Lab 5 A : MICROARRAY DATA ANALYSIS

In the following exercises we will take advantages of R/BioConductor packages. So before we get started, you may review the fundamental knowledge of R.

Exercise 1: Preprocessing: Reading the raw data, implementing quality control and normalization

1. Use the command library(affyPLM) to load simultaneously R packages, which include affy, Biobase and gcrma
2. Download the raw data from GEO, for example, GSE1000, with the extension .CEL (http://cbb.sjtu.edu.cn/course/bi390/data/GSE1000_RAW.tar)
3. Use the command abatch.raw=ReadAffy() to load the expression data from all the .CEL files in current directory and create an object abatch.raw in AffyBatch class. How big is the resulting matrix? What is the average intensity in the matrix? And for each of the samples? What other information can abatch.raw file provide?
4. Use MAplot(abatch.raw), hist(abatch.raw) and boxplot(abatch.raw) to generate graphs on the data. What do you see with quality data and is there some sort of deviation, inconsistency or error? Based on this information, do you think think the data require normalization?
5. Use RMA algorithm to process data: eset.rma=rma(abatch.raw). What is the dimensionality for this gene expression matrix?
6. Repeat the step 4) for eset.rma. Now what does the figure look like?

Exercise 2: GEO Database

1. Use the command library(GEOquery) to load the package for downloading GEO data (only for processed, for raw data .CEL, use the steps in Exercise 1)
2. Use eset.geo=getGEO("GSE1000")[[1]] to get the processed expression matrix. The [[1]] is required because getGEO() returns a list of matrices (in the case when experiments carried out on more than one platform)
3. Make graphs similar to those in step 4) of exercise 1 on this object. Do normalized data found in GEO have sufficient quality? Would you like to do any additional processing or normalization on the data?

Exercise 3: ArrayExpress Database

1. Search against the ArrayExpress web for experiment GSE1000. Can you find it? Which ID do you get? On which repository do you find more information, GEO or ArrayExpress?
2. Use the command library(ArrayExpress) to load the package for direct download expression data from ArrayExpress (unlike GEO, you can only download the raw data, no processed data are available)
3. Use eset.ae=ArrayExpress(id) to download the expression matrix (raw). Here "id" is the ArrayExpress ID like "GSE1000" in GEO as you saw in the Exercise 1 ("E-GEOD-1000"). Does eset.ae have the same dimensionality as abatch.raw in Exercise 1? Are the conditions for these two objects in same

or different order?

4. Now you can perform exploratory analysis as in step 4) in Exercise 1, but use another library (arrayQualityMetrics) to perform all the quality analysis, generating a report that indicates the conditions of dubious quality. To perform this analysis uses the command:
   arrayQualityMetrics(eset.ae, outdir="out")
5. Replace "out" with your own output directory. How many report files are generated? Does the result agree with that in Exercise 1?
6. Repeat the same quality analysis as we did on the RMA-normalized data, eset.rma, in Exercise 1. Any difference? Do any of the conditions show low quality?

Exercise 4: Scanning through the experiments and conditions

Explore the information about the experiment and its conditions. Even if a microarray experiment is mainly defined by the levels of expression, what really matters is what these levels are, i.e., the information we have on the rows and columns of the matrix.

Work on objects, eset.ae and eset.rma, which are generated in previous exercise:

1. For eset.rma, use commands annotation(eset.rma) and experimentData(eset.rma). What kind of information do you get? Does it complete? Repeat the above processes on eset.ae which is obtained from ArrayExpress. Which one holds more information?

2. Use command sampleNames(eset.rma) to get the condition names. What kind of information do you get? Would you give some information?
3. We see that eset.ae give more information. However, the normalized expression matrix eset.rma has higher quality. Can we combine the two objects? Here is one option:
   exprs(eset.ae)=exprs(eset.rma)[,sampleNames(eset.ae)]
4. replace the expression matrix of eset.ae by that of eset.rma. Remember in this case, we need to ensure that the two conditons for these two objects should be in the same order. That is the reason why we use [,sampleNames(eset.ae)] to reorder the conditions.

