Approximate time: 40 minutes

Learning Objectives

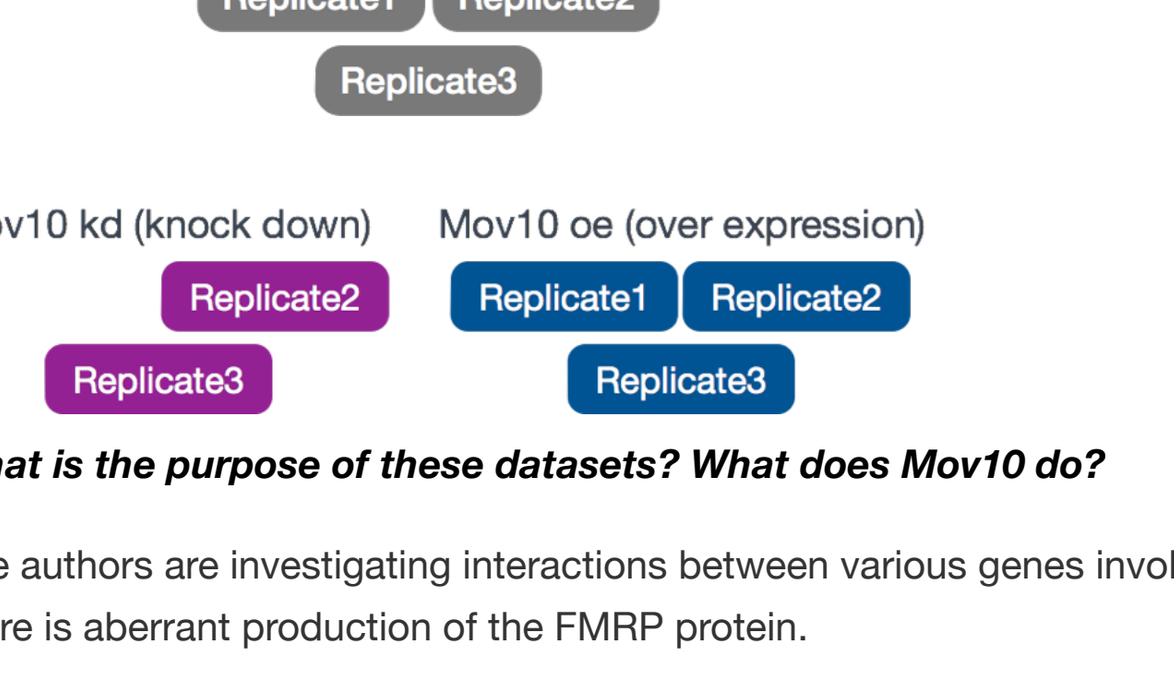
- Have a general idea of the experiment and its objectives
- Understand how and why we choose this dataset
- Getting setup in R (project setup, loading data, loading libraries)
- Becoming familiar with the `DESeqDataSet` object

Understanding the dataset

We will be using a real RNA-Seq dataset for today's class. It is part of a larger study described in Kenny PJ et al, Cell Rep 2014.

We are only using the RNA-Seq dataset which is publicly available in the SRA. The RNA-Seq was performed on HEK293F cells that were either transfected with a MOV10 transgene, or siRNA to knock down Mov10 expression, or non-specific (irrelevant) siRNA. This resulted in 3 conditions **Mov10 oe** (over expression), **Mov10 kd** (knock down) and **Irrelevant kd**, respectively. The number of replicates is as shown below.

Using these data, we will evaluate transcriptional patterns associated with perturbation of MOV10 expression. Please note that the irrelevant siRNA will be treated as our control condition.



