Bioinformatics III

Prof. Dr. Volkhard Helms
Daria Gaidar, Markus Hollander, Duy Nguyen, Thorsten Will
Summer Semester 2018

Saarland University
Chair for Computational Biology

Exercise Sheet 8

Due: June 19, 2018 10:15

Submit your solutions on paper, hand-written or printed at the *beginning* of the lecture. Alternatively, you can send an email with a single PDF attachment to markus-hollander@web.de. Your submission should include code listings for programming exercises. Additionally, hand in a .zip file with your source code via email. Comment your code!

Gene Expression and DNA Methylation

In this exercise sheet you use several correlation measures to investigate gene expression and DNA methylation data of several cell and tissue types, construct and visualise co–expression networks and examine if blood cells and skin tissues can be distinguished by their gene expression and DNA methylation.

Exercise 8.1: Data Preprocessing (20 points)

The supplement contains a gene expression and a DNA methylation data set of 100 genes from 19 samples. The samples HSC, MPP1, MPP2, CLP, CMP, GMP, MEP, CD4, CD8, B_cell, Eryth, Granu and Mono are from blood cells, whereas the samples TBSC, ABSC, MTAC, CLDC, EPro and EDif are from skin tissues.
The values in the data sets are already normalised, but still contain entries with empty or unknown values that need to be removed, as well as multiple entries for one gene that need to be merged before you can work with the data.

(a) **Data matrix:** The supplement contains the `data_matrix.py`–file with the outline of a *DataMatrix*–class in which you should implement the following functions:

  (1) *read_data()*: Read tab–separated tables where the first line gives the column names and the first column gives the row names. Remove rows without name or that contain empty or non–numerical values. If there are several rows with the same name, merge them into a single row by taking the mean at each position of those rows.

  (2) *get_columns()* and *get_rows()*: Return a dictionary with the row/column names as keys and corresponding observation lists as values. You need this for later exercises.

  (3) *not_normally_distributed(alpha, rows)*: Many statistical analysis methods make assumptions about the underlying distribution. The Shapiro–Wilk test is used to test the null–hypothesis that a list of observations comes from a normal distribution.

  Use the Shapiro–Wilk test to compute and return the names and p–values of rows with $p < alpha$. The parameter *rows* takes a boolean value that specifies whether this should be done for rows or columns. You can use the Shapiro–Wilk test from the *scipy* module.

  (4) *to_tsv(file_path)*: Write the processed matrix into a tab–separated file with the same format as the input matrices. The columns should be in the same order as in the input file and the rows should be in lexicographical order of their name.

(b) **Process expression and methylation data:** In the function *exercise_1()* in `main.py`, use your *DataMatrix*–class to read in the expression and methylation tables given in the supplement and write the processed matrices into files. Submit each matrix file with the following names:

- `lastname1_lastname2_expression.tsv`
- `lastname1_lastname2_methylation.tsv`

For each input file, report the number of genes and samples whose data does not follow a normal distribution with $\alpha = 0.05$.

Exercise 8.2: Correlation Measures (20 points)

Gene expression or DNA methylation are often investigated using various correlation coefficients. Implement the following functions in `correlation.py`:

(a) *rank(X)*: The Spearman and Kendall correlation coefficients consider the ranking (sort order) of values. To compute the ranking of a value list X:

(1) Compute a sorted version X_s of X, in descending order.
(2) Create a new list X_r that contains the index of each value of X in X_s.
(3) Return X_r.

If X contains a value v multiple times, then all occurrences of v are assigned the mean of their ranks. For example, for a list $X = [6, 6, 4, 2, 10]$ the ranking is $X_r = [1.5, 1.5, 3, 4, 0]$.

(b) *person_correlation(X, Y)*: The Pearson correlation coefficient ρ uses the sample covariation and sample standard deviation σ to compute the linear correlation between two observation lists X and Y of length n as follows

$$\rho = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}},$$

where $\bar{X}$ and $\bar{Y}$ are the means of the respective list. Return the Pearson correlation coefficient for the input lists.

(c) *spearman_correlation(X, Y)*: The Spearman correlation coefficient calculates a non–parametric correlation by computing the Pearson correlation coefficient on the ranking of two observation lists X and Y of length n. Return the Spearman correlation coefficient for the input lists.

(d) *kendall_correlation(X, Y)*: The Kendall correlation coefficient τ_B computes a non–parametric correlation by computing the concordant and discordant pairs in the ranking of two observation lists X and Y of length n, while considering tied pairs.

(1) Compute the rankings X_r and Y_r of the input lists.
(2) Pair the rankings as follows: $(X_{r,1}, Y_{r,1}), (X_{r,2}, Y_{r,1}), ..., (X_{r,n}, Y_{r,n})$.
(3) Compute the number of concordant pairs n_c and discordant pairs n_d by going through all (unique) combination of pairs $(X_{r,i}, Y_{r,i})$ and $(X_{r,j}, Y_{r,j})$ with $i \neq j$. A pair is *concordant* if

- $X_{r,i} < X_{r,j}$ and $Y_{r,i} < Y_{r,j}$ **or**
- $X_{r,i} > X_{r,j}$ and $Y_{r,i} > Y_{r,j}$.

