Introduction to gene expression microarray analysis in R and Bioconductor (./index.html)

Microarrays were widely used in the ’00s to interrogate the gene expression of cells in a transcriptome-wide manner. Although the decreasing costs of sequencing has led RNA-seq to become the method of choice for genome-wide transcriptomics, microarrays are still used due to the relative simplicity of analysis. In addition there are many existing data sets using microarrays that are still valuable for analysis. This lesson will introduce you to using analysing gene expression experiments on microarrays using linear models of differential expression.

Prerequisites

The first portion of the lesson assumes a basic knowledge of R, and theoretical knowledge of how gene expression microarray experiments are peformed. The second portion assumes theoretical familiarity with how differentially expressed genes are identified using microarrays.

Schedule

Time	Topic	Learning Objectives
Before start	Setup (./setup.html)	
14:00	Working with Bioconductor (./01-InstallingBioC/index.html)	• Become familiar with the basic Bioconductor setup. • Be able to install the appropriate Bioconductor packages for microarray analysis • Be able to use the help system and vignettes for Bioconductor packages

14:20	Importing processed microarray data into R from GEO (./02-GEODataImport/index.html)	• Be able to obtain data from GEO, including processed and raw data. • Be able to explain and use the differences between GEO data types. • Understand the concept of the ExpressionSet class of objects.
14:40	Importing raw (unprocessed) Affymetrix microarray data (./03-Supplemental-Affy/index.html)	• Be able to obtain supplemental data • Be able to explain and use the differences between GEO data types. • Understand the concept of the ExpressionSet class of objects.
14:55	Working with experimental metadata (./04-MetaData/index.html)	• Be able to use metadata from GEO objects to construct useful R data objects • Be able to use read.celfiles in combination with your own pData object to ensure data integrity
15:10	Microarray Data processing with RMA (./05-DataNormalisation/index.html)	• Understand and explain Background correct, normalise and summarisation steps for microarray data

15:50	coffee break (./05-break/index.html)	Break
16:00	Identifying differentially expressed genes using linear models (part 1) (./06-DifferentialGeneExpression/index.html)	• Be able to use limma to identify differentially expressed genes. • Understand the formula class of objects in R, and use it to specify the appropriate model for linear modeling.
16:40	Identifying differentially expressed genes using linear models (part 2, factorial designs) (./07-factorial_designs/index.html)	• Be able to use limma to identify differentially expressed genes. • Understand the formula class of objects in R, and use it to specify the appropriate model for linear modeling.
17:20	From features to annotated gene lists (./08-AnnotatingResults/index.html)	• Be able to use AnnotationDb methods to association annotations with platform data.
17:50	Basic downstream analysis of microarray data (./09-downstreamAnalysis/index.html)	• Be able to plot volcano plots and heatmaps in R. • Be able to interpret the above plots generated. • Be aware of some downstream analysis that are commonly done to interpret the results of differential expression analysis.
18:10	Finish	

Edit on GitHub (https://github.com/GTK-teaching/Microarrays-R/edit/gh-pages/index.md) / Source (https://github.com/GTK-teaching/Microarrays-R/) / Contact (mailto:dbsgtk@nus.edu.sg)

Using The GTK-teaching/lesson-theme (https://github.com/GTK-teaching/styles/) theme, derived from The Carpentries style (https://github.com/carpentries/styles/)

(../)

Introduction to gene expression microarray analysis in R and Bioconductor (../)

(../02-GEOD

Working with Bioconductor

Overview

Teaching: 10 min
Exercises: 10 min
Learning Objectives

- Become familiar with the basic Bioconductor setup.
- Be able to install the appropriate Bioconductor packages for microarray analysis
- Be able to use the help system and vignettes for Bioconductor packages

Bioconductor (https://www.bioconductor.org/) is an open-source project and software repository hosting a wide range of packages tailored for the analysis of biological data. As a repository, Bioconductor is a complement to CRAN (https://cran.r-project.org/), which is used for R package hosting *in general*. Importantly, Bioconductor provides widely used object classes (such as the ExpressionSet class, which we will discuss later) for representing and manipulating genomic data, as well as data packages such as annotation for both microarray platforms and genomes.

Installing Bioconductor packages

Bioconductor has its own package repository, like CRAN, for its packages. In order to install packages from Bioconductor, you will first need to set up the Bioconductor installer on your computer. This can be done using the code described in more detail at the Bioconductor installation page (http://www.bioconductor.org/install/), but basically this:

```
if (!requireNamespace("BiocManager"))
    install.packages("BiocManager")
BiocManager::install()
```

Notice the use of `if`. The `BiocManager` package will only be installed if it can't be loaded, because `requireNamespace()` returns a logical value depending on the success of loading the named library. After loading the BiocManager namespace, `BiocManager::install()` installs the required Bioconductor packages. To install an *optional* Bioconductor package named "foo", you can run `BiocManager::install('foo')` instead of `install.packages()`.

Setting up your computer for today's lesson.

For today's lesson we will be using data from GEO. There is a package in Bioconductor called GEOquery

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("GEOquery")
```

You will need to install the following packages which we will be using for the rest of the practical today. Some of them may be intalled by default, so only install them if they are not available.

affy (http://bioconductor.org/packages/release/bioc/html/affy.html)
A package for analysing Affymetrix platform data

oligo (http://bioconductor.org/packages/release/bioc/html/oligo.html)
A different package for analysing oligonucleotide platform data, including Affymetrix data

limma (http://bioconductor.org/packages/release/bioc/html/limma.html)
A package for analysing linear models of microarray data.

hgu133plus2.db (http://bioconductor.org/packages/release/data/annotation/html/hgu133plus2.db.html)
an annotation package for the Affymetrix microarrays of this lesson.

org.Hs.eg.db (https://bioconductor.org/packages/release/data/annotation/html/org.Hs.eg.db.html)
Annotation of the human genome

Once these packages are installed, load them using `library()`. Check that these packages are loaded using `sessionInfo()` or `search()`.

⚠ Masking objects with `library()`

when you attached a package with `library()` you may have noticed a bunch of messages beginning with `The following objects are masked ...`. Remember, `library()` adds packages at position 2 of the search path by default. This means that any time you attach a package using `library()` you risk *masking* objects of the same name in other packages.

You can always refer to an object `foo` from package `mypackage` using its fully qualified name: `mypackage::foo`.

Getting help for Bioconductor packages

Bioconductor workflows

Some great help is available at this page (http://bioconductor.org/packages/release/BiocViews.html#___Workflow) on Bioconductor, especially if you are unsure of how to approach a common problem. The site includes sample workflows for the most common analysis types, including sample code. Unlike vignettes (below) which deal with specific packages, these workflows often describe multiple packages commonly used together to answer a biological question.

Package vignettes

Many, if not most, packages that are deposited on Bioconductor includes a **vignette**, which is a short example of how to use the package to address a given problem. The vignettes will also provide example code, and use self-contained datasets to demonstrate use of the package.

To find the vignettes available for a package, type

```
browseVignettes('packagename')
```

at the R console.

Find out what you can do with an object

Bioconductor objects may seem complicated. Many of them have specific methods available, but you need to know what functions you can call on them. You can find the methods available for an object by using its *class*, via

```
## find the methods available for an object named foo
methods(class=class(foo))
```

Package reference manuals

This is usually a more technical document that details the usage of different functions within the package. Usually, this is what people refer to when they want to know the full range of arguments that can be passed into the function. Also, the reference manuals will include information about default options that are used if a particular argument is not supplied by the user.

Bioconductor mailing list

This is a community-driven resource for help. The Bioconductor mailing list is very active, with people providing not just help when one runs into errors in their scripts but also advice on statistical tests/methods for analyzing ones data. Questions on the mailing list can be found at https://support.bioconductor.org/ (https://support.bioconductor.org/) Please read the posting guide http://bioconductor.org/help/support/posting-guide/ (http://bioconductor.org/help/support/posting-guide/) prior to posting. Users on the forum assume that you are very familiar with R.

Key Points

- Bioconductor is an archive containing a wide range of packages for bioinformatics analysis.
- Installation of Bioconductor packages is done using the `BiocManager::install()` function, rather than through the usual `install.packages()`

(../)

(../02-GEOD

‹ (../01-InstallingBioC/index.html)

Introduction to gene expression microarray analysis in R and Bioconductor (../)

› (../03 Supp Affy/i

Importing processed microarray data into R from GEO

Overview

Teaching: 10 min
Exercises: 10 min
Learning Objectives

- Be able to obtain data from GEO, including processed and raw data.
- Be able to explain and use the differences between GEO data types.
- Understand the concept of the ExpressionSet class of objects.

In the class, we learned how the basic requirements of a complete data object for gene expression microarray studies consists of a tall, skinny matrix (for the experimental measurements) and a short, wide table (to describe the experiment). The Bioconductor class for this type of data is an `ExpressionSet`.

In this lesson we will be creating these objects in R from data available at GEO.

The ExpressionSet class

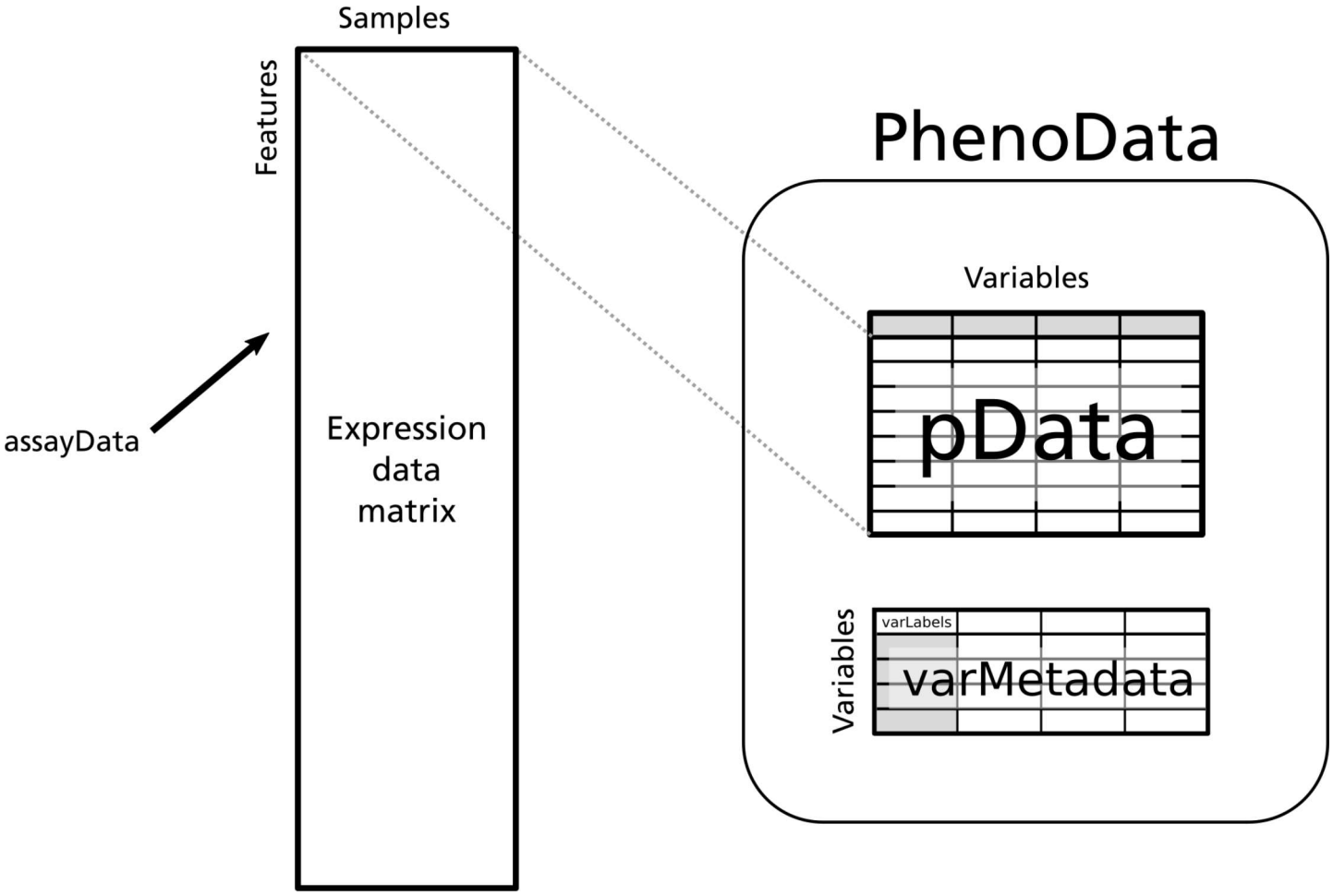
The ExpressionSet class of objects are commonly used to store microarray experiment data. A typical ExpressionSet class object contains the following data:

assayData
raw or processed intensities, where each row corresponds to one probe and each column corresponds to one sample

phenoData
experimental meta-data, where each row corresponds to one sample.

featureData
This is optional annotation of the features (e.g., genes or transcripts) measured in an experiment.

annotation
a character vector specifying the platform name



The first data set

Data for this lesson is taken from GEO (accession ID: GSE33146 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE33146)). The experiment was on a cancer cell line grown in culture. Cells are grown in either MEGM (where they retain an epithelial phenotype) or SCGM (where they undergo reversible epithelial-mesencymal transition, or EMT). Gene expression analysis using the hgu133plus2 (https://bioconductor.org/packages/release/data/annotation/html/hgu133plus2.db.html) microarray was performed to identify genes associated with the EMT process.

Getting processed data from GEO into R

Before any analysis, we need to get the data into R. If you navigate to GSE33146 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE33146), you will see that several different file formats are available for download.

The GEO file types

SOFT (https://www.ncbi.nlm.nih.gov/geo/info/soft.html) files
The SOFT file format is commonly used by submitters to GEO. It contains a lot of information about the experiment (MIAME compliance), and also contains the estimates for each gene or transcript, in a custom defined file format.

MINiML (https://www.ncbi.nlm.nih.gov/geo/info/MINiML.html) file
The MINiML file format uses XML, a computer-readable markup language. This has the same information as SOFT, but in a different format.

Series matrix file
The matrix file is meant to be readable directly into a spreadsheet. Generally these files don't have all the information provided by SOFT, but are easy to work with.

Supplemental files
The `.CEL` files have the original, unprocessed data. Since we want to process it ourselves, that's what we'll need.

Download the files or query them directly using GEOquery

The GEOquery package (definitely try `browseVignettes('GEOquery')`) provides tools for reading files directly from GEO based on the accession. GEOquery works for different kinds of GEO data, including Samples (GSM), platforms (GPL), data sets, (GDS), and series (GSE).

These data appear complicated.

GEOquery, like other Biconductor packages, creates data objects that seem somewhat complicated. If you are used to simple R lists, vectors, and data frames, the data structures may seem overwhelming. Keep in mind that these data stuctures, regardless of how complicated they are, are basically compositions of data structures you already know: lists, vectors, matrices, and data frames.

Getting a single GSM object for a single sample

We can create a single GSM object of a single sample with a call to getGEO (I'm using an example from the same GSE).

```
library(GEOquery)
gsm <- getGEO('GSM820817')
```

```
File stored at:
```

```
/var/tmp//Rtmpf45Tmb/GSM820817.soft
```

```
Rows: 54675 Columns: 2
```

```
── Column specification ────────────────────────────────────────────────────────
Delimiter: "\t"
chr (1): ID_REF
dbl (1): VALUE
```

```

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## What methods are available to call on this class of data?
methods(class=class(gsm))
```

```
[1] Accession Columns   dataTable Meta      show      Table
see '?methods' for accessing help and source code
```

```
## The Table method shows the estimates
head(Table(gsm))
```

```
     ID_REF     VALUE
1 1007_s_at 11.502651
2   1053_at  9.861017
3    117_at  6.299005
4    121_at  8.632274
5 1255_g_at  3.721331
6   1294_at  7.376385
```

```
## The Meta method shows the experimental design, etc.
head(Meta(gsm))
```

```
$channel_count
[1] "1"

$characteristics_ch1
[1] "cell line: Human breast cancer-derived cell line DKAT"
[2] "culture medium: MEGM"

$contact_address
[1] "MSRB Room 217, Research Dr."

$contact_city
[1] "Durham"

$contact_country
[1] "USA"

$contact_department
[1] "Pharmacology and Cancer Biology"
```

As you can see, the `Table` method returns numeric estimates for each probe (which is what we will eventually get after pre-processing), while the `Meta` method gives us information about our experiment.

Why does this GSM appear in *two* series?

One of the `Meta()` entries shows that the sample from GSE33146 actual belongs to *two* series. What is the difference between them?

Solution

```
Meta(gsm)[['series_id']]
```

```
[1] "GSE33146" "GSE33167"
```

Look up GSE33167 on the GEO web site. The series GSE33167 is a *super-series* composed of two series: the one we are looking at and another series comparing baseline expression among triple-negative breast cancer cell lines.

Retrieving the whole GSE as a list of ExpressionSets

The function `getGEO()` with a GSE identifier as a sole argument will download the series matrix files from GEO and return them as a *list of ExpressionSet objects*. In many cases, there will only be one experiment for a series, so the list will contain a single ExpressionSet, which can be retrieved as the first member of the list.

