

**Flexible model-based co-clustering – FlexiCoClustering manual**

Joint clustering of binary and continuous types of data

**Authors:** Fatin Zainul Abidin\*, David Westhead\*\*

**Contacts:**

\*bs12fnza@leeds.ac.uk

\*\*D.R.Westhead@leeds.ac.uk

**Algorithm Version:** 1

**Section 1: Introduction to algorithm**

We develop a method to cluster multiple data of different types that would be generically applicable to a range of different problems in biology. We chose a simple model based framework, using a joint probability distribution over binary and continuous variables that is a mixture over a variable number of clusters. It uses penalized maximum likelihood (ML) estimation of mixture model parameters using information criterion and meta-heuristic searching for optimum clusters or known as Monte-Carlo simulated annealing (SA). The program takes as input a mixture of binary data (e.g. presence/absence of mutations, motifs, regulatory input, epigenetic marks etc.) and continuous data (e.g. gene expression, protein abundance, metabolite levels) for a list of samples (e.g. genes, patients). This program works best with smaller and concise dataset thus, pre-filtered data to only important features is preferable. To date, this program works well with ~1000 rows in the input file and longer time might be needed for larger dataset or it might not converged into a desired solution. An example of pre-filtration of dataset would be, reducing number of genes to only highly variable or differentially expressed genes and mutation that occurs in at least two cancer patients. Upon taking the input files required, the program will run until either the termination or convergence criterion have met. The clusters solution is then refined using expectation maximization (EM) step by taking the simulated annealing solution as the priori input for EM and updating the marginal densities of each data points which is the probability of being in each and all other clusters.

**Platform Dependencies**

**Task Type:** Clustering of data points

**CPU Type:** any

**Operating System:** any

**Language:** Using GUI based (Java - JDK 1.8)

Using command-line interphase (any)

**Section 2: How to run the Program**

**? How to run the FlexiCoClustering using command-line interphase (FlexiCoClustering-CLI)**

1. Download all the files from <https://github.com/Fatinzaheera/FlexiCoClusteringPackage/tree/master/FlexiCoClustering-CLI>

2. To run the demo , use the following command:

java <path to the FlexiCoClustering/FlexiCoClustering-CLI>.ClustProg <Input.txt> <Output.txt>

Press enter

3. To re-submit the code after program terminated:

Change Nrun: '0' to Nrun: '1'

Increase the MaxTemps to a higher value

java <path to the FlexiCoClustering-CLI/FlexiCoClustering>.ClustProg <Input.txt> <Output.txt>

Press enter

**? How to run the FlexiCoClustering using GUI based (FlexiCoClustering-GUI)**

1. Download all the files from <https://github.com/Fatinzaheera/FlexiCoClusteringPackage/tree/master/FlexiCoClusteringGUI>

2. Run the program (using JDK 1.6 and above), use the following command:

On Windows command prompt

java -jar <path to the FlexiCoClusteringGUI/FlexiCoClustering.jar>\FlexiCoClustering.jar

On any Linux/Unix distributions

java -jar <path to the FlexiCoClusteringGUI/FlexiCoClustering.jar>/FlexiCoClustering.jar

3. A graphical user interphase (GUI) window will be opened and looks like this snapshot below:

|  |
| --- |
| C:\Users\bs12fnza\Downloads\Screenshot from 2016-09-09 18_37_28.png |

**Figure 1:** A snapshot of the FlexiCoClustering GUI upon submitting the java -jar command on the terminal/command prompt. Red arrow shows where user should change the NRun to ‘1’ after the initial run (NRun=0) have finished if it is required at all.

4. On the GUI options (using demo example):

"ClustFile" : Name the ClustFile as i.e. Clustfile.txt

"Outfile" : Name an Outfile as i.e. Output.txt

"Select inputFile" : Select <path to the FlexiCoClustering.jar>\example\input.txt

For the real run using user own data, please change parameters accordingly. Parameters are described in **table 2** on the next page.

Press the "Execute" button. On default, this program will run for 1000 temperature steps (MaxTemps) and produces and updates heat map image files (.png) in every 10 iterations interval for binary and continuous variables. The Output.txt and Clustfile.txt will be updated as the program progress and once the program terminated respectively. An EM refining file (EMRefinement.txt) containing all the marginal densities for the clusters will be produced automatically in the same working directory.

If more than initially specified number of maximum temperature steps (MaxTemps) is required after step 2 had finished, re-run the program by first by replacing ‘0’ with ‘1’ in the Nrun and then press the "Execute" button again. User can also terminate the run at any time by pressing ‘Stop’ button.

**Section 3: Package input and runtime parameters**

**Input File**

Input file (e.g. input.txt): A space separated formatted text file containing the binary and continuous inputs dataset.

|  |  |
| --- | --- |
| A. | C:\Users\bs12fnza\Downloads\Screenshot-293.png |
| B. | C:\Users\bs12fnza\Downloads\Screenshot-292.png |

**Table 1:** Example of an input file for **A.** command line interphase (CLI) and **B.** graphical user interphase (GUI) based package. From 3rd or 21st row onwards of GUI or CLN based package respectively, first column shows the data points (i.e. gene names, sample names) and the second column onwards are the binary inputs ("1" and "0") followed by continuous values.

**Parameters**

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Line number in Algorithm 1** |
| ClustOutFile | The name of the final clusters output file | - |
| OutFile | The name of the real-time output of the program file | - |
| InputFile | Select the input file to be used | - |
| Nitem | Number of data points (rows in the input files, excluding the first two rows) | - |
| Nbinary | Number of binary variables in the input files | - |
| NContinuous | Number of continuous variables in the input files | - |
| Initial | Initial solution as either all data points in their own cluster (agglomerative) or all data points in a same cluster (divisive) | 2 |
| MergeSplitProbability | Monte Carlo move: either an ordinary step (moving a data point between clusters) or a cluster merge/split | 15 |
| MaximumIterations | Loop over Monte-Carlo moves at every temperature | 14 |
| SinglePoint | Calculate the score for the known solution in the simulation during algorithm development stage. (always set to 0 in the real run) | - |
| Seed | The seed of the random number generator used to produce permutations | - |
| NClusters | Feed the algorithm with the known true solution as the starting point in the simulation during algorithm development stage. (always set to 0 in the real run) | - |
| Norm | Normalizes continuous inputs to zero mean and a standard deviation for each data points- z-scores. | 1 |
| IC | Objective function/Information criterion |  |
| StartTemp | Starting temperature of the simulated annealing. Higher temperature-more random solution will be accepted at the beginning of simulated annealing. | 9 |
| TempFactor | Temperature reducing factor | 11 |
| MaxTemp | Maximum number of temperature to be simulated (termination criterion) | 10 |
| EMIterations | EM convergence criterion | - |
| OutInterval | Intervals at which the solution is printed in the output and at which the heat maps are updated on GUI | - |
| MaxRepIters | Maximum number of best score repetition (convergence criterion) | 34 |
| NRun | Number of re-run of the program after initial run | - |

**Table 2:** Runtime parameters of both GUI and CLN based package.

**Output file (e.g. Output.txt)**

A text file containing all the runtime updates such as score, current best modules/clusters, etc.

**EMRefinement file (e.g. EMRefinement.txt)**

A text file containing all the marginal densities of each clusters found from the simulated annealinh procedure.

**Section 4: Mathematical representation of score calculation**

**Solution (clusters) score calculation:**

**** …………………… **(1)**

*N* : data points *i*, representing genes, tumour samples etc.

binary variables

continuous variables

are mixing coefficients.

*B* denotes the Bernoulli distribution with parameter, and *N* is a normal distribution with parameters and.

We assume a probability distribution (1) which is a mixture of *Nm* components (clusters).

In the case of genetic regulation the mixture components represent the well-known concept of a cluster of co-regulated genes, with, for example, Bernoulli parameters representing the probability of binding for particular transcription factors in promoter/enhancer elements, and the representing a shared average pattern of gene expression, which could be a time or developmental series but is not required to be. In the case of tumour samples, clusters could be related samples where Bernoulli parameters associate mutation probabilities at particular loci with shared patterns of oncogenic gene expression.

Since the number of clusters in unknown and difficult to estimate, an initial heuristic search was adopted for an approximately optimal model, followed by refinement of the solution by expectation maximization. The heuristic search employed a Monte-Carlo simulated annealing algorithm (see **Algorithm 1**) to optimize objective functions of the form

…………………… **(2)**

where L is the (maximized) log-likelihood from the distribution above, is a function of the number of data points and *k* is the number of parameters in the model.

Several different functions can be use with our algorithm as shown in table 3 below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Criterion** |  | **N** | **Equation** | **Reference** |
| a. | AIC2 | 2 | 1 | -2L + 2k | Akaike, 1973 |
| b. | AIC2.5 | 2.5 | 1 | -2L + 2.5k | - |
| c. | AIC3 | 3 | 1 | -2L + 3k | Bozdogan, 1993 |
| d. | HQC | 2 | ln ln(Ng) | -2L + 2k(ln(ln(Ng))) | Hannan and Quinn, 1979 |
| e. | AIC4 | 4 | 1 | -2L + 4k | - |
| f. | AIC5 | 5 | 1 | -2L + 5k | - |
| g. | BIC | 1 | ln Ng | -2L + k(ln(Ng)) | Scwarz,1978 |
| h. | CAIC | 1 | ln Ng +1 | -2L + k(ln( Ng )+1) | Bozdogan, 1987 |

**Table 3:** Different objective functions tested and can be chosen by user sorted ascendingly (from a. to h.) based on its stringency in penalizing free parameters and number of data points in the model.

|  |
| --- |
| **Monte-Carlo simulated annealing (SA) for clusters optimization.** |
| 1: Normalize expression (genes)\*  2: Clusters = Initialize clusters (agglomerative/divisive)  3: Best clusters = [list]  4: Score = 0.0  5: Old score = 0.0  6: Best score = 0.0  7: Difference = 0.0  8: Count temperature = 0  9: Temperature = Start temperature  10: While Count temperature < Maximum temperature:  11: Temperature \* Temperature decreasing factor  12: Score = Calculate modules score (clusters)  13: beta = 1.0/temp  14: While Iteration < Maximum iterations: \*\*  15: If random > Merge and split probability:  16: Change (clusters)  17: Else:  18: If random > 0.5:  19: Merge (clusters)  20: Else: Split (clusters)  21: Old Score = Calculate clusters score (clusters)  22: Difference = Difference + (Score-Old score)  23: If Difference < 0.0 or random < exponent(-beta\*Difference):  24: ‘accept’  25: if Score < Best score:  26: Best score = Score  27: Best clusters = clusters  28: Else:  29: ‘reject’  30: Score = Old score  31: Count temperature + 1  32: If Best score = Old Score:  33: Count score = Count score + 1  34: If Count score == 2000:  35: break  36: return Best score, Best clusters |

**Algorithm 1:** Pseudo-code for the Monte-Carlo Simulated annealing algorithm.

\*Optional

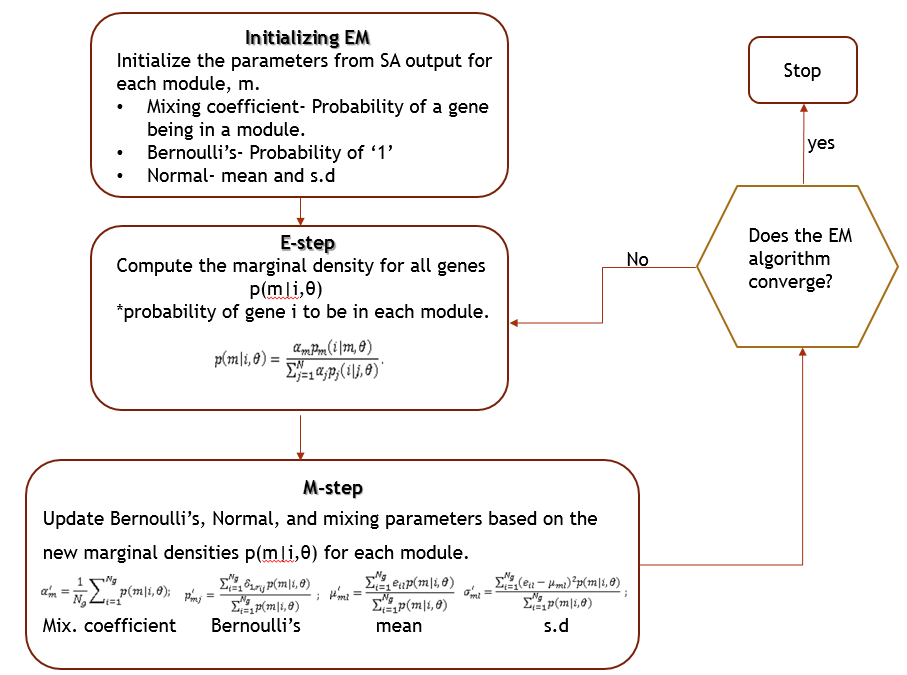
\*\*This step runs in parallel (minimum of 5 parallel threads).

**Expectation-maximization**

The parameters of model produced as the best solution from the heuristic search can be refined by EM, with the useful side effect of estimating the degree of mixing between modules through the probability density that data point *i* is generated from mixture component *m* (3).

…………………… **(3)**

This is derived from Bayes’ rule: *pm* is the probability density for mixture component *m* defined in 2.2, θ denotes the (current) vector of parameters for all modules and the mixing coefficients αm can be interpreted as prior probabilities for membership of each module. The steps of the EM algorithm are showed in the figure as followed:



**Figure 1:** Refinement of model parameters using EM which starts with the prior probability densities from the SA output and refinement of its parameters until convergence. Convergence here means, until the parameters values do not changed for 2 consecutive EM iterations or until the maximum number of iterations has been reached.