Lab 9. Microarray data analysis 1

1. Objective

* To learn how to use MATLAB.
* To preprocess microarray data for removing variation through MATLAB.
* To extract differentially expressed genes from preprocessed microarray data using statistical analysis through MATLAB.

1. Background
   1. Microarray

A microarray is a solid support (such as a glass microscope slide or a nylon membrane) on which DNA ofknown sequence is deposited in a regular grid like array. The DNA may take the form of cDNA(complementary DNA) or oligonucleotides that can hybridize complementally with target DNAs. Typically, RNA is extracted from biological sources of interest, such as cell lines with or without drug treatment, tissues from wild-type or mutant organisms, or samples studied across a time course (figure1). The RNA is often converted to cDNA, labeled with fluorescence dyes differently for cDNAs of different sources, and hybridized to the array.

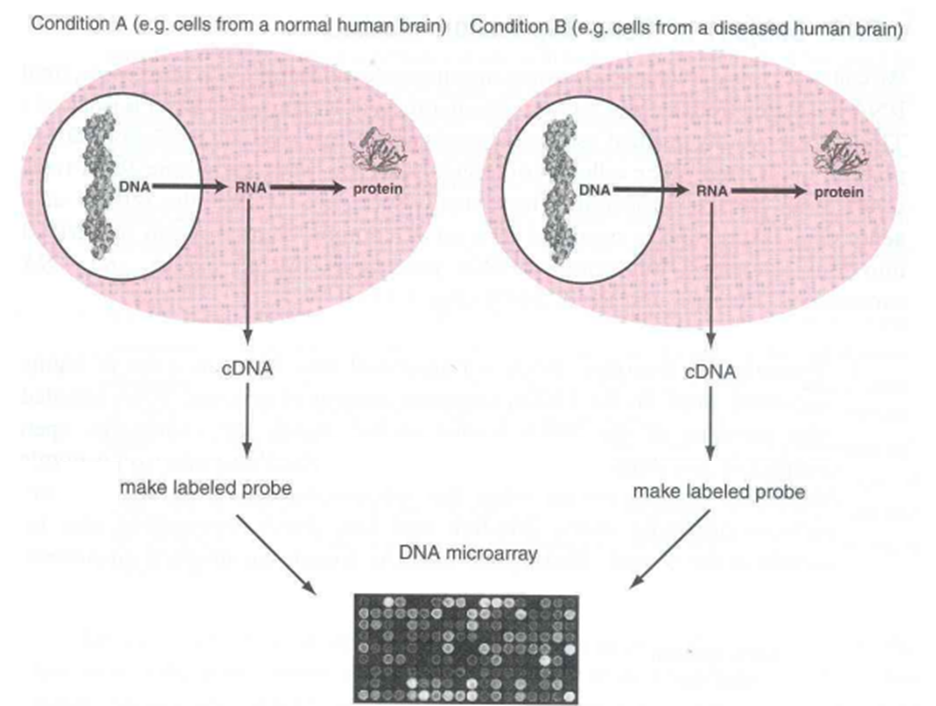


Figure1. Typical microarray experiment setup to find differential gene expression profiles in different conditions

cDNAs derived from RNA molecules in the biological starting material can hybridize selectively to their corresponding nucleic acids on the microarray surface. Following washing of the microarray, image analysis and data analysis are performed to quantitate each fluorescence signals that are detected from the hybridized cDNAs. Through this process, microarray technology allows the simultaneous measurement of the expression levels of thousands of genes represented on the array. Figure2 summarize the overview of this process.