```
library(GEOquery)
gse33146 <- getGEO('GSE33146')
```

```
Found 1 file(s)
```

```
GSE33146_series_matrix.txt.gz
```

```
Rows: 54675 Columns: 7
```

```
── Column specification ────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────────────────────────
──────────
Delimiter: "\t"
chr (1): ID_REF
dbl (6): GSM820817, GSM820818, GSM820819, GSM820820, GSM820821, GSM820822
```

```
ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
File stored at:
```

```
/var/tmp//Rtmpf45Tmb/GPL570.soft
```

```
length(gse33146)
```

```
[1] 1
```

```
class(gse33146[[1]])
```

```
[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
```

```
gse33146 <- gse33146[[1]]
```

Retrieving the whole GSE by parsing the SOFT file

When `getGEO()` is run with the named argument `GSEMatrix=FALSE`, it does not parse the Series Matrix files but does parse the SOFT files. The advantage of this method is that the SOFT files may contain more information than the series matrix files, but a disadvantage is that it is much slower and the resulting object is *not* a list of ExpressionSets. However, parsing the Series Matrix can also introduce errors; parsing the SOFT file is less error-prone.

The GSE is the most complicated object retrieved by getGEO, because a series is composed of sample (GSM) and platform (GPL) objects. Each is accessed by a *list*: `GSMList` and `GPLList`. You can download the whole series into an R object by accession, but you can also download the file outside of R and read it in as a local file.

```
library(GEOquery)
## if we have a good connection to the internet, we can download it directly from GEO
gse2 <- getGEO('GSE33146',GSEMatrix=FALSE)
# If it's available on a local file system, we can read the file directly
##gse2 <- getGEO(filename='data/GSE33146_family.soft')
names(GSMList(gse2))
```

📌 Using the processed data from a GSE or GDS as an R ExpressionSet

For a GEO Data Set (GDS) The GEOquery package provides a function `GDS2eSet` to converta a GDS directly to an ExpressionSet object. This function uses the *processed* data available on GEO, not the raw data.

There is no corresponding `GSE2eSet` function because a single GEO Series can contain data from multiple platforms, and an ExpressionSet object is data from one platform.

The two ways of getting series data from GEO using getGEO()

Retrieving a list of ExpressionSets

```
> gse33146 = getGEO('GSE33146')
> class(gse33146)
[1] "list"
> length(gse33146)
[1] 1
> class(gse33146[[1]])
[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
```

Retrieving a "GSE" object

```
> gse33146 = getGEO('GSE33146',GSEMatrix=FALSE)
> class(gse33146)
[1] "GSE"
attr(,"package")
[1] "GEOquery"
> methods(class=class(gse33146))
[1] GPLList GSMList Meta
see '?methods' for accessing help and source code
> names(GPLList(gse33146))
[1] "GPL570"
> names(GSMList(gse33146))
[1] "GSM820817" "GSM820818" "GSM820819" "GSM820820" "GSM820821" "GSM820822"
```

Get the second data set

Our second data set contains 12 samples, and is found in GSE66417 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE66417). Retrieve the processed data into an ExpressionSet object.

Solution

```
gse66417 <- getGEO('GSE66417')
```

```
Found 1 file(s)
```

```
GSE66417_series_matrix.txt.gz
```

```
Rows: 53617 Columns: 13
```

```
── Column specification ────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────────────────────────
──────────────
Delimiter: "\t"
dbl (13): ID_REF, GSM1622170, GSM1622189, GSM1622191, GSM1622194, GSM1622196...
```

```

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
File stored at:
```

```
/var/tmp//Rtmpf45Tmb/GPL16686.soft
```

```
Warning: One or more parsing issues, see `problems()` for details
```

```
length(gse66417) # this should be a list of length 1
```

```
[1] 1
```

```
gse66417 <- gse66417[[1]]
```

Key Points

- GEO data types have enough similarities to allow data access, but enough differences to require specific type-specific steps. and analysis.
- The ExpressionSet class of object contains slots for different information associated with a microarray experiment.

‹ (../01-InstallingBioC/index.html)

› (../03 Supp Affy/i

‹ (../02-GEODataImport/index.html)

Introduction to gene expression microarray analysis in R and Bioconductor (../)

(../04-Metal ›

Importing raw (unprocessed) Affymetrix microarray data

Overview

Teaching: 10 min
Exercises: 5 min
Learning Objectives

- Be able to obtain supplemental data
- Be able to explain and use the differences between GEO data types.
- Understand the concept of the ExpressionSet class of objects.

We now have processed data for two series, GSE33146 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE33146) and [GSE66417][GEE66417]. We need to get the *unprocessed* data to understand the processing.

Getting the raw data for the series using GEOquery

Getting the raw data (`CEL` files for Affymetrix data) is distinct from getting the processed ata. The `CEL` files can be pretty large, and sometimes the download can fail, even with a good connection.

Nonetheless, if you want to do it, `getGEOSuppFiles()` will download all the supplementary files for the GEO accession. Note that it doesn't *process* or even *parse* the files, since there are many different types of supplementary files on GEO and R doesn't know the format ahead of time. It does, however, return the file paths.

```
## If you have an awesome connection and a lot of time
filePaths <- getGEOSuppFiles('GSE33146')
filePaths <- getGEOSuppFiles('GSE66417')
```

In the interest of time, don't do that. Instead, use the files you have downloaded from LumiNUS and unpacked into a directory named for each series. You can leave them compressed.

Reading CEL data using the `oligo` package.

The `oligo` package provides functions for handling Affymetrix data, including CEL file data. The function `oligo::read.celfiles()` does the work. This function takes in a vector of filenames as the argument. We can manually type the file names into a vector, using the `c()` function. Alternatively, we can use `list.celfiles()` command provided by `oligoClasses` , which is installed when we install `oligo` . `list.celfiles()` will list all the files ending with the *.cel* extension (CEL files) in a directory, (by default the current working directory). Helpfully:

- the argument `listGzipped=TRUE` will find compressed CEL files.
- the argument `full.names=TRUE` will include the *directory* name, so the results can be passed to `read.celfiles()`.

Therefore, the following needs to be done:

1. Use `oligoClasseslist.celfiles(GSE33146)` with the appropriate arguments to generate a vector containing the name of all the CEL files.
2. Use `read.celfiles()` to read in the CEL files.

Try it!

Try to read the CEL files for both data sets into R using the information provided above.

Solution

```
library(oligo)
library(oligoClasses)
gse33146_celdata <- read.celfiles(list.celfiles('GSE33146',full.names=TRUE,listGzipped=TRUE))
```

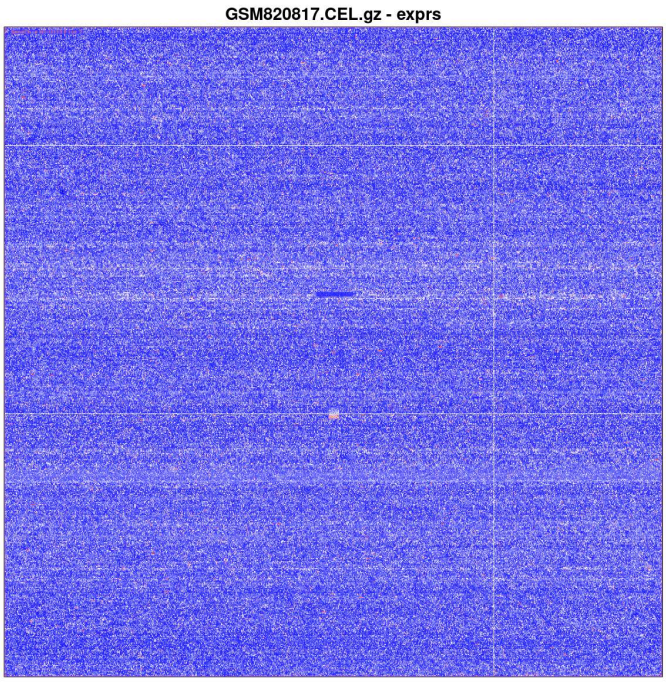
```
Reading in : GSE33146/GSM820817.CEL.gz
Reading in : GSE33146/GSM820818.CEL.gz
Reading in : GSE33146/GSM820819.CEL.gz
Reading in : GSE33146/GSM820820.CEL.gz
Reading in : GSE33146/GSM820821.CEL.gz
Reading in : GSE33146/GSM820822.CEL.gz
```

```
gse66417_celdata <- read.celfiles(list.celfiles('GSE66417',full.names=TRUE,listGzipped=TRUE))
```

```
Reading in : GSE66417/GSM1622170_Jurkat-Ctrl-RD1_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622189_Jurkat-Ctrl-RD2_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622191_Jurkat-Ctrl-RD3_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622194_Jurkat-Ixazomib-RD4_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622196_Jurkat-Ixazomib-RD5_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622198_Jurkat-Ixazomib-RD6_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622200_L540-Ctrl-RD7_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622202_L540-Ctrl-RD8_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622204_L540-Ctrl-RD9_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622206_L540-Ixazomib-RD10_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622209_L540-Ixazomib-RD11_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622211_L540-Ixazomib-RD12_HuGene-2_0-st_.CEL.gz
```

Once you have the CEL file data, you can get a pseudo-image of the chip intensities.

```
image(gse33146_celdata[,1])
```



Note that `read.celfiles` loaded data packages from Bioconductor containing information on the platform design for each chip type.

⚠ The risk of using `list.celfiles()`

Using `list.celfiles()` to provide the files to `read.celfiles()` can be risky. `list.celfiles()` provides a list of files in lexicographic order. This is *probably* the same order as the files in your GSE, but can you be sure? Ultimately, you need to use the metadata from the series to ensure that the rows of the phenoData match the columns of the assayData That is the subject of our next episode.

⚠ `oligo` or `affy`?

The `affy` package also has a `list.celfiles()` function (with a slightly different interface), and offers a `read.AffyBatch()` function to read celfiles. However, the `affy` package can only work with 3′ biased Affymetrix arrays, so `oligo` is preferred

🔑 Key Points

- GEO data types have enough similarities to allow data access, but enough differences to require specific type-specific steps.
- The ExpressionSet class of object contains slots for different information associated with a microarray experiment.

‹ (../02-GEODataImport/index.html)

› (../04-Metal

‹ (../03-Supplemental-Affy/index.html)

Introduction to gene expression microarray analysis in R and Bioconductor (../)

› (../05-DataN

Working with experimental metadata

Overview

Teaching: 10 min
Exercises: 5 min
Learning Objectives

- Be able to use metadata from GEO objects to construct useful R data objects
- Be able to use read.celfiles in combination with your own pData object to ensure data integrity

We have successfully read CEL files into an R object, but we haven't provided any information about the experimental design, which would normally be provided in a phenoData object. In the case of our two experiments, the experimental designs are pretty simple. Let's see if we can find the information we need from the metadata.

What you should have

If you have been following this tutorial, you should have the following objects:

- An *ExpressionSet* object for GSE33146 (the processed data). Let's assume this is called `gse33146`.
- An *ExpressionSet* object for GSE66417 (the processed data). Let's assume this is called `gse66417`.
- An *ExpressionFeatureSet* object for GSE33146 (the raw data). Let's assume this is called `gse33146_celdata`.
- A *GeneFeatureSet* object for GSE66417 (the raw data). Let's assume this is called `gse66417_celdata`.

Why do the raw data for the two experiments appear to be different classes?

The GSE33146 raw data is of class ExpressionFeatureSet, but the GSE66417 data is of class GeneFeatureSet. Raw Affymetrix data is all some form of "FeatureSet" data, but different array designs are *extensions* of the abstract FeatureSet class. The first series is from a 3′-biased array design, and the second is from a gene-based design. The classes represent those different design types.

GSE33146

In the simple case, we already have an ExpressionSet object, and we can use the pData from it.

```
library(tidyverse)
varLabels(gse33146)
```

```
 [1] "title"                  "geo_accession"
 [3] "status"                 "submission_date"
 [5] "last_update_date"       "type"
 [7] "channel_count"          "source_name_ch1"
 [9] "organism_ch1"           "characteristics_ch1"
[11] "characteristics_ch1.1"  "treatment_protocol_ch1"
[13] "growth_protocol_ch1"    "molecule_ch1"
[15] "extract_protocol_ch1"   "label_ch1"
[17] "label_protocol_ch1"     "taxid_ch1"
[19] "hyb_protocol"           "scan_protocol"
[21] "description"            "description.1"
[23] "data_processing"        "platform_id"
[25] "contact_name"           "contact_laboratory"
[27] "contact_department"     "contact_institute"
[29] "contact_address"        "contact_city"
[31] "contact_state"          "contact_zip/postal_code"
[33] "contact_country"        "supplementary_file"
[35] "data_row_count"         "cell line:ch1"
[37] "culture medium:ch1"
```

```
# It looks like the variables of interest are "cell line:ch1" and "treatment:ch1", and the sample accessions are in "geo_accession"
gse33146$supplementary_file
```

```
[1] "ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM820nnn/GSM820817/suppl/GSM820817.CEL.gz"
[2] "ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM820nnn/GSM820818/suppl/GSM820818.CEL.gz"
[3] "ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM820nnn/GSM820819/suppl/GSM820819.CEL.gz"
[4] "ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM820nnn/GSM820820/suppl/GSM820820.CEL.gz"
[5] "ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM820nnn/GSM820821/suppl/GSM820821.CEL.gz"
[6] "ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM820nnn/GSM820822/suppl/GSM820822.CEL.gz"
```

```
pd <- pData(gse33146)
pd['cel_file'] <- str_split(pd$supplementary_file,"/") %>% map_chr(tail,1)
```

With this simple trick, we now can re-read out CEL files, and *guarantee* that they are in the correct order as the experimental data we have from getGEO.

```
gse33146_celdata <- read.celfiles(paste0('GSE33146/',pd$cel_file),phenoData=phenoData(gse33146))
```

```
Reading in : GSE33146/GSM820817.CEL.gz
Reading in : GSE33146/GSM820818.CEL.gz
Reading in : GSE33146/GSM820819.CEL.gz
Reading in : GSE33146/GSM820820.CEL.gz
Reading in : GSE33146/GSM820821.CEL.gz
Reading in : GSE33146/GSM820822.CEL.gz
```

```
Warning in read.celfiles(paste0("GSE33146/", pd$cel_file), phenoData =
phenoData(gse33146)): 'channel' automatically added to varMetadata in phenoData.
```

We get a warning because some extra information about detection channels is added, but no matter: we have already gotten the treatment conditions attached to our data!

```
pData(gse33146_celdata)[,c("geo_accession","cell line:ch1","culture medium:ch1")]
```

```
          geo_accession                            cell line:ch1
GSM820817     GSM820817 Human breast cancer-derived cell line DKAT
GSM820818     GSM820818 Human breast cancer-derived cell line DKAT
GSM820819     GSM820819 Human breast cancer-derived cell line DKAT
GSM820820     GSM820820 Human breast cancer-derived cell line DKAT
GSM820821     GSM820821 Human breast cancer-derived cell line DKAT
GSM820822     GSM820822 Human breast cancer-derived cell line DKAT
          culture medium:ch1
GSM820817               MEGM
GSM820818               MEGM
GSM820819               MEGM
GSM820820               SCGM
GSM820821               SCGM
GSM820822               SCGM
```

GSE66417

We can do the same thing with our second data set, but we can see that the experimental layout is more complicated

```
library(tidyverse)
varLabels(gse66417)
```

```
 [1] "title"                  "geo_accession"
 [3] "status"                 "submission_date"
 [5] "last_update_date"       "type"
 [7] "channel_count"          "source_name_ch1"
 [9] "organism_ch1"           "characteristics_ch1"
[11] "characteristics_ch1.1"  "treatment_protocol_ch1"
[13] "growth_protocol_ch1"    "molecule_ch1"
[15] "extract_protocol_ch1"   "label_ch1"
[17] "label_protocol_ch1"     "taxid_ch1"
[19] "hyb_protocol"           "scan_protocol"
[21] "description"            "data_processing"
[23] "platform_id"            "contact_name"
[25] "contact_email"          "contact_institute"
[27] "contact_address"        "contact_city"
[29] "contact_state"          "contact_zip/postal_code"
[31] "contact_country"        "supplementary_file"
[33] "data_row_count"         "cell type:ch1"
[35] "treatment:ch1"
```

```
# It looks like the variables of interest are "cell type:ch1" and "treatment:ch1", and the sample accessions are in "geo_accession"
pd <- pData(gse66417)
pd['cel_file'] <- str_split(pd$supplementary_file,"/") %>% map_chr(tail,1)
```

We can repeat our simple trick to guarantee our cel files and our metadata are aligned.

