Manual of the MATLAB pipeline for the analysis of genomic time course data from the *Gene Expression Omnibus* (Version 1.55)

March 14, 2017

Contents

List of Figures

List of Tables

1 The Pipeline for Dynamic Gene Expression Data

The *Pipeline for Dynamic Gene Expression Data*, also referred to as `Pipeline4DGEData`, is the implementation of the pipeline proposed by Carey et al. (2017). It is written in `MATLAB` (R2016a). The tool is designed to analyze time course data from the *Gene Expression Omnibus* (GEO) in order to discover gene regulatory networks from high-throughput data.

Section 2 provides a very brief summary of the theoretical concepts involved in the usage of `Pipeline4DGEData`. However, the reader is strongly encouraged to refer to the GEO documentation and the work of Carey et al. (2017) for full details. Section 3 provides installation instructions and Section 4 illustrates the usage of `Pipeline4DGEData`. The example presented in the latter should be used as basis for other analyses to be performed independently by the reader.

2 Theoretical basis and terminology

2.1 The *Gene Expression Omnibus (GEO)*

The *Gene Expression Omnibus* (GEO)[1] is a public repository administered by the *National Center for Biotechnology Information* (NCBI)[2] for the storage of high throughput data. The GEO stores microarray data in MIAME compliant formats and makes it publicly available. The data is organized into series, samples and platforms.

2.2 GEO Series

A data series is a set of related samples. It may also provide information of the investigation associated to the contained samples. The GEO identifies each series by an accession number with the prefix 'GSE'. For example, series `GSE59015`.

[1] `https://www.ncbi.nlm.nih.gov/geo/`

[2] `https://www.ncbi.nlm.nih.gov`

2.3 GEO Samples

The data obtained from an experimental sample. It also describes how the experiment was conducted and how the data was handled or manipulated. The GEO identifies each sample by an accession number with the prefix 'GSM'. For example, samples `GSM1424445` and `GSM1424453`, which are associated to series `GSE59015`.

2.4 GEO Platforms

A description of the microarray used to collect a sample. The GEO identifies each platform by an accession number with the prefix 'GPL'. For example, platform `GPL17880`, which is associated to the samples in series `GSE59015`.

2.5 Experimental conditions

An experimental condition is the set of time-independent characteristics or parameters of a sample and/or the circumstances in which it was taken. Usually the main characteristic differentiating a condition from a related one is a stimulus (or lack of), which normally consists of a treatment of viral infection. Depending on the study, other characteristics may be present. These may include, the subjects gender or cell type used, for instance. The experimental condition excludes the time point, if any.

In series `GSE59015` there are two experimental conditions, namely `Wildtype` and `D10`, each one comprising eight samples.

2.6 Study

A study is a set of samples that refer to a common domain of experimental conditions. A common domain of experimental conditions may consist of, for instance, the presence of a stimulus (treatment or viral infection) and the absence of it. In most cases, the series and study coincide. In these cases, all the samples contained in the series belong to the same study. However, it is possible that the GEO stores series that encompass samples from varying domains. In these cases, the samples contained in the series are grouped into different studies.

2.7 Experimental macroconditions

2.8 Replicates

Replicates can be *biological* or *technical*.[3]

2.9 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 $x_{i,t}^s$ 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,[4] 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.

3 Installation of `Pipeline4DGEData`

The system requirements of `Pipeline4DGEData` are listed below.

- `MATLAB` R2016a or newer.

The sources of `Pipeline4DGEData` are available on its official `GitHub` repository. The latest version is always availabe for download from here as a compressed folder. Once downloaded, this folder must be decompressed. By default, the decompressed folder is named `Pipeline4DGEData-master/`. In

[3] Biological Replicates vs Technical Replicates.

[4] 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.

this manual it will be assumed that this folder has been renamed `Pipeline4DGEData/` and so it will be referred to as such throughout this document. This folder is the home directory of `Pipeline4DGEData`.[5] User-defined input for new analyses to be run must be provided in subfolder `Input` (details in Section 5) and the program returns the results in subfolder `Output` (details in Section 6).