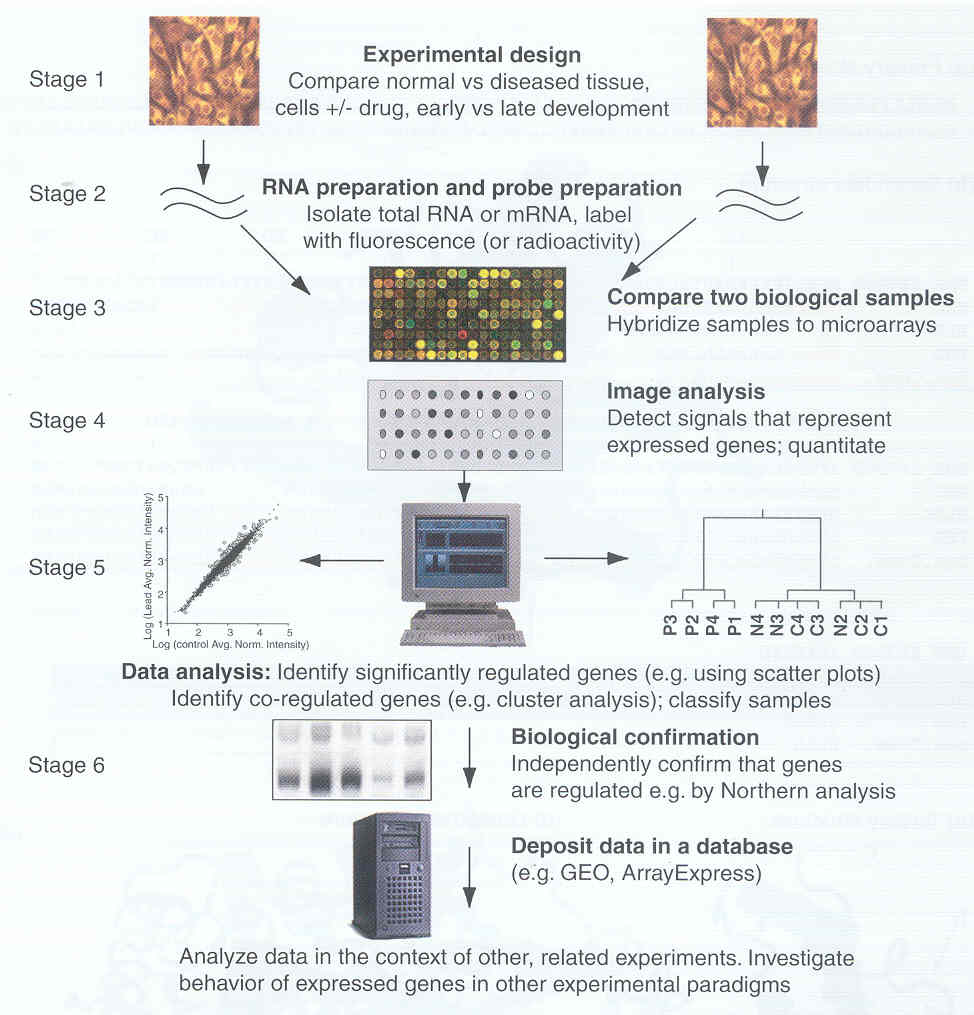


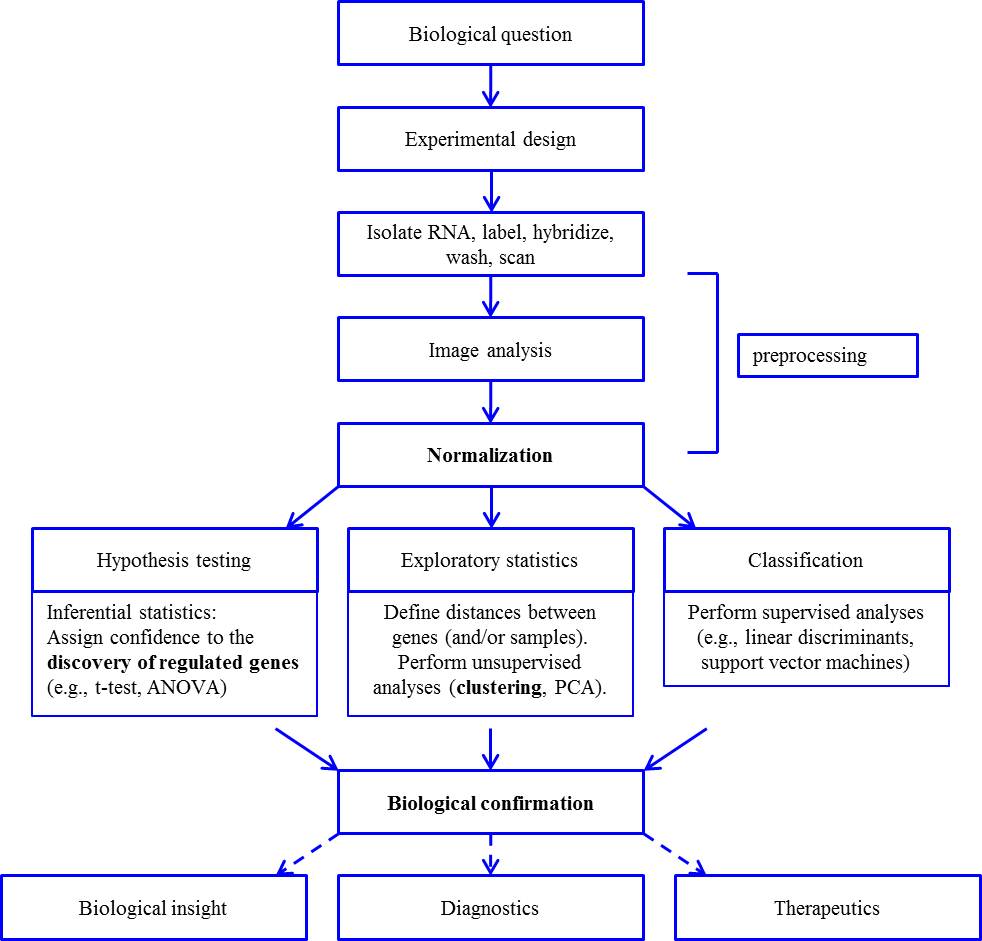
Figure2. Overview of the process of generating gene expression data

Microarray technique allows us to find the expression profiles of whole gene sets and systemic analysis of their changes in different conditions. Dozens of other application can be developed using its properties detecting the quantity and sequence specificity of target nucleic acids.