```
gse66417_celdata <- read.celfiles(paste0('GSE66417/',pd$cel_file),phenoData=phenoData(gse66417))
```

```
Reading in : GSE66417/GSM1622170_Jurkat-Ctrl-RD1_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622189_Jurkat-Ctrl-RD2_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622191_Jurkat-Ctrl-RD3_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622194_Jurkat-Ixazomib-RD4_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622196_Jurkat-Ixazomib-RD5_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622198_Jurkat-Ixazomib-RD6_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622200_L540-Ctrl-RD7_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622202_L540-Ctrl-RD8_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622204_L540-Ctrl-RD9_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622206_L540-Ixazomib-RD10_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622209_L540-Ixazomib-RD11_HuGene-2_0-st_.CEL.gz
Reading in : GSE66417/GSM1622211_L540-Ixazomib-RD12_HuGene-2_0-st_.CEL.gz
```

```
Warning in read.celfiles(paste0("GSE66417/", pd$cel_file), phenoData =
phenoData(gse66417)): 'channel' automatically added to varMetadata in phenoData.
```

```
pData(gse66417_celdata)[,c("geo_accession","cell type:ch1","treatment:ch1")]
```

```
           geo_accession    cell type:ch1                treatment:ch1
GSM1622170    GSM1622170  T-Cell Lymphoma                      Control
GSM1622189    GSM1622189  T-Cell Lymphoma                      Control
GSM1622191    GSM1622191  T-Cell Lymphoma                      Control
GSM1622194    GSM1622194  T-Cell Lymphoma 25nM of Ixazomib for 24 Hours
GSM1622196    GSM1622196  T-Cell Lymphoma 25nM of Ixazomib for 24 Hours
GSM1622198    GSM1622198  T-Cell Lymphoma 25nM of Ixazomib for 24 Hours
GSM1622200    GSM1622200 Hodgkin Lymphoma                      Control
GSM1622202    GSM1622202 Hodgkin Lymphoma                      Control
GSM1622204    GSM1622204 Hodgkin Lymphoma                      Control
GSM1622206    GSM1622206 Hodgkin Lymphoma 25nM of Ixazomib for 24 Hours
GSM1622209    GSM1622209 Hodgkin Lymphoma 25nM of Ixazomib for 24 Hours
GSM1622211    GSM1622211 Hodgkin Lymphoma 25nM of Ixazomib for 24 Hours
```

In this experiment, two coviariates are modified: cell type and treatment!

Key Points

- GEO metadata can be cast into R data objects for analysis. The details are up to the user.
- Using proper phenoData to describe an experiment helps to ensure reproducibility and avoid reading in files out of order

‹ (../03-Supplemental-Affy/index.html)

› (../05-DataN

‹ (../04-MetaData/index.html)

Introduction to gene expression microarray analysis in R and Bioconductor (../)

(../05 break ›

Microarray Data processing with RMA

Overview

Teaching: 20 min
Exercises: 20 min
Learning Objectives

- Understand and explain Background correct, normalise and summarisation steps for microarray data

The processes of RMA

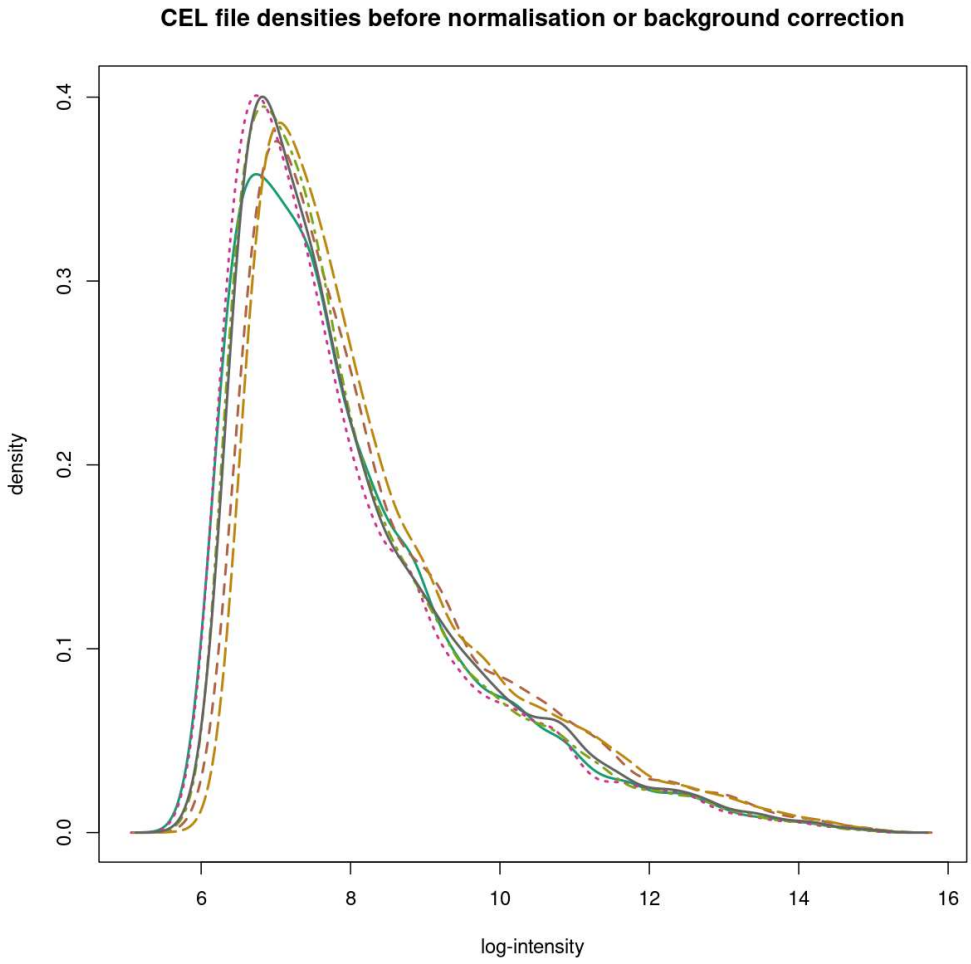
The RMA algorithm performs steps of data processing discussed in class.

1. Background correction.
2. Normalisation.
3. Summarisation (calculating feature-level data)

These steps are performed in order, and yield a processed data set size that is considerably smallar than the data set prior to processing. While the starting object is an AffyBatch object, the result is an ExpressionSet. `

Plotting the signal densities from raw CEL files

The `hist()` functions from the `oligo` package show the signal densities of data from each sample in overlaid line plots.



As you can see, each sample does look like it comes from a similar kind of distribution for all six CEL files, but the exact distributions clearly differ. The prcessing steps of RMA address these issues to provide estimates for each feaature (transcript).

The need for background correction

Microarray data contains ubiquitous background noise, so an important first task is removal of background noise. Different algorithms deal with background differently. Most Affymetrix expression microarrays are designed with *probe pairs*, where one probe is a *perfect match* (PM) to the target and one is a single base *mismatch* (MM) to the target. Methods like Affymetrix's own MAS5 algorithm use the difference between PM and MM hybridisation to addreess background noise. The RMA algorithm ignores the MM probes, and so must address background correction directly as a property of hybridisation.

The RMA model is to model the perfect match (PM) signal as a sum of the real signal for probe j, from probeset k, on array i.

$$\mathrm{PM}_{ijk} = \mathrm{bg}_{ijk} + s_{ijk}$$

The key part of the model is to assume that the true signal is drawn from an exponential distribution

$$s_{ijk} \sim \mathrm{Exp}(\lambda_{ijk})$$

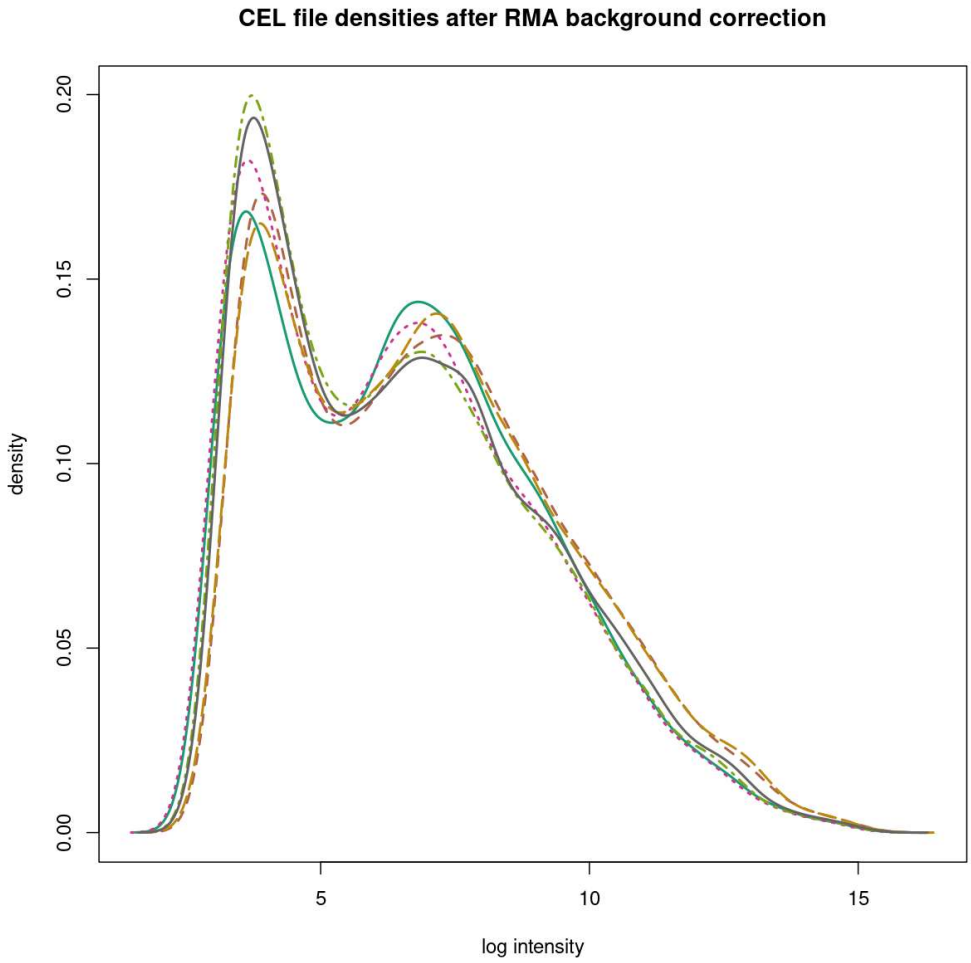
while the background component (both optical noise and non-specific hybridisation) is drawn from a normal distribution that depends only on the array:

$$\mathrm{bg}_{ijk} \sim \mathcal{N}(\beta_i, \sigma_i^2)$$

The background removal step of RMA estimates and removes this second component.

The *gcrma* algorithm differs from *rma* in assuming the background also depends on the GC content of the probes.

```
Background correcting... OK
```



The need for normalisation

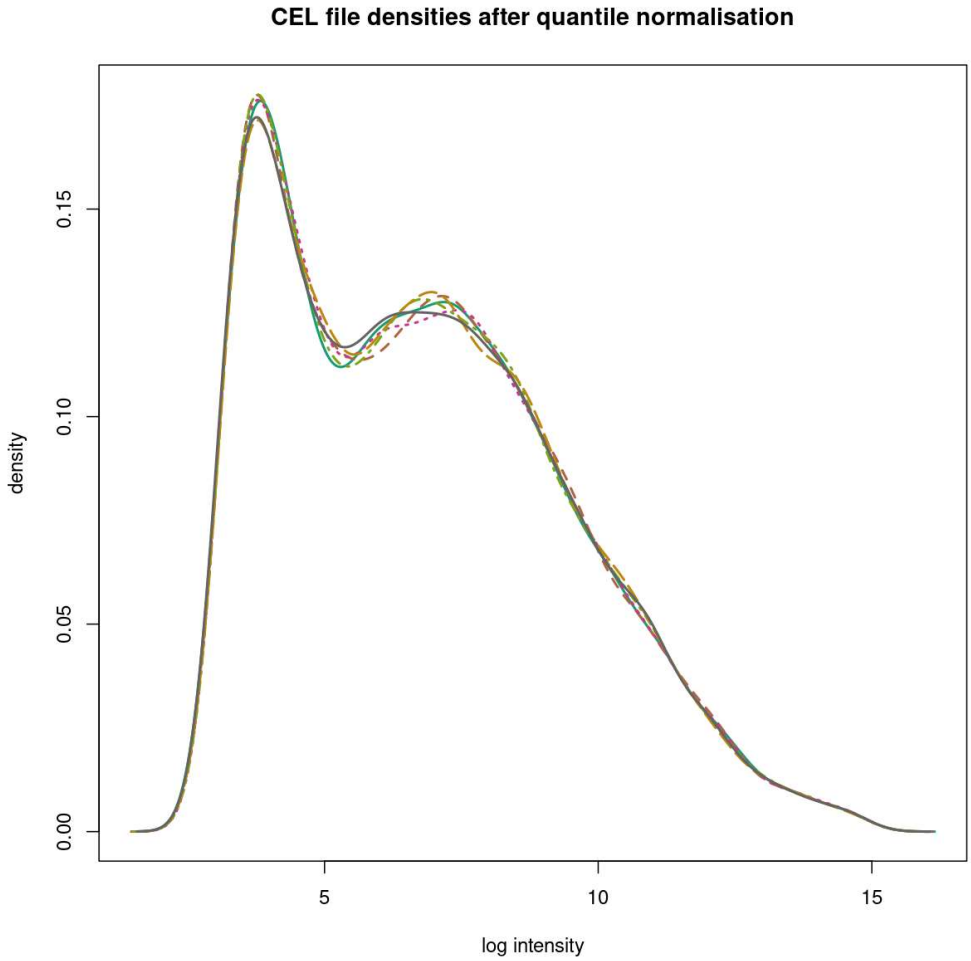
Sample normalisation in RMA is performed via quantile normalisation of the probe level data, as discussed in class. This allows samples to be compared to each other assuming the data arise from thesame parent distribution.

Quantile normalisation puts the data on a common empirical distribution.

```
oligo_normalised <- normalize(raw_nobg,method='quantile',which='pm')
```

```
Normalizing... OK
```

```
hist(oligo_normalised,lwd=2,xlab='log intensity', which='pm',
     main="CEL file densities after quantile normalisation")
```

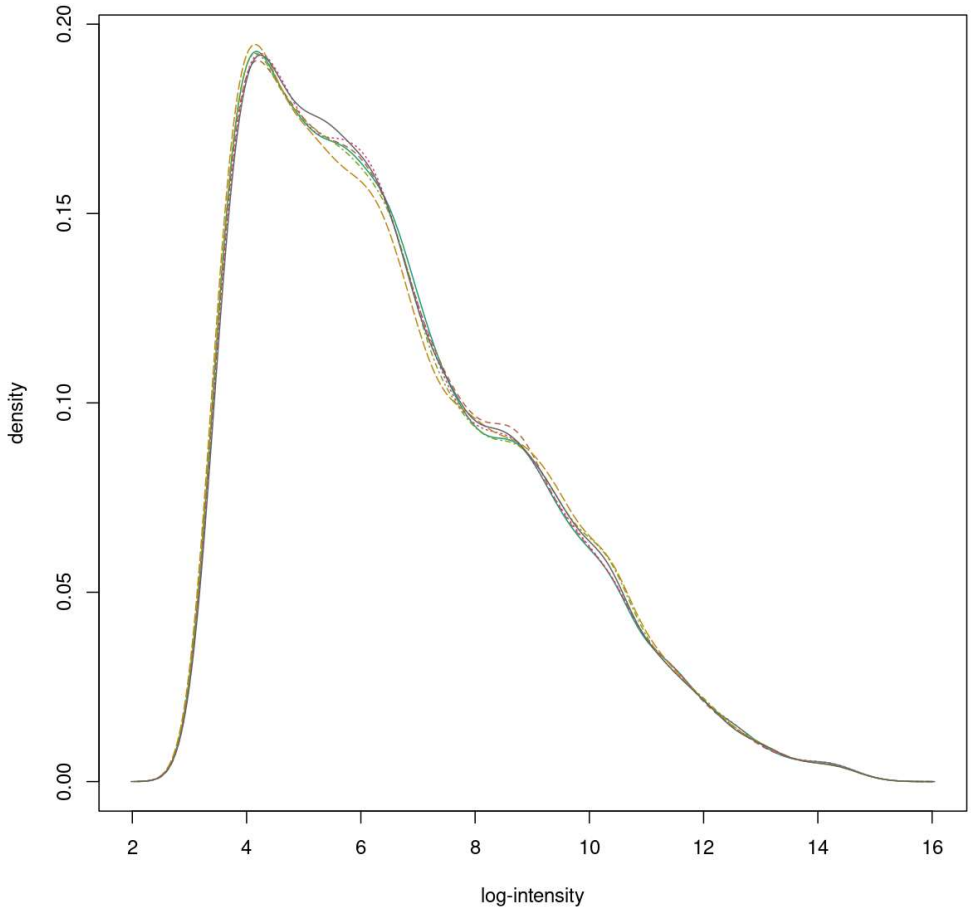


Summarisation: from probe-level to feature-level data.