`Pipeline4DGEData` uses the *Functional Data Analysis for MATLAB* library, which is available from its official site online. The library can be downloaded from this direct link as a compressed folder. Once downloaded, the decompressed folder (named `fdaM/`) must be copied to `Pipeline4DGEData/lib/`. That is to say, all the sources of the library must be in path `Pipeline4DGEData/fdaM/lib/`. After this, attached file `mean_grouped.m` must be copied into `Pipeline4DGEData/lib/fdaM/@fd/`.

`Pipeline4DGEData` also requires the *SBEToolbox (A Matlab Toolbox for Biological Network Analysis)* library (Konganti et al., 2013) (Version `1.3.3` or later). This library is available from its official `GitHub` repository. It can be downloaded from this direct link as a compressed folder. Once downloaded, the decompressed folder (named `SBEToolbox-1.3.3/`) must be copied to `Pipeline4DGEData/lib/`. That is to say, all the sources of the library must be in path `Pipeline4DGEData/lib/SBEToolbox-1.3.3/`.

Once all the above is done, `Pipeline4DGEData` will be ready for use. Details in Section 4.

3.1 Passive mode for FTP downloads

`MATLAB`'s native libraries do not include a passive mode for FTP connections. This may prevent `Pipeline4DGEData` from being able to download series from the `GEO` when running behind a firewall. For this reason, passive mode is enabled by default by using the community-contributed solution available here. These sources are included by default in `Pipeline4DGEData/lib/FTP/`. All credits and acknowledgements go to the original authors of these.

4 Usage of `Pipeline4DGEData`

This section shows how to use `Pipeline4DGEData` for the analysis of a set of experimental conditions by illustrating its usage on GEO series `GSE59015`.

[5] Care should be taken in order not to confuse `Pipeline4DGEData` (the tool) and `Pipeline4DGEData/` (the home folder of the tool).

Figure 1: Samples associated to series `GSE59015`.

Application of `Pipeline4DGEData` to any other GEO series should be analogous. It is assumed here that has been installed as described in Section 3.

GEO series `GSE59015` refers to a study about the resistance of *Plasmodium falciparum* (*i.e.*, the malaria parasite) to antimalarial drugs for the design of new medications. Two experimental conditions (Section 2.5 provides an introduction to experimental conditions) are considered: `Wildtype` and `D10`. The latter refers to *P. falciparum* cells that have been exposed to a medication and the former refers to a cells that haven't been exposed to any stimulus and thus serves as the control group. The list of samples is shown in Figure 1.

The following checklist provides the information that is required for running a pipeline analysis. This information should be compiled before starting `Pipeline4DGEData`.

1. The accession number of the GEO series.

   - In this example, this number is `GSE59015`.

2. The experimental conditions on which the analysis is to be performed and their corresponding GEO samples. And identifying name should be chosen in advance for each condition.

- In this example, the conditions' names and their samples are as follows.
  - Condition `Wildtype` comprising samples `GSM1424445` through `GSM1424452`, as shown in Figure 1.
  - Condition `D10` comprising samples `GSM1424453` through `GSM1424460`, as shown in Figure 1.

3. The condition files for the conditions on which the analysis is to be performed (Section 5.1 provides an introduction to condition files). These files must be located in folder `Input` and named as specified in Section 5.

   - In this example, the files for the two conditions must be as shown in Figure 3.

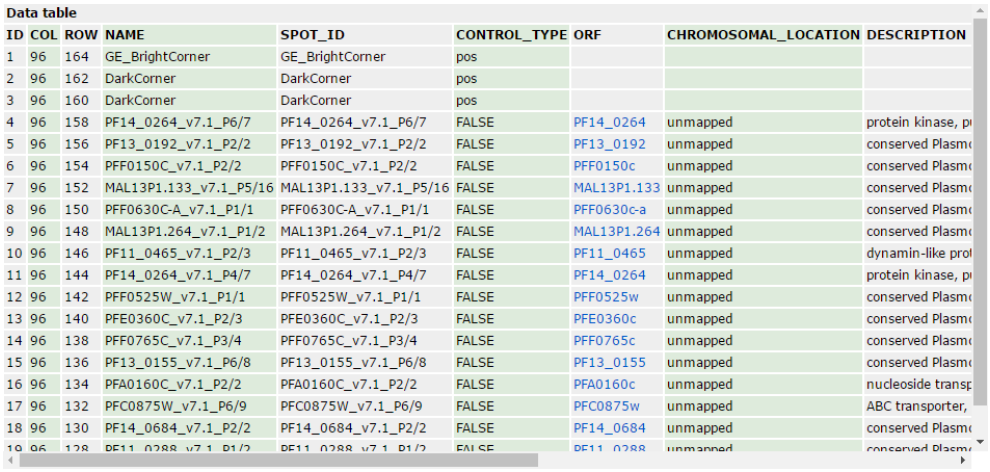
4. The accession number of the GEO platform associated to the GEO series.