* 1. Overview of Microarray Data Analysis

Figure 3 shows the overview of microarray data analysis. After microarray data are generated, preprocessing is performed in which raw image data are analyzed, normalized, and a matrix of genes and samples is created. There are three main stages of microarray data analysis – hypothesis testing, exploratory statistics, and classification. First, **hypothesis testing** is performed in which t-test, ANOVA, or other statistical tests are applied to determine which transcripts were significantly up- or down- regulated in the experiment. Second, **exploratory statistics** may be applied. This set of approaches includes clustering and principal components analysis and is used to inspect the complex data set for biologically meaningful patterns. The similarities of the data points are compared with a metric such as a correlation coefficient. This pattern of gene expression may be visualized using unsupervised approaches in which patterns are sought in the representation of genes. Third analysis is the **classification** using supervised analysis where samples are associated with labels from a preexisting classification such as normal vs. diseased tissue and gene expression measurements are used to predict which unknown samples are diseased. After results from the analysis one does a biological confirmation to see how valid the results are. This may lead to insight about biological processes, or to outcomes relevant to disease such as identifying diagnostic markers or strategies for therapeutic intervention.

The rest parts explain the detail background of selected topics involved in this lab activity which are normalization, gene identification using t-test and hypothesis testing.



genes

samples

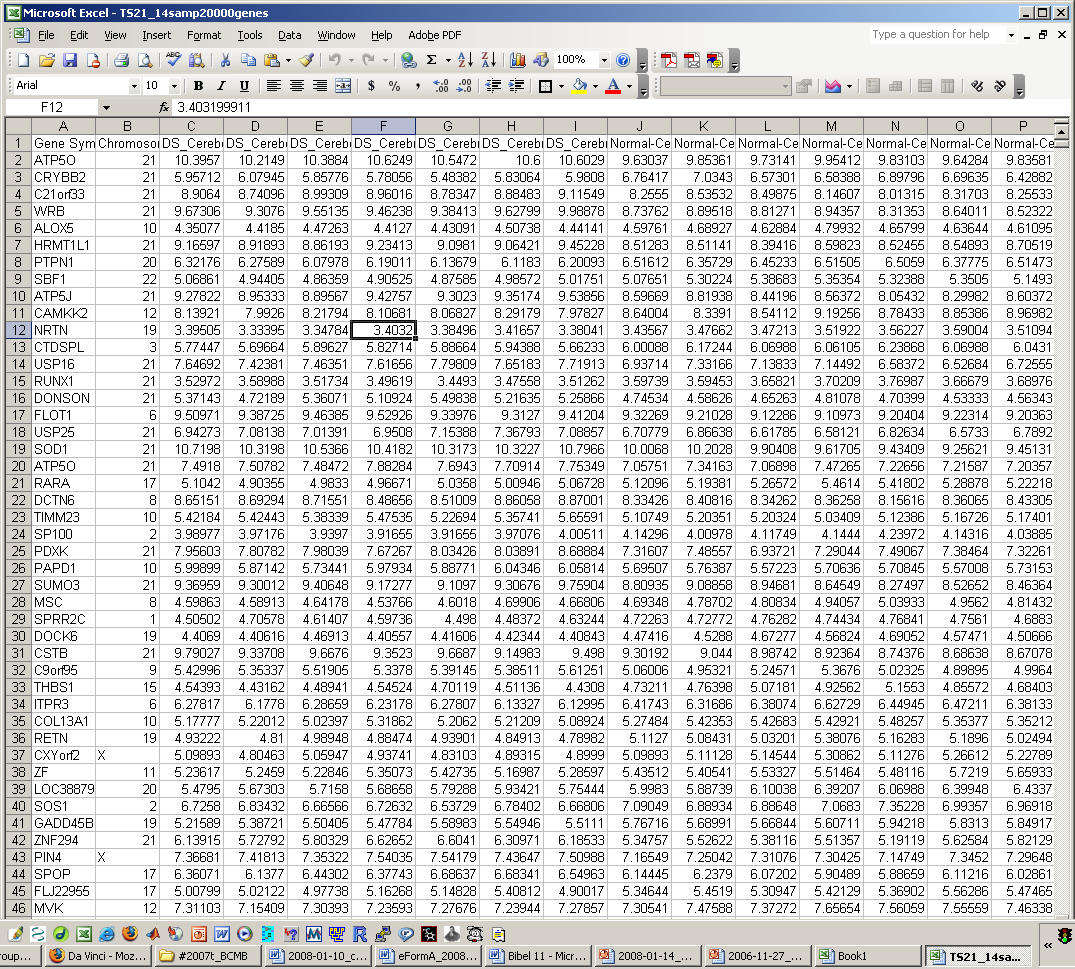


Figure3. Overview of microarray data analysis

* 1. Microarray data format

Following figure is sample data set used to demonstrate statistical approaches to microarray data analysis. Intensity values for eight genes (i.e. mRNA transcripts; see rows) were shown. The columns include Gene name (column A) and seven intensity values for seven Down syndrome samples (column G to M) and seven control samples (column N to T). The data were imported in to Microsoft Excel, and additional columns of information include the average Down syndrome expression value (column C), average control value (column D), the p-value resulting from a t-test (column E), and the Down syndrome to control ratio (column F, derived from C/D). Theoretical background about p-value and t-test is explained in section 2.5

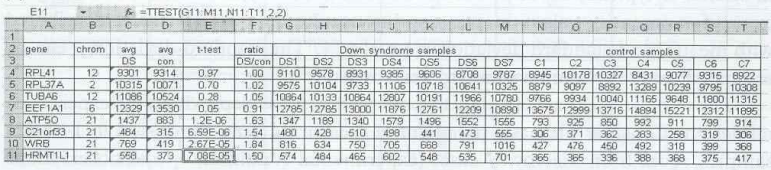


Figure4. Sample microarray dataset

There are three main databases that provide microarray data.

* Gene Expression Omnibus (GEO) at NCBI: [www.ncbi.nlm.nih.gov/geo/](http://www.ncbi.nlm.nih.gov/geo/)
* ArrayExpress at the EBI: [www.ebi.ac.uk/arrayexpress/](http://www.ebi.ac.uk/arrayexpress/)
* Stanford Microarray Database (SMD): <http://smd.stanford.edu/>
  1. Normalization