```
oligo_summarised <- rma(oligo_normalised,background=FALSE,normalize=FALSE)
```

```
Calculating Expression
```

```
hist(oligo_summarised)
```



The summarisation step reduces the size of the data for each sample to the number of measured transcripts (or genes, or exons, depending on the array). These are referred to as “features”, or “probesets”, and the resulting ExpressionSet is “feature-level” data. The summarisation step in rma is performed for each probe-set over all background-corrected, quantile-normalised samples using the median polish algorithm.

Performing RMA in R

Performing RMA in R is relatively straightforward. The function, `rma()` from the `affy` package takes in an **AffyBatch** object. The function `basicRMA()` from the `oligo` package takes in an **ExpressionFeatureSet** object. Hence, the following one line of code performs RMA normalisation, and returns an ExpressionSet object containing the background corrected, normalised, and summarised expression data.

```
gse33146_eset <- rma(gse33146_celdata)
```

```
Background correcting
Normalizing
Calculating Expression
```

```
gse66417_eset <- rma(gse66417_celdata)
```

```
Background correcting
Normalizing
Calculating Expression
```

Try it

If you get help on `rma()` using either `help(rma)` or `?rma`, you can see how to run the same process without background correction or normalisation. **Try it!**

Manipulating an ExpressionSet object

Earlier, we have alluded to the data contained in an **ExpressionSet** object. We can access each of these pieces of data using the following functions:

Data	Type of information	Function
Annotation	Chip information	`annotation()`
PhenoData	Phenotype data	`pData()`
Expression	Normalised gene expression	`exprs()`
Experiment data	Experimental information	`experimentData()`
Feature data	Probeset data	`featureData()`

The phenotype data for each of the samples are retained after rma, so you should be able to see that information

You can also try, running RMA *without* background correction, or without quantile normalisation, and plot the densities.

Key Points

- RMA, the most widely used processing algorithm for Affymetrix data, is implemented in R using the `rma()` function in the `oligo` or `affy` packages, depending on how the data was imported.
- The steps of background correction, quantile normalisation, and summarisation are performed in order to obtain feature-level data

< (../04-MetaData/index.html)

> (../05 break

Using The GTK-teaching/lesson-theme (https://github.com/GTK-teaching/styles/) theme, derived from The Carpentries style (https://github.com/carpentries/styles/)

(../05-break/index.html)

Introduction to gene expression microarray analysis in R and Bioconductor (../)

(../07-factori

Identifying differentially expressed genes using linear models (part 1)

Overview

Teaching: 20 min
Exercises: 20 min
Learning Objectives

- Be able to use `limma` to identify differentially expressed genes.
- Understand the formula class of objects in R, and use it to specify the appropriate model for linear modeling.

What you should have

At this point, you should have two fully processed ExpressionSet objects. We will use the first one (GSE33146) in this lesson.

The formula class of objects

The **formula** class is the work horse of statistical modeling in R. We use `~` in the specification of the model, where `y ~ x` means the response `y` is modelled by a linear variable `x`. More complex models are possible using the operators `+`, `*`, `:`, and others. Refer to the formula manual (https://stat.ethz.ch/R-manual/R-devel/library/stats/html/formula.html) for more information on the different operators and their meaning in a formula. Linear models expressed this way include an implicit 'intercept' term, which we can remove from the model by indicating `+ 0` in our model formula. For a variety of reasons, it's often convenient to remove the intercept term when doing differential gene expression analysis.

Identifying differentially expressed features

In a transcriptomics experiment, whether using microarrays or RNA-seq, we often fit the data from *every* transcript as a response against a common model of independent variables. The variables describe the experiment, and we specify just the right side of a formula (the left side being used to fit the data). For example, if we have a treatment `treat` that can take several values, and the actual value of `treat` varies across samples, we might specify a model by

```
~ treat
```

or possibly

```
~ 0 + treat
```

The second formula explicitly removes the intercept. More on that later.

Specifying our model for differential gene expression analysis

In order to identify differentially expressed genes using linear models, we need to do two things:

1. Generate a model matrix specifying the design, and
2. Fit the model to the design.

In all cases except for the simple two-class comparison, we must also specify a contrast matrix, which we'll get to shortly.

In our first data set, we only have one treatment: the change of culture conditions.

```
library(limma)
design <- model.matrix( ~ gse33146_eset[['culture medium:ch1']])
colnames(design)[2] <- "SCGM"
```

Notice that the design matrix has two columns: one specifiying the *intercept* and one specifying the change of culture conditions.

	(Intercept)	SCGM
	1	0
	1	0
	1	0
	1	1
	1	1
	1	1

The `lmFit()` function of limma fits a linear model for every row of our expression matrix. In the case of a two-class comparison, this is equivalent to a simple t-test.

```
fit <- lmFit(gse33146_eset,design)
```

Empirical Bayes correction in `limma`

Empirical Bayes (eBayes) is a method that borrows information about the distribution across genes to calculate a robust test statistic. In `limma`, this can be performed using the `eBayes()` function. The function requires that we provide an object returned from fitting a linear model (or contrast matrix) to the data. Performing eBayes correction is easily done in R using the following line of code:

```
fitted.ebayes <- eBayes(fit)
```

Extracting differentially expressed genes

We now have, model fits for each feature on the array, and we can arrange them in a table using `topTable()` By default, topTable will provide the top 10 features sorted by the "B" statistic, which is the log odds of differential expression.

```
topTable(fitted.ebayes)
```

```
Removing intercept from test coefficients
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
204268_at	-4.013515	12.416222	-64.54093	0	1e-07	17.05001
203691_at	-4.359692	9.194869	-62.85603	0	1e-07	16.93255
228335_at	5.036631	7.251430	62.22840	0	1e-07	16.88718
226560_at	-3.924272	7.475488	-61.58967	0	1e-07	16.84008
1558846_at	-3.910434	6.392854	-61.34606	0	1e-07	16.82186
201820_at	-5.150011	9.049732	-61.29092	0	1e-07	16.81772
210809_s_at	4.309197	10.827920	59.60159	0	1e-07	16.68715
204304_s_at	-3.724700	7.445306	-59.22172	0	1e-07	16.65680
204971_at	-4.622910	9.764728	-55.37340	0	1e-07	16.32707
206166_s_at	-4.268026	6.384286	-55.22246	0	1e-07	16.31326

See how it removed the intercept? In this case, the “SCGM” coefficient is telling us all that we need to know.

Another good function is `decideTests()`. This function returns which tested features of an array pass the test criteria. By default, this is a (Benjamini-Hochberg) p value of 0.05, and no log fold change cutoff. We can look at a summary with a log fold change cutoff of 1 (meaning 2 fold change in either direction).

```
summary(decideTests(fitted.ebayes[,"SCGM"],lfc=1))
```

```
       SCGM
Down    633
NotSig 53460
Up      582
```

Using contrasts

The intercept is basically useless when doing these sorts of tests, and what we really want is to be comparing experimental groups, right? So can we create a design that reflects the changes in group means? We don’t *have* do do this for only two conditions, but it becomes essential for more complex experimental designs.

To accomplish this we have to specify *contrasts* within our experimental design. Contrasts are the comparisons we wish to make. If `treat` represents two alternative treatments, we simply want to know if the gene expression is different for one treatment versus the other, we calculate the group means for each treatment, and the contrast (the difference) between them.

```
design <- model.matrix( ~ 0 + gse33146_eset[['culture medium:ch1']])
colnames(design) <- levels(as.factor(gse33146_eset[['culture medium:ch1']]))
```

MEGM	SCGM
1	0
1	0
1	0

	MEGM	SCGM
	0	1
	0	1
	0	1

If we remove the intercept, our coefficients now correspond to the conditions. This seems (to me) more natural, but it creates an extra step: deciding the contrasts between groups. In this case it's easy because there are only two groups. In other cases, we'll have more choices.

```
contrast_matrix <- makeContrasts(SCGM - MEGM, levels=design)
contrast_matrix
```

```
      Contrasts
Levels SCGM - MEGM
  MEGM          -1
  SCGM           1
```

In our contrast matrix, we are interested in finding out the difference between the group grown in SCGM (the EMT phenotype) and the control (grown in MEGM). For that reason, we used `SCGM-MEGM`, with the latter being the reference. This formulation means that when the log fold change is *positive*, expression is greater in SCGM than MEGM, and vice versa. Naturally, you can perform more complex analysis with multiple comparisons depending on the question of interest. You can read up more on using *limma* for more complex analysis from the *limma* user guide (available at https://bioconductor.org/packages/release/bioc/vignettes/limma/inst/doc/usersguide.pdf, and in particular, pages 35-64), which demonstrates the use of *limma* for a wide range of questions and even two-colored platforms.

Now we can move ahead to the fit.

```
fit <- lmFit(gse33146_eset,design)
fit2 <- contrasts.fit(fit,contrasts=contrast_matrix)
fit2 <- eBayes(fit2)
summary(decideTests(fit2,lfc=1))
```

```
       SCGM - MEGM
Down           633
NotSig       53460
Up             582
```

Notice we found the *exact same* number of differentially expressed genes, but used the contrasts explicitly in the second case.

The importance of a good model

While the process of fitting a model to the data is not difficult, the difficulty that is most often encountered is the choice of an appropriate model. Many times, there are underlying confounders that are not immediately apparent but have significant impact on the results. One such confounder frequently encountered in microarrays is *batch effect*, which arises when samples are analyzed on different days by different people, leading to the introduction of technical artifacts. For this reason, exploratory data analysis (EDA) is critical to understanding the nature of data prior to model fitting. While outside the scope of this practical, it is a worthwhile investment to find out some of these methods and also how one can correct for these technical differences in a statistically robust manner.

Key Points

- The `formula` class of objects in R enables us to represent a wide range of models to identify differentially expressed genes.

‹ (../05-break/index.html)

› (../07-factori

(../06-DifferentialGeneExpression/index.html)

Introduction to gene expression microarray analysis in R and Bioconductor (../)

(../08-Annota

Identifying differentially expressed genes using linear models (part 2, factorial designs)

Overview

Teaching: 20 min
Exercises: 20 min
Learning Objectives

- Be able to use `limma` to identify differentially expressed genes.
- Understand the formula class of objects in R, and use it to specify the appropriate model for linear modeling.

Experimental designs with more than one covariate

GSE66417 is an example of a *factorial* experimental design, in which two covariates are varied in a single experiment. In this case, the cell type is (Lymphoma or CTL cells) and the treatment is varied (control or Ixozamib). This is called a 2x2 factorial design.

In order to model the expression of each gene, we can model the *group* mean expression under each condition. We have four conditions, so the model is something like

$$Y = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \epsilon$$

where each value β represents a particular condition. We would like to identify genes where othe contrasts represent changes between different groups, i.e., the effect of drug treatment in each cell type. In order to do that, we need to specify contrasts explicitly. One way to do this is to create a “dummy” variable that represents the four groups.

```
library(dplyr)
pd <- pData(gse66417_eset)
pd <- rename(pd,cell_type="cell type:ch1",treatment="treatment:ch1")
pd$treatment <- as.factor(pd$treatment)
levels(pd$treatment) <- c("Ixazomib","Control")
pd$group <- as.factor(paste(pd$cell_type,pd$treatment))
levels(pd$group) <- c("Hodgkins.Control","Hodgkins.Ixazomib","TCL.Control","TCL.Ixazomib")
```

	cell_type	treatment	group
GSM1622170	T-Cell Lymphoma	Control	TCL.Control
GSM1622189	T-Cell Lymphoma	Control	TCL.Control
GSM1622191	T-Cell Lymphoma	Control	TCL.Control
GSM1622194	T-Cell Lymphoma	Ixazomib	TCL.Ixazomib
GSM1622196	T-Cell Lymphoma	Ixazomib	TCL.Ixazomib
GSM1622198	T-Cell Lymphoma	Ixazomib	TCL.Ixazomib
GSM1622200	Hodgkin Lymphoma	Control	Hodgkins.Control
GSM1622202	Hodgkin Lymphoma	Control	Hodgkins.Control

	cell_type	treatment	group
GSM1622204	Hodgkin Lymphoma	Control	Hodgkins.Control
GSM1622206	Hodgkin Lymphoma	Ixazomib	Hodgkins.Ixazomib
GSM1622209	Hodgkin Lymphoma	Ixazomib	Hodgkins.Ixazomib
GSM1622211	Hodgkin Lymphoma	Ixazomib	Hodgkins.Ixazomib

Now we can create a design representing the different groups

```
design <- model.matrix(~ 0 + pd$group)
colnames(design) <- levels(pd$group)
design
```

```
   Hodgkins.Control Hodgkins.Ixazomib TCL.Control TCL.Ixazomib
1                 0                 0           1            0
2                 0                 0           1            0
3                 0                 0           1            0
4                 0                 0           0            1
5                 0                 0           0            1
6                 0                 0           0            1
7                 1                 0           0            0
8                 1                 0           0            0
9                 1                 0           0            0
10                0                 1           0            0
11                0                 1           0            0
12                0                 1           0            0
attr(,"assign")
[1] 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$`pd$group`
[1] "contr.treatment"
```

Our contrasts can be formed by the usual `makeContrasts()` function, but we can easily specify five contrasts that might be interesting.

```
contrasts_matrix <- makeContrasts(drug_in_hodgkins=Hodgkins.Ixazomib - Hodgkins.Control,
              drug_in_TCL=TCL.Ixazomib - TCL.Control,
              cell_in_control=Hodgkins.Control - TCL.Control,
              cell_w_drug=Hodgkins.Ixazomib - TCL.Ixazomib,
              interaction=(Hodgkins.Control - TCL.Control) - (Hodgkins.Ixazomib - TCL.Ixazomib),
              levels=design)
```

```
kable(contrasts_matrix)
```

	drug_in_hodgkins	drug_in_TCL	cell_in_control	cell_w_drug	interaction
Hodgkins.Control	-1	0	1	0	1
Hodgkins.Ixazomib	1	0	0	1	-1
TCL.Control	0	-1	-1	0	-1
TCL.Ixazomib	0	1	0	-1	1

Now we can run the fit as usual

```
gse66417_fit <- lmFit(gse66417_eset,design)
gse66417_fit2 <- contrasts.fit(gse66417_fit,contrasts=contrasts_matrix)
gse66417_fit2 <- eBayes(gse66417_fit2)
summary(decideTests(gse66417_fit2,lfc=1))
```

```
       drug_in_hodgkins drug_in_TCL cell_in_control cell_w_drug interaction
Down                451           2            2330        2769         380
NotSig            52649       53531           49312       48453       52888
Up                  517          84            1975        2395         349
```

Key Points

- The `formula` class of objects in R enables us to represent a wide range of models to identify differentially expressed genes.

‹ (../06-DifferentialGeneExpression/index.html)

› (../08-Annota

(../07-factorial_designs/index.html)

Introduction to gene expression microarray analysis in R and Bioconductor (../)

(../09-downs

From features to annotated gene lists

Overview

Teaching: 15 min
Exercises: 15 min
Learning Objectives

- Be able to use AnnotationDb methods to association annotations with platform data.

Preparation

To make sure we are doing things right, let's get the identifiers for 10 probesets.

Try it: Get a limited number of probesets

Let's work with a limited number of probesets (say, 10) from our differential expression analysis. `topTable()` gives us 10 in a data.frame, so we can easily create a character vector of the top 10 probesets using methods from the last episode.

Solution

```
ps <- rownames(topTable(fitted.ebayes))
```

```
Removing intercept from test coefficients
```

```
ps
```

```
 [1] "204268_at"   "203691_at"   "228335_at"   "226560_at"   "1558846_at"
 [6] "201820_at"   "210809_s_at" "204304_s_at" "204971_at"   "206166_s_at"
```

Annotation of genomic data in AnnotationDb

Using our linear model, we have identified differentially expressed probesets between our two experimental condition. However, the results from `topTable()` only shows the probeset IDs, rather than the gene names. We need to map these IDs to gene symbols, which can then be further analyzed downstream. Fortunately, R has a wide range of *annotation packages* that allows us to do this.

To do so, we will use the annotation package *hgu133plus2.db*, where *hgu133plus2* is the array name. Intuitively, different arrays will have different annotation packages, but they will all end with *.db*.

Different AnnotationDB packages

Besides platform-specific annotation packages, there are also sequence annotation packages (the *BSgenome* packages) as well as UCSC transcript packages (the *UCSC.knownGenes* packages) and the organism annotation packages (the *org*) packages. Feel free to look up the different annotation packages available on BioConductor under the annotations tab.