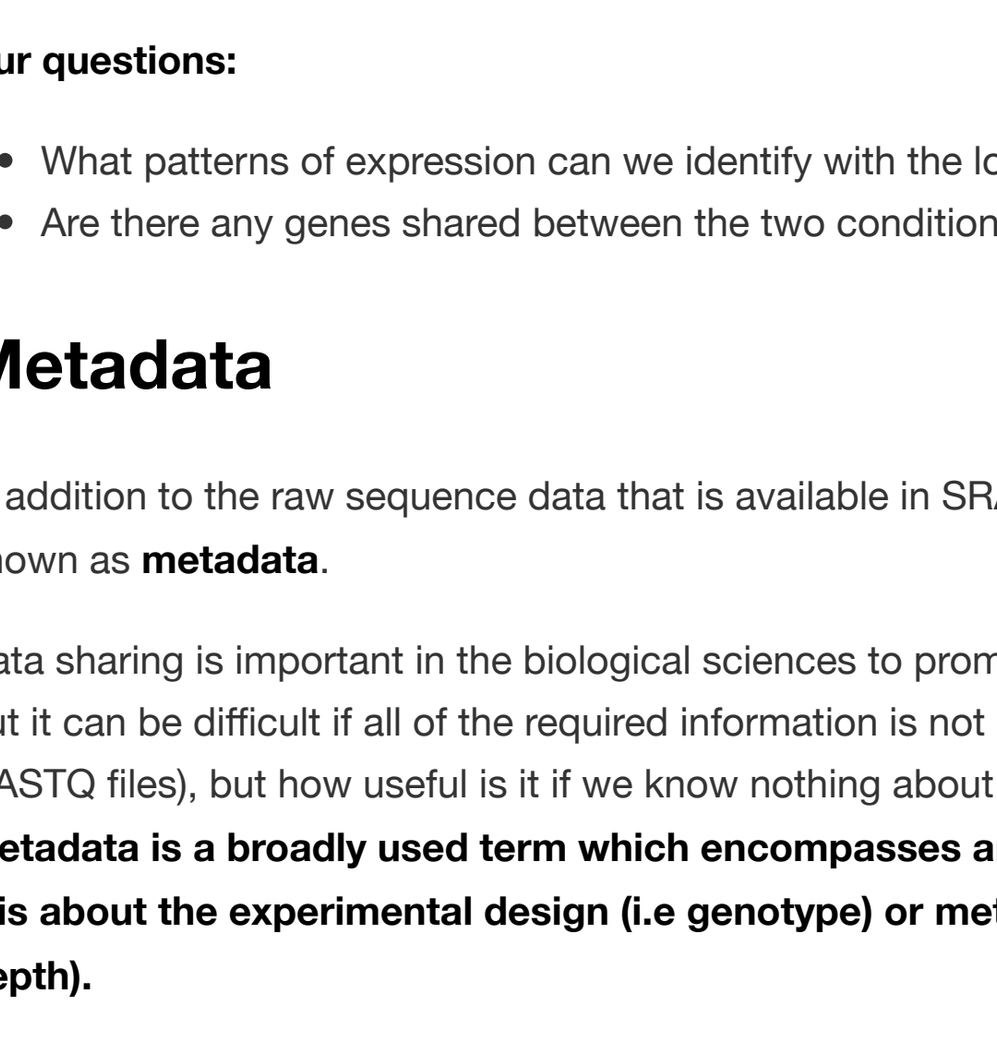
What is the purpose of these datasets? What does Mov10 do?

The authors are investigating interactions between various genes involved in Fragile X syndrome, a disease in which there is aberrant production of the FMRP protein.

> **FMRP** is "most commonly found in the brain, is essential for normal cognitive development and female reproductive function. Mutations of this gene can lead to fragile X syndrome, mental retardation, premature ovarian failure, autism, Parkinson's disease, developmental delays and other cognitive deficits." - from wikipedia

> **MOV10**, is a putative RNA helicase that is also associated with **FMRP** in the context of the microRNA pathway.

The hypothesis the paper is testing is that FMRP and MOV10 associate and regulate the translation of a subset of RNAs.



Our questions:

- What patterns of expression can we identify with the loss or gain of MOV10?
- Are there any genes shared between the two conditions?

Metadata

In addition to the raw sequence data that is available in SRA we also need to collect **information about the data**, also known as **metadata**.

Data sharing is important in the biological sciences to promote scientific integrity, and disseminate scientific discovery; but it can be difficult if all of the required information is not provided. From the SRA we can retrieve the sequence data (FASTQ files), but how useful is it if we know nothing about the samples that this sequence data originated from? **Metadata is a broadly used term which encompasses any kind of information that relates to our data, whether it is about the experimental design (i.e genotype) or metrics related to the sequence data (i.e sequencing depth).**

Here, we provide metadata for the data we are using today.