The term “normalization” as applied to microarray data does not refer to the normal (Gaussian) distribution, but instead it refers to the process of correcting two or more data sets prior to comparing their gene expression values. As an example of why it is necessary to normalize microarray data, the Cy3 and Cy5 dyes are incorporated into cDNA with different efficiencies. Without normalization, it would not be possible to accurately assess the relative expression of samples that are labeled with those dyes. Thus, normalization is required for microarray experiments.

Most investigators apply a global normalization to raw array element intensities so that the average ratio for gene expression is one. The main assumption of microarray data normalization is that the average gene does not change in its expression level in the biological samples being tested. For example, the procedure for global normalization can be applied to two-channel data sets (e.g., Cy3- and Cy5-labeled samples). Two-channel data are treated as two individual data sets such that each element signal intensity is divided by a correction factor specific to the channel. As an example, if the mean expression value for samples in the green channel is 10,000 arbitrary units and the mean value for samples in the red channel is 5,000, then the expression value for each gene in the red channel would be multiplied by 2. If the data are not log transformed, the mean ratio is then 1.

* + 1. Quantile normalization

Quantile normalization is an approach that produces the same overall distribution for all the arrays within an experiment. It is a nonparametric method. Parametric tests are applied to data sets that are sampled from a normal (Gaussian) distribution. Nonparametric tests do not make assumptions about the population distribution. To produce the same overall distribution for all arrays, quantile normalization method use average quantile value of genes among all arrays. First, all arrays’ genes are sorted and arranged in decreasing(increasing) order of gene expression values among that array. Second, average gene expression values of each order(quantile) are calculated. Third, each order(quantile)’s gene expression value is set to already calculated average value. Finally, all arrays’ genes are rearranged in original order. So, the highest gene expression value in all arrays becomes the average of the highest gene expression values, the second highest gene expression value becomes the average of the second highest gene expression values, and so on.

* 1. Gene selection with statistical test

Gene selection is the process to select the genes that differentially expressed in different conditions. Those selected genes may be involved in a key cellular mechanism to regulate or to be regulated by the conditions, which can be further developed for the disease markers or drug targets. To determine the significance of the differences of expression data in two conditions, we can use inferential statistics.

