MITRE 1.0 user's manual

Eli Bogart

October 22, 2018

Contents

Chapter 1

Introduction

MITRE learns interpretable predictive rules that classify hosts or predict host outcomes based on longitudinal observations of their microbiome. For a full explanation of what this means, why it is important, and how MITRE does it, refer to "MITRE: predicting host status from microbiota time-series data", Elijah Bogart and Georg Gerber (in preparation), and the associated supplementary technical note. For a quick tutorial introduction, see the `README.md` file, or go to `https://github.com/gerberlab/mitre`. This manual describes how to install and run MITRE, how to format your own data for use with MITRE, what types of output are available, and the options that may be specified in a MITRE configuration file to control its execution.

1.1 Installation

The recommended installation procedure is to use `pip` to download MITRE (and any dependencies not already installed) from the Python Package Index:

```
pip install mitre.
```

To install from source, run

```
git clone https://github.com/gerberlab/mitre.git
```

followed by

```
pip install mitre/
```

(note the trailing slash.) To verify that installation was succcessful, run

```
mitre --test
```

A series of status messages should be displayed, followed by 'Test problem completed successfully.'

You will need Python 2.7 installed to install and run MITRE. (Recent versions of Python 2.7 provide the `pip` command by default; the version of Python installed by default on OSX systems is an exception. Running `sudo easy_install pip` should fix this, but a better solution, which does not require `sudo` privileges, is to install your own Python interpreter. This can be conveniently done by installing the `anaconda` or `miniconda` distributions from `https://conda.io`, which also provide an improved package management and installation system.) Only Mac and Linux systems are supported at this time.

1.2 Usage

To use MITRE, create a configuration file specifying input data, the operations to be performed, and output types. Then run

```
mitre path_to_configuration_file
```

The MITRE configuration file options, and what happens when MITRE reads a configuration file, are described in detail in chapter 5 below. Rather than writing a configuration file from scratch, it may be easiest to modify a sample configuration file to suit your needs; an example file is provided in chapter 6 below.

Chapter 2

Formatting input data for MITRE: standard 16S data

This chapter describes, with examples, how to prepare input data files for MITRE when the data is (as is typical) 16S-based abundance data.

2.1 Required files

2.1.1 Abundance table

The abundance data should be provided as a comma-separated table, with the first row providing OTU IDs and the first column providing sample IDs.

Listing 2.1: Example abundance data table

```
"","OTU1","OTU2"
"SampleID1",768,46
"SampleID2",17,7545
```

(Quotes around strings are not necessary.)

2.1.2 Sample metadata table

The sample metadata table specifies an associated subject ID and timepoint for each sample ID. It should be given as a comma-separated table with no header row. No particular ordering is expected.

Listing 2.2: Example sample metadata table

```
SampleID1,SubjectID1,0
SampleID2,SubjectID1,27
SampleID3,SubjectID2,2
```

2.1.3 Subject data table

The subject data table gives information about each subject, (including the value of whatever variable will be used as the outcome that MITRE will try to predict, though that does not need to be explicitly marked.) It should be given as a comma-separated table with a header row, whose first column is the subject ID. (The first field in the header row is ignored.) Values may be either strings or numbers (but variables that appear to be Boolean may be converted to 0/1; see the note for the 'outcome_positive_value' option in section 5.4 below.)

Listing 2.3: Example subject data table

```
subjectID, delivery_mode, allergy, week_of_delivery,
   exposure_count
SubjectID1, cesarean, True, 38.1, 5
SubjectID2, vaginal, True, 39.9, 1
```

2.1.4 Pplacer results

This should be the `.jplace` file created by placing the representative seuqences from each OTU on a reference tree of known 16S dequences using the `pplacer` package in maximum likelihood mode.

2.2 Optional files

2.2.1 Sequence key

If this FASTA file of unique sequences is given, each OTU identifer in the abundance data table that exactly matches one of the sequences will be replaced with the identifier for that sequence in the FASTA file. The same will be done for OTU identifiers in the placement table described below, if it is given. This is useful when working with tables from DADA2 that use the inferred exact sequences as row and column headers.

2.2.2 Pplacer taxonomy table

A comma-separated-value taxonomy table from the `pplacer` reference package used for placement of the OTUs. See `http://fhcrc.github.io/taxtastic/refpkg.html` for a description of the `pplacer` reference package components. The following listing is provided as an aid to identifying the appropriate file, not as a complete example of its necessary properties.