Let's take a look at what's in the package:

```
library(hgu133plus2.db)
ls('package:hgu133plus2.db')
```

```
 [1] "hgu133plus2"              "hgu133plus2_dbconn"
 [3] "hgu133plus2_dbfile"       "hgu133plus2_dbInfo"
 [5] "hgu133plus2_dbschema"     "hgu133plus2.db"
 [7] "hgu133plus2ACCNUM"        "hgu133plus2ALIAS2PROBE"
 [9] "hgu133plus2CHR"           "hgu133plus2CHRLENGTHS"
[11] "hgu133plus2CHRLOC"        "hgu133plus2CHRLOCEND"
[13] "hgu133plus2ENSEMBL"       "hgu133plus2ENSEMBL2PROBE"
[15] "hgu133plus2ENTREZID"      "hgu133plus2ENZYME"
[17] "hgu133plus2ENZYME2PROBE"  "hgu133plus2GENENAME"
[19] "hgu133plus2GO"            "hgu133plus2GO2ALLPROBES"
[21] "hgu133plus2GO2PROBE"      "hgu133plus2MAP"
[23] "hgu133plus2MAPCOUNTS"     "hgu133plus2OMIM"
[25] "hgu133plus2ORGANISM"      "hgu133plus2ORGPKG"
[27] "hgu133plus2PATH"          "hgu133plus2PATH2PROBE"
[29] "hgu133plus2PFAM"          "hgu133plus2PMID"
[31] "hgu133plus2PMID2PROBE"    "hgu133plus2PROSITE"
[33] "hgu133plus2REFSEQ"        "hgu133plus2SYMBOL"
[35] "hgu133plus2UNIPROT"
```

This version of `ls()` allows us to quickly list the contents of any package. What we see are a bunch of *maps* from the hgu133plus2 probeset identifiers (which we have) to other identifiers (which we don't). We need to provide the probe identifiers of interest and then retrieve the other identifiers of interest.

Method 1: reaching into the databases

Now that we have the probesets in a character vector, let's use what we have. Each of the objects in `ls('package:hgu133plus2')` is an *environment*, which is a special kind of list. The way to extract the values we want is to use the function `mget()` with the probsets as an argument. If we want the result to be a vector, we need to `unlist()` it. Here's an example.

```
## get the symbols for our probesets
unlist(mget(ps,hgu133plus2SYMBOL))
```

```
  204268_at   203691_at   228335_at   226560_at  1558846_at   201820_at
   "S100A2"       "PI3"    "CLDN11"     "SGPP2"  "PNLIPRP3"      "KRT5"
210809_s_at 204304_s_at   204971_at 206166_s_at
    "POSTN"     "PROM1"      "CSTA"     "CLCA2"
```

This method works on all such annotation data packages, but is a bit cumbersome. Very often we want more than one piece of information, and unlisting might be dangerous if the mapping is not one-to-one.

Let's try a better method.

Method 2: The AnnotationDbi interface

A lot of Annotation data packages use a common interface through the AnnotationDbi package (see the vignette (https://bioconductor.org/packages/release/bioc/vignettes/AnnotationDbi/inst/doc/IntroToAnnotationPackages.pdf)). There are four key functions:

`select()`

run a query for selected columns with selected keys

`columns()`
identify what columns are available for a database

`keytypes()`
some, but occasionally not all, of the columns can be used as keys for a query

`keys ()`
list the keys

Let's first look at the available columns for our chip

```
columns(hgu133plus2.db)
```

```
 [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
 [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
[11] "GENETYPE"     "GO"           "GOALL"        "IPI"          "MAP"
[16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
[21] "PMID"         "PROBEID"      "PROSITE"      "REFSEQ"       "SYMBOL"
[26] "UCSCKG"       "UNIPROT"
```

We have a lot of columns to choose from. How about which can be used as keys

```
keytypes(hgu133plus2.db)
```

```
 [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
 [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
[11] "GENETYPE"     "GO"           "GOALL"        "IPI"          "MAP"
[16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
[21] "PMID"         "PROBEID"      "PROSITE"      "REFSEQ"       "SYMBOL"
[26] "UCSCKG"       "UNIPROT"
```

It looks like all the columns can be used as keys. One of the key types is "PROBEID". That looks right.

```
head(keys(hgu133plus2.db,keytype="PROBEID"))
```

```
[1] "1007_s_at" "1053_at"   "117_at"    "121_at"    "1255_g_at" "1294_at"
```

If we want to extract the symbols, gene identifiers, and gene names, it's as simple as using the `select()` function from AnnotationDbi with the probesets as our identifiers:

```
AnnotationDbi::select(hgu133plus2.db,ps,c("SYMBOL","ENTREZID","GENENAME"),keytype="PROBEID")
```

```
'select()' returned 1:1 mapping between keys and columns
```

```
       PROBEID   SYMBOL ENTREZID                             GENENAME
1    204268_at   S100A2     6273       S100 calcium binding protein A2
2    203691_at      PI3     5266                peptidase inhibitor 3
3    228335_at   CLDN11     5010                          claudin 11
4    226560_at    SGPP2   130367 sphingosine-1-phosphate phosphatase 2
5   1558846_at PNLIPRP3   119548   pancreatic lipase related protein 3
6    201820_at     KRT5     3852                            keratin 5
7  210809_s_at    POSTN    10631                           periostin
8  204304_s_at    PROM1     8842                           prominin 1
9    204971_at     CSTA     1475                           cystatin A
10 206166_s_at    CLCA2     9635          chloride channel accessory 2
```

Try it!

Using the given information, use `topTable()` to retrieve all genes that are differentially expressed with a adjusted p-value of less than 0.05, with at fold change of at least two (log fold change at least one). Restrict yourself to *upregulated* genes.

Solution

```
ps2 <- topTable(fitted.ebayes,number=Inf,p.value = 0.05,lfc=1)
```

```
Removing intercept from test coefficients
```

```
ps2_up <- rownames(ps2[ps2$logFC > 0,])
df <- AnnotationDbi::select(hgu133plus2.db,ps2_up,c("SYMBOL","ENTREZID","GENENAME"),keytype="PROBEID")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
dplyr::mutate(df,GENENAME=stringr::str_trunc(GENENAME,30))
```

	PROBEID	SYMBOL	ENTREZID	GENENAME
1	228335_at	CLDN11	5010	claudin 11
2	210809_s_at	POSTN	10631	periostin
3	211959_at	IGFBP5	3488	insulin like growth factor ...
4	223618_at	FMN2	56776	formin 2
5	210143_at	ANXA10	11199	annexin A10
6	203083_at	THBS2	7058	thrombospondin 2
7	227919_at	UCA1	652995	urothelial cancer associated 1
8	235521_at	HOXA3	3200	homeobox A3
9	213069_at	HEG1	57493	heart development protein w...
10	210095_s_at	IGFBP3	3486	insulin like growth factor ...
11	1555778_a_at	POSTN	10631	periostin
12	227450_at	ERP27	121506	endoplasmic reticulum prote...
13	203504_s_at	ABCA1	19	ATP binding cassette subfam...
14	229824_at	SHC3	53358	SHC adaptor protein 3
15	201744_s_at	LUM	4060	lumican
16	203440_at	CDH2	1000	cadherin 2
17	203854_at	CFI	3426	complement factor I
18	242005_at	<NA>	<NA>	<NA>
19	215729_s_at	VGLL1	51442	vestigial like family member 1
20	202481_at	DHRS3	9249	dehydrogenase/reductase 3
21	203304_at	BAMBI	25805	BMP and activin membrane bo...
22	202363_at	SPOCK1	6695	SPARC (osteonectin), cwcv a...
23	225202_at	RHOBTB3	22836	Rho related BTB domain cont...
24	1555471_a_at	FMN2	56776	formin 2
25	200665_s_at	SPARC	6678	secreted protein acidic and...
26	227396_at	PTPRJ	5795	protein tyrosine phosphatas...
27	223204_at	GASK1B	51313	golgi associated kinase 1B
28	203108_at	GPRC5A	9052	G protein-coupled receptor ...
29	206336_at	CXCL6	6372	C-X-C motif chemokine ligand 6
30	204422_s_at	FGF2	2247	fibroblast growth factor 2
31	223557_s_at	TMEFF2	23671	transmembrane protein with ...
32	231579_s_at	TIMP2	7077	TIMP metallopeptidase inhib...
33	214954_at	SUSD5	26032	sushi domain containing 5
34	226279_at	PRSS23	11098	serine protease 23
35	236277_at	<NA>	<NA>	<NA>
36	223484_at	C15orf48	84419	chromosome 15 open reading ...
37	203874_s_at	SMARCA1	6594	SWI/SNF related, matrix ass...
38	224480_s_at	GPAT3	84803	glycerol-3-phosphate acyltr...
39	213258_at	TFPI	7035	tissue factor pathway inhib...
40	218435_at	DNAJC15	29103	DnaJ heat shock protein fam...
41	202016_at	MEST	4232	mesoderm specific transcript
42	210762_s_at	DLC1	10395	DLC1 Rho GTPase activating ...
43	226905_at	RFLNB	359845	refilin B
44	231240_at	DIO2	1734	iodothyronine deiodinase 2
45	203505_at	ABCA1	19	ATP binding cassette subfam...
46	202838_at	FUCA1	2517	alpha-L-fucosidase 1
47	201288_at	ARHGDIB	397	Rho GDP dissociation inhibi...
48	221024_s_at	SLC2A10	81031	solute carrier family 2 mem...
49	203180_at	ALDH1A3	220	aldehyde dehydrogenase 1 fa...
50	234994_at	TMEM200A	114801	transmembrane protein 200A
51	212328_at	LIMCH1	22998	LIM and calponin homology d...
52	229242_at	TNFSF15	9966	TNF superfamily member 15
53	212667_at	SPARC	6678	secreted protein acidic and...
54	218332_at	BEX1	55859	brain expressed X-linked 1
55	206392_s_at	RARRES1	5918	retinoic acid receptor resp...
56	212143_s_at	IGFBP3	3486	insulin like growth factor ...
57	230869_at	FAM155A	728215	family with sequence simila...
58	1562484_at	MEIOC	284071	meiosis specific with coile...
59	208510_s_at	PPARG	5468	peroxisome proliferator act...
60	209291_at	ID4	3400	inhibitor of DNA binding 4,...
61	238689_at	ADGRF1	266977	adhesion G protein-coupled ...
62	202391_at	BASP1	10409	brain abundant membrane att...
63	203895_at	PLCB4	5332	phospholipase C beta 4
64	212325_at	LIMCH1	22998	LIM and calponin homology d...
65	225645_at	EHF	26298	ETS homologous factor
66	201565_s_at	ID2	3398	inhibitor of DNA binding 2
67	206391_at	RARRES1	5918	retinoic acid receptor resp...
68	215294_s_at	SMARCA1	6594	SWI/SNF related, matrix ass...
69	225817_at	CGNL1	84952	cingulin like 1
70	201566_x_at	ID2	3398	inhibitor of DNA binding 2
71	207826_s_at	ID3	3399	inhibitor of DNA binding 3,...
72	221085_at	TNFSF15	9966	TNF superfamily member 15
73	226751_at	CNRIP1	25927	cannabinoid receptor intera...
74	232231_at	RUNX2	860	RUNX family transcription f...
75	223315_at	NTN4	59277	netrin 4
76	203851_at	IGFBP6	3489	insulin like growth factor ...
77	204421_s_at	FGF2	2247	fibroblast growth factor 2
78	212327_at	LIMCH1	22998	LIM and calponin homology d...
79	202957_at	HCLS1	3059	hematopoietic cell-specific...
80	206382_s_at	BDNF	627	brain derived neurotrophic ...

81	209581_at	PLAAT3	11145	phospholipase A and acyltra...
82	203603_s_at	ZEB2	9839	zinc finger E-box binding h...
83	204159_at	CDKN2C	1031	cyclin dependent kinase inh...
84	209121_x_at	NR2F2	7026	nuclear receptor subfamily ...
85	204341_at	TRIM16	10626	tripartite motif containing 16
86	204070_at	PLAAT4	5920	phospholipase A and acyltra...
87	1566766_a_at	MACC1	346389	MET transcriptional regulat...
88	1555812_a_at	ARHGDIB	397	Rho GDP dissociation inhibi...
89	219014_at	PLAC8	51316	placenta associated 8
90	225673_at	MYADM	91663	myeloid associated differen...
91	212148_at	PBX1	5087	PBX homeobox 1
92	205923_at	RELN	5649	reelin
93	207480_s_at	MEIS2	4212	Meis homeobox 2
94	201859_at	SRGN	5552	serglycin
95	213170_at	GPX7	2882	glutathione peroxidase 7
96	243366_s_at	ITGA4	3676	integrin subunit alpha 4
97	212444_at	GPRC5A	9052	G protein-coupled receptor ...
98	235004_at	RBM24	221662	RNA binding motif protein 24
99	215073_s_at	NR2F2	7026	nuclear receptor subfamily ...
100	212233_at	MAP1B	4131	microtubule associated prot...
101	206307_s_at	FOXD1	2297	forkhead box D1
102	220468_at	ARL14	80117	ADP ribosylation factor lik...
103	213338_at	TMEM158	25907	transmembrane protein 158
104	202524_s_at	SPOCK2	9806	SPARC (osteonectin), cwcv a...
105	206595_at	CST6	1474	cystatin E/M
106	223103_at	STARD10	10809	StAR related lipid transfer...
107	202458_at	PRSS23	11098	serine protease 23
108	202976_s_at	RHOBTB3	22836	Rho related BTB domain cont...
109	205487_s_at	VGLL1	51442	vestigial like family member 1
110	202732_at	PKIG	11142	cAMP-dependent protein kina...
111	230250_at	PTPRB	5787	protein tyrosine phosphatas...
112	226622_at	MUC20	200958	mucin 20, cell surface asso...
113	204086_at	PRAME	23532	PRAME nuclear receptor tran...
114	228640_at	PCDH7	5099	protocadherin 7
115	213620_s_at	ICAM2	3384	intercellular adhesion mole...
116	228573_at	ANTXR2	118429	ANTXR cell adhesion molecule 2
117	204256_at	ELOVL6	79071	ELOVL fatty acid elongase 6
118	223044_at	SLC40A1	30061	solute carrier family 40 me...
119	203700_s_at	DIO2	1734	iodothyronine deiodinase 2
120	232235_at	DSEL	92126	dermatan sulfate epimerase ...
121	201042_at	TGM2	7052	transglutaminase 2
122	203167_at	TIMP2	7077	TIMP metallopeptidase inhib...
123	201939_at	PLK2	10769	polo like kinase 2
124	227123_at	RAB3B	5865	RAB3B, member RAS oncogene ...
125	209035_at	MDK	4192	midkine
126	204932_at	TNFRSF11B	4982	TNF receptor superfamily me...
127	227764_at	LYPD6	130574	LY6/PLAUR domain containing 6
128	215446_s_at	LOX	4015	lysyl oxidase
129	203828_s_at	IL32	9235	interleukin 32
130	202202_s_at	LAMA4	3910	laminin subunit alpha 4
131	205422_s_at	ITGBL1	9358	integrin subunit beta like 1
132	205844_at	VNN1	8876	vanin 1
133	231849_at	KRT80	144501	keratin 80
134	203510_at	MET	4233	MET proto-oncogene, recepto...
135	203875_at	SMARCA1	6594	SWI/SNF related, matrix ass...
136	215617_at	SPATS2L	26010	spermatogenesis associated ...
137	203767_s_at	STS	412	steroid sulfatase
138	232676_x_at	MYEF2	50804	myelin expression factor 2
139	227811_at	FGD3	89846	FYVE, RhoGEF and PH domain ...
140	212096_s_at	MTUS1	57509	microtubule associated scaf...
141	213416_at	ITGA4	3676	integrin subunit alpha 4
142	209470_s_at	GPM6A	2823	glycoprotein M6A
143	224560_at	TIMP2	7077	TIMP metallopeptidase inhib...
144	225061_at	DNAJA4	55466	DnaJ heat shock protein fam...
145	219179_at	DACT1	51339	dishevelled binding antagon...
146	212764_at	ZEB1	6935	zinc finger E-box binding h...
147	210220_at	FZD2	2535	frizzled class receptor 2
148	219937_at	TRHDE	29953	thyrotropin releasing hormo...
149	203896_s_at	PLCB4	5332	phospholipase C beta 4
150	213358_at	MTCL1	23255	microtubule crosslinking fa...
151	221872_at	RARRES1	5918	retinoic acid receptor resp...
152	218284_at	SMAD3	4088	SMAD family member 3
153	204149_s_at	GSTM4	2948	glutathione S-transferase mu 4
154	212822_at	HEG1	57493	heart development protein w...
155	212489_at	COL5A1	1289	collagen type V alpha 1 chain
156	201280_s_at	DAB2	1601	DAB adaptor protein 2
157	1555564_a_at	CFI	3426	complement factor I
158	1557779_at	LNCAROD	101928687	lncRNA activating regulator...
159	207463_x_at	PRSS3	5646	serine protease 3
160	216048_s_at	RHOBTB3	22836	Rho related BTB domain cont...
161	227688_at	LRCH2	57631	leucine rich repeats and ca...
162	224352_s_at	CFL2	1073	cofilin 2