   - In this example, this number is `GPL17880`.

5. The index of the column in the GPL record where the gene names are stored.

   - In this example, this index is 7, as shown in Figure 2.

Once the above is ready, the analysis can be started by running `pipeline` from the `MATLAB` console, as shown in Figure 4. The program will then prompt the user to enter the index of the column in the GPL record where the gene names are located (Item 5 in the checklist above). In this example, this index is 7 and must be entered as shown in Figure 5. After this the program will proceed to read and match all the probe ids to the corresponding gene names, informing the user that the process can take some time and notifying the progress as shown in Figure 6. When finished, the program will prompt the user to specify the gene ID type used in the dataset by selecting from a predefined list displayed on the `MATLAB` console. In this example, the type is `AGILENT_ID` and thus 5 must be entered, as shown in Figure 7. After this, the program will start the analysis for ALL the condition files present in folder `Input`. In this example, the program starts with condition `D10`



Data table

ID	COL	ROW	NAME	SPOT_ID	CONTROL_TYPE	ORF	CHROMOSOMAL_LOCATION	DESCRIPTION
1	96	164	GE_BrightCorner	GE_BrightCorner	pos			
2	96	162	DarkCorner	DarkCorner	pos			
3	96	160	DarkCorner	DarkCorner	pos			
4	96	158	PF14_0264_v7.1_P6/7	PF14_0264_v7.1_P6/7	FALSE	PF14_0264	unmapped	protein kinase, p
5	96	156	PF13_0192_v7.1_P2/2	PF13_0192_v7.1_P2/2	FALSE	PF13_0192	unmapped	conserved Plasm
6	96	154	PFF0150C_v7.1_P2/2	PFF0150C_v7.1_P2/2	FALSE	PFF0150c	unmapped	conserved Plasm
7	96	152	MAL13P1.133_v7.1_P5/16	MAL13P1.133_v7.1_P5/16	FALSE	MAL13P1.133	unmapped	conserved Plasm
8	96	150	PFF0630C-A_v7.1_P1/1	PFF0630C-A_v7.1_P1/1	FALSE	PFF0630c-a	unmapped	conserved Plasm
9	96	148	MAL13P1.264_v7.1_P1/2	MAL13P1.264_v7.1_P1/2	FALSE	MAL13P1.264	unmapped	conserved Plasm
10	96	146	PF11_0465_v7.1_P2/3	PF11_0465_v7.1_P2/3	FALSE	PF11_0465	unmapped	dynamin-like pro
11	96	144	PF14_0264_v7.1_P4/7	PF14_0264_v7.1_P4/7	FALSE	PF14_0264	unmapped	protein kinase, p
12	96	142	PFF0525W_v7.1_P1/1	PFF0525W_v7.1_P1/1	FALSE	PFF0525w	unmapped	conserved Plasm
13	96	140	PFE0360C_v7.1_P2/3	PFE0360C_v7.1_P2/3	FALSE	PFE0360c	unmapped	conserved Plasm
14	96	138	PFF0765C_v7.1_P3/4	PFF0765C_v7.1_P3/4	FALSE	PFF0765c	unmapped	conserved Plasm
15	96	136	PF13_0155_v7.1_P6/8	PF13_0155_v7.1_P6/8	FALSE	PF13_0155	unmapped	conserved Plasm
16	96	134	PFA0160C_v7.1_P2/2	PFA0160C_v7.1_P2/2	FALSE	PFA0160c	unmapped	nucleoside transp
17	96	132	PFC0875W_v7.1_P6/9	PFC0875W_v7.1_P6/9	FALSE	PFC0875w	unmapped	ABC transporter,
18	96	130	PF14_0684_v7.1_P2/2	PF14_0684_v7.1_P2/2	FALSE	PF14_0684	unmapped	conserved Plasm
19	96	128	PF11_0288_v7.1_P1/2	PF11_0288_v7.1_P1/2	FALSE	PF11_0288	unmapped	conserved Plasm

