R programming

José R. Valverde
jrvalverde@cnb.csic.es
CNB/CSIC

RNAseq

There are three kinds of lies: lies, damned lies, and statistics.

Attributed by Mark Twain to Benjamin Disraeli

Preparation

- Before we start, we need to prepare our R installation for this tutorial
- We need to install the required extensions (packages)
  - There are standard and non-standard packages
  - Most of the packages are a part of BioConductor

Standard tools

- We are going to make use of package “dplyr”
- **dplyr** provides advanced data manipulation tools, among them, tools to store statistical data in a database and access this database directly.
  - Install SQLite
  - Add the corresponding package for using it in R
    - `install.packages("RSQLite")`

Getting the latest version of **dplyr**

- You can install it from CRAN (stock version) or from GitHub (latest, bleeding-edge version)
  - Ensure you have devtools installed and that it is at least version 1.6

```
if (packageVersion("devtools") < 1.6) {
  install.packages("devtools")
}
devtools::install_github("hadley/lazyeval")
devtools::install_github("hadley/dplyr")
```

Bioconductor

- Some communities are specially active (e.g. the Geology and Biology communities) and produce huge amounts of data and analyses
- BioConductor is the specialized repository for Molecular Biology analyses
  - There are many standard packages for given tasks
  - But BioConductor contains a massive amount of data and tools
  - BioConductor also provides tools to ease installation

Prepare to install BioConductor

- You will not install “all” of Bioconductor
  - It's HUGE
- Instead, you install a minimal “installer”

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

- This will add a new function to your session
  - **?BiocManager::install**
  - **BiocManager::install( )** allows you to install anything you need from BioConductor or from CRAN.

Install BioConductor packages

- We will start our analysis making use of DESeq2
- DESeq2 is a BioConductor package. It is not in CRAN
- You install it with
  - `BiocManager::install("DESeq2")`
- Use the same command for other packages as needed during the session

Getting the data

- We are going to make use of two files:
  - GSE37704_feaurecounts.csv
  - GSE37704_metadata.csv
- You can try to download them directly from the Net by providing their URL to “read.csv( )”
  - URLs in the Net tend to shift spuriously
  - Chances are you may have trouble getting published data from outdated URLs.
  - It is best to always access the data from a DOI reference.

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)
```

- This has already been done for you

Get the gene data

```
library(dplyr)
library(DESeq2)
countDataURL =
"https://ndownloader.figshare.com/files/356
0183"
countData = read.csv(countDataURL,
    row.names=1) %>%
    dplyr::select(-length) %>%
    as.matrix()
countData = countData[rowSums(countData)>1,
]
head(countData)
```

The data

- This data set contains expression data from six samples, obtained by either
  - mapping to GRCh38 with STAR and counting the reads with "featureCounts" under a union-intersection model
  - alignment-free quantification using Sailfish summarized with the GTF file for GRCh38.
- Only protein-coding genes are included.
- We first select features with a gene count > 1 (i.e. we remove those with count=0 or count=1).

%>%

- This is an operation provided by dplyr
- It means "take the result of whatever is to the left of %>% and use it as first argument of whatever is at the wirght of %>%
- E.g.
  - a <- c(1,2,3,2,1) %>% mean()
  - is the same as
  - a <- c(1,2,3,2,1) ; mean(a)
- Some people prefers to write less

Get metadata

```
colData <-
read.csv("https://ndownloader.figshar
e.com/files/3560180", row.names=1)
head(colData)
```

The metadata file is a CSV or TAB-delimited file where we annotate characteristics of each sample.

As you can see, it only contains information on the siRNA control and an HoxA knock-down, which are the samples we want to analyze.

Run the DESeq2 pipeline

```
dds = DESeqDataSetFromMatrix(
      countData=countData,
      colData=colData,
      design=~condition)
dds = DESeq(dds)
dds
```

The DESeq pipeline automates all the process of Differential Gene Expression analysis, simplifying all the work for you.

Here we say we want to analyze DE by column "condition" in the metadata (i.e. whether data is control or knock-out)

DESeq2

- It is a fully automated differential expression protocol.
  - Easier to use
  - Requires that you provide raw counts (not normalized data).
  - Check DESeq2 documentation to learn more.
  - Some times it may be worth to pre-filter genes with very low reads), 10 may be a safe (lower) limit.
    - Note that DESeq2 will apply its own, more stringent filtering anyway
  - DESeq2 will take care of adjusting by library size.
  - Requires a *design formula*, which is a ~ followed by the variables to be considered (separated by +)
    - Factors are considered alphabetically by default.
  - If you change the formula, you will need to repeat all the analyses.
  - It is also possible to collapse technical replicates (several runs of same library) to increase counts.
    - Do **not** collapse biological replicates!

HoxA vs. Control siRNA

```
res = results(dds, contrast=
    c("condition",
      "hoxa1_kd", "control_sirna"))
res = res[order(res$padj),]
summary(res)
```

Now, we simply compare both samples, using the names in the metadata file (which are more convenient and easier to remember for us).

As with any other test, we get a p value, this time one for each gene. We use this p value to order the results by statistical significance (the lower the p value, the more significant).

We may also use non-adjusted p-values ($pvalue) instead (see ? results).

Independent Hypothesis Testing

- We can also use package IHW for p-value adjustment.

```
BiocManager::install('IHW')
library(IHW)
resIHW <- results(dds, filterFun=ihw)
```

- Easy, isn't it?
- NOTE: **by default DESeq2 adjusts the p-values for an alpha of <u>0.1</u>!**
  - You can change that with argument alpha
  - e.g. res <- resulst(dds, alpha=0.05, ...)

Plotting $\log_2$ Fold Changes

- We use plotMA()
- Red points have an adjusted p-value less than the alpha level (0.1 by default). Points out of the specified window are plotted as triangles pointing up or down at the borders.
- For normalized counts

```
plotMA(res, ylim=c(-2,2))
```

- For $\log_2$FC (use condition as shown by `resultNames(dds)`)

```
resultNames(dds)
BiocManager::install('apeglm')
resLFC <- lfcShrink(dds,
    "condition_hoxa1_kd_vs_control_sirna",
    type="apeglm")
plotMA(resLFC, ylim=c(-2,2))
```

Shrinking the data

- This is how DESeq2 calls taking logarithms to reduce variability and dispersion
- There are various methods:
  - apeglm: adaptive t prior estimator (requires package apeglm)
  - ashr: adaptive shrinkage estimator (requires ashr)
  - normal: adpatibe Normal distribution as prior.
- We could have used the index of the condition in resultNames as well (2 in this case) as "coef=2"

Assignment

- Try plotMA on the resLFC data calculated using different types of estimators.
  - You can generate new variables, one for each estimator
  - Then use par(mfrow=(1,3))
  - Then call plotMA
    - we can add other, general plotting parameters:
    - use parameter main to specify the title of each plot
    - e.g.

```
plotMA(resLFC,
       xlim=c(1,1e5), ylim=c(-3,3),
       main="apglm")
```

Plot Counts

- Sometimes we'd like to see the counts for a single gene across groups. Here we only have two groups, but, nevertheless:

```
plotCounts(dds,
     gene=which.min(res$padj),
     intgroup="condition")
```

- Here we plot the one with minimal adjusted p value. We could have used the row name or a numeric index.

Load ENSEMBL information for H.sapiens

```
library("AnnotationDbi")
# install the human database
BiocManager::install("org.Hs.eg.db")
library("org.Hs.eg.db")

columns(org.Hs.eg.db)
```

We will use ENSEMBL annotation to access the KEGG data. We access it using "AnnotationDbi" and a specific database (org.Hs.eg.db for *Homo sapiens*).

For other organisms you will need to search if there is an ESEMBL database available. If there is none, you will need to build one yourself from a GFF3 file corresponding to the annotated genome for the organism (if there is one) using ensembldb::createEnsDbForSpecies().

Use ENSEMBL to map KEGG data

```
# gene symbols
res$symbol = mapIds(org.Hs.eg.db,
                    keys=row.names(res),
                    column="SYMBOL",
                    keytype="ENSEMBL",
                    multiVals="first")
```

We will now proceed to fetch relevant annotation and add it to our DESeq results as additional columns. Here we retrieve the ENSEMBL GeneIDs.

Note that sometimes one feature may match multiple IDs. In that case, multiVals specifies which one(s) we want.

Use ENSEMBL to map KEGG data

```
# get Entrez Ids
res$entrez = mapIds(org.Hs.eg.db,
                    keys=row.names(res),
                    column="ENTREZID",
                    keytype="ENSEMBL",
                    multiVals="first")
```

ENSEMBL uses GeneIDs, but KEGG uses EntrezIDs. So, we need to retrieve those as well, so we may later use them to find the relevant KEGG information for each gene.

Use ENSEMBL to map gene names

```
# get gene names
res$name =  mapIds(org.Hs.eg.db,
            keys=row.names(res),
            column="GENENAME",
            keytype="ENSEMBL",
            multiVals="first")
# check we added the columns
head(res, 10)
```

And finally we also get gene names (these may help us interpret results in long listings)

Getting actual KEGG data

- As we said, KEGG uses EntrezIDs. We got those from the ENSEMBL annotation.
- Now we are ready to use the EntrezIDs to retrieve pathway information from KEGG.
- For this we will make use of 'gage' (Generally Applicable Gene-set Enrichment for Pathway Analysis) and 'gageData' (the data needed to map KEGG pathways and GO terms for common organisms).
  - kegg.sets.hs is a list of pathway data: each element contains the EntrezIDs in that pathway.

Pathway detection using KEGG

```
BiocManager::install(c("pathview",
    "gage", "gageData"))
library(pathview)
library(gage)
library(gageData)
data(kegg.sets.hs)
data(sigmet.idx.hs)
kegg.sets.hs =
        kegg.sets.hs[sigmet.idx.hs]
head(kegg.sets.hs, 3)
```

Pathway analysis with “gage”

```
foldchanges = res$log2FoldChange
names(foldchanges) = res$entrez
head(foldchanges)
keggres = gage(foldchanges,
          gsets=kegg.sets.hs,
          same.dir=TRUE)
lapply(keggres, head)
```

same.dir=T means that we will get separate lists for up- and down-regulated genes ($greater and $less)

Select top 5 upregulated pathways

```
keggrespathways = data.frame(
      id=rownames(keggres$greater),
      keggres$greater) %>%
      tbl_df() %>%
      filter(row_number()<=5) %>%
      .$id %>%
      as.character()
keggrespathways
keggresids = substr(keggrespathways,
                    start=1, stop=8)
keggresids
```

We extract the IDs so we can use them later for plotting.

Exercise

- Go back to the last slide
- Try to reproduce it step by step instead of using %in%
  - Use variable names to store the intermediate results
  - print each of these variables before going to the next step
  - try to understand what each step does (use help when in doubt)

Select top 5 upregulated pathways

```
keggrespathways = data.frame(
      id=rownames(keggres$greater),
      keggres$greater)
krp_as_df <- tbl_df(keggrespathways)
krp_top_5 <- filter(krp_as_df, row_number()<=5)
krp_top_5_names <- krp_top_5$id
as.character(krp_top_5_names)

keggrespathways
keggresids = substr(keggrespathways,
                    start=1, stop=8)
keggresids
```

We extract the IDs so we can use them later for plotting.

Make the plots

```
plot_pathway = function(pid)
    pathview(gene.data=foldchanges,
             pathway.id=pid,
             species="hsa",
             new.signature=FALSE)
tmp = sapply(keggresids,
             plot_pathway)
```

Here we us a trick: we define a plotting function and then apply it to all the selected (5) pathways at once using sapply.

Time to have a look at the plots now. How cool is that?

Gene Ontology

```
data(go.sets.hs)
data(go.subs.hs)
gobpsets = go.sets.hs[go.subs.hs$BP]

gobpres = gage(foldchanges,
               gsets=gobpsets,
               same.dir=TRUE)

lapply(gobpres, head)
```

We can do the same with GO. We load the Hs GO terms and use gage to mine them as well.

What next?

- Statistics, statistics and more statistics...
  - We can use the $\log_2$FC ($\log_2$ of fold change) to compare expression patterns (lfcShrink)
  - We can then apply statistical tests to decide which changes are *statistically significant*.
  - we can cluster samples and genes, etc...
- Thinking, thinking, thinking...
  - All of this will give us a lot of information
  - Statistical significance does not imply biological significance
  - We need to study the results to draw meaningful biological conclusions.

There is more, much more...

- There are other packages and workflows to carry out an RNAseq analysis
- You may be interested in seeing our alternative presentation on RNAseq, where we introduce a different workflow and several statistical methods.
- And, in any case, there is much, much more to annotation, analysis, etc... that can be done with these and other packages.
- But this should be enough to get you started.

Thank you

Questions?