Manual of the `MATLAB` pipeline for the analysis of genomic time course data from the *Gene Expression Omnibus*

Juan Camilo Ramírez

January 5, 2017

Contents

List of Figures

List of Tables

1 Summary

1. The pipeline analysis is performed individually for each experimental condition.

2. An experimental condition is composed of several GEO samples, all associated to a GEO series.

3. The information of each sample must be provided in an input file located in folder Input.

4. Several condition files can be deposited in folder Input and the pipeline analysis will be performed on all the conditions.

5. The analysis results for all condition files given as input are output in folder Results.

6. Section 1 explains how to prepare the condition files that will be given as input.

7. Section 2 explains how to run the analysis on the condition files created in Section 1.

2 Theoretical basis and terminology

2.1 Experimental conditions

2.2 Experimental macroconditions

2.3 Replicates

Replicates can be *biological* or *technical*.[1]

2.4 Genes' between-replicate noise

Genes may exhibit variation across replicates from the same condition. This variation, referred to as the *gene's between-replicate noise* and denoted by δ, is measured for each gene as follows. Let $x^s_{i,t}$ be the expression level of replicate s's i-th gene at time t and let S be the total number of replicates. Let $\sigma_{i,t}$ be the standard deviation of the expression level of the i-th gene at time t across the S replicates. The gene i's between-replicate noise, denoted by δ_i,[2] is given by

$$\delta_i = \frac{1}{T} \sum_{t \in t_1}^{t_T} \sigma_{i,t}. \tag{1}$$

Section 9 introduces the `MATLAB` function provided for the calculation of the genes' between-replicate noise from a group of replicates.

[1]Biological Replicates vs Technical Replicates.

[2]Another possibility is to define δ_i in terms of the coefficient of variation instead of the standard deviation. In other words, the between-replicate noise could be defined as $\delta_i = \frac{1}{T} \sum_{t \in t_1}^{t_T} \frac{\sigma_{i,t}}{\mu_{i,t}}$, where $\mu_{i,t}$ is the mean of the expression level of the i-th gene at time t across the S replicates. However, this measure is unreliable given that expression levels are interval scale measures and the coefficient of variation is reliable only in ratio scale measures.

3 The pipeline analysis

4 Input files

4.1 Condition files

Condition files are the input files for the pipeline analysis of one or more experimental conditions. The information of each experimental condition must be provided in a text file. One file for each condition. The condition files must be stored in folder `Input`. Each condition file must be named with the following format: `[GEO SERIES NUMBER]_-_[NAME OF EXPERIMENTAL CONDITION]_-_[NUMBER OF TOP DRGs FOR CLUSTERING].txt`.

All the samples associated with the experimental condition must be provided along with the time points using the format described as follows. Each line of the file must consist of the time point, including the time unit (e.g., 2 hours), followed by a comma (,) and then followed by the accession number of one sample. For example, the file for condition "D10" of *GEO* series `GSE59015` must be named `GSE59015_-_D10_-_3000.csv` and the contents must be as follows.

```
0 hours,GSM1424453
6 hours,GSM1424454
12 hours,GSM1424455
18 hours,GSM1424456
24 hours,GSM1424457
30 hours,GSM1424458
36 hours,GSM1424459
42 hours,GSM1424460
```

The condition file can group different replicates that represent the same experimental condition. In this case the pipeline will run one single analysis with the average of all the replicates specified in the file. For example, the file for all the replicates in *GEO* series `GSE41067` must be named `GSE41067_-_ALL_-_10000.csv` and the contents must be as follows.

```
0 hours,GSM1008154,GSM1008162,GSM1008170
1 hours,GSM1008155,GSM1008163,GSM1008171
2 hours,GSM1008156,GSM1008164,GSM1008172
4 hours,GSM1008157,GSM1008165,GSM1008173
6 hours,GSM1008158,GSM1008166,GSM1008174
8 hours,GSM1008159,GSM1008167,GSM1008175
10 hours,GSM1008160,GSM1008168,GSM1008176
12 hours,GSM1008161,GSM1008169,GSM1008177
```

The condition files can be written manually, following strictly the format described above. Optionally, this task can be carried out more easily by using script `create_input_files.m`. In order to do this, `create_input_files.m` must be run and the instructions provided thereafter by the program must be followed. More details in Section 5.