163	203697_at	FRZB	2487	frizzled related protein
164	208937_s_at	ID1	3397	inhibitor of DNA binding 1,...
165	204667_at	FOXA1	3169	forkhead box A1
166	213711_at	KRT81	3887	keratin 81
167	201010_s_at	TXNIP	10628	thioredoxin interacting pro...
168	202554_s_at	GSTM3	2947	glutathione S-transferase mu 3
169	208712_at	CCND1	595	cyclin D1
170	214091_s_at	GPX3	2878	glutathione peroxidase 3
171	1556300_s_at	SIM1	6492	SIM bHLH transcription fact...
172	232151_at	MACC1	346389	MET transcriptional regulat...
173	205387_s_at	CGB3	1082	chorionic gonadotropin subu...
174	205387_s_at	CGB5	93659	chorionic gonadotropin subu...
175	205387_s_at	CGB7	94027	chorionic gonadotropin subu...
176	226084_at	MAP1B	4131	microtubule associated prot...
177	243521_at	ZXDA	7789	zinc finger X-linked duplic...
178	203325_s_at	COL5A1	1289	collagen type V alpha 1 chain
179	212509_s_at	MXRA7	439921	matrix remodeling associated 7
180	221577_x_at	GDF15	9518	growth differentiation fact...
181	212386_at	TCF4	6925	transcription factor 4
182	205016_at	TGFA	7039	transforming growth factor ...
183	228642_at	HOTAIRM1	100506311	HOXA transcript antisense R...
184	235256_s_at	GALM	130589	galactose mutarotase
185	226997_at	ADAMTS12	81792	ADAM metallopeptidase with ...
186	213906_at	MYBL1	4603	MYB proto-oncogene like 1
187	221911_at	ETV1	2115	ETS variant transcription f...
188	221031_s_at	APOLD1	81575	apolipoprotein L domain con...
189	226713_at	CCDC50	152137	coiled-coil domain containi...
190	228005_at	ZXDB	158586	zinc finger X-linked duplic...
191	202800_at	SLC1A3	6507	solute carrier family 1 mem...
192	236831_at	CCDC50	152137	coiled-coil domain containi...
193	204036_at	LPAR1	1902	lysophosphatidic acid recep...
194	225524_at	ANTXR2	118429	ANTXR cell adhesion molecule 2
195	212339_at	EPB41L1	2036	erythrocyte membrane protei...
196	225003_at	TMEM205	374882	transmembrane protein 205
197	244353_s_at	SLC2A12	154091	solute carrier family 2 mem...
198	209120_at	NR2F2	7026	nuclear receptor subfamily ...
199	203184_at	FBN2	2201	fibrillin 2
200	209267_s_at	SLC39A8	64116	solute carrier family 39 me...
201	212067_s_at	C1R	715	complement C1r
202	224663_s_at	CFL2	1073	cofilin 2
203	1557051_s_at	HOTAIRM1	100506311	HOXA transcript antisense R...
204	227599_at	MB21D2	151963	Mab-21 domain containing 2
205	238029_s_at	SLC16A14	151473	solute carrier family 16 me...
206	228693_at	CCDC50	152137	coiled-coil domain containi...
207	227966_s_at	CCDC74A	90557	coiled-coil domain containi...
208	227966_s_at	CCDC74B	91409	coiled-coil domain containi...
209	230130_at	SLIT2	9353	slit guidance ligand 2
210	237435_at	<NA>	<NA>	<NA>
211	235591_at	SSTR1	6751	somatostatin receptor 1
212	203632_s_at	GPRC5B	51704	G protein-coupled receptor ...
213	239202_at	RAB3B	5865	RAB3B, member RAS oncogene ...
214	1555852_at	PSMB8-AS1	100507463	PSMB8 antisense RNA 1 (head...
215	235230_at	PLCXD2	257068	phosphatidylinositol specif...
216	203060_s_at	PAPSS2	9060	3'-phosphoadenosine 5'-phos...
217	229269_x_at	SSBP4	170463	single stranded DNA binding...
218	227444_at	ARMCX4	100131755	armadillo repeat containing...
219	205535_s_at	PCDH7	5099	protocadherin 7
220	203939_at	NT5E	4907	5'-nucleotidase ecto
221	229194_at	PCGF5	84333	polycomb group ring finger 5
222	213413_at	STON1	11037	stonin 1
223	204908_s_at	BCL3	602	BCL3 transcription coactivator
224	202179_at	BLMH	642	bleomycin hydrolase
225	203585_at	ZNF185	7739	zinc finger protein 185 wit...
226	222803_at	PRTFDC1	56952	phosphoribosyl transferase ...
227	228737_at	TOX2	84969	TOX high mobility group box...
228	203636_at	MID1	4281	midline 1
229	228107_at	C8orf88	100127983	chromosome 8 open reading f...
230	202664_at	WIPF1	7456	WAS/WASL interacting protei...
231	228098_s_at	MYLIP	29116	myosin regulatory light cha...
232	213400_s_at	TBL1X	6907	transducin beta like 1 X-li...
233	213421_x_at	PRSS3	5646	serine protease 3
234	212423_at	ZCCHC24	219654	zinc finger CCHC-type conta...
235	213150_at	HOXA10	3206	homeobox A10
236	204339_s_at	RGS4	5999	regulator of G protein sign...
237	204994_at	MX2	4600	MX dynamin like GTPase 2
238	204237_at	GULP1	51454	GULP PTB domain containing ...
239	214639_s_at	HOXA1	3198	homeobox A1
240	201348_at	GPX3	2878	glutathione peroxidase 3
241	204337_at	RGS4	5999	regulator of G protein sign...
242	235050_at	SLC2A12	154091	solute carrier family 2 mem...
243	204338_s_at	RGS4	5999	regulator of G protein sign...
244	202388_at	RGS2	5997	regulator of G protein sign...

```
245   1566764_at        MACC1    346389 MET transcriptional regulat...
246    208782_at        FSTL1     11167             follistatin like 1
247  210664_s_at         TFPI      7035 tissue factor pathway inhib...
248    219895_at     TMEM255A     55026     transmembrane protein 255A
249    204748_at        PTGS2      5743 prostaglandin-endoperoxide ...
250  204298_s_at          LOX      4015                  lysyl oxidase
251    227719_at        SMAD9      4093           SMAD family member 9
252    204819_at         FGD1      2245 FYVE, RhoGEF and PH domain ...
253    219902_at        BHMT2     23743 betaine--homocysteine S-met...
254  200999_s_at        CKAP4     10970 cytoskeleton associated pro...
255    226137_at        ZFHX3       463         zinc finger homeobox 3
256    228933_at          NHS      4810 NHS actin remodeling regulator
257  233825_s_at       CD99L2     83692           CD99 molecule like 2
258    227846_at       GPR176     11245 G protein-coupled receptor 176
259    242417_at      PLEKHA7    144100 pleckstrin homology domain ...
260  201058_s_at         MYL9     10398           myosin light chain 9
261  228128_x_at        PAPPA      5069                   pappalysin 1
262    204955_at         SRPX      8406 sushi repeat containing pro...
263  209198_s_at        SYT11     23208               synaptotagmin 11
264    201506_at         TGFBI      7045 transforming growth factor ...
265  209356_x_at       EFEMP2     30008 EGF containing fibulin extr...
266  203441_s_at         CDH2      1000                     cadherin 2
267  223218_s_at       NFKBIZ     64332            NFKB inhibitor zeta
268    205316_at      SLC15A2      6565 solute carrier family 15 me...
269    215263_at         ZXDA      7789 zinc finger X-linked duplic...
270    215263_at         ZXDB    158586 zinc finger X-linked duplic...
271    232239_at    LINC00865    643529 long intergenic non-protein...
272    212531_at         LCN2      3934                   lipocalin 2
273    213844_at        HOXA5      3202                   homeobox A5
274  210065_s_at         UPK1B      7348                  uroplakin 1B
275    227458_at         CD274     29126                CD274 molecule
276    230372_at         HAS2      3037          hyaluronan synthase 2
277    200962_at         RPL31      6160          ribosomal protein L31
278    218573_at       MAGEH1     28986          MAGE family member H1
279    242051_at          <NA>      <NA>                          <NA>
280    226152_at        TTC7B    145567 tetratricopeptide repeat do...
281    229437_at     MIR155HG    114614               MIR155 host gene
282    209676_at         TFPI      7035 tissue factor pathway inhib...
283    205884_at        ITGA4      3676       integrin subunit alpha 4
284    204639_at          ADA       100            adenosine deaminase
285   1554171_at         ZMYM3      9203 zinc finger MYM-type contai...
286    235391_at       CIBAR1    137392 CBY1 interacting BAR domain...
287  205398_s_at         SMAD3      4088           SMAD family member 3
288  234192_s_at         GKAP1     80318   G kinase anchoring protein 1
289  222016_s_at       ZSCAN31     64288 zinc finger and SCAN domain...
290  222942_s_at         TIAM2     26230     TIAM Rac1 associated GEF 2
291    219450_at      C4orf19     55286 chromosome 4 open reading f...
292  210064_s_at         UPK1B      7348                  uroplakin 1B
293    218729_at           LXN     56925                       latexin
294  229461_x_at        NEGR1    257194    neuronal growth regulator 1
295    230788_at        GCNT2      2651 glucosaminyl (N-acetyl) tra...
296    209651_at      TGFB1I1      7041 transforming growth factor ...
297  206116_s_at         TPM1      7168                 tropomyosin 1
298    229354_at   PDCD6-AHRR 116412618 PDCD6-AHRR readthrough (NMD...
299    229354_at         AHRR     57491 aryl-hydrocarbon receptor r...
300    226876_at        RFLNB    359845                     refilin B
301  241484_x_at       FRG1CP 100289097 FSHD region gene 1 family m...
302    226875_at       DOCK11    139818    dedicator of cytokinesis 11
303    238032_at          <NA>      <NA>                          <NA>
304    204279_at        PSMB9      5698  proteasome 20S subunit beta 9
305    224823_at         MYLK      4638      myosin light chain kinase
306    226534_at        KITLG      4254                    KIT ligand
307    242093_at        SYTL5     94122           synaptotagmin like 5
308    224990_at       SMIM14    201895 small integral membrane pro...
309  232322_x_at      STARD10     10809 StAR related lipid transfer...
310  204933_s_at    TNFRSF11B      4982 TNF receptor superfamily me...
311    201510_at         ELF3      1999 E74 like ETS transcription ...
312    205130_at          MOK      5891             MOK protein kinase
313 1557961_s_at      C8orf88 100127983 chromosome 8 open reading f...
314    205259_at        NR3C2      4306 nuclear receptor subfamily ...
315    206432_at         HAS2      3037          hyaluronan synthase 2
316  201009_s_at         TXNIP     10628 thioredoxin interacting pro...
317    219148_at          PBK     55872             PDZ binding kinase
318  229465_s_at          <NA>      <NA>                          <NA>
319    227503_at          <NA>      <NA>                          <NA>
320    205978_at            KL      9365                       klotho
321  210751_s_at          RGN      9104                    regucalcin
322  214519_s_at         RLN2      6019                     relaxin 2
323    225847_at         NCEH1     57552 neutral cholesterol ester h...
324    227449_at         EPHA4      2043               EPH receptor A4
325    222772_at         MYEF2     50804     myelin expression factor 2
326  220979_s_at    ST6GALNAC5     81849 ST6 N-acetylgalactosaminide...
```

```
327    228035_at        STK33     65975     serine/threonine kinase 33
328    235171_at LOC100505501 100505501   uncharacterized LOC100505501
329    238423_at        SYTL3     94120           synaptotagmin like 3
330  215913_s_at        GULP1     51454 GULP PTB domain containing ...
331    209292_at          ID4      3400 inhibitor of DNA binding 4,...
332  203699_s_at         DIO2      1734     iodothyronine deiodinase 2
333  210868_s_at       ELOVL6     79071    ELOVL fatty acid elongase 6
334    206698_at           XK      7504        X-linked Kx blood group
335    226066_at         MITF      4286 melanocyte inducing transcr...
336    210002_at        GATA6      2627         GATA binding protein 6
337    212771_at     FAM171A1    221061 family with sequence simila...
338    227491_at       ELOVL6     79071    ELOVL fatty acid elongase 6
339  209293_x_at          ID4      3400 inhibitor of DNA binding 4,...
340  214651_s_at        HOXA9      3205                    homeobox A9
341  214651_s_at HOXA10-HOXA9 100534589       HOXA10-HOXA9 readthrough
342    204037_at        LPAR1      1902 lysophosphatidic acid recep...
343  224964_s_at         GNG2     54331      G protein subunit gamma 2
344    228450_at      PLEKHA7    144100 pleckstrin homology domain ...
345    226425_at        CLIP4     79745 CAP-Gly domain containing l...
346    237086_at        FOXA1      3169                forkhead box A1
347  205890_s_at          UBD     10537                    ubiquitin D
348  205890_s_at       GABBR1      2550 gamma-aminobutyric acid typ...
349  232079_s_at      NECTIN2      5819 nectin cell adhesion molecu...
350    229092_at        NR2F2      7026 nuclear receptor subfamily ...
351    227341_at        BEND7    222389        BEN domain containing 7
352    212488_at       COL5A1      1289  collagen type V alpha 1 chain
353    238877_at         EYA4      2070 EYA transcriptional coactiv...
354  228850_s_at        SLIT2      9353         slit guidance ligand 2
355  205925_s_at        RAB3B      5865 RAB3B, member RAS oncogene ...
356   1554062_at           XG      7499 Xg glycoprotein (Xg blood g...
357  201150_s_at        TIMP3      7078 TIMP metallopeptidase inhib...
358    207177_at        PTGFR      5737       prostaglandin F receptor
359  236918_s_at       LRRC34    151827 leucine rich repeat contain...
360    227554_at    MAGI2-AS3 100505881          MAGI2 antisense RNA 3
361  205005_s_at         NMT2      9397       N-myristoyltransferase 2
362    201425_at        ALDH2       217 aldehyde dehydrogenase 2 fa...
363  210675_s_at        PTPRR      5801 protein tyrosine phosphatas...
364  227703_s_at        SYTL4     94121           synaptotagmin like 4
365     52837_at      SHISAL1     85352                   shisa like 1
366    226326_at        PCGF5     84333   polycomb group ring finger 5
367  209782_s_at          DBP      1628 D-box binding PAR bZIP tran...
368    227763_at        LYPD6    130574  LY6/PLAUR domain containing 6
369    212636_at          QKI      9444 QKI, KH domain containing R...
370     35626_at         SGSH      6448 N-sulfoglucosamine sulfohyd...
371  222771_s_at        MYEF2     50804     myelin expression factor 2
372    220014_at        PRR16     51334                proline rich 16
373  201008_s_at        TXNIP     10628 thioredoxin interacting pro...
374  226933_s_at          ID4      3400 inhibitor of DNA binding 4,...
375    228608_at        NALCN    259232 sodium leak channel, non-se...
376    201278_at         DAB2      1601          DAB adaptor protein 2
377    212151_at         PBX1      5087                 PBX homeobox 1
378  201137_s_at     HLA-DPB1      3115 major histocompatibility co...
379    204646_at         DPYD      1806 dihydropyrimidine dehydroge...
380    212336_at      EPB41L1      2036 erythrocyte membrane protei...
381    236769_at         <NA>      <NA>                          <NA>
382  203637_s_at         MID1      4281                      midline 1
383    229480_at    MAGI2-AS3 100505881          MAGI2 antisense RNA 3
384    201069_at         MMP2      4313      matrix metallopeptidase 2
385  211458_s_at    GABARAPL3     23766 GABA type A receptor associ...
386  211458_s_at    GABARAPL1     23710 GABA type A receptor associ...
387    235988_at       ADGRF1    266977 adhesion G protein-coupled ...
388  201809_s_at          ENG      2022                       endoglin
389  216014_s_at         ZXDA      7789 zinc finger X-linked duplic...
390  216014_s_at         ZXDB    158586 zinc finger X-linked duplic...
391    205924_at        RAB3B      5865 RAB3B, member RAS oncogene ...
392  204235_s_at        GULP1     51454 GULP PTB domain containing ...
393    214033_at        ABCC6       368 ATP binding cassette subfam...
394    214033_at      ABCC6P1    653190 ATP binding cassette subfam...
395    214033_at      ABCC6P2    730013 ATP binding cassette subfam...
396    224941_at        PAPPA      5069                   pappalysin 1
397    206029_at       ANKRD1     27063        ankyrin repeat domain 1
398  201858_s_at         SRGN      5552                      serglycin
399  206580_s_at       EFEMP2     30008 EGF containing fibulin extr...
400  207836_s_at         RBPMS     11030 RNA binding protein, mRNA p...
401  217388_s_at         KYNU      8942                   kynureninase
402    209710_at        GATA2      2624         GATA binding protein 2
403    203196_at        ABCC4     10257 ATP binding cassette subfam...
404  204529_s_at          TOX      9760 thymocyte selection associa...
405    241789_at        RBMS3     27303 RNA binding motif single st...
406    235759_at         <NA>      <NA>                          <NA>
407 1554334_a_at       DNAJA4     55466 DnaJ heat shock protein fam...
408  202975_s_at      RHOBTB3     22836 Rho related BTB domain cont...
```

