1 CSB2 – Exercise 3: differential expression

Exercise assignments for this week covers the essentials of identifying differential expression in microarray data using the two-sample t-test. The lectures regarding differential expression provided (hopefully) sufficient theoretical background, in the demonstration session these results were utilized in a simulation context, and now, in these exercises, the methods are tested with real data.

The outline of this exercise is thus very similar to what it was in the demonstration session: read data, perform the t-test, and choose a critical region, based on False Discovery Rate (FDR) estimation, where the null hypotheses are rejected and genes are declared differentially expressed.

First, we need to start Matlab and make everything prepared for importing the data.

Table 1: **Set up Matlab**

(1) Open Matlab with the terminal by typing 'matlab -desktop'.

(2) If there is not enough free space in your personal directory (approx. 23MB required), change the work directory of Matlab to 'worktmp'.

(3) Start editing a new script by typing 'edit scriptDiffExpr.m' in the Matlab command prompt.

1.1 Importing data to Matlab

We use the same data that has already been preprocessed with methods introduced earlier in this course, namely the tumor data measured with Illumina bead arrays. The data is stored in a tab-delimited text file 'illuminaData.txt', and it can be downloaded from the course web pages.

Table 2: **Download data**

(1) Download the data from `http://www.cs.tut.fi/courses/SGN-6156/Exercise2.zip`.

(2) Save the data either to your personal directory or to 'worktmp', depending on which work directory you are using.

(3) Unzip the package and delete all files except 'illuminaData.txt'.

(4) Open 'illuminaData.txt' and study the formatting and contents of the file (number of rows, number of columns, column names, probe IDs, table separator, etc.).

(5) Is the data linear or log-transformed?

(6) Find out which columns correspond to cancer tissues and which to healthy tissues.

The Illumina data consists of four replicated measurements of both tissue types. Next we import the numeric data and probe labels into Matlab using the function 'dlmread'.

Table 3: **Import delimited data**

(1) Read the data in 'illuminaData.txt' into matrix 'Data' using the function 'dlmread'; take all columns, including probe IDs, but leave the column headers (first row) out.

(2) Hint: delimiter type is *tab*, *i.e.*, '\t'.

(3) Once read, check that the imported data matrix matches with the data in the text file.

(4) Calculate the number of probes in the data and store that information into the variable 'N'; you can use the function 'size' on the first dimension of 'Data'.

In total, there are 9 columns and 22184 rows to be read, and only the first row, *i.e.*, headers, are excluded. Thus, check that the data matrix has dimensions <22184x9>.

Once this is done, it is time to separate the data in 'Data' into three distinct data matrices, 'ProbeIDs', 'DataHealthy', and 'DataCancer'.

Table 4: **Separate data entities**

(1) As an example, probe IDs can be read from 'Data' with a command 'ProbeIDs = Data(:,1)'.

(2) Use similar notation for creating other required data matrices, 'DataHealthy' and 'DataCancer'.

It is always good to see what your data looks like, and a very easy way to perform this sanity check is to plot the histogram and scatter of the data.

Table 5: **Visualize data**

(1) Plot the histogram of the whole data using the function 'hist'.

(2) Hint: the command '[DataHealthy(:);DataCancer(:)]' produces one long column vector, and it can be used as input for the function 'hist'. Use, for instance, 30 bins.

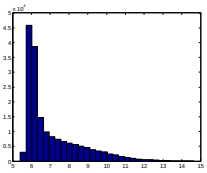
(3) Prepare a colormap 'Cmap' with a command 'Cmap = zeros(N,3)'. This means that, for each probe, black color is used.

(4) Make a scatter plot with the function 'scatter'. Reduce circle size to 3 and use the colormap 'Cmap'.

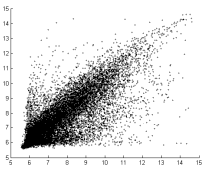
(5) Hint: if you want just one scatter plot for visualizing all data at the same time, use means (*e.g.*, 'mean(DataHealthy,2)' and 'mean(DataCancer,2)') over replicates as inputs for the function 'scatter'.

(6) Hint: for creating a proper log-ratio scatter plot, you can "rotate" the data by replacing the data arguments for 'scatter' with 'mean(DataHealthy,2)+mean(DataCancer,2)' and 'mean(DataHealthy,2)-mean(DataCancer,2)'.

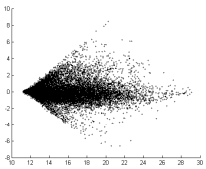
The visualization you get should look like the ones below.



(a) Data histogram



(b) Scatter plot



(c) Rotated scatter plot

1.2 Performing the two-sample t-test

The data is now in a form that makes later tasks easy to execute. Next, we extract p-values from the data using the two-sample t-test.

Table 6: **Extract p-values**

(1) Use the function 'ttest2' to perform the test, and put the p-values into a column vector 'pValues'; in order to make it a column vector, it must be transposed.

(2) Note: both data sets, given as inputs to the function, need to be transposed; you can do transposing on the input line directly.

(3) The significance level, α, can be set to any value between 0 and 1, as here that parameter is not meaningful; we will choose α later on based on our FDR estimate.

(4) Use both tails when calculating the p-values, and assume equal variances between the populations.

Again, visualization is the key to see if everything is in order. p-value histogram plot, for instance, reveals whether our assumptions regarding the data have been relevant or not. More specifically, if a peak near small p-values is visible and the histogram is otherwise flat, everything should be in order.