The goal of inferential statistical analysis of microarray data is to test the hypothesis that some genes are differentially (up- or down-) expressed in an experimental comparison of two or more conditions. We formulate the null hypothesis H0 that there is no difference in signal intensity across the conditions being tested. The alternative hypothesis H1 is that there are differences in gene expression levels. To determine whether accept the null hypothesis or not, test statistic is calculated by using observed gene expression signal intensity across the condition. And then we accept or reject the null hypothesis based on the results of a test statistic. The probability of rejecting the null hypothesis when it is true is reffered as the significance level *α* and this probability is determined by the above test statistic. In scientific filed, the significance level *α* is typically set at *α* =0.05. If the probability under the null hypothesis is below the significance level *α*, we can say that the null hypothesis is rejected because the probability that the null hypothesis occurs is below the significance level (very low).

* + 1. *t*-test

The test statistic that you apply to a microarray study depends on the experimental design. Consider the basic paradigm of measuring gene expression in 14 brain samples, seven from trisomy 21 cases (experimental condition, with observations ) and seven form apparently normal individuals (control condition, with observations ). You can calculate the mean and standard deviation for the expression of each gene represented on the microarray. A *t*-test is performed to test the null hypothesis that there is no difference in gene expression levels, considered one gene at a time, between the two populations. Compute the mean expression value for each gene from control (*x*1) and experimental (*x*2) conditions, estimate the variance, and divide them. The average for each sample (e.g., 1) is given by:

The variance (square of the standard deviation, *s*2) for *x*1 is given by:

The t-test essentially measures the signal-to-noise ratio in your experiment by dividing the signal (difference between the means) by the noise (variability estimated in the two conditions).

From the *t*-statistic we can calculate a *p*-value. For calculating p-value, the basic assumption that gene expression values are following gaussian distribution is introduced. However, in the biological field, typically sample size is small and the poplulation standard deviation is unknown. In this situation, student’s t-distribution can be used instead and above t-statistic is following this ditribution. Thus, p-value under the null hypothesis that mean expression value between two conditions is zero is computed using t-distribution. P(T t) is calculated from the t-distribution and this is the probability of rejecting the null hypothesis when it is true. This allow us to either reject or accept the null hypothesis that the control and experimental conditions have equal gene expression values (i.e., the null hypothesis is that there is no differential expression). For a *t*-test that provides a *p*-value of 0.01, this means that one time in 100 the observed difference between the control and experimental groups will occur by chance assuming the null hypothesis, therefore we can safely reject the null hypothesis. Figure 4 presents the result of *t*-tests performed on eight genes. Four genes assigned to chromosome 21 have very low *p*-values (about to ), one has a *p*-value of 0.05, and three have non-significant *p*-values (0.97, 0.70, 0.28).

* 1. Introduction to MATLAB

MATLAB stands for “MATrix LABoratory” and supports interactive environment incorporating such as programming language, mathematical calculations, data visualization and large number of inbuilt routines corresponding to many mathematical, engineering & science problems.

* + 1. Basic Mathematical Operators, Expressions & Comments

The following basic mathematical operators are supported by Matlab:

|  |  |
| --- | --- |
| **Operation** | **Symbol** |
| Addition (a + b) | + |
| Subtraction (a-b) | - |
| Multiplication (a . b) | \* |
| Division (a ¸ b) | / or \ |
| Exponentiation (ab) | ^ |

Single lines (calculations or commands) of Matlab have one of two general forms:

variable = expression

|  |
| --- |
| y=sin(3.14);  z=(y\*4.6)^12.9 – 13.456; |

or

expression

|  |
| --- |
| plot(x,y,’-+’);  z^3.5 |

The results of calculations do not need to be saved to a variable explicitly. If no variable is specified then the result is automatically saved to the ans (answer) variable. In general you should not use ans but your own meaningfully named variables.

|  |
| --- |
| >> 91.3 – 14  ans =  77.3  >> z=ans \* 2  z =  144.6 |

Simply typing a variable’s name alone is interpreted as a command to show the value stored in that variable.

|  |
| --- |
| >> z  z =  144.6 |

By default Matlab interprets the end of a line as the end of a statement/expression.

Semicolon

Semicolon(;) terminates the current expression and suppresses output (to the screen) of the result

|  |
| --- |
| e.g., volume = pi \* radius^2 \* height;  The value is calculated and stored in volume but not echoed back to the screen. |

A semicolon should be used on most lines of code as we are not interested in intermediary results.

Comma

The comma(,) can be used to separate multiple statements on the same line.