409	212093_s_at	MTUS1	57509	microtubule associated scaf...
410	202087_s_at	CTSL	1514	cathepsin L
411	235051_at	CCDC50	152137	coiled-coil domain containi...
412	209897_s_at	SLIT2	9353	slit guidance ligand 2
413	233496_s_at	CFL2	1073	cofilin 2
414	205885_s_at	ITGA4	3676	integrin subunit alpha 4
415	235367_at	MYPN	84665	myopalladin
416	218469_at	GREM1	26585	gremlin 1, DAN family BMP a...
417	1552502_s_at	RHBDL2	54933	rhomboid like 2
418	227486_at	NT5E	4907	5'-nucleotidase ecto
419	238447_at	RBMS3	27303	RNA binding motif single st...
420	205006_s_at	NMT2	9397	N-myristoyltransferase 2
421	231941_s_at	MUC20	200958	mucin 20, cell surface asso...
422	241869_at	APOL6	80830	apolipoprotein L6
423	218885_s_at	GALNT12	79695	polypeptide N-acetylgalacto...
424	231841_s_at	JCAD	57608	junctional cadherin 5 assoc...
425	59697_at	RAB15	376267	RAB15, member RAS oncogene ...
426	1554997_a_at	PTGS2	5743	prostaglandin-endoperoxide ...
427	238983_at	NSUN7	79730	NOP2/Sun RNA methyltransfer...
428	225978_at	RIMKLB	57494	ribosomal modification prot...
429	225242_s_at	CCDC80	151887	coiled-coil domain containi...
430	218468_s_at	GREM1	26585	gremlin 1, DAN family BMP a...
431	231842_at	JCAD	57608	junctional cadherin 5 assoc...
432	226603_at	SAMD9L	219285	sterile alpha motif domain ...
433	203886_s_at	FBLN2	2199	fibulin 2
434	230720_at	RNF182	221687	ring finger protein 182
435	213640_s_at	LOX	4015	lysyl oxidase
436	229441_at	PRSS23	11098	serine protease 23
437	211990_at	HLA-DPA1	3113	major histocompatibility co...
438	214457_at	HOXA2	3199	homeobox A2
439	214825_at	FAM155A	728215	family with sequence simila...
440	202531_at	IRF1	3659	interferon regulatory factor 1
441	223723_at	MELTF	4241	melanotransferrin
442	1553997_a_at	ASPHD1	253982	aspartate beta-hydroxylase ...
443	232270_at	AOPEP	84909	aminopeptidase O (putative)
444	213158_at	ZBTB20	26137	zinc finger and BTB domain ...
445	232361_s_at	EHF	26298	ETS homologous factor
446	209469_at	GPM6A	2823	glycoprotein M6A
447	221232_s_at	ANKRD2	26287	ankyrin repeat domain 2
448	213547_at	CAND2	23066	cullin associated and neddy...
449	214247_s_at	DKK3	27122	dickkopf WNT signaling path...
450	202555_s_at	MYLK	4638	myosin light chain kinase
451	224189_x_at	EHF	26298	ETS homologous factor
452	204726_at	CDH13	1012	cadherin 13
453	204115_at	GNG11	2791	G protein subunit gamma 11
454	210176_at	TLR1	7096	toll like receptor 1
455	218029_at	RIPOR1	79567	RHO family interacting cell...
456	210757_x_at	DAB2	1601	DAB adaptor protein 2
457	235350_at	C4orf19	55286	chromosome 4 open reading f...
458	223253_at	EPDR1	54749	ependymin related 1
459	200998_s_at	CKAP4	10970	cytoskeleton associated pro...
460	206291_at	NTS	4922	neurotensin
461	214078_at	PAK3	5063	p21 (RAC1) activated kinase 3
462	211003_x_at	TGM2	7052	transglutaminase 2
463	201279_s_at	DAB2	1601	DAB adaptor protein 2
464	218613_at	PSD3	23362	pleckstrin and Sec7 domain ...
465	204284_at	PPP1R3C	5507	protein phosphatase 1 regul...
466	209159_s_at	NDRG4	65009	NDRG family member 4
467	218893_at	ISOC2	79763	isochorismatase domain cont...
468	219944_at	CLIP4	79745	CAP-Gly domain containing l...
469	204198_s_at	RUNX3	864	RUNX family transcription f...
470	203058_s_at	PAPSS2	9060	3'-phosphoadenosine 5'-phos...
471	232825_s_at	DSEL	92126	dermatan sulfate epimerase ...
472	226103_at	NEXN	91624	nexilin F-actin binding pro...
473	243041_s_at	<NA>	<NA>	<NA>
474	1555097_a_at	PTGFR	5737	prostaglandin F receptor
475	1568838_at	WASIR2	100132169	WASH and IL9R antisense RNA 2
476	224940_s_at	PAPPA	5069	pappalysin 1
477	235570_at	RBMS3	27303	RNA binding motif single st...
478	201482_at	QSOX1	5768	quiescin sulfhydryl oxidase 1
479	203698_s_at	FRZB	2487	frizzled related protein
480	223614_at	MMP16	4325	matrix metallopeptidase 16
481	211573_x_at	TGM2	7052	transglutaminase 2
482	201147_s_at	TIMP3	7078	TIMP metallopeptidase inhib...
483	202724_s_at	FOXO1	2308	forkhead box O1
484	206600_s_at	SLC16A5	9121	solute carrier family 16 me...
485	226771_at	ATP8B2	57198	ATPase phospholipid transpo...
486	225562_at	RASA3	22821	RAS p21 protein activator 3
487	211599_x_at	MET	4233	MET proto-oncogene, recepto...
488	228360_at	LYPD6B	130576	LY6/PLAUR domain containing 6B
489	1559506_x_at	MIR137HG	400765	MIR137 host gene
490	1559506_x_at	MIR2682	100616452	microRNA 2682

491	211958_at	IGFBP5	3488	insulin like growth factor ...
492	209589_s_at	EPHB2	2048	EPH receptor B2
493	227808_at	DNAJC15	29103	DnaJ heat shock protein fam...
494	228948_at	EPHA4	2043	EPH receptor A4
495	236369_at	<NA>	<NA>	<NA>
496	205872_x_at	PDE4DIP	9659	phosphodiesterase 4D intera...
497	236523_at	C4orf54	285556	chromosome 4 open reading f...
498	223878_at	INPP4B	8821	inositol polyphosphate-4-ph...
499	219855_at	NUDT11	55190	nudix hydrolase 11
500	212419_at	ZCCHC24	219654	zinc finger CCHC-type conta...
501	236798_at	LINC00888	100505687	long intergenic non-protein...
502	229842_at	ELF3	1999	E74 like ETS transcription ...
503	213156_at	ZBTB20	26137	zinc finger and BTB domain ...
504	207069_s_at	SMAD6	4091	SMAD family member 6
505	209723_at	SERPINB9	5272	serpin family B member 9
506	207818_s_at	HTR7	3363	5-hydroxytryptamine receptor 7
507	1558048_x_at	<NA>	<NA>	<NA>
508	231993_at	ITGBL1	9358	integrin subunit beta like 1
509	206828_at	TXK	7294	TXK tyrosine kinase
510	206876_at	SIM1	6492	SIM bHLH transcription fact...
511	236281_x_at	HTR7	3363	5-hydroxytryptamine receptor 7
512	213325_at	NECTIN3	25945	nectin cell adhesion molecu...
513	222773_s_at	GALNT12	79695	polypeptide N-acetylgalacto...
514	242396_at	NR2F2	7026	nuclear receptor subfamily ...
515	210612_s_at	SYNJ2	8871	synaptojanin 2
516	213712_at	ELOVL2	54898	ELOVL fatty acid elongase 2
517	229372_at	GOLT1A	127845	golgi transport 1A
518	231248_at	CST6	1474	cystatin E/M
519	216098_s_at	HTR7P1	93164	5-hydroxytryptamine recepto...
520	216098_s_at	HTR7	3363	5-hydroxytryptamine receptor 7
521	226592_at	ZNF618	114991	zinc finger protein 618
522	213807_x_at	MET	4233	MET proto-oncogene, recepto...
523	229464_at	MYEF2	50804	myelin expression factor 2
524	228481_at	<NA>	<NA>	<NA>
525	1557050_at	<NA>	<NA>	<NA>
526	1554333_at	DNAJA4	55466	DnaJ heat shock protein fam...
527	219850_s_at	EHF	26298	ETS homologous factor
528	214927_at	ITGBL1	9358	integrin subunit beta like 1
529	235236_at	INSYN2B	100131897	inhibitory synaptic factor ...
530	204035_at	SCG2	7857	secretogranin II
531	230183_at	EXT1	2131	exostosin glycosyltransfera...
532	222168_at	<NA>	<NA>	<NA>
533	214500_at	MACROH2A1	9555	macroH2A.1 histone
534	209119_x_at	NR2F2	7026	nuclear receptor subfamily ...
535	240407_at	NAV2-AS6	100126784	NAV2 antisense RNA 6
536	205421_at	SLC22A3	6581	solute carrier family 22 me...
537	235122_at	HIVEP3	59269	HIVEP zinc finger 3
538	212095_s_at	MTUS1	57509	microtubule associated scaf...
539	209905_at	HOXA9	3205	homeobox A9
540	209905_at	HOXA10-HOXA9	100534589	HOXA10-HOXA9 readthrough
541	205466_s_at	HS3ST1	9957	heparan sulfate-glucosamine...
542	236917_at	LRRC34	151827	leucine rich repeat contain...
543	225355_at	NEURL1B	54492	neuralized E3 ubiquitin pro...
544	224989_at	SMIM14	201895	small integral membrane pro...
545	223217_s_at	NFKBIZ	64332	NFKB inhibitor zeta
546	223226_x_at	SSBP4	170463	single stranded DNA binding...
547	230329_s_at	NUDT6	11162	nudix hydrolase 6
548	235643_at	SAMD9L	219285	sterile alpha motif domain ...
549	1557080_s_at	ITGBL1	9358	integrin subunit beta like 1
550	230831_at	FRMD5	84978	FERM domain containing 5
551	206805_at	SEMA3A	10371	semaphorin 3A
552	208711_s_at	CCND1	595	cyclin D1
553	210875_s_at	ZEB1	6935	zinc finger E-box binding h...
554	229435_at	GLIS3	169792	GLIS family zinc finger 3
555	206941_x_at	SEMA3E	9723	semaphorin 3E
556	1559361_at	MACC1	346389	MET transcriptional regulat...
557	1559361_at	LOC101927668	101927668	uncharacterized LOC101927668
558	219451_at	MSRB2	22921	methionine sulfoxide reduct...
559	1557236_at	APOL6	80830	apolipoprotein L6
560	1552309_a_at	NEXN	91624	nexilin F-actin binding pro...
561	224920_x_at	MYADM	91663	myeloid associated differen...
562	225575_at	LIFR	3977	LIF receptor subunit alpha
563	236029_at	FAT3	120114	FAT atypical cadherin 3
564	206330_s_at	SHC3	53358	SHC adaptor protein 3
565	206290_s_at	RGS7	6000	regulator of G protein sign...
566	236489_at	ADGRF1	266977	adhesion G protein-coupled ...
567	209750_at	NR1D2	9975	nuclear receptor subfamily ...
568	235331_x_at	PCGF5	84333	polycomb group ring finger 5
569	224822_at	DLC1	10395	DLC1 Rho GTPase activating ...
570	244852_at	DSEL	92126	dermatan sulfate epimerase ...
571	222932_at	EHF	26298	ETS homologous factor
572	213590_at	SLC16A5	9121	solute carrier family 16 me...

```
573    227053_at      PACSIN1     29993 protein kinase C and casein...
574  214647_s_at          HFE      3077     homeostatic iron regulator
575    242629_at        RAB3B      5865 RAB3B, member RAS oncogene ...
576    230036_at       SAMD9L    219285 sterile alpha motif domain ...
577    235030_at        NXPE3     91775 neurexophilin and PC-estera...
578  201867_s_at         TBL1X      6907 transducin beta like 1 X-li...
579    204749_at        NAP1L3      4675 nucleosome assembly protein...
580  213816_s_at          MET      4233 MET proto-oncogene, recepto...
581    235563_at       GPRC5A      9052 G protein-coupled receptor ...
582    230003_at      SLC16A7      9194 solute carrier family 16 me...
583    229963_at         BEX5    340542     brain expressed X-linked 5
584    206084_at        PTPRR      5801 protein tyrosine phosphatas...
585    210173_at        PTPRJ      5795 protein tyrosine phosphatas...
586    228697_at        HINT3    135114 histidine triad nucleotide ...
587    238520_at       TRERF1     55809 transcriptional regulating ...
588    207029_at         KITLG      4254                     KIT ligand
589  215506_s_at       DIRAS3      9077          DIRAS family GTPase 3
590    211020_at        GCNT2      2651 glucosaminyl (N-acetyl) tra...
591  219557_s_at        NRIP3     56675 nuclear receptor interactin...
592  203355_s_at         PSD3     23362 pleckstrin and Sec7 domain ...
593  224496_s_at      TMEM107     84314      transmembrane protein 107
594 1555229_a_at          C1S       716                 complement C1s
595    226533_at        HINT3    135114 histidine triad nucleotide ...
596  211124_s_at         KITLG      4254                     KIT ligand
597    242134_at          <NA>       <NA>                          <NA>
```

Key Points

- BioConductor has a rich annotation infrastructure, with different data type being stored in different annotation packages.
- The `select()` function allows us to efficiently query annotation databases.
- Using `topTable()` in conjunction with `rownames()` allows us to retrieve all the probes which are differentially expressed between our experimental conditions.

‹ (../07-factorial_designs/index.html)

› (../09-downs

(../08-AnnotatingResults/index.html) Introduction to gene expression microarray analysis in R and Bioconductor (../) (../)

Basic downstream analysis of microarray data

Overview

Teaching: 10 min
Exercises: 10 min
Learning Objectives

- Be able to plot volcano plots and heatmaps in R.
- Be able to interpret the above plots generated.
- Be aware of some downstream analysis that are commonly done to interpret the results of differential expression analysis.

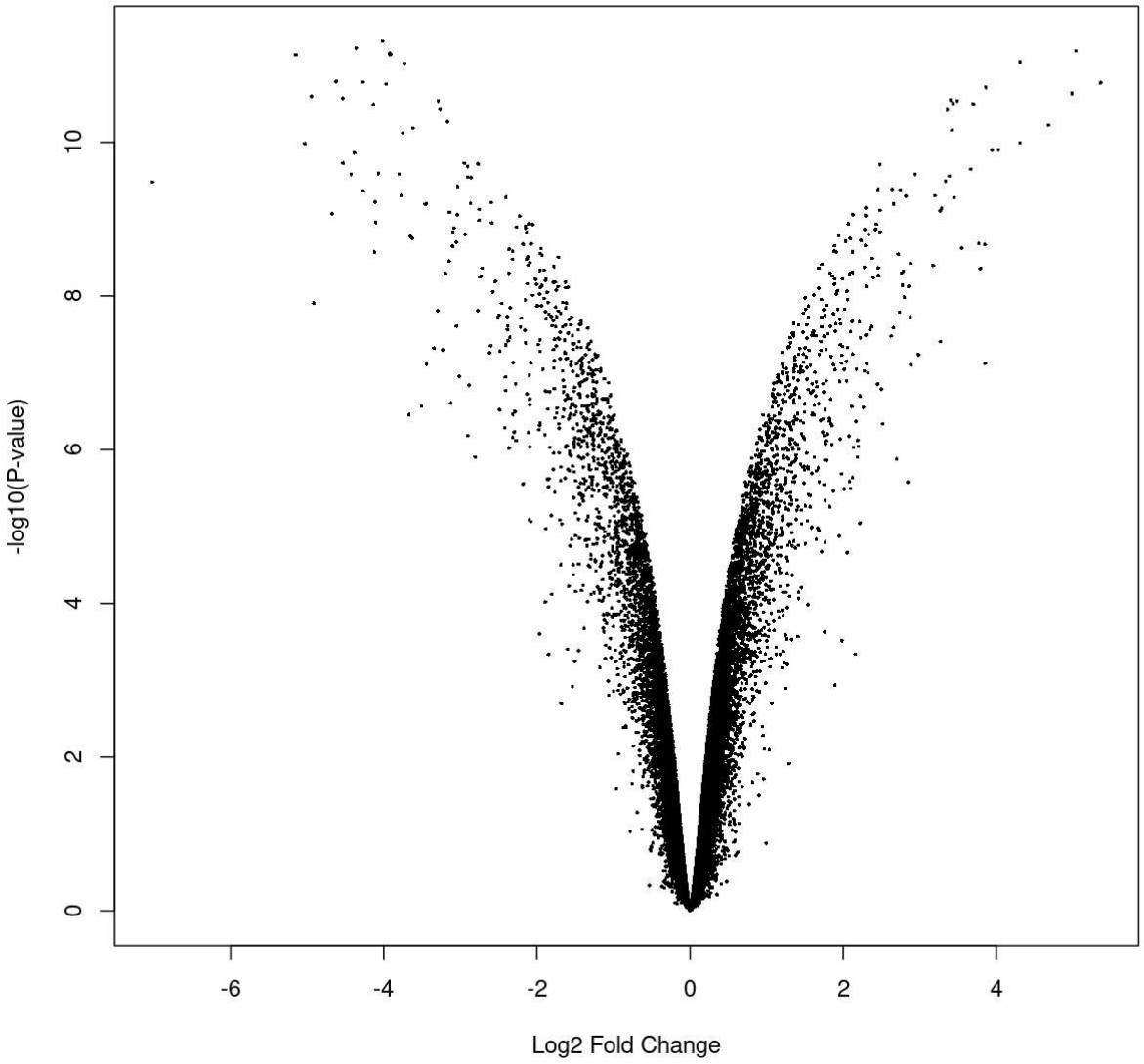
Downstream analysis of microarray data

So far in our analysis, we have obtained a list of differentially expressed genes. However, having a list of genes is not usually the most informative. Here, we will discuss some plots that are commonly used to visualize the results of differential gene expression analysis. We will also discuss (but not go through their implementation) some commonly employed strategies to interpret the results of differential gene expression analysis.

Volcano plots

A volcano plot is a helpful visualization that shows the log fold-change versus the negative log p-value. Usually, we will expect that genes larger absolute fold changes will have larger negative log p-values and hence implying greater statistical confidence.

```
volcanoplot(fitted.ebayes,coef=2)
```

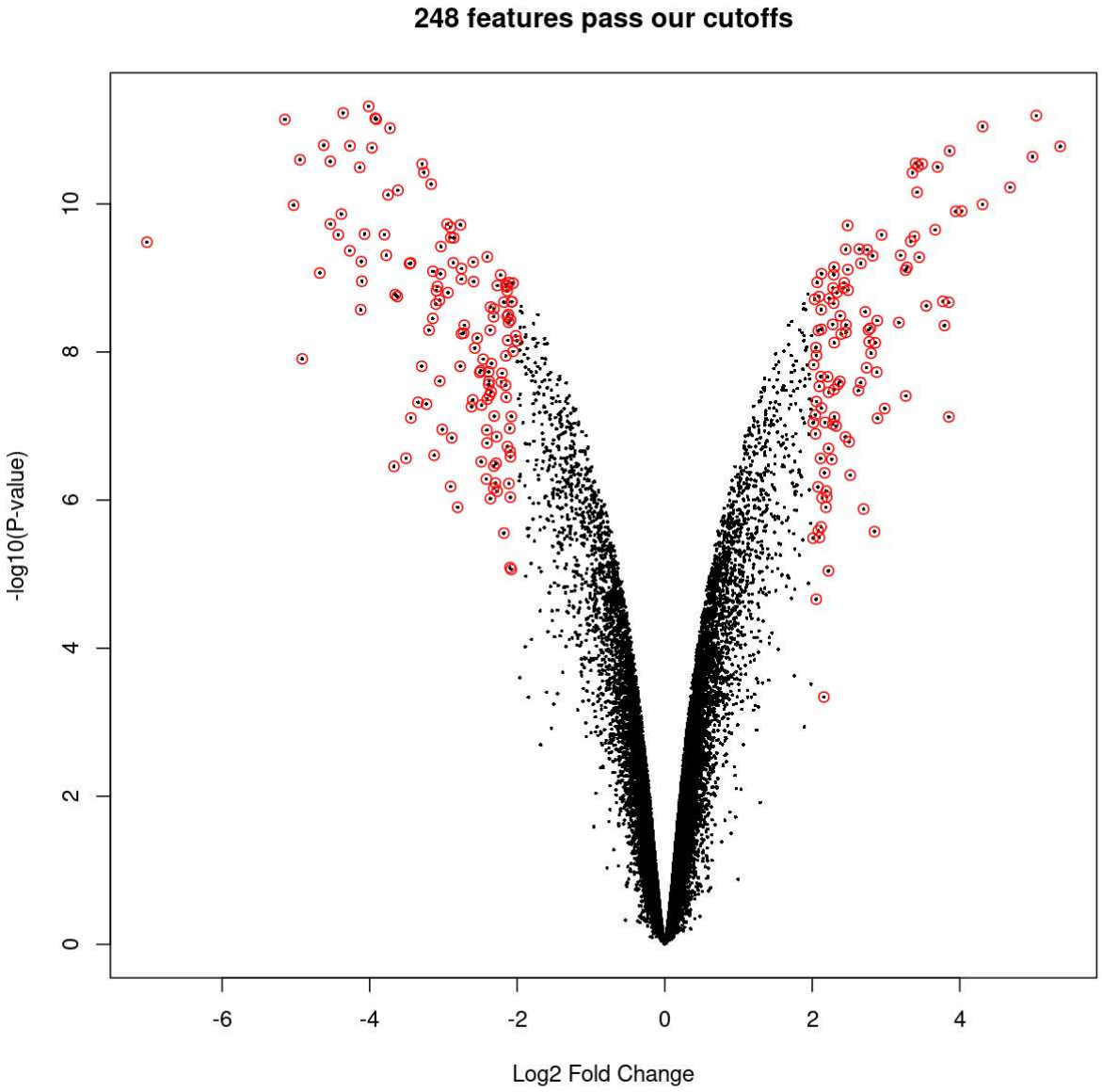


But we can use some R features to highlight the points with cutoffs of interest. Let's decide the interesting genes are those with $|\log_2(\text{fold change})| \geq 2$ and adjusted $p < 0.05$, thus:

```
interesting_genes <- topTable(fitted.ebayes,number=Inf,p.value = 0.05,lfc=2)
```

```
Removing intercept from test coefficients
```

```
volcanoplot(fitted.ebayes, coef=2, main=sprintf("%d features pass our cutoffs",nrow(interesting_gene
s)))
points(interesting_genes[['logFC']],-log10(interesting_genes[['P.Value']]),col='red')
```



From the plot above, we can see that many genes tend to show small but statistically insignificant changes in expression level between the two conditions. On the other hand, a small subset of the genes show very large fold changes with relatively small p-values. These “interesting” genes tend to be further analyzed for their potential role in the biological phenomenon of interest.

Heatmaps

Heatmaps are among the most widely used plots for expression data matrices of expression data.The idea is to draw a false colour image of a data matrix (typically red is “hot” or high expression), and to use a clustering algorithm to sort the rows (features) and columns (samples) to highlight natural groupings of the data.

📌 Choosing the genes of interest for analysis.

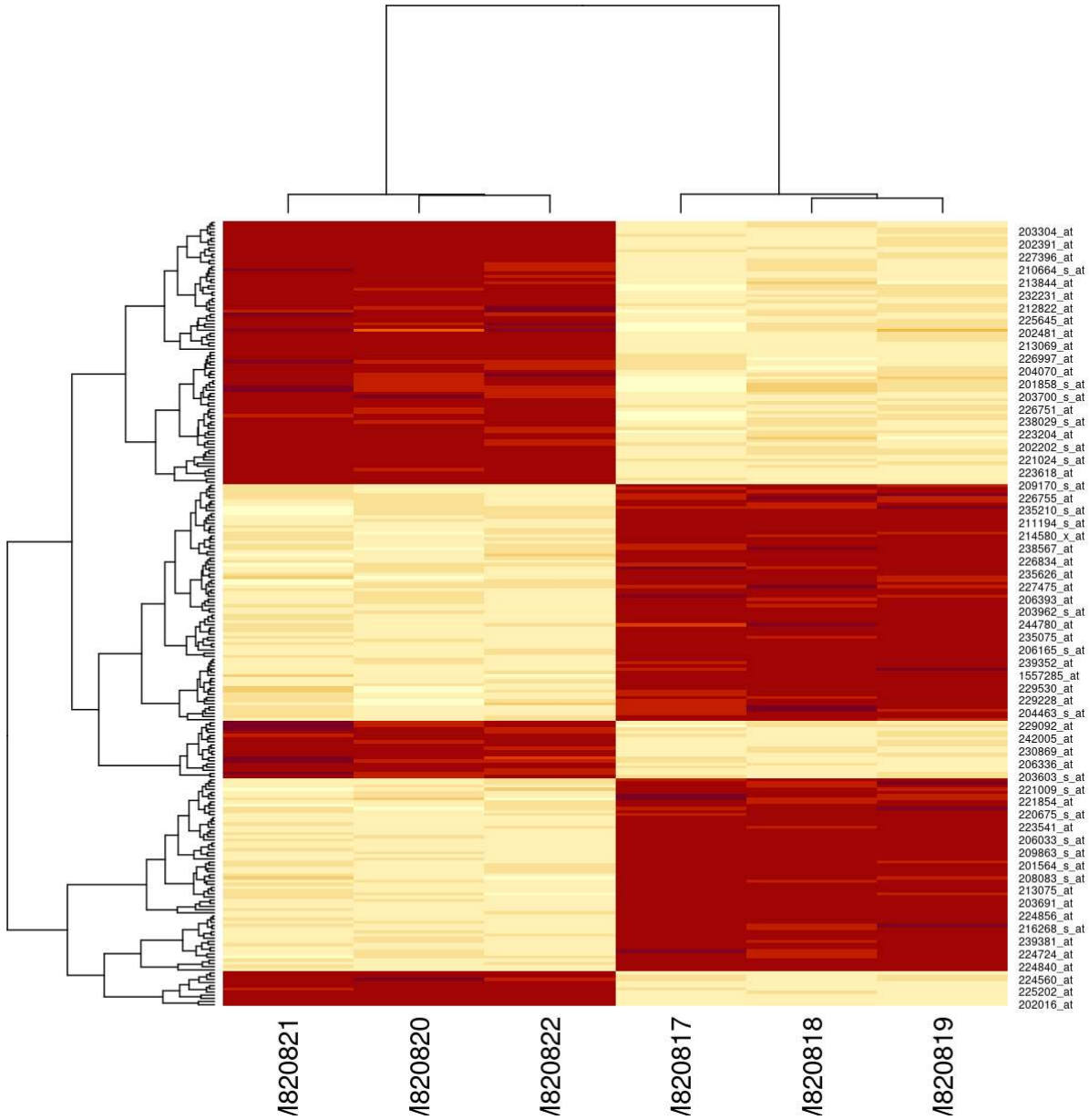
How do we choose which genes we should be included in our heatmap? Ideally, we will have all the differentially expressed genes on it. However, in some cases, that is may not be feasible (especilly if there are simply too many). Another strategy might be to use genes that show the most variance, as it is most likely that we can use these highly variable genes to cluster our samples into distinct groups. Finally, we can visualize simply the top most significantly expressed genes, which we will do here.

For simplicity, we will visualize the expression of features that pass our cutoffs. To do so, we will need the following:

1. The list of all the probesets most differentially expressed between our experimental condition, and;
2. Their normalized expression value.

Let's use the genes that passed our previous cutoffs and create a dataset of just those genes

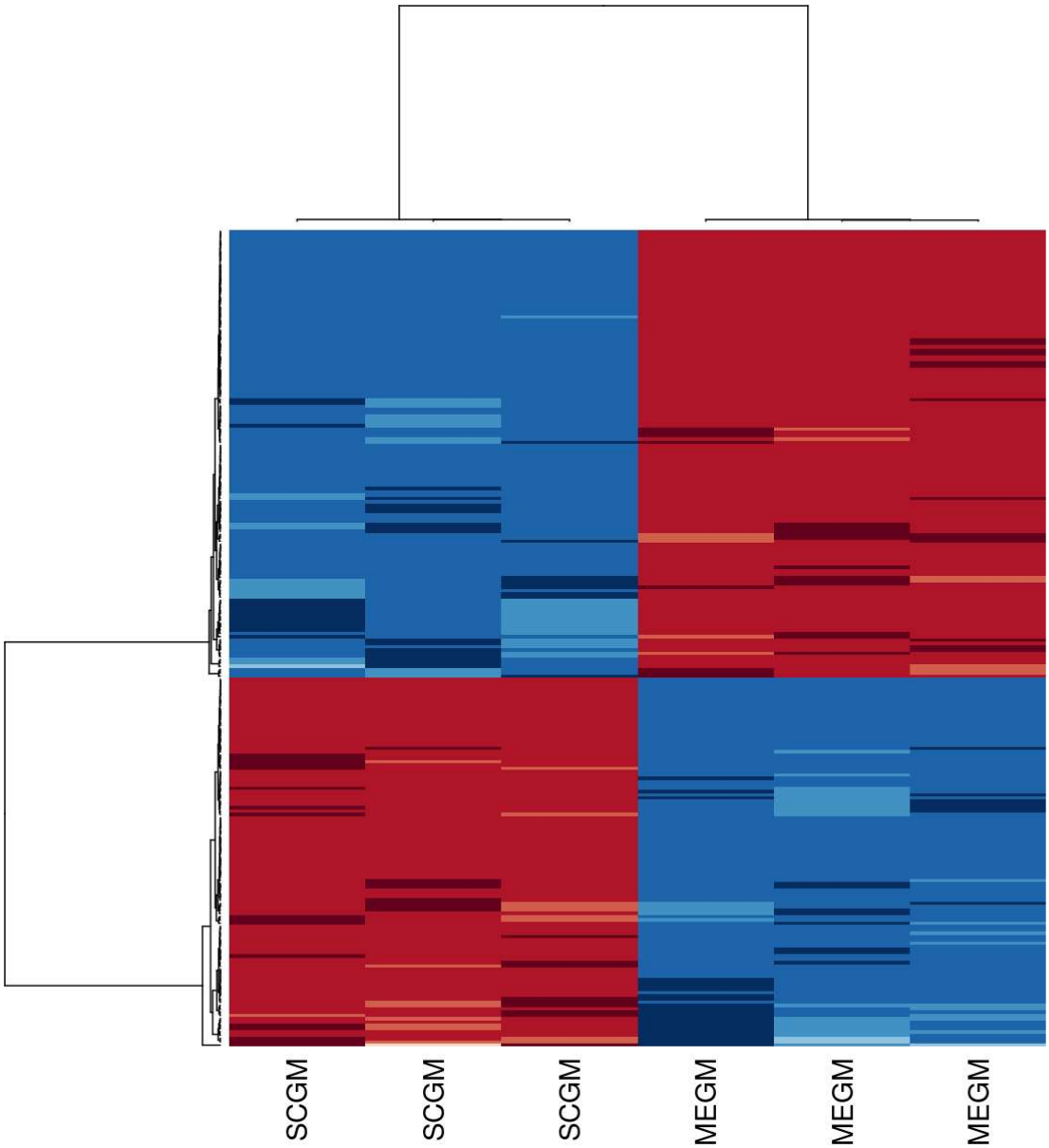
```
eset_of_interest <- gse33146_eset[rownames(interesting_genes),]
heatmap(exprs(eset_of_interest))
```



The `heatmap()` function does a lot of work. The denodrograms on the top and the left represent the ordering of rows and columns. This is done using a distance calculation, by default Euclidian distance. It doesn't seem quite right, because we only have two groups of genes: they should either be up or down in each group.

We can make a few changes to fix and beautify the heatmap. The most useful is to use *correlation*, rather than euclidean distance, for the clustering. The function requires a distance as input, so we'll use `1 - correlation`. We can also change the colours, label the samples according to culture conditions, and remove the gene labels.

```
library(RColorBrewer)
heatmap(exprs(eset_of_interest),
        labCol=gse33146_eset[['culture medium:ch1']] ,labRow=NA,
        col       = rev(brewer.pal(10, "RdBu")),
        distfun   = function(x) as.dist(1-cor(t(x))))
```



What's next?

Although we have discussed the volcano plot and heatmaps, there are numerous avenues for downstream analysis. For instance, one commonly used analysis toolkit is the gene set enrichment analysis (GSEA) where we test for whether particular sets of genes are enriched in our list of differentially expressed genes. Ultimately, what downstream analysis is performed is dependent on the question that one wishes to

address. New analysis methods are constantly being developed by others, and hence it is important for one to keep abreast on these developments in order to get the most out of our analysis.

Key Points

- Volcano plots can be used to infer relationships between fold changes and statstical confidence.
- Heatmaps can be used to visualize distinct trends in gene expression patterns between different experimental conditions by clustering.

‹ (../08-AnnotatingResults/index.html)

^ (../)