Listing 2.4: Beginning of an example taxonomy table file

```
"tax_id","parent_id","rank","tax_name","root","below_root",
   "superkingdom","superphylum","phylum","subphylum",
   "class","subclass","order","below_order",
   "suborder","family","below_family","subfamily",
   "tribe","genus","below_genus","subgenus",
   "species_group","species_subgroup","species"
"1","1","root","root","1","","","","","","",
   "","","","","","","","","","","","","",""
"131567","1","below_root","cellular
   organisms","1","131567","","","","","","","",
   "","","","","","","","","","","",""
"2157","131567","superkingdom","Archaea","1","131567",
   "2157","","","","","","",
   "","","","","","","","","","","",""
"2","131567","superkingdom","Bacteria","1","131567",
   "2","","","","","","","","",
   "","","","","","","","",""
```

2.2.3 Pplacer sequence information

A comma-separated-value table from the `pplacer` reference package used for placement of the OTUs, giving information about the sequences included in the reference package. See `http://fhcrc.github.io/taxtastic/refpkg.html` for a description of the `pplacer` reference package components. The following listing is provided as an aid to identifying the appropriate file, not as a complete example of its necessary properties.

Listing 2.5: Beginning of an example sequence information file

```
seqname,tax_id,species_name
S000438419,53635,Acidimicrobium ferrooxidans
S001416053,121039,Ferrimicrobium acidiphilum
S000750001,209649,Ferrithrix thermotolerans
```

2.2.4 Placement table

A comma-separated table giving predicted taxonomies for the OTUs in the abundance table, with a header row listing the taxonomic levels at which placements are predicted, from coarsest to finest (last two entries must be "Genus" and "Species"), and with the OTU identifiers in the first column, e.g.:

Listing 2.6: Example placement table

```
"","Kingdom","Phylum","Class","Order","Family","Genus",
    "Species"
"OTU1","Bacteria","Bacteroidetes","Bacteroidia",
    "Bacteroidales","Bacteroidaceae","Bacteroides","dorei"
"OTU2","Bacteria","Verrucomicrobia","Verrucomicrobiae",
    "Verrucomicrobiales","Verrucomicrobiaceae",
    "Akkermansia",NA
```

When a placement for an OTU was not predicted at a particular level, the entry should be NA or empty. Note that this is the format that results from passing the output of `dada2`'s `addSpecies` function to `write.csv`.

Chapter 3

Formatting input data for MITRE: metaphlan results

This chapter describes how to prepare input data files for MITRE when the data is taxonomic abundance estimates generated by metaphlan.

3.1 Required files

3.1.1 Abundance table

The abundance table should be Metaphlan's standard tab-delimited clade abundance table; the upper left corner might look like, e.g.,

Listing 3.1: Example abundance data table

```
Taxonomy          S001      S002
k__Bacteria       100       100
k__Bacteria|p__Actinobacteria    5.6       2.7
```

Note that percentages will internally be divided by 100 to convert to relative abundance fractions. The clade name formatting will be used to establish a tree relating the clades in which each edge has length 1.0.

3.1.2 Sample metadata table

Same format as described in 2.1.2.

3.1.3 Subject data table

Same format as described in 2.1.3.

Chapter 4

Quick guide to available MITRE output types

MITRE offers a number of different output options. This section provides a quick guide to how each may be specified in the configuration file.

To generate...

- **An interactive visualization of the MITRE results as an HTML file:** Set ‘gui_output’ in section ‘postprocessing’ to True.
- **The point estimate of the best rule set, distribution of rule set lengths, and the Bayes factor in favor of the empty rule set, as a text file:** Set ‘quick_summary’ in section ‘postprocessing’ to True.
- **All of the above, plus the most frequent clusters of similar rule sets seen in the posterior distribution, as a text file:** Set ‘full_summary’ in section ‘postprocessing’ to True.
- **A table of Bayes factors indicating the strength of the evidence that particular variables or detectors are involved in the true rule list, as a text file:** Set ‘bayes_factor_table’ in section ‘postprocessing’ to True.
- **A table describing taxonomically the OTU or group of bacteria corresponding to each variable in the model, as a text file:** Specify an appropriate ‘taxonomy_source’ and input data files

in section ‘data’, and set ‘aggregate_on_phylogeny’ to True in section ‘preprocessing’.

- **Cross-validated accuracy assessments for the MITRE point and ensemble estimators, as text files:** Include either a ‘crossvalidation’ or ‘leave_one_out’ section in the configuration file, with all required options given there (see sections 5.10 and 5.11 below.)

- **Convergence diagnostic plots as PDF files:** Set ‘mixing_diagnostics’ in section ‘postprocessing’ to True.

- **Convergence diagnostic measures:** Set ‘mixing_diagnostics’ in section ‘postprocessing’ to True, and use the options described in section 5.7 below to run multiple independent MCMC chains. Then, use the separate command line utility `mitre_mcmc_diagnostics` to calculate $\hat{R}$ values (after Gelman *et al.*, *Bayesian Data Analysis*, 3rd ed, pp. 284-285) for assorted variables using the output tables from the independent runs (refer to that utility’s usage message by running `mitre_mcmc_diagnostics -h`).

- **The full suite of outputs necessary to compare MITRE performance against alternative methods:** See section 5.14 below.