The value of any variables will be echoed to the screen.

|  |
| --- |
| e.g., x=5.7, y=89.12  will echo those variables and their names back on the screen. |

Period

Long statements can be split over lines by 3 periods.

|  |
| --- |
| e.g., incomeInHand = salary – tax – unionFee …  - superAnnuation; |

* + 1. Manipulation & calculation of matrices

Matlab was designed to strongly support arrays. Most operations treat scalars and arrays in exactly the same way. In most programming languages the size of an array is fixed at compile time and may not be altered in the program; however, arrays can be dynamically created and altered in size in MATLAB.

|  |
| --- |
| arr1 = 1:10;  arr1(11) = 11; |

Matlab provides a number of mechanisms for the explicit creation of arrays:

square brackets ([])

When the individual elements are listed

colon operator (:)

When a starting value, final value and step-size are known (sequence)

linspace & logspace

To create linear and exponential sequences of a set size

|  |
| --- |
| A = [1 2; 3 4]; % Creates a 2x2 matrix  B = [1,2; 3,4]; % The simplest way to create a matrix is  % to list its entries in square brackets.  % The ";" symbol separates rows;  % the (optional) "," separates columns.  N = 5 % A scalar  v = [1 0 0] % A row vector  v = [1; 2; 3] % A column vector  v = v' % Transpose a vector (row to column or column to row)  v = 1:0.5:3 % A vector filled in a specified range:  v = pi\*[-4:4]/4 % [start:stepsize:end], brackets are optional  v = [] % Empty vector |

An example code:

|  |
| --- |
| >> theta=[0 pi/4 pi/2 3\*pi/4 pi]  theta =  0 0.7854 1.5708 2.3562 3.1416  >> theta = [theta pi+theta(2:end)]  theta =  Columns 1 through 7  0 0.7854 1.5708 2.3562 3.1416 3.9270 4.7124  Columns 8 through 9  5.4978 6.2832  >> sinValues=sin(theta)  sinValues =  Columns 1 through 7  0 0.7071 1.0000 0.7071 0.0000 -0.7071 -1.0000  Columns 8 through 9  -0.7071 -0.0000 |

Indexing vectors & matrices:

|  |
| --- |
| % Warning: Indices always start at 1 and \*NOT\* at 0!  v = [1 2 3];  v(3) % Access a vector element  m = [1 2 3 4; 5 7 8 8, 9 10 11 12; 13 14 15 16]  m(1, 3) % Access a matrix element matrix(ROW #, COLUMN #)  m(2, :) % Access a whole matrix row (2nd row)  m(:, 1) % Access a whole matrix column(1st column)  m(1, 1:3) % Access elements 1 through 3 of the 1st row  m(2:3, 2) % Access elements 2 through 3 of the 2nd column  m(2:end, 3) % Keyword "end" accesses the remainder of acolumn or row  m = [1 2 3; 4 5 6]  size(m) % Returns the size of a matrix  size(m, 1) % Number of rows  size(m, 2) % Number of columns  m1 = zeros(size(m)) % Create a new matrix with the size of m |

Simple operations & dot(.) modifier

|  |
| --- |
| a = [1 2 3 4]'; % A column vector  2 \* a % Scalar multiplication  a / 4 % Scalar division  b = [5 6 7 8]'; % Another column vector  a + b % Vector addition  a – b % Vector subtraction  a .^ 2 % Element-wise squaring (note the ".")  a .\* b % Element-wise multiplication (note the ".")  a ./ b % Element-wise division (note the ".") |

* + 1. Handling Scripts

Re-entering the same program (set of instructions) multiple times is wasteful and Script (M-Files) are enables to record MATLAB scripts. Basic approach for using M-Files is as follows:

1. Create m-file with a text editor (or Matlab's built-in Script editor).

2. In Matlab command window enter name of script file

(e.g., if file called example1.m then enter example1) to run it.

3. While still errors in the script

3.1 Modify & save script using editor

3.2 Rerun script in Matlab

M-files (scripts) must all have the suffix ".m". When a script name is entered, Matlab looks for a file of that name.

|  |
| --- |
| e.g., >> amoeba  Matlab searches for a file called amoeba.m |

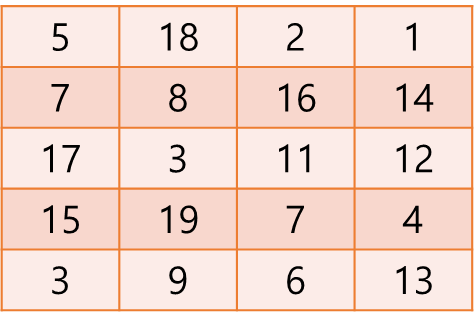
1. Prelab activities
   1. Explain about pros and cons of the microarray technology.
   2. What is the purpose of the microarray data preprocessing? Explain how the microarray expression data are processed from the raw images.
   3. Answer the following questions about quantile normalization

(a) Explain about quantile normalization.

(b) How the microarray expression data will be changed after quantile normalization?

(c) Apply quantile normalization technique to the dataset below. Explain the key step of procedure and corresponding results.