Table 7: **Visualize p-values**

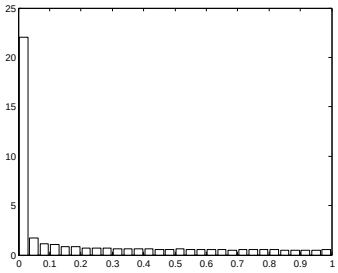
(1) Use the command '[y,x] = hist(pValues,30)' to store the histogram data into vectors x and y.

(2) Normalize the counts in y with the estimated area under the histogram as follows: 'ya = y / trapz(x,y)'.

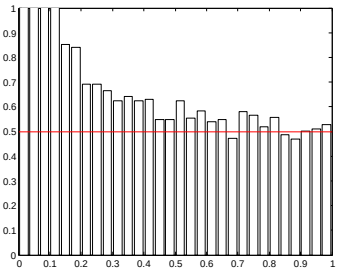
(3) Finally, visualize the normalized histogram with the function 'bar'; use white color, *i.e.*, 'w'.

(4) Estimate the fraction of non-differentially expressed genes, π_0, from the normalized histogram; the estimate is the height of the rectangle that extends from 0 to 1.

(5) Note: the estimate needs to be visualized, but store it into the variable 'pi0Est'.



(d) p-value histogram



(e) Zoomed p-value histogram with π_0 estimate visualized

1.3 FDR estimation

It was derived in the lectures that an estimator for the expected fraction of false positives within the set of significant genes (false positives divided by positives),

$$\mathrm{FDR}(\alpha) = \mathbb{E}\left[\frac{\mathrm{FP}(\alpha)}{\mathrm{P}(\alpha)}\right],$$

can be given, according to Storey *et al.*, as

$$\widehat{\mathrm{FDR}}(\alpha) = \frac{N\alpha\hat{\pi}_0}{S(\alpha)},$$

where $S(\alpha)$ is the number of genes declared differentially expressed. Next, we implement the estimator and go through different α values and calculate the estimated FDR.

Table 8: **Estimate FDR**

(1) Make a column vector 'alphaVec' that contains values $0.01, 0.02, \ldots, 0.99, 1.00$: 'alphaVec = (0.01:0.01:1.00)'; the result needs to be transposed.

(2) Make a column vector of zeros: 'FDREstVec = zeros(100,1)'; 'alphaVec' has 100 elements, too.

(3) Make a loop in where one 'alpha' at a time from 'alphaVec' is picked, estimated FDR is computed, and the result is stored into 'FDREstVec'.

(4) Hint: $S(\alpha)$ can be calculated simply with the command 'S = sum(pValues < alpha)'; the number of genes having p-value below 'alpha'.

You should now have a column vector of α-values, and for each element in the α-vector, α_i, $\widehat{\text{FDR}}(\alpha_i)$ is associated. Now, let us plot this mapping from αs to FDRs and vice-versa.

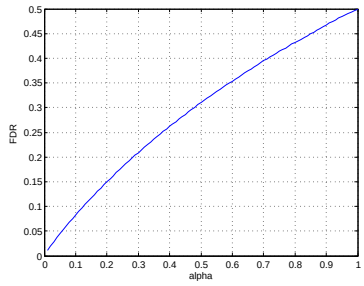
Table 9: **Visualize the mapping between αs and FDRs**

(1) Use the function 'plot' to plot the mapping between 'alphaVec' and 'FDREstVec'.

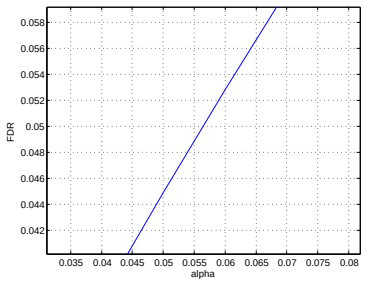
(2) Use the functions 'xlabel' and 'ylabel' to attach labels on the axes.

(3) Use the command 'grid on' to visualize a grid.

(4) Zoom closer where 'FDREstVec' is close to 0.05 and see what does that value correspond to in 'alphaVec', and store that α-value into the variable 'alphaFinal'.



(f) Mapping



(g) Zoomed mapping

1.4 Identify differentially expressed genes

It is now time to actually identify the differentially expressed genes, and it is possibly the simplest step to do.

Table 10: **Identify differentially expressed genes**

(1) Compute a list of binary indicators whether a p-value is below the calculated significance level, 'alphaFinal'.

(2) Hint: try the command 'H = pValues < alphaFinal'.

(3) Note: FDR should now be fixed to 0.05.

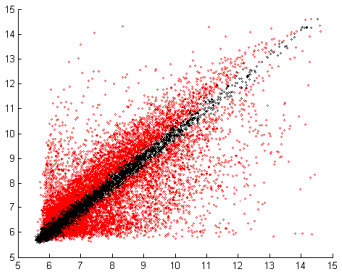
It is also nice to see what does the result look like, *i.e.*, what points in the scatter plot correspond to differentially expressed genes.

Table 11: **Visualize data with differentially expressed genes**

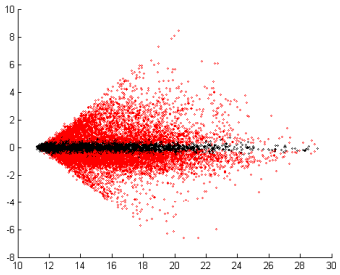
(1) Update the colormap 'Cmap' with the command 'Cmap(H,1) = 1'; this makes all the points corresponding to differentially expressed genes red.

(2) Use otherwise same inputs for the function 'scatter'.

(3) It appears that by this approach we find a huge number of differentially expressed genes. Are the results still satisfying?



(h) Scatter plot



(i) Rotated scatter plot