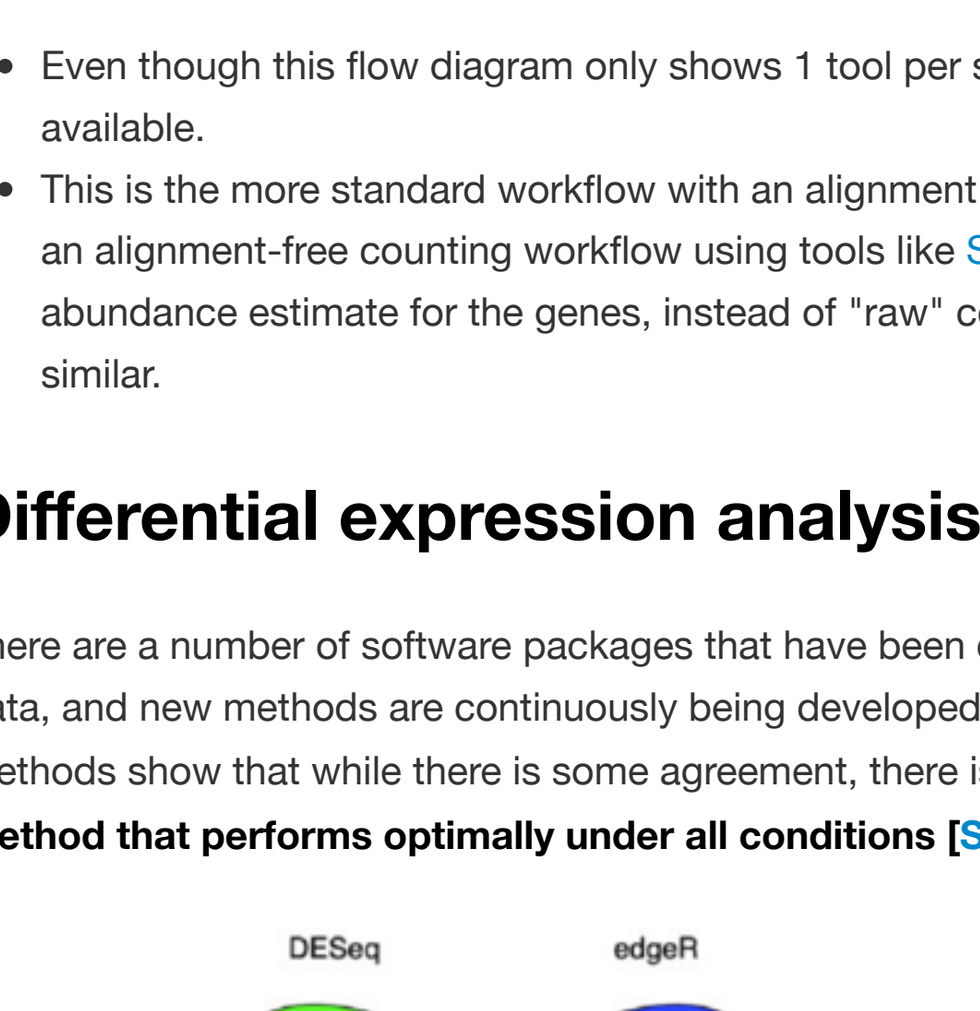
- The RNA was extracted from treated **HEK293F cells**.
- The cDNA libraries for this dataset are **stranded** and were generated using the **TruSeq Stranded mRNA Library Prep Kit** from Illumian.
- Sequencing was carried out on the **Illumina HiSeq-2500 for 100bp single end** reads.
- **~40 million reads** per sample were generated.

Exercise

1. What types of metadata are used in your experimental design (any experiment)?
2. What other kinds of metadata might a sequencing project generate?
3. Why is this type of information important?

From Sequence Data to Count Matrix

Arguably the most common use for transcriptome data is to search for differentially expressed genes. Finding genes that are differentially expressed between conditions is an integral part of understanding the molecular basis of phenotypic variation. The following steps briefly describe the steps and give examples of tools that one might use to obtain gene counts to perform differential expression (DE analysis) on RNA-Seq data.