<Dataset>



* 1. Explain about the student t-test. What kind of t-test should be implemented to compare the means of gene expression values between disease and normal samples in the microarray data? Explain with proper formula.

\* You can refer to the course textbook and lecture material, but you must write down the answer in your own words.

1. Mainlab activities

* To preprocess microarray data for removing variation
* To select differentially expressed genes(DEGs) from microarray data
  1. Data preparation.

1. Access to NCBI GEO database (<https://www.ncbi.nlm.nih.gov/geo/>)
2. Download series matrix file of “GEO accession code GSE18842” (Non-small cell lung cancer (n=46) vs. healthy control (n=45))

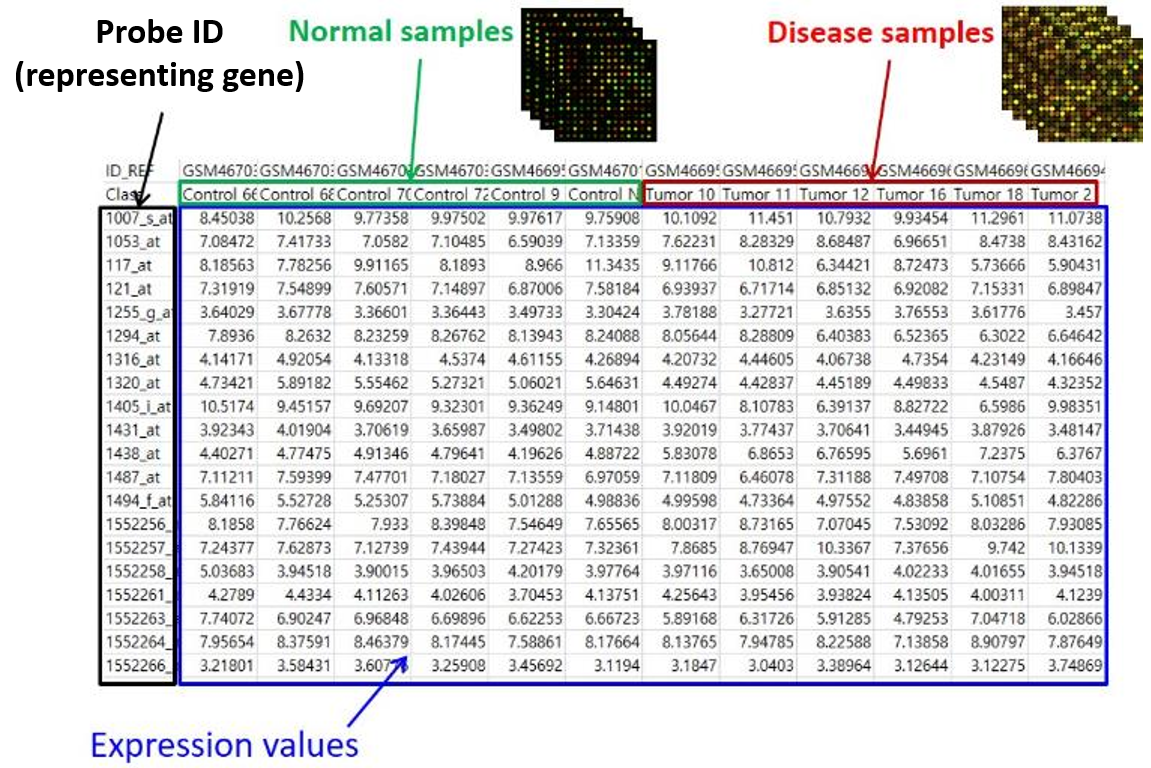


Figure4. Organization of series matrix file (log2-normalized gene expression data)

1. Make data matrix files with “Sample\_characteristics\_ch1” row and expression data. And Remove last line (!series\_matrix\_table\_end )

