

### Assignment Goals

- i. Use mutual information to reconstruct gene expression networks.
- ii. Develop working knowledge of genome-wide association studies (GWAS).
- iii. Understand and experiment with multiple testing correction procedures.
- iv. Use Gaussian processes to model biological time series.

### Submission Instructions

- To turn in your assignment, please log in to the server **pluto.biostat.wisc.edu** using your **pluto** server username and password.
- Copy all relevant files to the directory

**/home/medinfo/bmi776-2022/hw2/<USERNAME>**

where **<USERNAME>** is your NetID. Submit all of your Python source code and test that it runs on the biostat server.

- For the rest of the assignment, compile all of your answers in a single file and submit as **solution.pdf**.
- Write the number of late days you used at the top of **solution.pdf**.
- For the written portions of the assignment, show your work for partial credit.

### Part 1: Mutual information in regulatory networks (50 points)

In class, we saw how FIRE uses mutual information to detect relationships between sequence motifs and gene expression levels. Information-theoretic algorithms are also popular for reconstructing transcriptional regulatory networks from gene expression profiles. Given the expression levels for a set of genes measured in a sufficient number of biological conditions, mutual information (MI) can detect certain types of pairwise dependencies that may suggest one gene is a regulator (e.g., a transcription factor) and another is its target. For this assignment, we will create simple undirected gene-gene networks by ranking and thresholding the MI of gene pairs.

Write a program, **CalcMI.py**, that takes as input the expression data for a set of genes across a number of biological conditions and outputs the list of gene-gene dependencies and their MI. It should only consider the dependencies between unique genes, not the MI of a gene and itself (i.e., the entropy of that gene's expression). Compute MI by discretizing the gene expression levels, mapping continuous values into discrete bins. For a pair of genes  $G1$  and  $G2$ , construct a count matrix that tracks the number of times  $G1$ 's expression is in some bin  $a$  and  $G2$ 's expression is in some bin  $b$ . Add a pseudocount of 0.1 to all entries in the count matrix. From this count matrix, estimate  $P(G1 = a)$ ,  $P(G2 = b)$  and  $P(G1 = a, G2 = b)$  needed for calculating the MI between  $G1$  and  $G2$ .

Please implement the following two binning strategies:

- *Equal size binning.* This results in equal-sized bins. For example, if a gene's expression values lie in the range [1, 11] and **we assign them into four bins**, the bins would be [1, 3.5), [3.5, 6), [6, 8.5), and [8.5, 11].
- *Equal density binning.* This uses a percentile-based assignment to discretize expression values. With **two bins**, the lowest 50% of a gene's expression values would be mapped to bin 0 and the highest 50% would be mapped to bin 1.

Your program should be callable from the command line as follows:

```
python CalcMI.py \  
    --bin_num=<bins> \  
    --bin_str=<strategy> \  
    --out=<out> \  
    <dataset>
```

*For example :*

```
python3 CalcMI.py \  
    --bin_num=5 \  
    --bin_str="size"\  
    --out="example1_uniform_5_bins.txt"\  
    "example1.txt"
```

where

- **<dataset>** is a text file containing gene expression values. The first line of the file provides the column labels. The first column is the time point, which you will not need. The other columns contain the temporal expression profile of genes, each labeled with a numeric index. The file is in a tab-delimited format.
- **<bins>** is the number of bins into which you should assign the continuous gene expression values when calculating the MI.
- **<strategy>** is the binning strategy ("*size*" for equal size binning and "*density*" for equal density binning).
- **<out>** is the name of the text file into which the program will print *all* unique gene pairs and their MI values line by line. Round MI to three decimal places, and print the lines in descending order of the rounded MI values. Break ties

based on the index of the first gene and then the index of the second gene if needed, sorting gene indexes in ascending order.

Example input files (example1.txt and example2.txt), their corresponding output files, and the template CalcMI.py with argument parsing code can be found in the *hw2\_files* directory. Your program will be evaluated on the example inputs and additional datasets that will be kept private.

## **Part 2: Genome-Wide Association Study (15 points)**

Please suppose that we perform a GWAS for a disease of interest (e.g. Type 1 diabetes or Coronary Heart Disease) and obtain the summary tables below for 1 particular SNP (Single Nucleotide Polymorphism) in the genome. The subjects in this study come from 1 of 2 distinct populations, A and B.

Population A			
	CC	CG	GG
<i>Disease</i>	42	44	58
<i>Control</i>	110	31	14

Population B			
	CC	CG	GG
<i>Disease</i>	480	582	458
<i>Control</i>	455	611	505

- (A) Use a Pearson's  $\chi^2$ -test and an Armitage test for linear trend to determine if there is an association between the genotype at this SNP and disease status within population A. For each test, state the null and alternative hypotheses, calculate the test statistic by hand, report the *p*-value, and state your conclusion.
- (B) Please repeat the same tests for population B using the provided code in **gwas\_tests.py**.
- (C) Suppose that we did not record which population each subject came from. Repeat the same tests for the entire set of subjects using the provided code (**gwas\_tests.py**). Please discuss your results in (C) in light of your results from (A) and (B).