   Could you figure out another option?
5. The phenotypic data of the experiment can be obtained by command pData(eset.ae). The returned object is a data.frame, whose row contains a lot of information on each of the experiment condition. Which information, do you think, is more relevant (indicates the column names)?
6. Two types of information are very important features (Characteristics..XXX.) And experimental factors (Factor.Value..XXX.). The first indicate features common to all experimental samples. The second examines the factors which affect the experiment. For example, if we are studying the variation in expression in the human brain **regarding sex**, **tissue (brain) and the (human)** are fixed characteristics, while **sex (male, female)** is the variable factor. What are the characteristics and experimental factors eset.ae (indicating their names as they appear in the

data.frame) and what values do they take in the experimental factors?

7. The command `pData(eset.ae)[,"Factor.Value..Time."]` returns the experimental values of the time factor. For conditions that have value 6 we can use:

   `which(pData(eset.ae)[,"Factor.Value..Time."]==6)`

   What is the mean expression value for time 6 and time 32, repectively?

Exercise 5: Exploring related information and mapping probes

The names of the first 10 probes in our experiment can be obtained by `featureNames (eset.rma) [1:10]`. The probe names, `xxxxx_at` are Affymetrix probe identifiers, generally refer to portions of genes coding sequences (although there are also some control probes, etc.). To map probe identifiers to corresponding genes we have annotation packages, one for each Affymetrix platform

1. Get the expression value for probe 202709_at at contition 3.
2. Compute the mean expression values for column “GSM15719.cel” and “GSM15794.cel” ranging from 1000 to 2000
3. In our case, the platform is “hgu133a” (this information can be obtained by `annotation(eset.ae))`. Download the package “hgu133a.db” to map the probes to genes.
4. Use `HGU133A()` to display all existing mappings. You will see many identifiers mappings known as UniGene, UniProt, Entrez, etc.. Which of them do you find are most useful? Are there any that you do not know?

5. To map the probe 202709_at to the corresponding gene, use the following command: mget("202709_at", hgu133aGENENAME). Likewise, we can obtain the information for a set of probes using the similar commands, for example, the first 10 probes.

Exercise 6: Differential Expression

Let's do some analysis on a simple data set: GSE17636/E-GEOD-17636.

This is the experiment on two cell lines of breast cancer, one for (wt) and another treated by NF1iC2 (nf), a tumor inhibitor. There are 3 replicates for each cell line, resulting in 6 conditions.

Here is the R code for executing differential expression:

```
source("http://www.bioconductor.org/biocLite.R")
biocLite(c("ArrayExpress", "hgu133plus.db", "affyPLM"))
library(ArrayExpress) # get microarray data
ae.raw=ArrayExpress("E-GEOD-17636")
library(affyPLM) #normalization
ae.rma=rma(ae.raw)
# wildtype and nf1-c2
wt=which(pData(ae.rma)[,"Characteristics..genotype."]=="wild type")
nf=which(pData(ae.rma)[,"Characteristics..genotype."]!="wild type")
# getting the gene names for each probe
library(hgu133plus2.db)
gn=unlist(mget(featureNames(ae.rma), hgu133plus2SYMBOL, ifnotfound=NA))
brca=grep("BRCA", gn) # probes corresponding to BRCA
gn[brca]
```

Can you understand well what is done in each step? Could you explain it? You can explore the data in more depth or conduct additional treatment.

Perform the following tasks:

1. Build two vetors, one containing the average expression of each probe

under condition “wildtype (wt)” and another with the average expression of each probe under condition “nf1ic1 (nf)”.

2. Calculate the mean expressions for probes corresponding to the genes BRCA1 and BRCA2, in both groups. Does the condition “nf1ic1” affect the gene expression?

Now let's see which genes are differentially expressed in general.