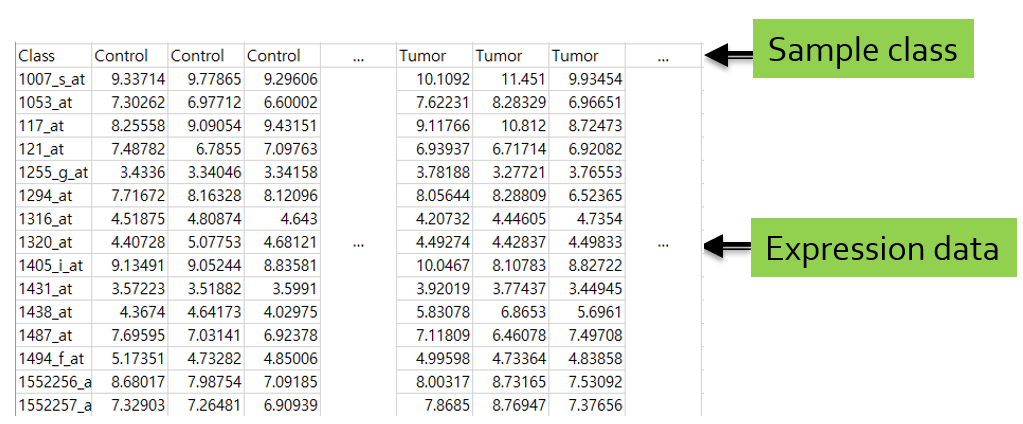


Figure5. Reconstruction of series matrix file for further process

1. Using Excel or other spreadsheet program, export data matrix in the tab-delimited text file format
   1. Quantile Normalization
2. Perform quantile normalization of the data using MATLAB quantile normalization function (refer below “useful functions for this lab”).

**Q. Explain about an effect of quantile normalization**

- Use boxplot about expression data of each sample

* 1. Split dataset

1. In the next session (lab10), we will evaluate selected DEGs from this lab. Discovery dataset and validation dataset must be independent because of the overfitting problem.

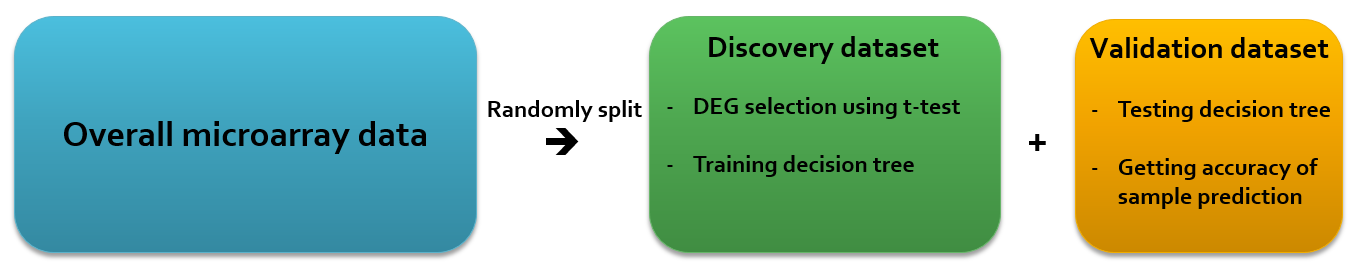


Figure6. Usage of the splitted dataset

1. Divide the quantile normalized dataset randomly into discovery(cancer 35, control 35) and validation dataset(cancer 11, control 10). There should be no overlap between discovery and validation dataset. (validation dataset will be used in lab10)
   1. Get differentially expressed genes
2. Using the above quantile normalized “discovery” dataset, find differentially expressed genes between cancer and control samples based on the student t-test. Perform t-test for each gene between the conditions and select differentially expressed genes which have p-value < 0.05 (refer below “useful functions for this lab”).

**Q. Get DEGs of the dataset and their p-values**

- You should obtain DEGs from the discovery set only.

- Show the text file output

- P-values should be sorted (low to high) and choose your top 100 probe candidates.

Example)

|  |  |
| --- | --- |
| Probe | pvalue |
| 1552473\_at | 1.12E-17 |
| 1563853\_at | 1.65E-14 |
| 200057\_s\_at | 9.03E-12 |
| … | … |

**Q. Export the expression data of top 100 DEGs**

- Show the text file output (which contains normalized value of each genes)

- Prepare ‘discovery dataset’ and ‘validation dataset

Example)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Cancer | Cancer | Cancer | …  …  …  … | Control | Control |
| 1007\_s\_at | 13.4073 | 13.0307 | 12.854 | 12.1972 | 12.2694 |
| 1053\_at | 8.4921 | 8.4 | 9.4716 | 8.0717 | 8.3959 |
| 1255\_g\_at | 6.2366 | 6.5821 | 8.2705 | 7.7682 | 7.5675 |
| … | … | … | … | … | … | … |

**Q. Explain about the results of t-test with proper diagrams**

- Use parameters of mattest() function (quantile plot, histogram, …)

* 1. Useful functions for this lab

tblread() : data import

quantilenorm(): quantile normalization

find(): finding elements within Boolean(TRUE/FALSE) type variables

strncmp()/strmatch()/ismember etc. : compare character vector elements  choose and use for finding cancer and tumor sample in dividing samples

randsample(): random sampling of vector object

mattest(): t-test

tblwrite(): data export

boxplot(): boxplot of the data

You can use any other functions in the MATLAB or you implemented.

1. References
2. Bioinformatics: Sequence and Genome Analysis, 2nd edition, David W. Mount, 2004, Cold Spring Harbor Laboratory Press.