Note that in almost all output text files, rules will be expressed in terms of internal variable identifiers (which may be the names of input OTUs, or numbers representing edges in the phylogenetic tree.)

These may be looked up in the variable annotations file (if it has been generated) which provides a taxonomy-based label for each such variable and, for each variable that represents a higher-level group, a list of the OTUs included in that group.

The interactive visualization provides descriptions of the variables visualized detector refers to; where a visualized detector represents a higher-level group, the interface will draw the subtree of the phylogeny corresponding to the group.

Chapter 5

Configuration options

5.1 How a configuration file is processed

When MITRE reads a configuration file, it attempts to carry out each of the following operations, if the corresponding sections of the configuration file are present:

preprocessing Loading, filtering, transforming, and/or annotating the input data, (section 5.5)

model Preparing a model object with specified parameters (section 5.6)

sampling Drawing samples from the posterior distribution (section 5.8)

postprocessing Generating output files, including summaries of the posterior distribution, mixing diagnostics, or the interactive visualization (section 5.9)

crossvalidation K-fold crossvalidation: splitting the data across a set number of folds, creating and sampling from a model for each fold, assessing the accuracy of the resulting predictive rules for the held-out data, and writing a summary report (section 5.10)

leave_one_out As above, but performs leave-one-out crossvalidation instead (section 5.11).

comparison_methods Applying random forest and L1-regularized logistic regression models to predict the outcome, assessing their accuracy by cross-validation, and writing a report (section 5.12).

All sections are optional. When applicable, the results of the 'preprocessing' operation will be supplied to the 'model' and 'comparison_methods' operations, the results of the 'model' step will be supplied to the 'sampling' step and the cross-validation methods, and the results of the 'sampling' step will be supplied to the 'postprocessing' step.

For each operation that depends on an earlier step, however, it is possible to oomit the earlier step and instead specify a file of previously generated results to be loaded.

The 'data' section (5.4) tells MITRE where to find input data files, and is required if the 'preprocessing' section is present.

The 'description' section (5.3) provides information about the problem that is used to label output files, and is required. The 'general' section (5.2) is optional; it controls options for MITRE operation that apply across all steps in the pipeline.

The 'benchmarking' section (5.14) is special. It modifies the normal sequence of operations to apply the comparison methods to the dataset after some steps of filtering, but before the phylogenetic aggregation process; preprocessing is then completed, and cross-validation is performed. This is a convenience method which facilitated the benchmarking calculations presented in the MITRE manuscript.

5.2 General

Options allowed in section 'general' are:

verbose Boolean, optional. If true, MITRE will print progress messages for most steps, including every MCMC iteration. Internally this sets the log level to `logging.INFO`. Default: false.

5.3 Description

Options allowed in section 'description' are:

tag A short string used to generate output filenames. Required.

5.4 Data

Options allowed in section ‘data’ are:

load_example Allowed values are ‘bokulich’, ‘david’, ‘digiulio’, ‘karelia’, or ‘t1d’. If one of these values is given, MITRE will load one of the pre-packaged example datasets, consisting of data from, respectively,

- Bokulich, N. A., et al., “Antibiotics, birth mode, and diet shape microbiome maturation during early life”, *Science Translational Medicine* 8(343): 343ra82 (2016)
- David, L. A., et al., “Diet rapidly and reproducibly alters the human gut microbiome”, *Nature* 505(7484): 559 – 563 (2014)
- DiGiulio, D. B., et al., “Temporal and spatial variation of the human microbiota during pregnancy”, *PNAS* 112(35): 11060 – 11065 (2015)
- Vatanen, T., et al., “Variation in Microbiome LPS Immunogenicity Contributes to Autoimmunity in Humans”, *Cell* 165(4): 842 – 853 (2016),
- Kostic, A., et al., “The Dynamics of the Human Infant Gut Microbiome in Development and in Progression toward Type 1 Diabetes.”, *Cell Host & Microbe* (2015): `doi:10.1016/j.chom.2015.01.001`.

The first four datasets were reprocessed as described in the MITRE supplementary note. For Kostic et al, metaphlan results and WGS sample metadata from was downloaded from the DIABIMMUNE website (`https://pubs.broadinstitute.org/diabimmune/t1d-cohort`) and minimally reformatted using a script distributed with MITRE. If this option is given, no other options in this section are required.

data_type ‘16s’ (the default) or ‘metaphlan’. Optional. If ‘metaphlan’, MITRE will expect Metaphlan clade abundance estimates as input and handle them specially.

abundance_data Filename or path to a table of OTU abundance data, formatted as described in section 2.1.1 (or 3.1.1) if ‘data_type’ is ‘metaphlan’. Required, unless ‘load_example’ is set.

sample_metadata Filename or path to a table giving a subject and timepoint for each sample in the abundance table, formatted as described in section 2.1.2. Required, unless 'load_example' is set.