A pair is *discordant* if

- $X_{r,i} < X_{r,j}$ and $Y_{r,i} > Y_{r,j}$ **or**
- $X_{r,i} > X_{r,j}$ and $Y_{r,i} < Y_{r,j}$.

Also compute the number of tied pairs n_X with $X_{r,i} = X_{r,j}$ and the number of tied pairs n_Y with $Y_{r,i} = Y_{r,j}$. A pair with $X_{r,i} = X_{r,j}$ **and** $Y_{r,i} = Y_{r,j}$ does not count towards n_X and n_Y.

(4) Compute the Kendall correlation coefficient as

$$\tau_B = \frac{n_c - n_d}{\sqrt{(n_c + n_d + n_X)(n_c + n_d + n_Y)}}.$$

Return the Kendall correlation coefficient for the input lists.

Exercise 8.3: Gene Co–Expression Networks (30 points)

Co–expression of genes is a possible indicator that those genes are part of the same process or pathway, functionally related, or regulated by the same transcriptional programs.

(a) **Network construction:** `Correlation.py` contains the already implemented *CorrelationMatrix*–class, and `network.py` contains the outline of the *CorrelationNetwork*–class. In the latter, implement the following functions:

(1) *init(correlation_matrix, threshold)*: Use the *CorrelationMatrix* to add undirected edges between nodes with absolute correlation $\geq$ *threshold*.

(2) *to_sif(file_path)*: The simple interaction format (SIF) is a basic, tab–separated format without header that can be read by many network visualisation tools.

- **Column 0:** label of the source node
- **Column 1:** interaction type
- **Columns 2+:** label of target node(s)

The interaction type should be the correlation value rounded to two decimal places. The file should include interactions only once, meaning that if you already included "n_1 0.75 n_2", do not include "n_2 0.75 n_1".

(b) **Network visualisation:** In the function *exercise_3()* in `main.py`, use your implementation to construct gene co–expression networks for the expression and methylation data tables with the Pearson, Spearman and Kendall correlation coefficient with *threshold* $= 0.75$. This should give you a total of 6 SIF files that you should submit with the following names:

- `lastname1_lastname2_expression_network_pearson.sif`
- `lastname1_lastname2_expression_network_spearman.sif`
- `lastname1_lastname2_expression_network_kendall.sif`
- `lastname1_lastname2_methylation_network_pearson.sif`
- `lastname1_lastname2_methylation_network_spearman.sif`
- `lastname1_lastname2_methylation_network_kendall.sif`

Visualise each network file with the open source network visualisation software Cytoscape. (You do not have to submit images of the networks, but it will help with the discussion.)

(c) **Discussion:** Briefly comment on the similarities and difference between the networks. Explain and discuss your results.

Exercise 8.4: Hierarchical Clustering (30 points)

In the previous task you investigated the correlation between genes. In this task you explore the correlation between samples and use hierarchical clustering to examine if gene expression and DNA methylation can be used to correctly distinguish between tissue types.
Hierarchical clustering methods use a distance metric $d(a, b)$ that measures how similar two individual observations a and b are, and a linkage criterion to determine the similarity of sets of observations and which clusters to combine next.
In this task you are going to implement a bottom–up (also called agglomerative) approach that uses the average linkage criterion and correlation as the metric:

(a) **Implementation:** Implement the following functions in the *CorrelationClustering*–class in `cluster.py`:

(1) *average_linkage(cluster_A, cluster_B)*: Given two clusters A and B with m and n observations, respectively, return the average linkage

$$l(A,B) = \frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} |d(A_i, B_j)|.$$

(2) *cluster()*: The bottom–up approach starts with each observations as a single cluster and performs the following steps until there is only one cluster containing all observations:

i. Compute the linkage criterion for all pairs of clusters.

ii. Select the two clusters with the highest linkage criterion.

iii. Add the two clusters and their linkage value to the trace.

iv. Merge the two clusters.

(3) *trace_to_tsv(file_path)*: Write the clustering trace into a tab–separated file, where each line represents a clustering step. Columns 0 and 1 should each contain the names in a cluster, separated by a comma. Column 2 should contain the linkage value.
For example: "HSC, MPP1 MMP2, CLP, CMP 0.93"

You can use the *Cluster*–class in `cluster.py` to represent clusters.

(b) **Application:** In the function *exercise_4()* in `main.py`, use your implementation to hierarchically cluster the expression and methylation data tables with the Pearson, Spearman and Kendall correlation coefficient. This should give you a total of 6 TSV files that you should submit with the following names:

- `lastname1_lastname2_expression_cluster_pearson.tsv`
- `lastname1_lastname2_expression_cluster_spearman.tsv`
- `lastname1_lastname2_expression_cluster_kendall.tsv`
- `lastname1_lastname2_methylation_cluster_pearson.tsv`
- `lastname1_lastname2_methylation_cluster_spearman.tsv`
- `lastname1_lastname2_methylation_cluster_kendall.tsv`

(c) **Discussion:** Can hierarchical clustering be used to differentiate between blood cells and skin tissues? Are there differences between the correlation coefficients or data type? Why?