
plotMDS(y,dim=c(3,4), col=col.status, pch=char.celltype,
        cex=2)
legend("topright", legend=levels(sampleinfo$Status),
       col=cols, pch=16)
legend("bottomright", legend=levels(sampleinfo$CellType),
       pch=c(1,4))
```

Once more, using Glimma

```
labels <- paste(sampleinfo$SampleName,
     sampleinfo$CellType, sampleinfo$Status)
group <- paste(sampleinfo$CellType,
      sampleinfo$Status,sep=".")
group <- factor(group)
glMDSPlot(y, labels=labels, groups=group, folder="mds")
# The result will be an HTML page
```

Select top differentially expressed genes ...

```
# We estimate the variance for each row in the
# logcounts matrix
var_genes <- apply(logcounts, 1, var)
head(var_genes)
# Get the gene names for the top 500 most variable genes
sorted_var <- names(sort(var_genes, decreasing=TRUE)
                    )
select_var <- sorted_var[1:500]
# Subset logcounts matrix
highly_variable_lcpm <- logcounts[select_var,]
dim(highly_variable_lcpm)
```

...with heatmaps

```
## Get some nicer colours
mypalette <- brewer.pal(11,"RdYlBu")
morecols <- colorRampPalette(mypalette)
# Set up colour vector for celltype variable
col.cell <- c("purple","orange")[sampleinfo$CellType]

# Plot the heatmap
heatmap.2(highly_variable_lcpm,col=rev(morecols(50)),
    trace="none",
    main="Top 500 most variable genes across samples",
    ColSideColors=col.cell,scale="row")
```

Normalisation for composition bias

```
# Apply normalisation to DGEList object
#  NOTE that here we overwrite it, you may not want to
y <- calcNormFactors(y)
y$samples

par(mfrow=c(1,2))
plotMD(logcounts,column = 7)
abline(h=0,col="grey")
plotMD(logcounts,column = 11)
abline(h=0,col="grey")
```

Plot using y

```
par(mfrow=c(1,2))
plotMD(y,column = 7)
abline(h=0,col="grey")
plotMD(y,column = 11)
abline(h=0,col="grey")
```

Getting into detail

- Next, we would like to know what are the specific differences across each two specific groups.
  - You can visualize the process as doing first an ensemble group comparison (ANOVA, Kruskal-Wallis, etc.) followed by multiple comparison *post hoc* tests at a large scale.

Differential expression with limma-voom

- First we'll create a design matrix tailored to our analysis (see limma --*linear models for microarrays*)

```
# Look at group variable again
group
# Specify a design matrix without an
# intercept term
design <- model.matrix(~ 0 + group)
design
# Make the column names of the design matrix
# look nicer
colnames(design) <- levels(group)
design
```

Voom transform the data

```
par(mfrow=c(1,1))
v <- voom(y,design,plot = TRUE)
v
# What is contained in this
# object?
names(v)
```

More plots

```
par(mfrow=c(1,2))
boxplot(logcounts, xlab="",
    ylab="Log2 counts per million",
    las=2, main="Unnormalised logCPM")
## Let's add a blue horizontal line that
## corresponds to the median logCPM
abline(h=median(logcounts),col="blue")

boxplot(v$E, xlab="", ylab="Log2 counts per million",
    las=2, main="Voom transformed logCPM")
## Let's add a blue horizontal line that corresponds to
## the median logCPM
abline(h=median(v$E),col="blue")
```

If you got here...

- You can now continue alone.
- Simply follow any of the fine tutorials available on the web
- Such as this one
  - http://combine-australia.github.io/RNAseq-R/