subject_data Filename or path to a table of data about each subject, formatted as described in section 2.1.3. Required, unless 'load_example' is set.

jplace_file Filename or path to pplacer results in `.jplace` format, as described in section 2.1.1. Required if 'aggregate_on_phylogeny' is set to True in section 'preprocessing', or 'taxonomoy_source' is 'pplacer' or 'hybrid', unless 'load_example' is set.

sequence_key Filename or path to a FASTA file to be used to rename the OTUs listed in the abudance table, in the case where the name of each OTU is simply a sequence, as in some DADA2 output; see 2.2.1. Optional.

outcome_variable String, specifying which of the columns in the subject data table encodes the outcome that MITRE should try to predict. Subjects for whom this data is not available will be dropped from the calculation. Required, unless 'load_example' is set.

outcome_positive_value The value of the outcome variable that corresponds to a positive outcome. All other (non-missing) values will be considered a negative outcome. Due to a minor limitation of MITRE's data parsing system, *if the outcome variable is given as 'true'/'false', 'True'/'False', etc., in the subject data table, the positive value must be given as '1' or '0' in the configuration file.* Other strings or numerical values should work as expected. Required, unless 'load_example' is set.

taxonomy_source String specifying how taxonomic annotations for each variable (i.e., subtree of the overall phylogeny) should be generated. Valid options are 'table', 'pplacer', and 'hybrid' (see the supplement to the MITRE manuscript for a full explanation of these options; in short, 'table' labels OTUs from a table and higher clades based on the OTUs they contain, 'pplacer' labels all variables based on the pplacer results, and 'hybrid' labels variables based on pplacer results except where a species-level placement is given in a table.) If this option is not present, no such annotation will be done. If given, note that the table of annotations will be written to `[tag]_variable_annotations.txt` at the conclusion of the preprocessing step, unless phylogenetic aggregation is not performed.

placement_table Filename or path to a table of taxonomic placements for OTUs, as described in section 2.2.4.

pplacer_taxa_table Filename or path to a table from the pplacer reference package, formatted as described in section 2.2.2. Required if 'taxonomy_source' is 'pplacer' or 'hybrid'; otherwise ignored.

pplacer_seq_info Filename or path to a table from the pplacer reference package, formatted as described in section 2.2.3. Required if 'taxonomy_source' is 'pplacer' or 'hybrid'; otherwise ignored.

additional_subject_covariates String or comma-separated list of strings specifying additional variables (columns in the subject data table) to be included in the regression calculation. See "Incorporating additional covariates", section 1.6 of the supplementary note to the MITRE manuscript for full details. These variables should be categorical. For quantitative covariates, see below. Subjects where data for any of these variables are missing will be dropped from the calculation. Optional.

additional_covariate_default_states String or comma-separated list of strings specifying, in the appropriate order, the values of the additional covariates to be treated as the default state. For each such covariate, the model will incorporate a change in the odds of the outcome associated with every other value of the covariate, but not this value. Required if 'additional_subject_covariates' is specified.

additional_subject_continuous_covariates String or comma-separated list of strings specifying additional variables (columns in the subject data table) to be included in the regression calculation as quantitative covariates. The name is somewhat misleading, as integer-valued variables are fine here. See "Incorporating additional covariates", section 1.6 of the supplementary note to the MITRE manuscript for full details. Subjects where data for any of these variables are missing will be dropped from the calculation. Optional.

metaphlan_do_weights Optional Boolean, default false, ignored unless 'data_type' is 'metaphlan'. If this is set, variables corresponding to clades will be weighted according to the length of the subtree beneath them (effectively, the number of descendant clades for which a row is present in the metaphlan input file); otherwise, all clades will have the same weight.

metaphlan_weight_scale Optional float, default 1.0, ignored unless 'data_type' is 'metaphlan'. If 'metaphlan_do_weights' is set, scale the subtree lengths by this value.

5.5 Preprocessing

See section 2.1 of the MITRE supplementary note for a detailed discussion of the data preprocessing and filtering procedure controlled by the 'preprocessing' section of the configuration file. Note that the ordering of the options in the configuration file does not matter. Steps will be carried out in the order the corresponding options are presented below, skipping any for which the corresponding options are not given in the configuration file. Except where specified, any of these options may be omitted.

min_overall_abundance Numeric value ($N_{\text{counts,OTU}}$ in the notation of section 2.1). If this is specified, all OTUs with lower total abundance data than this value, summed across all samples, are dropped. (Abundance data is assumed to be measured in 16S read counts at this stage, but this is not enforced.)

min_sample_reads Numeric value ($N_{\text{counts,sample}}$ in the notation of section 2.1). If this is specified, all samples where the total abundance data across all (remaining) OTUs sum to less than this value are dropped. (Abundance data is assumed to be measured in 16S read counts at this stage, but this is not enforced.)