### **Part 3: Multiple testing correction (20 points)**

In a study that produces thousands of  $p$ -values, such as GWAS and RNA-Seq differential expression analysis, we need to apply a multiple testing correction procedure to adjust the  $p$ -values based on the number of tests performed. Here, you will investigate the statistical power of several such procedures for the task of identifying differentially expressed genes from a microarray dataset.

The dataset, **log2counts.csv**, consists of the  $\log_2$ -counts of 3,170 genes measured from seven *BRCA1*- and eight *BRCA2*-mutation-positive tumor samples. The genes with outlier counts have been removed. The nominal  $p$ -values are calculated using a permutation test as detailed in Remark C in the Appendix of Storey & Tibshirani (2003).

The base code in **MTC.py** guides you through data loading, permutation testing and plotting of  $p$ -values. Your task is to implement Bonferroni, Benjamini-Hochberg, and Storey-Tibshirani multiple testing correction procedures. All procedures take as input a list of *sorted*  $p$ -values and a significance threshold  $\alpha$ , and return the indices of genes that are considered differentially expressed (i.e., significant). The Storey-Tibshirani procedure takes an additional parameter,  $\lambda$ , estimated visually from a *density* histogram of the  $p$ -values.

Your program should be callable from the command line as follows:

```
python MTC.py \  
    --procedure=<procedure> \  
    --alpha=<alpha> \  
    --lamb=<lambda> \  
    --fig=<fig> \  
    <counts>
```

where

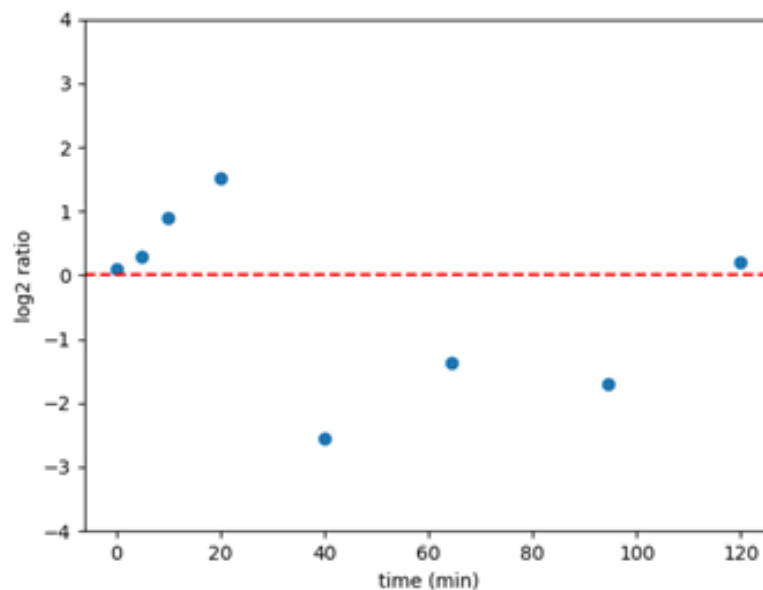
- **<counts>** is a text file containing the counts for the samples, one gene per row.
- **<procedure>** is the correction procedure ("*bf*" for Bonferroni, "*bh*" for Benjamini-Hochberg and "*st*" for Storey-Tibshirani) for processing the  $p$ -values.
- **<alpha>** is the significance threshold, default to 0.05.
- **<lambda>** is the estimated  $\lambda$  used by the Storey-Tibshirani procedure. It is not used by the other two procedures.
- **<fig>** is the path where the generated  $p$ -value plot is saved.

The program outputs a plot of sorted  $p$ -values in which the significant genes are colored in red. Run the program with all three procedures. Please discuss the plots.

**Part 4: Gaussian processes for biological time series (15 points)**

Suppose we are studying how neural stem cells respond to an environmental toxin. We perform RNA-Seq on cells in the control and treatment groups to obtain gene expression data at 0, 5, 10, 20, 40, 60, 90 and 120 min. Gaussian processes (GP) with a squared exponential kernel are well suited for modeling biological data collected over a time course. Following GP regression, the posterior mean is smooth over time, and the confidence intervals track uncertainty between the measured time points.

- (A) Shown below is a time series of  $\log_2$  fold change of gene expression (i.e.,  $\log_2$  ratio between treatment and control). Assume a GP prior with zero mean and a squared exponential kernel. Please sketch the mean and 95% confidence interval of the GP posterior (note the irregular time intervals). You may further assume that the kernel parameters (i.e., length scale  $l$  and variance  $\sigma^2$ ) have been optimized to maximize the data likelihood. Explain your reasoning. (5 points)



- (B) Differential expression analysis and clustering are the natural first steps toward understanding gene functions. Describe a GP-based statistical test that can be applied separately to each gene to assess whether its temporal expression profile in the normal condition differs from that under drug exposure, or a GP-based

probabilistic model that clusters genes by their temporal expression profiles to reveal shared biological functions. You are encouraged (but not required) to work with the  $\log_2$  ratios as in (A). Clearly state and justify your test/model formulation as well as your assumptions. **(10 points)**