3. To do this, we first need to calculate exchange ratios are. The log ratio is calculated as ln (expression1/expression2). Calculator the log ratios for all probes using as expression1 and expression2 the means obtained in the first point for nf and w2, respectively.
4. Determine which probes have a log-ratio greater than 0.8. Stores your positions in a vector. What genes are mapped?
5. Use t.test function to determine the statistical significance of differential expression of the probes obtained in the previous step. For example, if one of the probes is 1061, should be done:
   t.test(exprs(ae.rma)[1061,wt], exprs(ae.rma)[1061,nf])
6. Would you be able to do it for all differentially expressed probes, using sapply()? Beware, if you select many probes, this can be very slow.

Exercise 7: Differential Expression Analysis using linear fit.

We continue with the previous example, we now use the limma package, which models the data using a linear fit and estimates the significance.

1. Use the command library(limma) to load the package limma

2. Use ?lmFit and ?eBayes to see how to use this method. In particular, you can also consult the user guide with limmaUsersGuide() function. In this guide, carefully read Section 8.1 (Two groups) for differential analysis. The following sections, with combinations of more groups and time series analysis, are also very interesting.

Exercise 8: Hierarchical Clustering

We start from the set of genes differentially expressed above a log ratio of 0.8 found in Exercise 6:

1. Calculate the distance between probes (rows) dr <- dist(exprs(ae.rma [degs,])). Explore the different distances that can be used (by default, Euclidean distance)
2. Making a hierarchical clustering rows with hr <- hclust (dr) . Again, we can test different hierarchical clustering methods.
3. Print the result with plot(hr)
4. Repeat for the conditions, generating dc and hc (distance and clustering for columns). Hint: for this we will need to use the expression matrix transpose t().
5. Generate a heat map (heatmap) with dendrograms. A heat map represents the expression matrix as a color grid, each square of the grid represents the expression of a probe / gene for a condition, using a color scale. Hint: use the heatmap and heatmap.2 functions (the latter is in the package

gplots) . Look at your help document as they have a lot of arguments to be configured.

Exercise 9: Functional Annotation with GO Terms

Now we know which genes are differentially expressed, the corresponding statistical significance, and clustering of these genes. The last thing we want to know is whether these genes share any known biological function. We will use the GOstats package with the Gene Ontology.

Here we will conduct a *hypergeometric* test.

Remember that any significance test basically checks whether the fact that n genes out of m distinct genes are annotated with a term is significant or not.

1. First, choose, for example, the top 50 probes differentially expressed according to the analysis in the Exercise 7 (limma) using topTable() function.
2. Determines Entrez IDs for these probes, using hgu133plus2.db package and the function mget().
3. Entrez IDs are determined for ALL probes of the experiment, using again the package hgu133plus2.db and mget() function. This set is called the "universe" for the problem.
4. Prepare an object of class GOHyperParams where geneIds will be the first group of identifiers that have sought and universeGeneIds the second. It uses ontology = "BP" and annotation = "hgu133plus2rÉÇ".

```
params <- new("GOHyperGParams",
    geneIds=selectedEntrezIds, universeGeneIds=entrezUniverse,
    ontology = "BP", annotation = "hgu133plus2rÉÇ", pvalueCutoff=0.001,
    conditional=FALSE, testDirection="over")
```

5. Run the test with the function hyperGTest() inspects the result with summary().

What groups appear as enriched? Is there any relationship between them? Are there biological reason regarding the experiment (evaluation of the effect of a tumor repressor)? Would you say, seeing the numbers of genes annotated in the group with respect to all annotated in the universe, that the enrichment of these groups is reasonable?

Change the group of differentially expressed genes, selecting for example those below a given p-value (topTable()'s p.value argument) or those obtained with simpler analysis of differential expression in Exercise 6. What is the relationship between these groups? Did they show similar enrichment of the same GO terms?

Final submission

Execute the similar analysis on the E-MTAB-62 microarray data in ArrayExpress.