trim_start Numeric value (t_i in the notation of section 2.1). The time to consider as the beginning of the study. (By default, this will be the timepoint of the earliest (remaining) sample.) Samples before this time will be dropped. If this is given, 'trim_stop' must be given also.

trim_stop Numeric value (t_f in the notation of section 2.1). The time to consider as the end of the study. (By default, this will be the timepoint of the latest (remaining) sample.) Samples after this time will be dropped.

density_filter_n_samples Numeric value (N_s in the notation of section 2.1). If this is given, 'density_filter_n_intervals' and 'density_filter_n_consecutive' must be given also. The duration of the study will be divided into the specified number of time intervals, and subjects from whom at least the specified number of samples are available in *every* time window formed from the specified number of consecutive intervals are retained; samples from all other subjects are dropped.

density_filter_n_intervals Integer ($N_{\mathrm{w,filter}}$ in the notation of section 2.1). See above.

density_filter_n_consecutive Integer (N_c in the notation of section 2.1). See above.

take_relative_abundance Boolean. If True, abundance data will be converted to relative abundances by normalizing so that the sum of the data for each sample is 1.0.

do_internal_normalization Boolean. If True, abundance data will be normalized relative to the abundance of a specified set of variables. This option is mutually exclusive with 'take_relative_abundance'. If this is true, 'normalization_variables_file' or 'normalize_by_taxon' must be given.

internal_normalization_min_factor Numeric, optional. Sets the minimum normalization factor for internal normalization (default: 1.) An informative error is raised if we ever try to normalize abundance data by dividing by a factor less than this.

normalization_variables_file String. This should be the path to a file containing one variable name per line. If 'do_internal_normalization' is true, the input data for these variables will be added within each sample to produce a factor by which the data for each individual variable will be divided. Note, the input data is added, and the factor calculated, at the point in the preprocessing at which conversion to relative abundance is usually done; this is after the filter for ‘min_overall_abundance’ is applied but before any other filtering is done which could cause variables to be dropped from the analysis. Conceptually, this makes sense as the primary intent of the ‘min_overall_abundance’ filter is to discard data that is thought to be possibly spurious, and so unreliable; dropping variables for which data is believed to be reliable, but which are not abundant enough that it is desirable to treat them as possibly independent predictors of the outcome of interest, should instead be done using the temporal abundance filtering options described below. In any case, if any variable listed in this file is not found as a variable in the model at the point of normalization, an error occurs. Likewise, if the total abundance of these variables is zero for any sample (that has not been discarded by the time normalization is done), an error occurs. This option is mutually exclusive with ‘normalize_by_taxon’.

normalize_by_taxon String, mutually exclusive with ‘normalization_variables_file’. If this is given, a file formatted as described in section 2.2.4 must be specified as option ‘placement_table’ in section ‘data’. That file will be searched for OTUs which contain this exact string as their placement at at least one taxonomic level. The total abundance of all variables of the model at the point of normalization which belong to the resulting list of target OTUs will be used to normalize the data. It is an error if no such variables exist. To allow the user to monitor which variables are found and used for normalization when using this setting, an output table is written containing the placement information for the matching variables (ending ‘variables_used_for_normalization.csv’)

aggregate_on_phylogeny Boolean. If True, introduce new variables corresponding to subtrees of the overall phylogeny of the OTUs, inferred from `pplacer` results given in the ‘jplace_file’ option above, as discussed in section 2.2 of the supplement. The data for each new variable will be the sum of the data for the OTUs contained in the subtree.

log_transform Boolean. If True, take the natural log of all abundance data. Values less than 10^{-10} will be replaced by 10^{-6} before taking the log.

temporal_abundance_threshold Numeric value (a in the notation of section 2.1). If this option is given, ‘temporal_abundance_consecutive_samples’ and ‘temporal_abundance_n_subjects’ must be also. Variables which exceed the abundance threshold in at least the specified number of consecutive samples in the data from each of at least the specified number of subjects will be kept; others will be dropped. If the data has been log-transformed, this should be specified on a log scale.

temporal_abundance_consecutive_samples Numeric value (N_a in the notation of section 2.1). See above.

temporal_abundance_n_subjects Numeric value (N_i in the notation of section 2.1). See above.

discard_surplus_internal_nodes Boolean. If true, variables corresponding to subtrees rooted at internal nodes which have only one (remaining) child node after the filtration steps have been applied will be dropped from the analysis.

pickle_dataset Boolean. If true, the MITRE datastructure corresponding to the processed and filtered dataset will be serialized to `[tag]_dataset_object.pickle` at the conclusion of the preprocessing step.

5.6 Model

5.6.1 Required values

Required or conditionally required options section ‘model’ are:

load_data_from_pickle String giving the location of a `dataset_object.pickle` file from a previous preprocessing step. Must be given if no ‘preprocessing’ section in this file. If given, a model for that dataset will be built; otherwise, a model will be built for the result of the ‘preprocessing’ step specified in this file.

n_intervals Integer, required. (N_w in the notation of chapter 1 of the MITRE supplementary note.) The study duration will be divided into this many intervals and rules will apply to time windows formed one or more consecutive such intervals.

t_min Numeric value, required. ($W_{\min}$ in the notation of chapter 1 of the MITRE supplementary note.) Rules may apply only to time windows this long or longer.

t_max Numeric value, required. ($W_{\max}$ in the notation of chapter 1 of the MITRE supplementary note.) Rules may apply only to time windows this long or shorter.

5.6.2 Saving the model object

An optional setting allows saving the model for reuse:

pickle_model Boolean. If true, the MITRE datastructure representing the model will be serialized to `[tag]_model_object.pickle` at the conclusion of the model setup step.

5.6.3 Advanced settings

The following advanced options, all numeric, are not required. See the indicated sections of the MITRE supplementary note for explanations of their significance.

prior_coefficient_variance σ_b^2 in the notation of section 1.3. (default 100.0)

hyperparameter_alpha_m α_m in the notation of section 1.4 (default 0.5)

hyperparameter_beta_m β_m in the notation of section 1.4 (default 2.0)

hyperparameter_alpha_primitives α_n in the notation of section 1.4 (default 2.0)

hyperparameter_beta_primitives β_n in the notation of section 1.4 (default 4.0)

hyperparameter_a_empty a_Θ in the notation of section 1.4.1 (default 0.5)

hyperparameter_b_empty b_Θ in the notation of section 1.4.1 (default 0.5)

max_thresholds N_θ in the notation of section 1.2.4 (default, no maximum)

max_rules $m_{\max}$ in the notation of section 1.4 (default 10)

max_primitives $n_{\max}$ in the notation of section 1.4 (default 10)

window_concentration_typical $c_{w,\text{typical}} = 1/\lambda_w$ in the notation of section 1.4.3 (default 5.0)

window_concentration_update_ratio Controls the standard deviation of the proposal distribution for the update δ_w to c_W relative to $c_{w,\text{typical}}$ (default 0.2); see the Appendix.

window_fraction_update_std Controls the standard deviation of the proposal distribution for the update δ_f to f_W (default 0.2); see the Appendix.

delta_l_scale_mean Variance of the normal prior distribution of μ_L in the notation of section 1.4.3, relative to Δ_L^2 (default 50.0).

delta_l_scale_sigma Upper bound on the range of allowed values for σ_L in the notation of section 1.4.3, relative to Δ_L (default 25.0)

lambda_l_offset Amount by which Λ_L in the notation of section 1.4.3 differs from the median of the logarithms of the subtree weights L_i (default 0.0).

5.7 Replicates

If this section is specified, sampling and postprocessing will be done as specified in relevant sections for multiple independent runs, optionally parallelized. Settings for comparison, crossvalidation, benchmarking, etc., will be ignored. Output files will be tagged with '_replicate_0', etc.

Options allowed in section 'replicates' are:

n_replicates Integer, required. Number of replicates to perform.

parallel_workers Integer, optional, default 1. Number of processes to run in parallel.

5.8 Sampling

Options allowed in section 'sampling' are:

load_model_from_pickle String giving the location of a `model_object.pickle` file from a previous model setup step. Must be given if no 'model' section in this file. If given, that model will be sampled from; otherwise, the model specified by the 'model' section of this file will be used.

total_samples Total number of samples to draw. Either this or 'sampling_time' must be given.

sampling_time Number of seconds to run the sampling process. If present, overrides 'total_samples'.

pickle_sampler Boolean. If true, the MITRE datastructure representing the sampler will be serialized to `[tag]_sampler_object.pickle` at the conclusion of sampling.

5.9 Postprocessing

The following settings, all optional, may be specified in section 'postprocessing':

load_sampler_from_pickle String giving the location of a `sampler_object.pickle` file from a previous sampling step. Must be given if no 'sampling' section in this file. If given, generate output from the samples drawn by that sampler; otherwise, from the samples drawn as specified in the 'sampling' section of this file.

burnin_fraction Number between 0 and 1 giving the fraction of samples to discard as burn-in before analysis. Defaults to 0.05.

quick_summary Boolean. If True, write the point summary, the distribution of rule set lengths, and the Bayes factor in favor of the empty rule set `[tag]_quick_summary.txt`.

full_summary Boolean. If True, write the point summary, the distribution of rule set lengths, the Bayes factor in favor of the empty rule set, and the highest-posterior-probability rule set clusters (see section 3.3 of the MITRE supplement) to `[tag]_full_summary.txt`.

bayes_factor_table Boolean. If True, write the 10 variables and 10 detectors with the highest Bayes factors supporting their inclusion in the true model, and those factors, to `[tag]_bayes_factor_table.txt`. See section 3.4 of the MITRE supplement.

