MDSINE User Manual

Table of Contents

1. INTRODUCTION 3

2. SOFTWARE INSTALLATION 4

LINUX/UNIX 4
MAC 4
WINDOWS 4

3. EXECUTION 5

MATLAB 5
MAC 5
LINUX/UNIX 5
WINDOWS 5

4. EXECUTING THE ANALYSES IN THE MANUSCRIPT 5

4.1 CLOSTRIDIUM DIFFICILE INFECTION DATASET 6
UNIX 6
MAC 6
WINDOWS 7
PLOTTING 7
4.2 PROBIOTIC COCKTAIL DATASET 7
UNIX 7
MAC 7
WINDOWS 7
PLOTTING 8

5. INPUT FILES 8

5.1: SEQUENCING COUNTS FILE 8
5.2: BIOMASS FILE 8
5.3: METADATA FILE 9

6. CONFIGURATION FILE 11

6.1: GENERAL PARAMETERS 11
6.2: PREPROCESSING 12
GENERAL PREPROCESSING 12
BAYESIAN SPLINES FOR BIOMASS DATA 12
BAYESIAN SPLINES FOR COUNT (SEQUENCING) DATA 12
6.3 INFERENCE 13
BAYESIAN ADAPTIVE LASSO (BAL) 13
BAYESIAN VARIABLE SELECTION (BVS) 13

Maximum Likelihood Ridge Regression (MLRR and MLCRR) 14
6.4: Simulations **15**
6.5: Linear Stability **16**
6.6: Post Processing: Visualizations and Keystoneness Analysis Outputs **16**
7.1 Parameters **18**
7.2 Simulation Trajectories **18**
7.3 Stability Analysis **18**
7.4 Keystone Analysis **19**
7.5 Cytoscape **19**

8. R-PLOTTING UTILITIES **20**

1. Introduction

The purpose of this document is to describe how to install the software, execute the MDSINE software, format input files, edit the configuration file, and plot results.

The MDSINE package implements three novel algorithms for gLV parameter inference: a maximum-likelihood method, constrained ridge regression (MLCRR); a Bayesian Adaptive Lasso (BAL) method, and a Bayesian variable selection (BVS) method. A maximum-likelihood unconstrained ridge regression algorithm (MLRR), is also implemented in MDSINE for comparison. MDSINE also enables you to perform simulations to predict the behavior of microbial ecosystems; a linear stability analysis; and analysis of species "keystoneness" with respect to the community. The software also outputs files for plotting in R.

The MDSINE project is organized into the following modules: - Data import - Maximum Likelihood ridge regression - Bayesian Inference - Simulation of Trajectories - Linear stability - Post processing – Utilities.

The behavior of the software is completely defined by the configuration file. This eliminates the need for extensive command line options and also provides complete documentation of the analysis for purposes of reproducibility.

The configuration file is just a text file in the following format:

```
[Section] # comment
parameter = value  # comment 2
```

For example, the first section of the configuration file is the general section, which dictates the overall execution of the software. For example, if you would like to run the inference and then simulate the system based on initial conditions, then your configuration might look like:

```
[General] # 1 or 0 for true or false
run_inference = 1
run_simulations = 1
run_linear_stability = 0
run_post_processing = 1

# seed can be empty or a number
seed = 100

algorithm = BVS
output_dir = output   # output directory

metadata_file = example1/metadata.txt   # metadata in format specified
in manual
counts_file = example1/counts.txt   # biom converted to text, qiime or
mothur out
biomass_file = example1/biomass.txt # biomass file in format specified
in readme
```

Note that there is no correspondence between the order in which the parameter sections are specified, and the order in which the pipeline runs the various algorithms.

2. Software installation

Download the latest release at https://bitbucket.org/MDSINE/mdsine/downloads.

Linux/Unix

WARNING: There is an issue with the Linux installer such that it may claim that you have an issue with your connection, and the installation will halt. The installation may require a few tries, but once it starts, it should complete successfully.

1. Run the executable provided.
2. Select an installation directory for the MDSINE executable.
3. Select an installation directory for MATLAB runtime library (required for MDSINE software).
4. When complete, navigate to the installation directory of MDSINE and execute the following: `./run_mdsine.sh <MATLAB Runtime directory> <Configuration file> ./run_mdsine.sh /usr/local/MATLAB/MATLAB_Runtime/v90 <data folder>/parameters.cfg`

The MATLAB runtime directory will have a version; e.g., /usr/local/MATLAB/MATLAB_Runtime/v90

Two example configuration files are in subdirectories of the installation directory ("/data_diet" and "/data_cdiff").

Mac

This is analogous to the Linux/Unix installation and usage. The Mac default installation directory is /Applications/mdsine. The mac default MATLAB runtime library is /Applications/MATLAB/MATLAB_runtime.

Windows

Run the Windows installer and go through the following steps:

1. Select an installation directory for the MDSINE executable.
2. Select an installation directory for the MATLAB runtime library (on subsequent installs, this step will be skipped automatically).
3. Navigate to the installation directory in the file explorer, shift+right click -> Open command prompt here.

To execute: `mdsine <configuration file>`. Note that the default example parameter files will need to be edited to specify the correct paths, specifically: output_dir = output will need to be a path such as: output_dir = C:/Users/USERNAME/Documents/mdsine/output.

3. Execution

Matlab

To use mdsine within MATLAB, call the mdsine function with the configuration file as an argument:

```
>> mdsine('my.cfg')
```

Mac

Navigate to the mdsine directory in a terminal, and call mdsine with the MATLAB runtime directory (automatically installed alongside MDSINE) and the configuration file as the command line parameters. (Note that you may have to change the path.):

```
# ./run_mdsine.sh /Applications/MATLAB/MATLAB_Runtime/v90
/Users/Me/Project/my.cfg
```

Linux/Unix

Navigate to the mdsine directory in a terminal (or add it to your path), and call mdsine with the MATLAB runtime directory (automatically installed alongside MDSINE) and the configuration file as the command line parameters:

```
# ./run_mdsine.sh /usr/local/MATLAB/MATLAB_Runtime/v90
/home/Me/Project/my.cfg
```

Windows

Use the file explorer to navigate to the installation directory, and shift+right click to open a command prompt at that location. Call mdsine with the configuration file as the command line parameter:

```
> mdsine C:/Users/Me/Project/my.cfg
```

Note that a normal user may not write to the default windows install location, so if you want to run the examples you will need to edit the configuration file's output to write to a location in your user directory. For example:

```
output_dir = C:/Users/Me/Project/output
```

Additionally, note that while Windows uses backslash to determine directories (i.e., `C:\Users\Me`), MATLAB uses forward slashes (`C:/Users/Me`).

4. Executing the analyses in the manuscript

Data files and configuration files are provided to analyze the data presented in the manuscript. The commands for executing these analyses are described below, as well as details on the output files. To execute these analyses, please first navigate to the installation directory of the MDSINE package, as noted in Section 3.

When MDSINE is run on each of the provided main analysis configuration files, five files are output for subsequent visualizations:

```
BVS.results.parameters.txt
BVS.results.stability_analysis.txt
BVS.results.keystone_analysis.txt
BVS.results.simulations.txt
BVS.results.cytoscape.txt
```

The first four files listed are visualized using provided R scripts, and the fifth file is visualized using Cytoscape. The filenames starts with the name of the inference algorithm chosen (so if you change the algorithm chosen, make sure to change the plotting commands accordingly.) The output files go into the output_dir specified in the general section of the configuration file (see Section 6). For more details on the Rscripts and the output files, see Sections 7 and 8.

The cross validation procedure produces a single file as output for the *Clostridium difficile* dataset analysis:
`crossvalidation_cdiff/cdiff_crossvalidation_rmse.txt`
This file contains the root mean squared error for each holdout experiment.

The corresponding file for the cross validation procedure for the probiotic cocktail dataset is:
`crossvalidation_diet/diet_crossvalidation_rmse.txt`

Note that for the execution commands described below, it is assumed the location of the MATLAB runtime library is in the following locations (if not then edit the command accordingly):
Mac: `/Applications/MATLAB/MATLAB_Runtime/v90/`
Unix: `/usr/local/MATLAB/MATLAB_Runtime/v90/`

4.1 Clostridium difficile infection dataset

Unix

Main analyses:
```
# ./run_mdsine /usr/local/MATLAB/MATLAB_Runtime/v90/
data_cdiff/parameters.cfg
```

Cross-validation analyses:
```
# ./run_mdsine /usr/local/MATLAB/MATLAB_Runtime/v90/
cdiff_crossvalidation
```

Mac

Main analyses:
```
# ./run_mdsine /Applications/MATLAB/MATLAB_Runtime/v90/
data_cdiff/parameters.cfg
```

Cross-validation analyses:
```
# ./run_mdsine /Applications/MATLAB/MATLAB_Runtime/v90/
cdiff_crossvalidation
```

The cross validation produces a single file as output:
`crossvalidation_cdiff/cdiff_crossvalidation_rmse.txt`

Which contains the RMSE for each holdout and the correlation of the trajectories.

Windows

Main analyses:

```
> mdsine data_cdiff/parameters.cfg
```

Cross-validation analyses:

```
> mdsine cdiff_crossvalidation
```

Plotting

Plotting commands are the same across all platforms. The following commands assume that you're in the installation directory of MDSINE, and that the output directory field of the configuration file was not modified.

```
# Rscript utilities/plot_model_parameters.R
output_cdiff/BVS.results.parameters.txt
output_cdiff/BVS.parameters.pdf
```

4.2 Probiotic cocktail dataset

Unix

Main analyses:

```
# ./run_mdsine /usr/local/MATLAB/MATLAB_Runtime/v90/
data_diet/parameters.cfg
```

Cross-validation analyses:

```
# ./run_mdsine /usr/local/MATLAB/MATLAB_Runtime/v90/
diet_crossvalidation
```

Mac

Main analyses:

```
# ./run_mdsine /Applications/MATLAB/MATLAB_Runtime/v90/
data_diet/parameters.cfg
```

Cross-validation analyses:

```
# ./run_mdsine /usr/local/MATLAB/MATLAB_Runtime/v90/
diet_crossvalidation
```

Windows

Main analyses:

```
> mdsine data_diet/parameters.cfg
```

Cross-validation analyses:

```
> mdsine diet_crossvalidation
```

Plotting
Plotting commands are the same across all platforms. The following commands assume that you're in the installation directory of MDSINE, and that the output directory field of the configuration file was not modified.

```
> Rscript utilities/plot_model_parameters.R
output_diet/BVS.results.parameters.txt
output_diet/BVS.parameters.pdf
> Rscript utilities/plot_simulated_trajectories.R
BVS.results.simulations.txt output_diet/BVS.trajectories.pdf
> Rscript utilities/plot_stability_analysis.R
BVS.results.stability_analysis.txt output_diet/BVS.stability
> Rscript utilities/process_keystone_analysis_results.R
BVS.results.keystone_analysis.txt output_diet/BVS.keystoneness
```

5. Input Files
There are three input files that contain the sequencing counts data, biomass data, and the metadata for the experiments. We describe here the specification for each of those files. Please see the data_cdiff and data_diet directories in the repository or the installation directory for examples. These example files are also what were used for generating the results shown in the main manuscript. Each input file is tab delimited, and must have the same number of delimiting characters (tabs) on each line. Some rows may need additional tabs, and any empty spaces may be filled with a -1, as in data_diet/metadata.txt.

5.1: Sequencing Counts File
The counts input is effectively trimmed text output from qiime or mothur: there is a header row with a label per sample, and then each row after should be the OTU ID with counts for each sample. The OTU identifiers are listed in the first column. For example, with *N* samples and *M* OTU, the format is as follows:

```
#OTU ID    1     2     3     4 ... N
SpeciesName1     ...
SpeciesName2     ...
...
SpeciesNameM     ...
```

5.2: Biomass File
The values for this file are the masses corresponding to the samples (nrows = nsamples), one column per qPCR replicate. The mass can be in whatever units are desired. If there are three replicates (see data_cdiff), the file would look like:

```
mass1 mass2 mass3
4.46e+09   4.75e+09   6.93e+09
...
```

If the data were preprocessed prior to input into the MDSINE package (e.g., averaged qPCR data), you may use a single column for biomass data. However, it's best to include replicates as MDSINE takes into account the variability across the replicates.

5.3: Metadata File

There are two sections to the metadata file. The first six columns are information about each sample, and the last set of columns are information about the OTUs. In other words, the number of rows in the first six columns are equal to the number of samples. The number of columns remaining is equal to the number of different experimental blocks, and the number of rows for those columns is equal to the number of OTUs (the number of rows in the counts file, excluding the header).

Experimental blocks define groups of experiments that are not synchronized in time, have substantially different numbers of time-points, or otherwise differ significantly in the experimental protocol. For instance, if you have a control and treatment group, these should be specified as different experimental blocks. The Bayesian spline estimation procedures and other parts of the pre-processing pipeline will process different experimental blocks separately, as the assumption is that different blocks have substantially different statistical properties.

The first six columns of the metadata file must have their appropriate headers:

1. sample ID (string) - such as 1, 2, 3... or Subject1_t1, Subject1_t2, etc.
2. isIncluded (boolean) - for each sample, 1 if the sample is to be included, 0 if not. This column is provided for convenience, if the user desires to not have certain time-points included in the analysis.
3. subject ID (int) - unique identifier for each subject. Note that if a subject belongs to two different experimental blocks, they must be assigned different subjects IDs in the input file.
4. measurement ID (float) - timepoint at which the sample was obtained.
5. perturb ID (int) - indicates which perturbation is present for the sample, 0 if none.
6. experimental block ID (int) - indicates which experimental block the sample belongs to.

The remaining columns in the metadata file specify experimental interventions in which particular OTUs were introduced into the system; if no such interventions were performed then the values for these columns are omitted. The values for these columns are the indices of the measurement ID (time point) when the OTU was introduced. An OTU that is not an "intervention" is defined to be introduced at time index t = 0. (Hence if every OTU is entered as having been introduced at index t = 0 for a given experimental block, that is equivalent to omitting interventions for that block). There is one column of interventions per experimental block, and one row of interventions per OTU. See data_diet for an example with 2 experimental blocks and 13 OTU's.

It is recommended to construct the metadata file in a spreadsheet program and export it as a tab-separated file, so that there is the correct number of delimiters on each row.

An example header, and few representative rows (from the **data_diet** example):

```
sample ID   isincluded     subject id     measurementID  perturb ID     expt block
    intv    intv2
1   1       1       1      0       1      0       0
2   1       1       2      0       1      0       0
...
```

```
13  1       1       13      0       1       0       0
14  1       1       14      0       1
...
330 1       7       29      0       2
```

Note that there are two tab characters after the last value, to ensure the same number of delimiting characters on each line (7 in this case, for 8 columns)

6. Configuration File

The configuration file drives the behavior of the MDSINE software completely. There are *no* run time flags or any other options. The sections of the configuration file are primarily split between modules (Inference, Simulation, and Post Processing) and sub-modules, but there are a few shared sections.

Note that the examples given in the following sections are from the configuration files used to actually analyze the data in the manuscript. The user is encouraged to look at the supplied configuration files, as they illustrate common analysis scenarios and the necessary parameter settings.

Basic parameters that users will likely need to set are **bolded**. Non-bolded parameters are advanced options, and generally will not need to be set by the user. Parameters are numbered in this manual to make it easier to explain their function, but in the actual configuration file should not be numbered and do not need to be in the order shown in this manual.

6.1: General Parameters

There are two sections for 'general' parameters, the first is under [General], and the second is [Parallel], which has one field.

```
    [General]
1.  run_inference = 1
2.  run_simulations = 1
3.  run_linear_stability = 1
4.  run_post_processing = 1
5.  seed =
6.  algorithm = BVS
7.  output_dir = output_diet
8.  metadata_file = data_diet/metadata.txt
9.  counts_file = data_diet/counts.txt
10. biomass_file = data_diet/biomass.txt
    [Parallel]
11. cores = 4
```

The four 'run_' parameters (1-4) determine which of the respective modules (described in subsequent sections) to run. The seed parameter (5) allows the user to specify a random seed for purposes of debugging or to produce a deterministic run; this should normally be left blank. The algorithm choice (6) should be one of these values: "BAL", "BVS", "MLRR", or "MLCRR". Note that the BVS algorithm uses the output of the BAL algorithm for initialization so if "BVS" is selected, "BVS" will also run. The output and input names (7-10) are relative paths as shown in the example, but can be absolute paths. For the format of the three input files, see the input specification. The final parameter (11) is the number of cores to use for parallelization, which is currently used only in MLRR/MLCRR and the linear stability analysis.

6.2: Preprocessing

These parameters are used for preprocessing of the input data for the inference routines. Preprocessing includes filtering of OTUs by minimal median number of counts across time-points; biomass trajectory estimates from qPCR data; and OTU trajectory and gradient estimation prior to parameter inference (see section on Bayesian Splines for details).

General Preprocessing

```
    [Preprocessing]
1. minMedCount = 10
2. numReplicates = 3
3. useSplines = 1
```

Option (1) specifies a minimum filtering threshold for median OTU counts across all time-points to determine which OTUs to include in the analysis. Option (2) specifies the number of biomass qPCR replicates in the data. The Bayesian spline estimation methods (for biomass and counts data) can be disabled by setting option (3) to a value of zero. Note that the spline estimation procedures are available only if the Bayesian inference methods are selected.

Bayesian Splines for Biomass Data

```
    [bayesian spline biomass]
1. numIters = 10000
2. numBurnin = 2000
3. smoothnessOrder = 1
4. tauScale = 100
5. init_lambda = 1.00E-03
6. gpA = 1.00E-09
7. gpB = 1.00E+07
```

These are all advanced parameters that the user generally will not need to set (see the Methods section of the manuscript for details). Option (1) specifies the number of MCMC samples and option (2) specifies the number of burnin iterations. The order of temporal adjacency is specified by (3), with order 1 or order 2 currently supported. Parameters (4-7) specify hyperparameter and parameter initializations as explained in the Methods section of the manuscript.

Bayesian Splines for Count (Sequencing) Data

```
     [bayesian spline counts]
1.   numIters = 10000
2.   numBurnin = 2000
3.   smoothnessOrder = 1
4.   tauScale = 100
5.   lambda_omega_init = 1.00E-03
6.   gpA_omega = 1.00E-09
7.   gpB_omega = 1.00E+07
8.   eps_a1_init = 1.00E-03
```

```
 9.  tune_eps_a1_factor = 1.00E+02
10.  tune_eps_a0_factor = 1.00E+01
11.  numInitEstimate = 1.50E+02
12.  v_prop = 2.50E-01
```

These are all advanced parameters that the user generally will not need to set (see the Methods section of the manuscript for details). Option (1) specifies the number of MCMC samples and option (2) specifies the number of burnin iterations. The order of temporal adjacency is specified by (3), with order 1 or order 2 currently supported. Parameters (5-8) specify hyperparameter and parameter initializations as explained in the Methods section of the manuscript. Parameters (9, 10) control the width of the jump kernel for sampling the Negative Binomial variance parameter. To ensure stable parameter estimation, inference using the Negative Binomial Distribution is not performed for a small number of initial MCMC iterations (150 by default), as specified by parameter (11), and instead assumes Normally distributed noise for log transformed data with variance initialized by (12) during these first iterations.

6.3 Inference

Bayesian Adaptive Lasso (BAL)

This algorithm will produce a set of MCMC samples of dynamical system parameters (growth, self-regulation, and interaction terms, across all OTUs). Additionally, it will provide an indicator matrix of inferred presence/absence of interactions. These outputs are used in downstream analyses.

The following is the set of parameters in the configuration file relevant to using this algorithm:

```
    [Bayesian Lasso]
1.  numIters = 10000
2.  numBurnin = 2000
3.  data_std_init = 10
4.  lambda_interact_init = 1.00E+13
5.  gpB_lambda_interact = 1.00E+21
6.  gpA_lambda_interact = 1.00E-09
```

These are all advanced parameters that the user generally will not need to set (see the Methods section of the manuscript for details). Option (1) specifies the number of MCMC samples and option (2) specifies the number of burnin iterations. The remaining parameters are initializations for parameters (4) and settings for hyperparameters (5-6).

Bayesian Variable Selection (BVS)

This algorithm will produce a set of MCMC samples of dynamical system parameters (growth, self-regulation, and interaction terms, across all OTUs) and an indicator matrix of inferred presence/absence of interactions. Additionally, this algorithm provides Bayes factors on the confidence of the edges predicted. These outputs are used in downstream analyses. It is necessary to run this algorithm to produce an interaction network with confidence values on the edge calls (via Bayes factors), as shown in the manuscript.

The following is the set of parameters in the configuration file relevant to using this algorithm:

```
   [Bayesian Select]
1. numIters = 25000
2. numBurnin = 2500
3. data_std_init = 1.00E+05
4. interact_beta_a = 0.5
5. interact_beta_b = 0.5
6. perturb_beta_a = 0.5
7. perturb_beta_b = 0.5
```

Options (1-3) are advanced options that the user generally will not need to set (see the Methods section of the manuscript for details). Option (1) specifies the number of MCMC samples and option (2) specifies the number of burnin iterations. Option (3) specifies an initialization value for the variance of the OTU concentration derivatives.

The remaining options specify hyperparameters on the prior belief for microbe-microbe interactions (4-5) and microbe-perturbation interactions (6-7). Alternate settings for these hyperparameters are recommended depending on whether the user is interested in estimating the underlying qualitative interaction network or alternately using the dynamical system model to perform quantitative predictions, as described in the Methods section for the manuscript.

Maximum Likelihood Ridge Regression (MLRR and MLCRR)

The overall goal of the maximum likelihood algorithms is to infer point estimates of the growth and interaction parameters for the gLV model. As a result, running these algorithms produces a single set of parameters, as opposed to the Bayesian approaches that provide an approximation to the distribution of the parameters. Note that the configuration for MLRR and MLCRR is the same, so there is only one configuration section. To select between MLRR and MLCRR, use the algorithm field of the [General] section, as described above.

The relevant section in the configuration file is:

```
   [Ridge Regression]
1. normalize_counts = 1
2. scaling_factor = 1
3. differentiation = 1
4. mix_trajectories = 1
5. k = 30
6. min = -3
7. max = 2
8. N = 15
9. replicates = 15
```

These are all advanced parameters that the user generally will not need to set (see the Methods section of the manuscript for details). Option (1) allows the user to disable (=0) creating ratios from the counts data, in the case where the supplied data is already in the form of concentrations (i.e., for the probiotic cocktail dataset). In this case, the biomass data should be specified as a column of ones. The scaling factor (2) is added for convenience, if your biomass needs to be converted into different units; this option is generally not used. Option (3) specifies the method for estimating gradients from the OTU concentration trajectories, with options for forward, backward, or central (=1, 2 and 3 respectively).

Option (4), by default (=1) specifies that the *k*-fold cross-validation procedure will break up each subject's data into groups, and that the groups span subjects (i.e., some of both subject 1 and subject 2's data will be part of group A.) Set this parameter to 0 if you have many subjects with few time points, to 1 if there are fewer subjects but more time points. Generally the default (=1) will be work well for most datasets. Option (5) specifies the number of groups *k* for cross-validation. If option (4) is turned off (=0), then *k* must be at most (#subjects - 1). If option (4) is on, it must be at most (#samples - 1). In general, however, these values should be considerably lower than the maximum possible settings. For example, 20 groups for 100 samples.

The next three parameters (6-8) define the numerical region over which the algorithm will search for regularization parameters settings. The region is from 10^min to 10^max in a logarithmically-spaced vector. For example, (min, max, N)=(-3, 2, 5) yields [0.0010, 0.0178, 0.3162, 5.6234, 100.0000]. The computation time will scale cubically with the number of spans in the vector, as there are three regularization parameters in the search space.

Options (9) is the number of different "shuffles" or replications for cross-fold-validation. Note that these replications will be parallelized if cores > 1 in the [Parallel] section. This will speed up your analysis considerably (by a factor of the number of cores used), so we recommend parallelizing when possible.

6.4: Simulations

The simulations module depends on your having run the inference module. It will use the output from whatever inference algorithm is specified in option (6) in the general parameters section.

This module enables the user to run simulations based on the inferred dynamical system model Trajectories are generated by numerically integrating each of a set of MCMC samples from the Bayesian algorithms (or the single estimate from the maximum likelihood algorithms), and summarizing the integrated trajectories using order statistics (the median is reported).

The relevant section in the configuration is (e.g., for **data_cdiff** as provided):

```
    [Simulation]
1.  start_time = 30
2.  end_time = 56
3.  time_step = 0.1
4.  thin_rate = 1
5.  assume_stiff = 1
```

Options (1) and (2) specify the starting and ending times respectively for the simulation. Option (3) specifies the time increment for generating output from the numerical integration. Note that increasing the size of this step is not the same as increasing the step size used by the numerical integration solver, rather it specifies the temporal resolution of the output trajectories. The MCMC thinning rate (4) specifies how many of the MCMC samples to use: thin_rate = *k* would indicate that 1 of every *k* samples is to be used for producing a trajectory via numerical integration. Option (5) indicates whether to allow for stiff differential equations in the numerical integration routine; we recommend the default value (=1), which generally avoids numerical issues and has better runtime performance.

If more than one experimental block is specified, then the start_time, end_time, and time_step options must have the same number of arguments as the number of experimental blocks. For example, this is what is used for **data_diet**:

```
    [Simulation]
 1. start_time = 5 5
 2. end_time = 65 29
 3. time_step = 0.1 0.1
 4. thin_rate = 5
 5. assume_stiff = 1
```

6.5: Linear Stability

The linear stability analysis depends on your having run the inference module. It will use the output from whatever inference algorithm is specified in option (6) in the general parameters section.

This module generates a data structure encompassing all possible combinations of OTUs (2^(#OTU's) - 1), and provides two basic outputs: the probability (frequency) of stability based on MCMC samples, and the median steady state concentrations of OTUs in the stable states. In the case of MLRR/MLCRR, there is one "sample", as it outputs a maximum-likelihood point estimate rather than a set of MCMC samples. As a result, frequency of stability is 0 or 1 for MLRR/MLCRR, but in the range of 0 to 1 for BAL/BVS.

The only relevant parameter for linear stability is:

```
[linear stability]
sample_step = 100
```

The sample step parameter simply describes how many MCMC samples to skip between samples (thinning rate) for use in stability analyses. The MCMC samples used will be from 1:sample_step:num_samples, so if there are 20,000 samples the total samples used for linear stability (using sample_step = 100) will be 200.

Note that because this is a combinatorial analysis, runtime will be extremely long (and extremely large amounts of data generated) if you have a large number of OTUs in your analysis (scaling exponentially). For this reason as well as reasons mentioned previously, we recommend against using more than 20 OTUs for these analyses.

6.6: Post Processing: Visualizations and Keystoneness Analysis Outputs

The post-processing section is not a module *per se*; rather, it specifies what output is to be produced by MDSINE.

```
   [Post Processing]
1. write_parameters = 1
2. write_cytoscape = 1
3. write_trajectories = 1
4. write_stability_analysis = 1
5. perform_keystone_analysis = 1
6. keystone_cutoff = 0.75
```

Most of the options are clear, where 1 is on and 0 is off. Parameters (1), trajectories (3), and stability analysis (4) refer to the output of the inference module, simulations module, and linear stability module, respectively. These analyses will produce files in the output folder named: <Algorithm>.results.<post_processing>.txt. For example, BVS.results.simulations.txt for the trajectories, and BVS.results.parameters.txt for the parameters.

Option (2) will write a file for visualization of the qualitative interaction network in the format for cytoscape's input. The last two parameters (5, 6) for the keystone analysis require the stability analysis to have been performed and written (4). Option (5) is the threshold probability of stability above which the state will be accepted (e.g., a value of 0.75 corresponds to including states with >=0.75 probability of being stable).

7. Output File Formats

7.1 Parameters

Filename: <algorithm>.results.parameters.txt

This is a flat (tab-delimited) file that contains all parameters inferred by the selected algorithm. That is, for each OTU (names included as originally provided), the file contains its inferred growth and self-regulation parameters; interaction parameters with respect to every other OTU in the system; and interaction parameter with perturbation(s), if applicable. For the Bayesian algorithms, the parameters written are the means over all MCMC samples. For BVS, the Bayes factor for each interaction parameter is additionally output.

Example output:

```
parameter_type source_taxon   target_taxon   value   significance   MCMC_std
growth_rate    NA      Strain4 0.92807 224990  0.071163
...
interaction    Strain4 Strain4 -0.002025      224990  0.00016425
interaction    Strain4 Strain6 0       0       0
interaction    Strain4 Strain7 -1.3952e-05    0.018468       0.00010553
...
perturbation   Perturbation1  Strain4 -0.44018       224990  0.051392
perturbation   Perturbation1  Strain6 0.46674 224990  0.076585
...
```

`parameter_type` is either `growth_rate`, `interaction`, or `perturbation`. `source_taxon` is which taxon or perturbation is affecting the `target_taxon`, and `value` the magnitude of the effect (mean of MCMC samples). Significance refers to the Bayes factor, and `MCMC_std` is the standard deviation across all MCMC samples.

7.2 Simulation Trajectories

Filename: <algorithm>.results.simulations.txt

This is a flat (tab-delimited) file that contains the results of the simulations (if that module has been run). That is, for each subject, the predicted trajectory (concentration at each time point) is written to the file for each OTU. If the option has been specified, the actual data (concentrations) for each subject will also be written to file.

Example output:

```
trajectory_ID  taxon   abundance      time    type
1       Strain4 112.6   5       simulation
1       Strain4 117.623 5.1     simulation
...
7       Strain29       60.6    16      data
7       Strain29       72.4    17      data
...
```

7.3 Stability Analysis

Filename: <algorithm>.results.stability_analysis.txt

This is a flat (tab-delimited) file that contains the results of the linear stability analysis (if that module has been run). For each possible subset of OTU's in the system (2^(#OTU's)-1), the file contains the probability (frequency) of stability of that state, as well as the median steady-state concentrations of each OTU in the stable states.

Example output:
```
ProfileID       PerturbationID N_species      frequency       Strain1 ... StrainN
1       0       1       1       0       ...     238.89
...
8190    1       12      0.551111        138.784 ...     0
...
```

7.4 Keystone Analysis

Filename: <algorithm>.results.keystone_analysis.txt
This file contains the results of the keystoneness analysis (if that module has been run). For each experimental regime (i.e., perturbation(s) or control setting), all of the steady states containing one fewer than the maximally-sized stable configuration, and which are a *subset* of that maximal configuration, are written to file. The threshold for inclusion as a stable configuration (default probability of >=0.75 stability) is specified in the configuration file, as described above.

As the keystone analysis is a subset of the stability analysis, it is in the same format as the stability analysis.

7.5 Cytoscape

Filename: <algorithm>.results.cytoscape.txt
This is a flat (tab-delimited) file that contains information about the inferred interaction network needed for Cytoscape visualization. That is, for each OTU (names included as originally provided), the file contains its interaction parameters with respect to every other OTU in the system; and interaction parameter with perturbation(s), if applicable. For the Bayesian algorithms, the parameters written are the means over all MCMC samples. For BVS, the Bayes factor for each interaction parameter is additionally written.

There is no header for this file. Some example lines look like:
```
      Strain6 Strain4 -1 0.0072626 28.8013
      Strain7 Strain4 -1 3.0781e-07 0.00022227
      Strain9 Strain4 -1 3.8794e-07 0.00013335
      Strain13 Strain4 1 8.2323e-07 0.00017781
      Strain14 Strain4 1 0.00038352 0.0052721
```

8. R-Plotting Utilities

There are four included R scripts that will process the output from the post-processing module to produce figures similar to those in the manuscript.

To use the scripts, you must have R installed, and Rstudio is recommended.

The general format for Mac or Linux is:

```
Rscript <plot utility> <input file> <output filename>
```

The four included scripts and their inputs are

plot_model_parameters.R
<algorithm>.results.parameters.txt
plot_simulated_trajectories.R
<algorithm>.results.simulations.txt
plot_stability_analysis.R
<algorithm>.results.stability_analysis.txt
process_keystone_analysis_results.R
<algorithm>.results.keystone_analysis.txt

The plotting of the stability analysis will take some time, and has a third option (number of processes, default=2) to potentially speed up some of the processing. For example:

```
Rscript plot_stability_analysis.R BVS.results.stability_analysis.txt 4
```

The plotting of the trajectories also has an optional argument, which says whether to group by taxa or by subjects. By default, the grouping is by subject. For the trajectory grouping,

```
Rscript plot_simulated_trajectories.R BVS.results.simulations.txt taxon
```

Note that the libraries used for the plotting scripts will be automatically installed if the first part of the script is run from within Rstudio.