Introduction

This document is designed as a User's Guide for running the MDI software. It does not provide any details regarding the underlying model(s), nor does it provide a developer's guide for the MDI code.

Before reading this document, it is highly recommended that you read the README file provided with MDI. This will probably save you a lot of time!

The easiest way to get up and running with MDI is to run the MDISim example that is provided with MDI. To get this to run, type the following at the Matlab command line:

```
>> MDISim(1)
```

Note that the argument (here, 1) is arbitrary, and simply provides an identifier for the current run. This is important if you want to run MDI multiple times without overwriting the output of a previous run. The `MDISim` function provides an example of how MDI can be called. Probably the best way to start is to open up `MDISim` in the Matlab editor, and to use this as a template. You can also see the file format expected by MDI by looking at the example data files in the Data subdirectory.

If things are still unclear, please read on.

Running MDI

There is just one function that you will need to call: MDI. MDI requires a number of inputs. These are (in order):

1. **An output directory.** This is the directory into which all of the output generated by MDI will be written. To write into the current directory (i.e. the one from which you are running MDI), this can be specified as: `''` (an empty string).
2. **The number of components.** This is the maximum number of clusters that may appear in the data. Current recommendation: set this number to be equal to half the number of genes in the dataset. If it turns out that the number you specify is too small (i.e. if MDI "thinks" there should be more clusters in the dataset than the number you specify here), then MDI will stop, and you will have to restart your analysis (specifying a larger number of components).
3. **The number of Gibbs sampling iterations.**
4. **The names of all of the data files.** This should be a cell array of length K, where K is the number of datasets you seek to integrate. The first element is the name of the data file containing the data for the first dataset, the second element is the name of the data file containing the data for the second dataset, …, and so on.

5. **The data type for each of the datasets.** This should be a cell array of length K, where K is the number of datasets you seek to integrate. MDI permits real-valued, discrete, time course, etc. datasets to be integrated. In this cell array, we specify the data type of each dataset that we are trying to integrate. Currently allowed values are:
    a. `'Multinomial'`**:** This is the data type to use for a dataset that contains only categorical values (e.g. 1,2,3 or 0,1).
    b. `'BagOfWords'`**:** If we have a categorical dataset that has only two discrete values (0 and 1), then we may use the BagOfWords data type as an alternative to the Multinomial. This runs more quickly than the Multinomial type, but will result in a different clustering. [My personal preference is to use Multinomial, and to resort to BagOfWords only if things are running very slowly].
    c. `'TimeCourseUnequalSpacing'`**:** This is the data type to use if you have real-valued time course data. Note that TimeCourseUnequalSpacing should be used for all time courses, including those with equally spaced time points (there used to be a TimeCourseEqualSpacing data type, but this is not currently in use).
    d. `'Gaussian'`**:** This is the data type to use if you have real-valued data, where the features can be treated as independent (so *not* time course data).
    e. `‘Poisson’`**:** Data points are assumed to be drawn from a Poisson distribution. Note that the test data files Dat/PoissonTestData1.csv and Dat/PoissonTestData2.csv where created using the file GenerateSyntheticCountsData.m
6. **The sampling frequency for the Gaussian process hyperparameters.** This input is only used if one or more of your datasets is of type `'TimeCourseUnequalSpacing'`. If none of your datasets are of this type, just set this input to 1. If you have time courses, increasing this parameter (to, say, 2 or 5) can speed up the time it takes for MDI to run.
7. **The frequency with which the MCMC output should be written to file.** If this value is set to 10 (say), this means that MDI will only write every 10th sample to file. This automatically takes care of “thinning” the MCMC output, and also reduces the proportion of time spent writing output to file.
8. **A unique identifier to attach to the output of the current run.** It may happen that you wish to run MDI repeatedly over the same collection of datasets (i.e. you may wish to run multiple MCMC chains in parallel). If this is the case, then you should specify a distinct identifier for each run using this input. If you do not intend to run MDI repeatedly, then set this input to be (say) 1.
9. **A logical value indicating if you are starting a new MDI run, or if you are continuing a previous run.** This is the “initialisation” variable. If set to true, MDI will start a new run. If set to false, MDI will look for the output from a previous run, and pickup from where the last run left off.
10. **A vector of length K. Non-zero indicates that feature parameters should be randomly initialized with either one or zero for the corresponding data file.** This applies only to multinomial and Gaussian

data types. This input is optional. If absent then all feature parameters are initialized with one.

Example:

```
outDir       = '';     % 1. output directory
nComponents = 50;     % 2. number of components
nSamples     = 10000; % 3. number of iterations
% 4. Names of files containing the data:
fileName1    = 'Data/MDItestdata1.csv';
fileName2    = 'Data/MDItestdata2.csv';
fileName3    = 'Data/MDItestdata3.csv';
fileName4    = 'Data/MDItestdata4.csv';
fileName5    = 'Data/MDItestdata5.csv';
fileName6    = 'Data/MDItestdata6.csv';
% 5. Data types for each of the datasets:
dataType1    = 'TimeCourseUnequalSpacing';
dataType2    = 'TimeCourseUnequalSpacing';
dataType3    = 'TimeCourseUnequalSpacing';
dataType4    = 'TimeCourseUnequalSpacing';
dataType5    = 'TimeCourseUnequalSpacing';
dataType6    = 'TimeCourseUnequalSpacing';
% 6.  Sampling frequency for GP hyperparameters:
hyperSamplingFrequency = 1; % so, we should re-sample on every
iteration
% 7.  Frequency with which we should write to file:
samplingFrequency = 10; % so, we should save down every 10th
sample
% 8. Specify a unique ID for the current run:
uniqueIdentifier = 1;

% 9.  Is this the first time we are running MDI for these datasets?
initialise = true;

% Finally, the call to the MDI function:
MDI(outDir,nComponents,nSamples, {fileName1,fileName2,...
      fileName4, fileName5, fileName6},...
     {dataType1, dataType2, dataType3, ...
      dataType4, dataType5, dataType6},...
     hyperSamplingFrequency,...
     samplingFrequency, uniqueIdentifier, initialise);
```

Preparing the data

In this section we specify the required data format for MDI. Dataset files can be CSV text files or MATLAB mat files.

For CSV files

- This file should have a header that contains the feature names. In the case of time course data, the feature names should be numbers (e.g. 0, 160, 320, …) that correspond to the times at which the measurements were taken.
- The first column specifies the gene names.

For mat files

- The file should contain a single structure with fields 'data', the matrix of data points; 'featureNames', corresponding feature names for each data column; and 'geneNames', corresponding gene names for each row.

Note that MDI assumes that the genes are given in the same order for all of the datasets that you are trying to integrate.

Example: Suppose we are trying to integrate 2 datasets. We must provide 2 data files, e.g. Dataset1.csv and Dataset2.csv. These should look similar to the table below. Note that the first column of Dataset1.csv is identical to the first column of Dataset2.csv. In particular, the genes **must** appear in the same order.

Dataset1.csv

	0	1	2	3	…
YER170W	-0.24	-0.22	0.72	0.47	…
YNL102W	-0.37	0.04	1.05	0.80	…
YAR007C	-0.47	-0.01	1.04	0.32	…
YBR088C	-0.36	0.06	0.79	0.49	…
YDL164C	-0.46	0.04	0.87	0.38	…
YKL113C	-0.67	0.06	1.00	0.80	…
YNL312W	-0.23	0.05	0.79	0.55	…
YDR097C	-0.41	-0.28	0.98	0.57	…
…	…	…	…	…	…

←Feature names

↑
Gene names

Dataset2.csv

	P1	P2	P3	P4	…
YER170W	-0.78	-0.21	0.63	0.31	…
YNL102W	-0.38	0.10	0.76	0.51	…
YAR007C	-0.07	0.18	0.97	0.46	…
YBR088C	-0.47	0.02	0.80	0.41	…
YDL164C	-0.71	-0.18	0.86	0.74	…
YKL113C	-0.49	0.24	1.14	0.63	…
YNL312W	-0.20	-0.36	0.05	-0.12	…
YDR097C	-0.19	-0.35	0.10	-0.12	…
…	…	…	…	…	…

← Feature names

↑ Gene names

Running MDI on a cluster

MDI can be run on a cluster via a job scheduler. To facilitate this we have included some example cluster scripts for the PBS scheduler. These will probably need modifying to suit your own cluster environment.

MDISim.sh creates an array of 20 jobs and submits them to the scheduler. Each job is a call to MDISim.pbs, which runs MDISim, using the array index as the unique identifier.

Analysing the output

MDI produces 3 output files (2 .mat files, and one .csv file). It is only the CSV file that is important for subsequent analysis. The following assumes some basic familiarity with the MDI model.

Let K be the number of datasets that you are trying to integrate. The first K columns of the output CSV file provide the sampled values for the K mass parameters associated with each Dirichlet prior. The next K(K-1)/2 columns provide the "phi" (dataset association) parameters. These are in the order phi_{1,2}, phi_{1,3}, ..., phi_{1,K}, phi_{2,3}, phi_{2, 4}, ... and so on. The next p columns provide the component allocation labels for the p genes in Dataset 1. The next p columns contain the component allocation labels for the p genes in Dataset 2, ... and so on. Currently, MDI does not save down the sampled hyperparameter values (in the time course case), just because to do so would come with associated file I/O expense, and would increase the size of the output CSV file.

One of the mat files will be named by concatenating all the input file names in order and the unique identifier, all separated by underscores. This file contains the current state of the sampler. If MDI is run subsequently with the same input files, same unique identifier, and initialize = false, then this file expected to be present to provide the initial state for the new run.