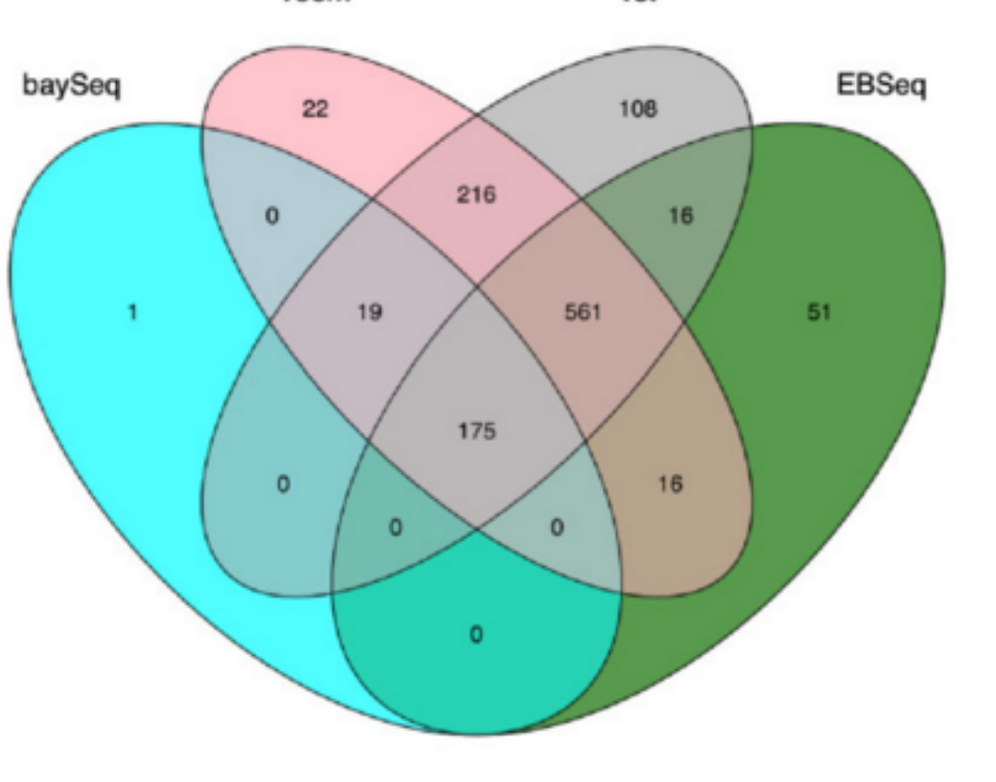


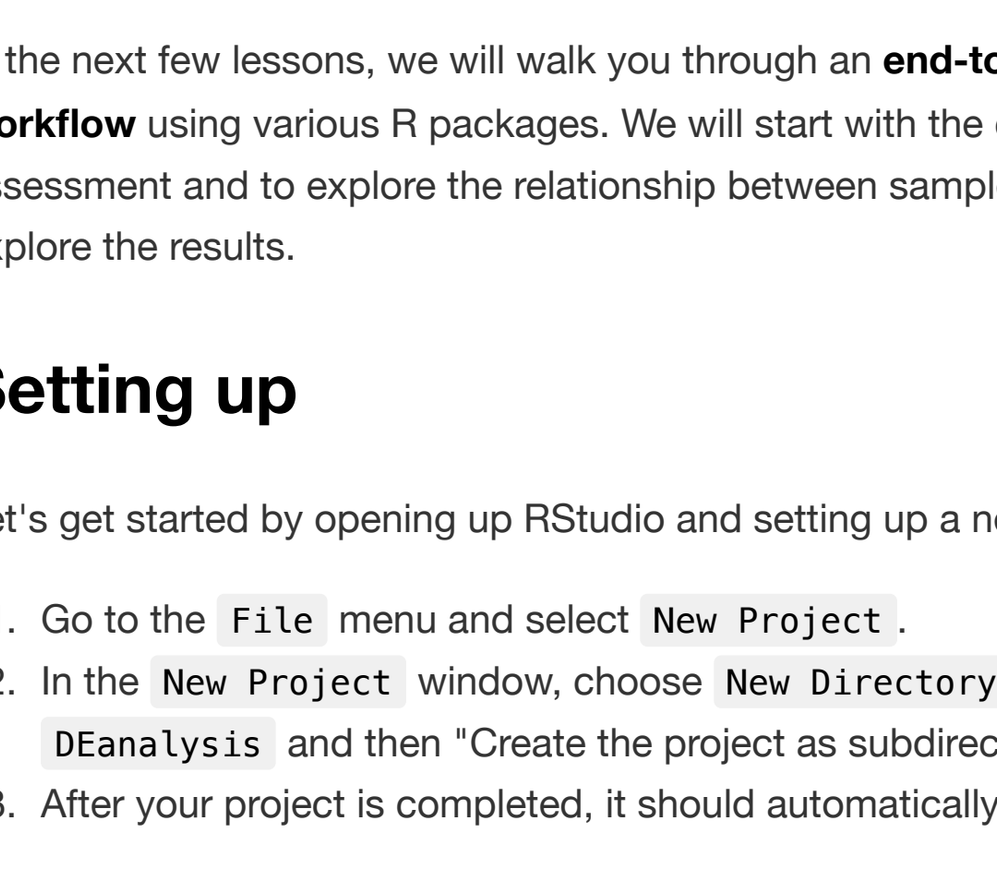
An in-depth explanation of these steps is outside the scope of today's class, but a couple of points:

- Even though this flow diagram only shows 1 tool per step after the sequencing step, there are several tools available.
- This is the more standard workflow with an alignment + a counting step, but more recently people are moving to an alignment-free counting workflow using tools like Salmon and Kallisto. These newer tools will generate an abundance estimate for the genes, instead of "raw" counts, but the downstream steps for statistical analysis are similar.

Differential expression analysis

There are a number of software packages that have been developed for differential expression analysis of RNA-seq data, and new methods are continuously being developed. Many studies describing comparisons between these methods show that while there is some agreement, there is also much variability. **Additionally, there is no one method that performs optimally under all conditions [Soneson and Dleorenzi, 2013].**





In the next few lessons, we will walk you through an **end-to-end gene-level RNA-seq differential expression workflow** using various R packages. We will start with the count matrix, perform exploratory data analysis for quality assessment and to explore the relationship between samples, perform differential expression analysis, and visually explore the results.

Setting up

Let's get started by opening up RStudio and setting up a new project for this analysis.

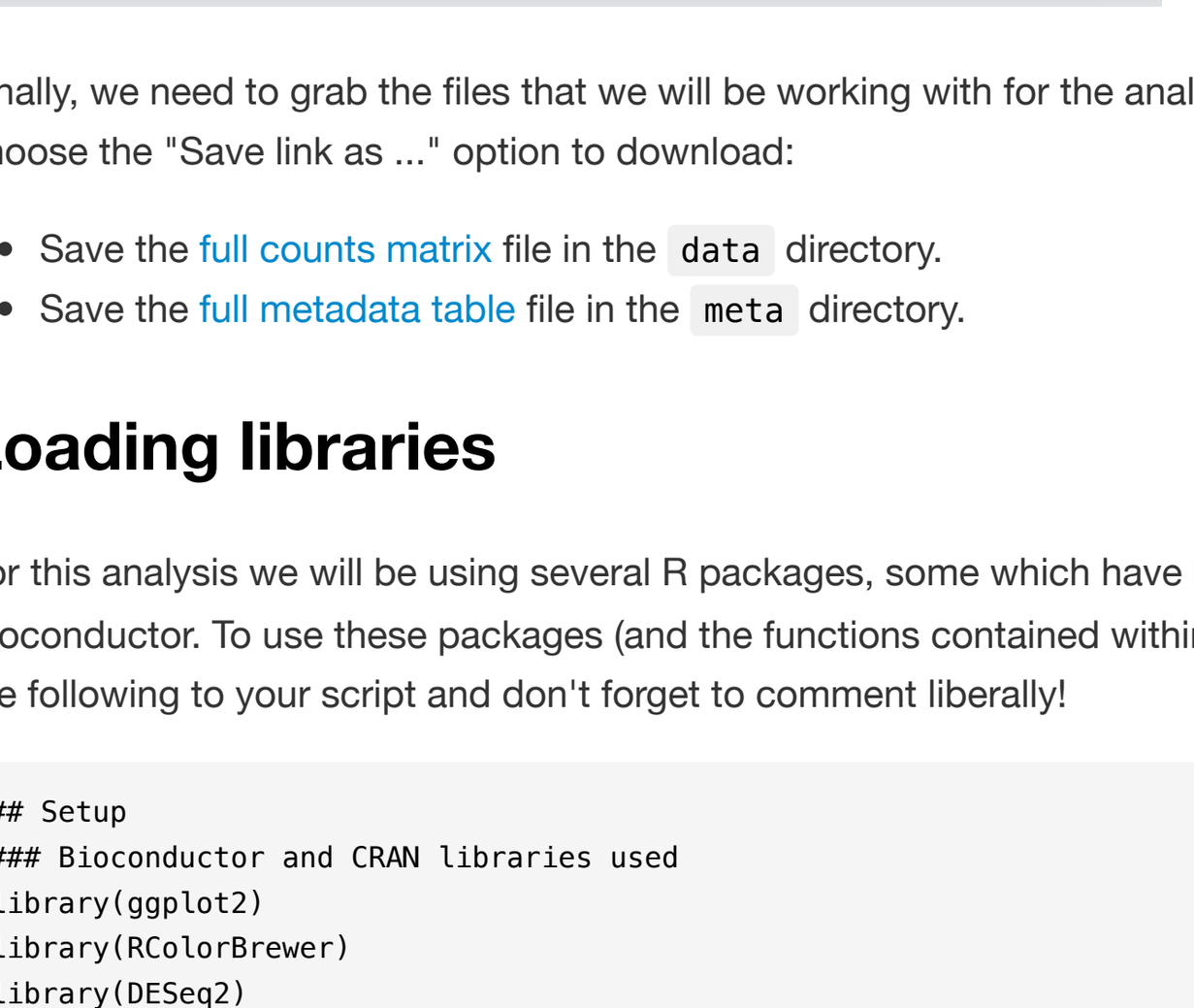
1. Go to the `File` menu and select `New Project`.
2. In the `New Project` window, choose `New Directory`. Then, choose `Empty Project`. Name your new directory `DEanalysis` and then "Create the project as subdirectory of:" the Desktop (or location of your choice).
3. After your project is completed, it should automatically open in RStudio.

To check whether or not you are in the correct working directory, use `getwd()`. The path `Desktop/DEanalysis` should be returned to you in the console. Within your working directory use the `New folder` button in the bottom right panel to create three new directories: `data`, `meta` and `results`. Remember the key to a good analysis is keeping organized from the start!