gui_output Boolean. If true, prepare an interactive visualization of the results as described in section 3.5 of the MITRE supplement and save it to `[tag]_visualization.html`. Note, this output cannot be generated if 'aggregate_on_phylogeny' is not set.

bayes_factor_samples Integer. Draw this many samples to estimate the prior distribution over the space of detectors in the process of estimating Bayes factors, if the other options in this section require the estimation of Bayes factors (beyond the Bayes factor for the empty rule set.) Default 10000.

mixing_diagnostics Boolean. If true, plot the sampled values of the likelihood, prior, rule set length (total number of detectors), an arbitrary subset of the auxiliary variables ω, and parameters f_w, c_w, μ_L, and σ_L (defined in the MITRE supplementary note) versus MCMC iteration. Results will be saved in `[tag]_likelihood.pdf`, `[tag]_prior.pdf`, etc., and a table (which may grow large, as it contains one row per MCMC iteration) is written to `[tag]_mcmc_traces.csv`.

5.10 K-fold crossvalidation

Required or conditionally required options in section 'crossvalidation' are:

load_model_from_pickle String giving the location of a `model_object.pickle` file from a previous model setup step. Must be given if no 'model' section in this file. If given, that model will be crossvalidated; otherwise, the model specified by the 'model' section of this file will be used.

n_folds Integer; how many folds of cross-validation should be done.

parallel_workers Integer; how many parallel worker processes should be launched.

burnin_fraction Number between 0 and 1 giving the fraction of samples to discard as burn-in before calculating accuracy metrics for each fold.

total_samples Total number of samples to draw for each fold. Either this or 'sampling_time' must be given.

sampling_time Number of seconds to run the sampling process for each fold. If present, overrides 'total_samples'.

Non-mandatory options are:

write_reports_every_fold Boolean. If True, the quick summary described in section 5.9 will be written after sampling concludes for every fold (`[tag]_0_qr.txt`, `[tag]_1_qr.txt`, etc.) Note that accuracy metrics in these reports are calculated based on the *training* set for the fold, not the held-out test set.

write_full_summaries_every_fold Boolean. If True, the full summary described in section 5.9 will be written after sampling concludes for every fold (`[tag]_0_full_summary.txt`, `[tag]_1_full_summary.txt`, etc.) Note that accuracy metrics in these reports are calculated based on the *training* set for the fold, not the held-out test set.

pickle_results Boolean. If true, the true outcomes and predicted probabilities of a positive outcome for each subject in each test set under the point and ensemble classifiers will be written (as serialized Python objects) to `[tag]_cv_point_results.pickle` and `[tag]_cv_ensemble_results.pickle` for further analysis.

At the conclusion of crossvalidation, a table of accuracy metrics for the point and ensemble classifiers learned in each fold, applied to the held-out test data for each fold, will be written to `[tag]_mitre_cv_report.txt`.

5.11 Leave-one-out crossvalidation

The options for section 'leave_one_out' are the same as those for section 'crossvalidation', above, except that 'n_folds' is not required (and is ignored if given.)

At the conclusion of crossvalidation, a table of accuracy metrics for the predictions made on the left-out subjects for each fold will be written to `[tag]_mitre_leave_one_out_report.txt`.

5.12 Comparison

This section supports benchmarking the MITRE models against alternative methods. See the MITRE manuscript for the details of the comparison methods used and an explanation of these parameters. Options in section 'comparison_methods' are:

load_data_from_pickle String giving the location of a `dataset_object.pickle` file from a previous preprocessing step. Must be given if no 'preprocessing' section in this file. If given, a model for that dataset will be built; otherwise, a model will be built for the result of the 'preprocessing' step specified in this file.

cv_type String, optional. If 'leave_one_out', assess the performance of the methods by leave-one-out crossvalidation; otherwise, use k-fold crossvalidation.

n_folds Integer, required unless 'cv_type' is 'leave_one_out', specifying how many folds to use for k-fold crossvalidation.

n_intervals Integer, required. $N_{\mathrm{w,comparison}}$ in the notation of the MITRE manuscript.

n_consecutive Integer, required. $N_{\mathrm{c,comparison}}$ in the notation of the MITRE manuscript.

5.13 Simulated data

5.13.1 Required values

If this section is present, a synthetic dataset will be generated and MITRE will then be applied to the synthetic data set as specified in the other sections of the configuration file. An ordinary input dataset must be specified as usual in the 'data' section, to be used as a template that informs the simulation of the synthetic data.

Options in section 'simulated_data', all mandatory, are:

num_subjects Integer. Number of subjects to simulate.

num_times_sim Integer. Number of timepoints to simulate per subject.

time_window_width Numeric. Width of time window for the introduced perturbation.

min_time_points_in_data Numeric value. In order to be used as a template subject in the simulated data, an original subject must have at least this many timepoints.

time_point_std_percent Float. Standard deviation of truncated Gaussian noise used to add irregularity to time point spacing, as fraction of the average interval between timepoints.

include_start_time Float. In order to be used as a template subject in the simulated data, an original subject's data must start no later than this time. The simulated data starts at this time.

include_end_time Float. In order to be used as a template subject in the simulated data, an original subject's data must end no earlier than this time. The simulated data stops at this time.

num_perturbations Integer, allowed values are 1 and 2. Perturb this many clades in subjects with positive outcome.

data_std_percent Float. Standard deviation for measurement error (as proportion of mean starting value).

time_std_percent Float. Standard deviation for the abundance of the perturbed clade outside the perturbaton window at each time step (as proportion of exp(control_log_mean)).

min_clade_abundance Float. When picking clades to perturb, minimum abundance in any subject/time point.

max_clade_abundance Float. When picking clades to perturb, maximum abundance in any subject/time point.

max_otus_in_clade Integer. When picking clades to perturb, maximum number of OTUs allowed.

pickle_simulation_result Boolean. If true, save simulated dataset to pickle before proceeding.

save_perturbation_info Boolean. If true, write a text file listing the clades where the perturbations have been injected and the corresponding time windows.

counts_concentration Float. Concentration parameter for Dirichlet-multinomial distribution for count data.

num_counts Int. Number of sequencing reads to simulate.

case_log_mean Float. The abundance X of the perturbed clade during the perturbation period is drawn from a log-normal distribution. ($Z \sim \text{Normal}(\mu, \sigma)$, $X = e^Z$). This option specifies the mean parameter (μ) for the log-normal distribution.

case_log_std Float. The standard deviation parameter (σ) for the case log-normal distribution.

control_log_mean Float. In the control subjects, the clade that is being perturbed in the case subjects also has its abundance drawn from a log-normal distribution, but with different parameters. This option sets the mean parameter.

control_log_std Float. The standard deviation parameter for the control log-normal distribution.

control_gets_one_pert Boolean. When the case subjects are receiving 2 perturbations, setting this option to True will result in all of the control subjects receiving one of the perturbations that is being given to the cases. If False, the control subjects will receive no perturbation.

5.13.2 Multiple simulations

Multiple simulations with different numbers of subjects and timepoints can be performed to study the effect of these study design parameters on the performance of MITRE and the comparator methods. The following steps indicate a procedure for performing an investigation of this type.

1. Generate the master dataset with the maximum numbers of subjects and timepoints to analyze. Prepare a configuration file with the desired simulated_data settings (see 5.13.1). With the setting pickle_simulation_result = True, this master dataset will be saved as a pickle file.

2. The MITRE software distribution includes a script (subsample.py) which will generate subsampled versions of the master dataset with user-specified numbers of subjects and timepoints. This script is run from the command line. The first command line argument must contain the pickle file of the original simulated dataset which will be subsampled. To choose how many subjects and timepoints to subsample, use the arguments `--timepoints` and `--subjects` followed by the desired numbers of timepoints and subjects. The final optional argument is `--filename_add`, which can be used to supply a string that will be appended to each filename when the subsampled datasets are saved. To subsample only subjects or only timepoints, do not use the other argument. The program will save the subsampled datasets as pickles. For example, to subsample master.pkl, selecting 48, 32, and 20 subjects, and 24, 18, 12, and 6 timepoints, run: `python subsample.py master.pkl --subjects 48 32 20 --timepoints 24 18 12 6`

3. Run MITRE using a configuration file with the benchmarking section present (see 5.14) on each of these subsampled datasets as well as the master dataset. To load the pickles, use the optional setting load_previous_simulation_result in the simulated_data section of the configuration file.

The 'simulated_data' section of the configuration file has an optional setting which facilitates the loading of these subsampled simulated datasets:

load_previous_simulation_result String. This option can be used to load a pickle file of the sort that is outputted when pickle_simulation_result is set to True. If this option is supplied, actual simulation is bypassed and all other 'simulated_data' options can be ignored. The purpose of this option is to allow a simulated dataset to be loaded at the correct stage of the MITRE benchmarking algorithm. The load_data_from_pickle option in the 'benchmarking' section assumes that the dataset has already been subjected to the preprocess_step1 function, which applies a number of filters and transformations. Therefore, to load simulated dataset pickles in the form that is outputted by the simulator (i.e., before these filters and transformations have been applied), it is more convenient to use this option.

5.14 Benchmarking

If this section is present:

1. Data is loaded following the options in the 'preprocessing' section normally, through the step of conversion to relative abundance.

2. The first comparison-methods crossvalidation is run immediately, following the options specified in the 'comparison' section of the configuration file as usual. (If log-transformation is specified in the 'preprocessing' section, this is applied to the data passed to the comparison methods.) The resulting report will be labeled `[tag]_benchmark_step3_comparison.txt`.

3. