Day 27: Bioconductor

Larry Kalesinskas

Bioconductor: a package for bioinformatics

- An open-source package repository (in R) specifically targeted towards bioinformaticians.
- Contains a large, curated, (usually) well-documented set of packages to perform bioinformatics analyses.
- Packages for all sorts of uses: sequencing, expression and other microarrays, flow cytometry, mass spectrometry, image analysis, etc
- Bioconductor site: https://www.bioconductor.org/

`Bioconductor`: types of packages

- `Software`: algorithms, access to resources, visualizations. e.g: DeSeq2 for RNA-seq analysis.
- `Annotations`: Database-like packages that provide identifiers and other information. e.g: hg38 with Annotated Genes
- `Experiment Data`: Datasets that are used by packages for different analyses. e.g. Iris dataset.
- There are a total of 1823 packages in release 3.10!

`Bioconductor`: Notable Packages

- Most popularly downloaded packages: https://bioconductor.org/packages/stats/
- RNA-seq: edgeR, DESeq2
- Microarray: affy, Rsamtools, limma, GEOquery
- Visualizations: geneplotter, enrichplot, clusterProfiler

Bioconductor: Installing Bioconductor

- To use any Bioconductor packages, you first need to have a working, up-to-date implementation of Bioconductor, which can be installed using this:

```
install.packages("BiocManager")

BiocManager::install()
```

- `install.packages("BiocManager")`: Installs the Bioconductor packages from CRAN
- `BiocManager::install()`: installs and updates Bioconductor and dependencies

Bioconductor: Installing Bioconductor Packages

- To install any bioconductor package you can pass the package name to `BiocManager::install()`.

```
BiocManager::install("biomaRt")
```

- Once installed, a Bioconductor package acts like any other and can be be loaded using `library(package_name)`.

Bioconductor: Figuring out how packages work

- Many Bioconductor packages have helpful vignettes that demonstrate how to use the package and its utility.

```r
library(biomaRt)
browseVignettes("biomaRt")
```

- This is in addition to the typical R function documentation.

biomaRt: A library for getting data and identifers

- biomaRt is a package designed to query biomaRt-style databases (http://www.biomart.org/community.html).
- This package is your best friend for accessing general genomic data and converting between data from Ensembl, UniProt and HapMap.
- Example question: What is the gene name for a certain protein? What is the gene symbol assosciated with this identifier?

biomaRt: Ensembl Setup

- To use biomaRt and get data, we need to specify a mart (database) and a dataset (within database) to use.

```
library(biomaRt)
ensembl = useEnsembl(biomart="ensembl")
nrow(listDatasets(ensembl))
[1] 202
```

- Let's pick the human dataset.

```
ensembl = useEnsembl(biomart="ensembl",
                     dataset="hsapiens_gene_ensembl")
```

biomaRt: Example

- Let’s say I’m interested in finding genes assosciated with Insulin Binding (GO:0043559).

```
queryResults <- getBM(
      attributes = c('entrezgene_id', 'hgnc_symbol'),
      filters = 'go',
      values = 'GO:0043559',
      mart = ensembl)
queryResults
  entrezgene_id hgnc_symbol
1          3416         IDE
2          9854      C2CD2L
3          3480       IGF1R
4          5295      PIK3R1
5          3643        INSR
6          3329       HSPD1
```

- Attributes -> what data do you want to pull down? All attributes can be listed by using `listAttributes(ensembl)`. In this case, I want the Entrez Gene ID and Symbol.
- Filters -> What type of data am I inputting? All filters can be listed by using `listFilters(ensembl)`. In this case, I’m inputting a Gene Ontology term.
- Values -> What are the values I want to filter by? In this case, I want to grab anything with ‘GO:0043559’.

biomaRt: Example

- Now, let's say I want to get a sequence:

```
proteins <- getSequence(id=queryResults$entrezgene_id,
                        type="entrezgene_id",
                        seqType="peptide",
                        mart=ensembl)
head(proteins, 1)

1 MDPGWGQRDVGWAALLILFAASLLTVFAWLLQYARGLWLARARGDRGPGPALAGEPAGSLRELGVWRSL
  entrezgene_id
1          9854
```

- ID -> input data/identifiers
- Type -> data type of input - (ensembl, entrezgene, refseq, etc)
- seqType -> output data type - (peptide, gene_exon, etc)

`biomaRt`: Practice

- Let’s pull down the gene symbol assosciated with Ensembl gene id ENSG00000012048.
- Hint: Use `listFilters(ensembl)` to find the filter for Ensembl gene ids.

biomaRt: Solution

```
queryResults <- getBM(
      attributes = c('ensembl_gene_id', 'hgnc_symbol'),
      filters = 'ensembl_gene_id',
      values = 'ENSG00000012048',
      mart = ensembl)
queryResults
  ensembl_gene_id hgnc_symbol
1 ENSG00000012048       BRCA1
```

DEseq2: Package Installation

- Differential gene expression analysis based on the negative binomial distribution
- A commonly used package for testing for differential gene expression in RNA-seq data

```r
BiocManager::install("DESeq2")
library(DESeq2)
```

DESeq2: Import Data

- In this example, we will compare the gene expression between 4 ALS patients and 4 controls (modified from GSE52202).
- DESeq2 can take many forms of input (check out their vignette) - in this case, we are importing a counts matrix (sample x gene table) from a `.csv`.
- Generally, DESeq2 needs the raw RNA-seq pre-processed and filtered before you do differential analysis.

```
data_dir <- "https://web.stanford.edu/class/somgen223/data/"
countData <- read.csv(str_c(data_dir, "countData.csv"), row.names=1)
head(countData)
         ctr.1 ctr.2 ctr.3 ctr.4 als.1 als.2 als.3 als.4
A1BG         4     4     4     6     3     4     3     4
A1BG-AS1     2     0     2     2     2     1     2     1
A1CF         0     0     0     0     0     0     0     0
A2M         64     5    54    57     9     3    22    17
A2M-AS1      1     0     2     0     3     0     1     1
A2ML1        0     0     0     0     0     0     0     0
```

DESeq2: Process Data

- DESeq needs two things:
  1. Counts Matrix with Columns as Samples - we imported this!
  2. Condition Matrix with sample groups - we need to make this!

```
expData <- data.frame(row.names = colnames(countData),
      condition = factor(c(rep("ctl", 4), rep("exp", 4))))
head(expData)
      condition
ctr.1       ctl
ctr.2       ctl
ctr.3       ctl
ctr.4       ctl
als.1       exp
als.2       exp
```

DESeq2: Process Data

- In order to run DESeq2, we need to put the data into the right format that the package will expect.
- There are a few functions to do this, depending on your input data - check vignette!

```
deSeqData <- DESeqDataSetFromMatrix(countData = countData,
                                    colData = expData,
                                    design = ~ condition)
deSeqData
class: DESeqDataSet
dim: 21284 8
metadata(1): version
assays(1): counts
rownames(21284): A1BG A1BG-AS1 ... ZZEF1 ZZZ3
rowData names(0):
colnames(8): ctr.1 ctr.2 ... als.3 als.4
colData names(1): condition
```

DESeq2: Run DESeq2

- Now with the data in the right format, we can finally run DESeq!

```
deSeqData <- DESeq(deSeqData)
```

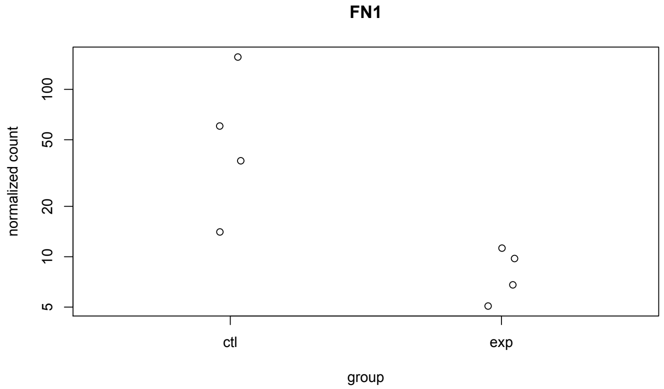
- However, the results actually aren't in `deSeqData`. To get results, you have to call... `results()`

```
deSeqFinalResults <- results(deSeqData)
head(deSeqFinalResults, 1)
log2 fold change (MLE): condition exp vs ctl
Wald test p-value: condition exp vs ctl
DataFrame with 1 row and 6 columns
             baseMean    log2FoldChange             lfcSE
            <numeric>         <numeric>         <numeric>          <num
A1BG 4.05477575863463 -0.47000842795273 0.672457421665092 -0.6989415430
                pvalue      padj
             <numeric> <numeric>
A1BG 0.484588563937088        NA
```

DESeq2: Gene Plotting

- What is the gene that separates the groups the most?

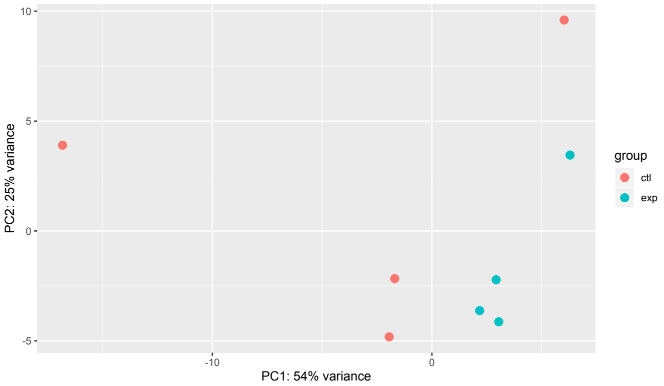
```
plotCounts(deSeqData,
           gene=which.min(deSeqFinalResults$padj),
           intgroup="condition")
```



DESeq2: PCA Plots

- Do my conditions cluster together?

```
DESeq2::plotPCA(rlogTransformation(deSeqData),
                intgroup="condition")
```



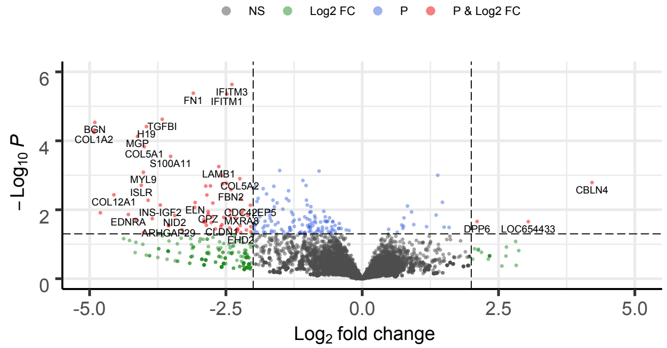
DESeq2: Volcano Plots

- What would an RNA-seq experiment be without volcano plots?
- Could make your own using ggplot… But there’s a package for it!

```r
BiocManager::install("EnhancedVolcano")
library(EnhancedVolcano)
```

Enhanced Volcano: Volcano Plots

```
volPlot <- EnhancedVolcano(deSeqFinalResults,
    lab = rownames(deSeqFinalResults),
    x = 'log2FoldChange',
    y = 'pvalue',
    xlim = c(-5, 5),
    ylim = c(0, 6))
volPlot
```



Enhanced Volcano: Extract Signficant Genes

```
significantGenes <- volPlot$data %>%
  filter(Sig == "FC_P") %>%
  dplyr::select(lab)
head(significantGenes)
       lab
1 ARHGAP29
2      BGN
3     BST2
4    CBLN4
5 CDC42EP5
6    CLDN1
```

Pathway Analysis: What do my significant genes do?

- We can use a pathway analysis library built into Bioconductor to get an idea of what biological processes the enriched genes are involved in.
- One such package is `ReactomePA`.

```
BiocManager::install("ReactomePA")
library(ReactomePA)
```

Pathway Analysis: What do my significant genes do?

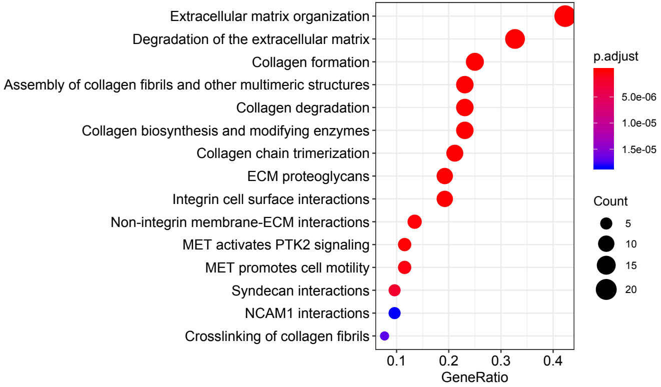
- First, need to convert gene symbols to Entrez IDs

```
genes <- getBM(
  attributes=c("hgnc_symbol","entrezgene_id"),
  filters = "hgnc_symbol",
  values = significantGenes$lab,
  mart = ensembl)
head(genes)
  hgnc_symbol entrezgene_id
1     ARHGAP29          9411
2          BGN           633
3         BST2           684
4        CBLN4        140689
5     CDC42EP5        148170
6        CLDN1          9076
```

Pathway Analysis: What do my significant genes do?

- Then we can run Pathway Analysis with the Entrez IDs extracted from my genes.

```
x <- enrichPathway(gene=genes$entrezgene_id,
                   pvalueCutoff=0.05)
dotplot(x, showCategory=15)
```



Summary

- Bioconductor has many powerful packages that have been coded specifically for bioinformatics analyses and are great for analyzing a wide variety of data.
- Many packages have vignettes, which (if well-written) step you through the package and functions built within
- Don’t reinvent the wheel - if it’s been developed already - use it!