Go to the `File` menu and select `New File`, and select `R Script`. This should open up a script editor in the top left hand corner. This is where we will be typing and saving all commands required for this analysis. In the script editor type in a header line:

```
## Gene-level differential expression analysis using DESeq2
## NGS Data Analysis 2016
```

Now save the file as `de_script.R`. When finished your working directory should now look something like this:



Finally, we need to grab the files that we will be working with for the analysis. Right click on the links below, and choose the "Save link as ..." option to download:

- Save the full counts matrix file in the `data` directory.
- Save the full metadata table file in the `meta` directory.

Loading libraries

For this analysis we will be using several R packages, some which have been installed from CRAN and others from Bioconductor. To use these packages (and the functions contained within them), we need to **load the libraries.** Add the following to your script and don't forget to comment liberally!

```
## Setup
### Bioconductor and CRAN libraries used
library(ggplot2)
library(RColorBrewer)
library(DESeq2)
library(pheatmap)
```

Dowload packages

```
# Install DESeq2
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

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

```
# Install pheatmap
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("pheatmap")
```

Loading data

To load the data into our current environment, we will be using the `read.table` function. We need to provide the path to each file and also specify arguments to let R know that we have a header (`header = T`) and the first column is our row names (`row.names =1`). By default the function expects tab-delimited files, which is what we have.

```
## Load in data
data <- read.table("data/Mov10_full_counts.txt", header=T, row.names=1)

meta <- read.table("meta/Mov10_full_meta.txt", header=T, row.names=1)
```

Use `class()` to inspect our data and make sure we are working with data frames:

```
### Check classes of the data we just brought in
class(data)
class(meta)
```

As a sanity check we should also make sure that we have sample names that match between the two files, and that the samples are in the right order.