Figure 2: Table of platform `GPL17880` showing that the gene names are stored in the 7-th column (titled `'ORF'`).

as shown in Figure 8. When the analysis has been completed for all the experimental conditions, the program will confirm that the process has been finished successfully, as shown in Figure 9. The results are output to folder `Output` and organized as explained in Section 6.

5 Input files

5.1 Condition files

Condition files are the input files for the pipeline analysis of one or more experimental conditions (See Section 2.5 regarding 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* se-

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

(a) Condition file for condition `Wildtype`.

```
0 hours,GSM1424445
6 hours,GSM1424446
12 hours,GSM1424447
18 hours,GSM1424448
24 hours,GSM1424449
30 hours,GSM1424450
36 hours,GSM1424451
42 hours,GSM1424452
```

(b) Condition file for condition `D10`.

Figure 3: Figure 3a shows the contents of the condition file for condition `Wildtype` and Figure 3b shows the contents of the condition file for condition `D10`.

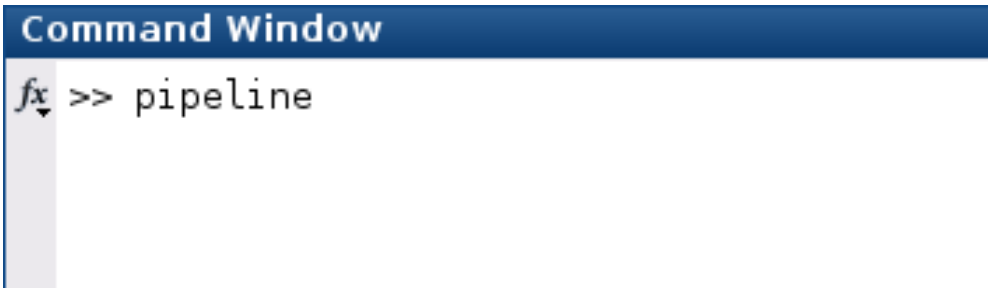


Figure 4

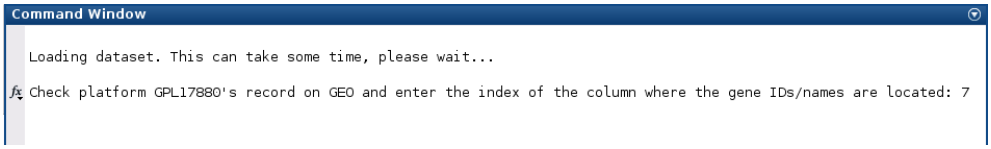


Figure 5

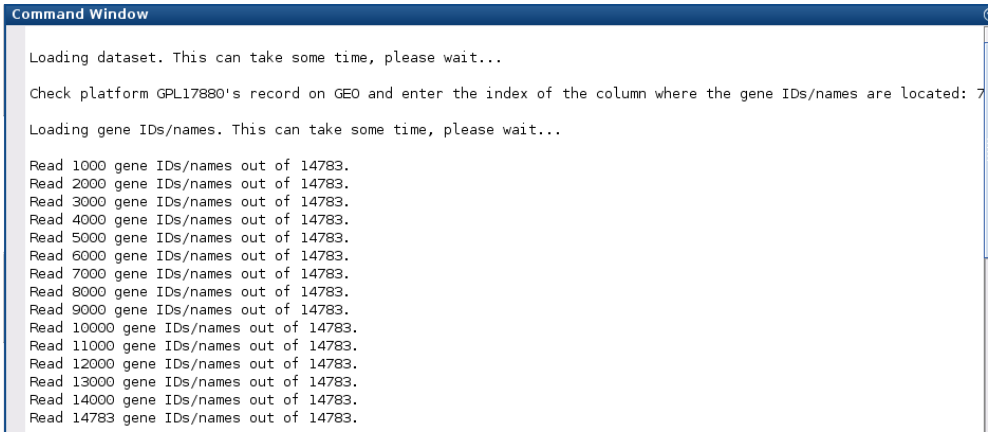


Figure 6

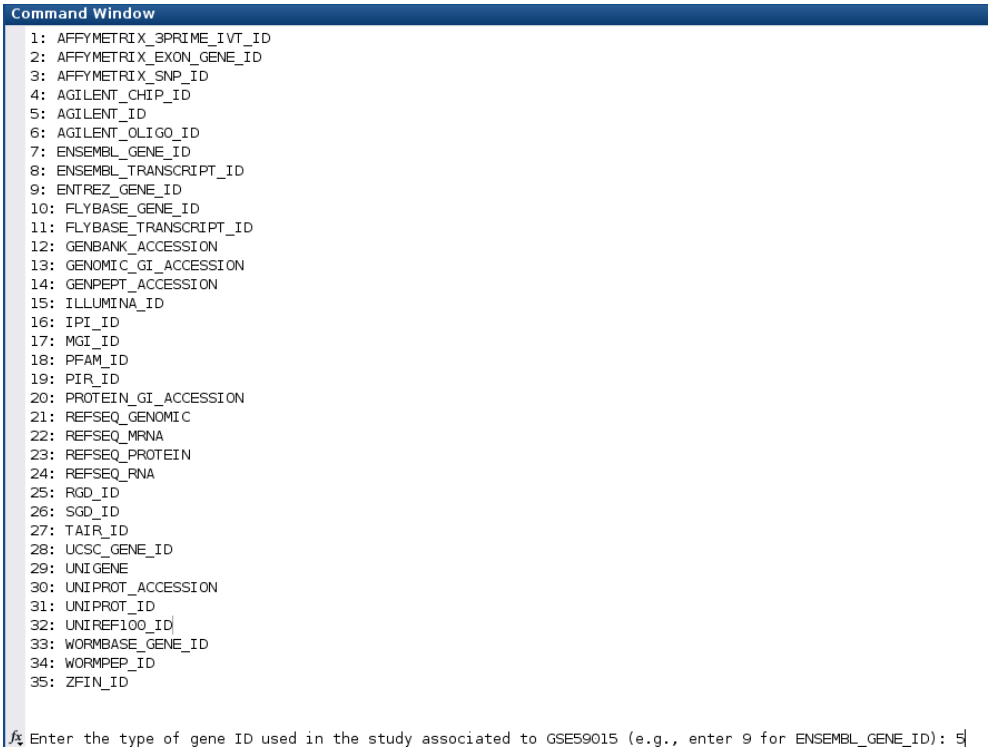


Figure 7

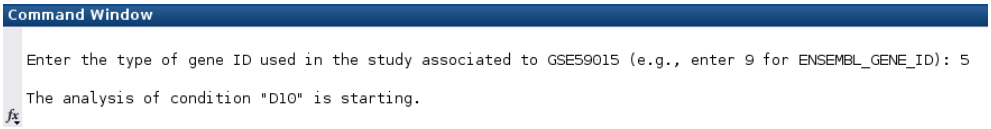


Figure 8

Figure 9

ries `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 7.

Condition files can contain different number of replicates at each time point. In these cases, each line of the condition file must contain the blank fields corresponding to the missing time points so that all lines in the file have the same number of comma-separated values. Suppose that in the previous example Replicate 2 and Replicate 3 do not have a sample at time point ``1 hours'' and that Replicate 1 does not have a sample at time point ``10 hours''. Thus the condition file should look as follows.

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

5.2 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.

6 Output of results

`Pipeline4DGEData` outputs all of its results to subfolder `Output`. Each pipeline analysis is carried out on an experimental condition and results from several conditions are grouped in folder by the accesion number of the associated GEO series.

7 Guided creation of condition files

The (experimental) condition files for the pipeline analysis can be created manually with the format described in Section 5.1 on page 9 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.

Prior to running the pipeline analysis for the two conditions (Subsection 5.1 provides an introduction to conditions files), the corresponding condition files must be created. In order to do this, `MATLAB` must be started inside folder `Pipeline4DGEData/` and in the console the command `create_input_files` must be called, as shown in Figure 10 and then pressing `Enter`.

The program will then request the accession number of the GEO series on which the analysis is to be performed. This information must be entered

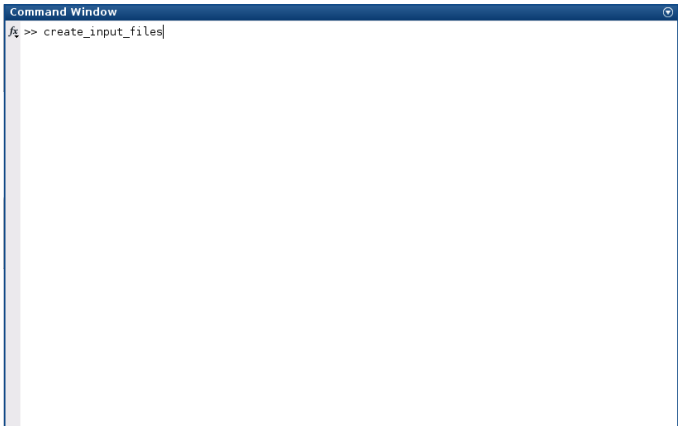


Figure 10: Calling `create_input_files` to create condition files.

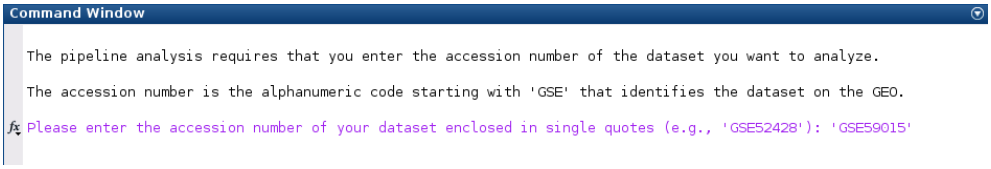


Figure 11: Entering the accession number for the creation of the condition files.

as single quotes (’), as shown in Figure 12. In this example, the accession number entered is `GSE59015`.

After hitting `Enter`, the program will read the series data from the GEO and will request the index of the column where the gene names are located in the platform record.

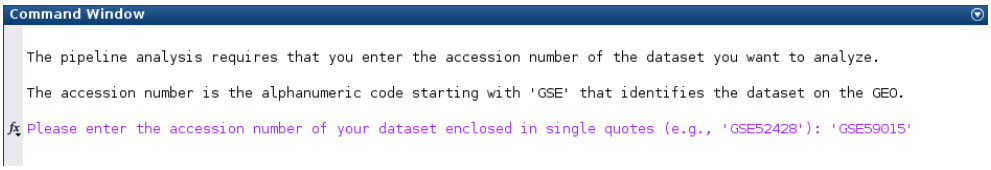


Figure 12: Entering the accession number for the creation of the condition files.

8 compare.m

A comparison can be performed between each pair of experimental conditions from a GEO series. This comparison includes an examination of how one gene behaves under one condition against the other and what modules in one condition match closest in expression pattern another module from the other condition.

This comparison can be performed once the regular pipeline analysis is completed for the given GEO series and results are stored in folder `Output`. The comparison can be launched by executing `compare.m` whose input is a macrocondition file. See Subsection 5.2.

An important requirement is that all the conditions to be compared (*i.e.*, those listed in the macrocondition file) have the same number of time points. Some series in the GEO include conditions with different numbers of time points. If a comparison is needed between these, then the pipeline analysis must be done on these including the samples where the to-be-compared conditions agree on the time points.

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 2.9. Its input is a macrocondition file, as defined in Subsection 5.2. 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.csv`, 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 Output of the pipeline

All results are output to folder `Output`. Results are grouped by the *GEO* series' accession number and by the name of the experimental condition, as illustrated in Figure 13.

11 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

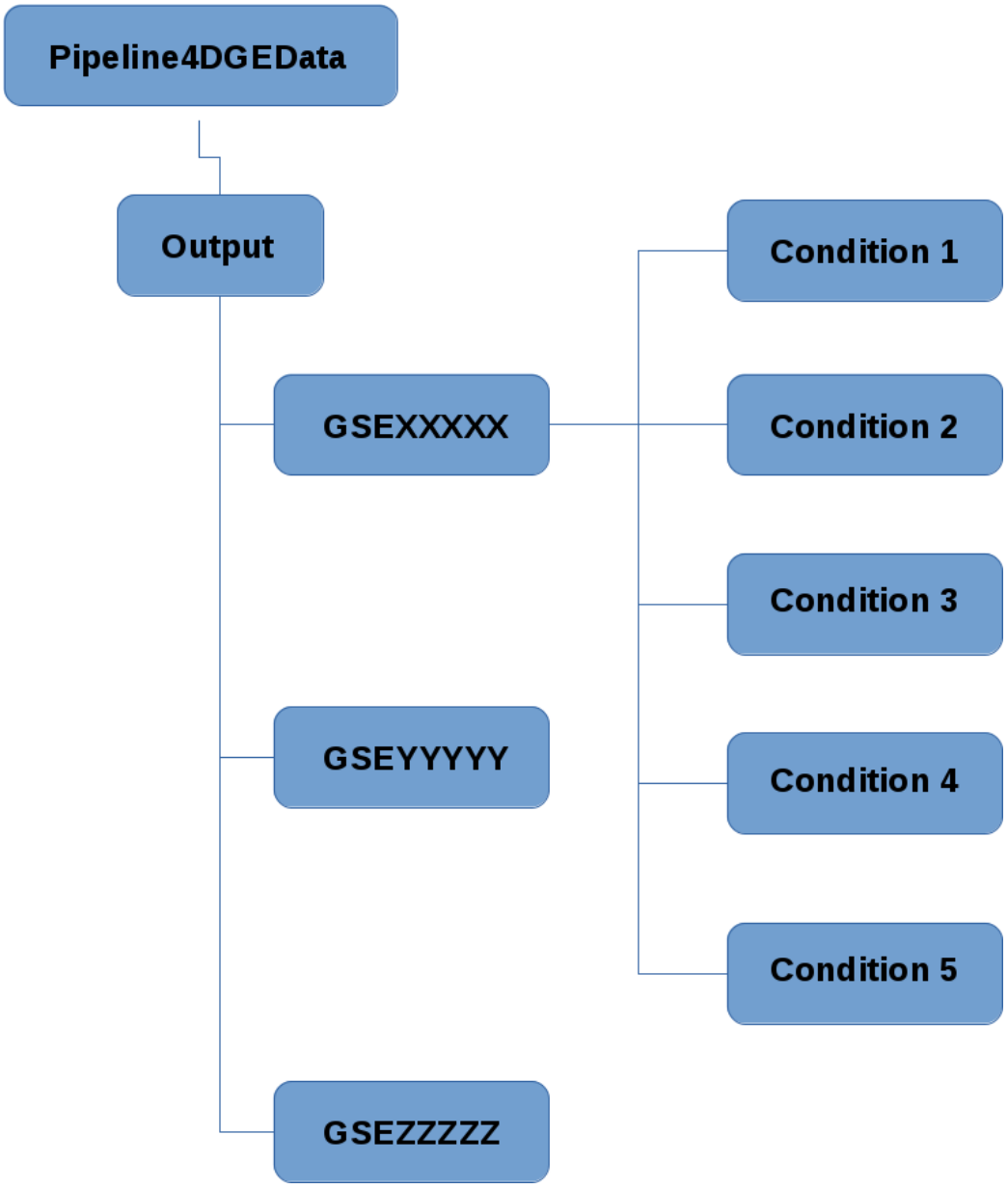


Figure 13: Folder hierarchy of results.

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.

References

Carey, Michelle, Ramirez, Juan Camilo, Wu, Shuang, and Wu, Hulin, 2017. A Big Data Pipeline: Identifying Dynamic Gene Regulatory Networks from Time Course GEO Data with Applications to Influenza Infection. *Statistical Methods in Medical Research*, 1–14.

Konganti, Kranti, Wang, Gang, Yang, Ence, and Cai, James J, 2013. SBEToolbox: a Matlab toolbox for biological network analysis. *Evolutionary Bioinformatics*, 9:355.