4.2 Macrocondition files

The conditions comprising the macrocondition must be provided in a text file that must be located in folder `Input`. This file must be named using the following format: `[GEO SERIES NUMBER]_-_[NAME OF MACRO CONDITION].txt`. For example, the following are the contents of a macrocondition of five H3N1 subjects from `GSE52428`.

```
H3N1_001
H3N1_002
H3N1_003
H3N1_004
H3N1_005
```

This file can be constructed manually, or with BASH script `prepare_input.sh`. This option requires running the script with the following syntax.

```
./prepare_input.sh [GEO SERIES] [NAME OF MACRO CONDITION].txt
```

Example

```
./prepare_input.sh GSE52428 H1N1.txt
```

The above will read ALL the conditions whose analyses have been completed for the GEO series indicated and write the file with the format described earlier.

5 `create_input_files.m`

The (experimental) condition files for the pipeline analysis can be created manually with the format described in Section 4.1 or with the help of script `create_input_files.m`. In order to do the latter, the command `create_input_files` must be executed on the `MATLAB` console from the pipeline directory, as shown in Figure ??.

Once all condition files are located in folder Input, then the analysis can be started by running pipeline.m. The script will read ALL the condition files in folder Input and run the analysis for each one of them.

6 `integrated_analisis.m`

`integrated_analisis.m`: input is a macrocondition file. See Subsection 4.2.

7 `annotate.m`

`annotate.m`: input is a condition file. See Subsection 4.1.

8 `compare.m`

`compare.m`: input is a macrocondition file. See Subsection 4.2.

9 `measure_fit_of_replicates.m`

This is a script that calculates the between-replicate noise for all genes across a number of replicates, as shown in Subsection 2.4. Its input is a macrocondition file, as defined in Subsection 4.2. Results will be output in folder `Pipeline_HOME/Output/[GSE Series]/Comparison_of_replicates/[Macrocondition]/`.

This folder contains a file named `[Macrocondition]_noise_per_gene_ALL_GENES.csv`. Each line of this file corresponds to a probe in the GSE matrix and contains three values separated by commas (,). The first field is the probe ID, the second is the gene name, and the third is the gene's between-replicate noise. The probes are listed in this file in the same order they appear in the GSE matrix. A secondary file, named `[Macrocondition]_noise_per_gene_ALL_GENES_SORTED_BY_NOISE.cs` provides the same information with the probes listed by their between-replicate noise in decreasing order. That is to say, probes that appear on top are the ones whose expression is most consistent across the probes provided in the macrocondition file whereas those at the bottom are the ones exhibiting the most discrepancy in expression across these replicates.

10 Simulation of time course data

The robustness of the pipeline process can be validated by simulating gene expression data and verifying if the pipeline returns the same model the simulated data originate from. More specifically, expression data can be simulated from given a GRN and set of GRMs and given as input to the pipeline in order to verify that the algorithm approximates the original GRN and GRMs. In order to achieve this, two types of the gene expression curves need to be generated: those from DRGs and those from non-DRGs. First, the expression curves of the DRGs are simulated from the mean curves of the GRMs as follows. Let K be the matrix representing the expression of one gene response module composed of n genes over t time points, given by

$$K = \begin{pmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,t} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,t} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n,1} & X_{n,2} & \cdots & X_{n,t} \end{pmatrix}.$$

Also let M be the tuple of per-column means of K, given by

$$M = \begin{pmatrix} \mu_1 & \mu_2 & \cdots & \mu_t \end{pmatrix},$$

and let S be the per-column standard deviation of K, given by

$$S = \begin{pmatrix} \sigma_1 & \sigma_2 & \cdots & \sigma_t \end{pmatrix}.$$

Then n simulated curves are obtained where the i-th element is a normal deviate of μ_i with standard deviation σ_i. This process is repeated for all the GRMs until obtaining the simulated expression curves of all the DRGs. Thereafter, the non-DRG curves are simulated as normal deviates of the per-time point means of all the DRG curves, with standard deviation 1.0. The simulated DRG and non-DRG expression curves can then be passed as arguments to the pipeline and see if the GRMs and GRN obtained approximate the originals.

The procedure described above is performed by script `run_simulation.m`, which reads the results of a previously-completed analysis from folder `Output`, constructs the simulated data and runs the pipeline in folder `Output/[Condition]_simulated`, where `[Condition]` is the name of the original experimental condition.