```
### Check that sample names match in both files
all(names(data) %in% rownames(meta))
all(names(data) == rownames(meta))
```

`DESeq2DataSet`

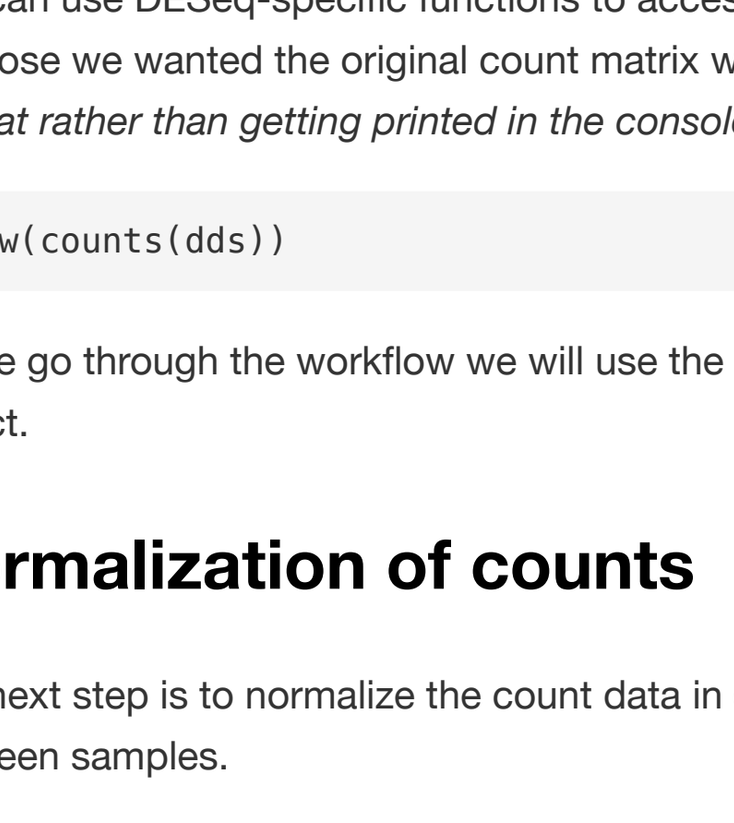
Now that we have all the required files and libraries loaded we are ready to begin with the exploratory part of our analysis.

Amongst the DE tools that work on count data directly, a popular one is DESeq2. The methodology of DESeq2 has high sensitivity and precision, while controlling the false positive rate. It also has various functions for QC assessment conveniently built-in.

The first thing we need to do is create `DESeqDataSet` object. Bioconductor software packages often define and use a custom class for storing data that makes sure that all the needed 'data slots' are consistently provided and fulfill the requirements. These objects are similar to `lists` in that the `data slots` are analogous to components as they store a number of different types of data structures. These objects are **different from lists** in that the slots are designated for specific information and access to that information (i.e. selecting data from the object) is by using object-specific functions as defined by the package.

Let's start by creating the `DESeqDataSet` object and then we can talk a bit more about what is stored inside it. To create the object we will need the **count matrix** and the **metadata** table as input. We will also need to specify a **design formula**. The design formula specifies the column(s) in the metadata table and how they should be used in the analysis. For our dataset we only have one column we are interested in, that is `~sampletype`. This column has three factor levels, which tells DESeq2 that for each gene we want to evaluate gene expression change with respect to these different levels.

```
## Create DESeq2Dataset object
dds <- DESeqDataSetFromMatrix(countData = data, colData = meta, design = ~ sampletype)
```



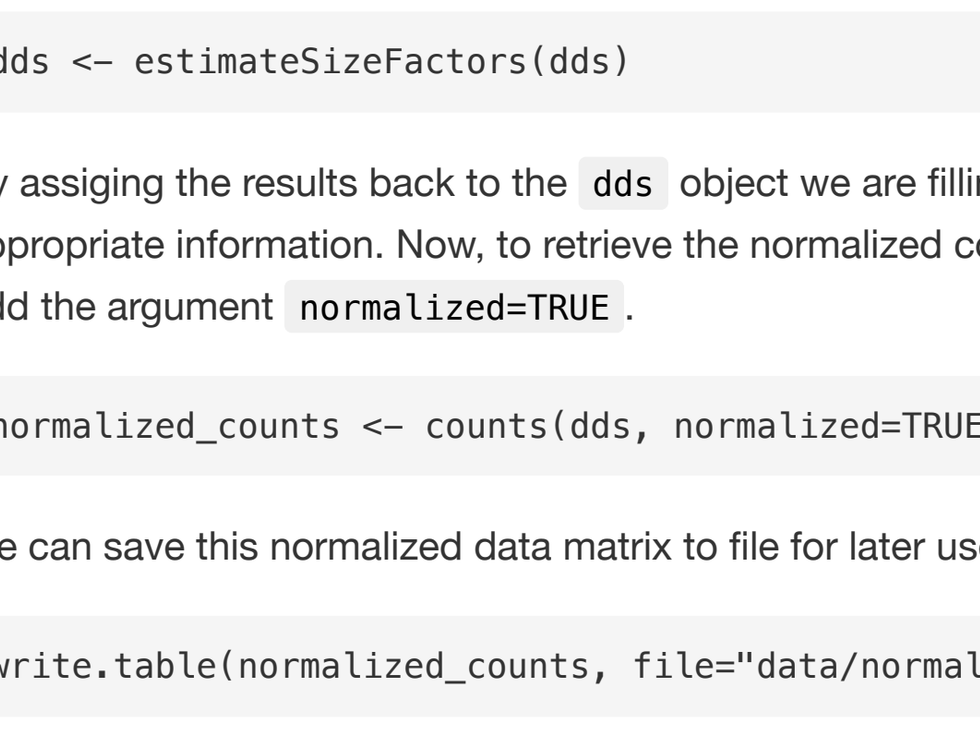
You can use DESeq-specific functions to access the different slots and retrieve information, if you wish. For example, suppose we wanted the original count matrix we would use `counts()` (*Note: we nested it within the `View()` function so that rather than getting printed in the console we can see it in the script editor*) :

```
View(counts(dds))
```

As we go through the workflow we will use the relevant functions to check what information gets stored inside our object.

Normalization of counts

The next step is to normalize the count data in order to be able to make fair gene comparisons both within and between samples.



Remember, that there are other factors that are proportional to the read counts in addition to the gene expression that we are interested in. In DESeq2, `sizeFactors` are computed based on the median of ratios method. This method accounts for gene length and sequencing depth. To generate these size factors we can use the `estimateSizeFactors()` function:

```
dds <- estimateSizeFactors(dds)
```

By assiging the results back to the `dds` object we are filling in the slots of the `DESeqDataSet` object with the appropriate information. Now, to retrieve the normalized counts matrix from `dds`, we use the `counts()` function and add the argument `normalized=TRUE`.

```
normalized_counts <- counts(dds, normalized=TRUE)
```

We can save this normalized data matrix to file for later use:

```
write.table(normalized_counts, file="data/normalized_counts.txt", sep="\t", quote=F, col.names=NA)
```

Transformation of counts

Many common statistical methods for exploratory analysis of multidimensional data, e.g. clustering and principal components analysis (PCA), work best for data that generally have the **same range of variance at different ranges of the mean values**. For RNA-seq raw counts, however, the variance grows with the mean. So the results of a PCA will be largely driven by genes with many counts.

A simple and commonly used strategy to avoid this is to take the logarithm of the normalized count values plus a small pseudocount (for 0 counts); however, now the genes with the very lowest counts will tend to dominate the results.

The DESeq2 solution to this is the **regularized log transform** [Love, Huber, and Anders 2014]. For genes with high counts, the `rlog` transformation will give similar result to the ordinary log2 transformation of normalized counts. For genes with lower counts, however, the values are shrunken towards the genes' means across all samples.

```
### Transform counts for data visualization
rld <- rlog(dds, blind=TRUE)
```

The `rlog` function returns a `DESeqTransform`, another type of DESeq-specific object. The reason you don't just get a matrix of transformed values is because all of the parameters (i.e. size factors) that went in to computing the rlog transform are stored in that object. We will be using this object to plot figures for quality assessment.
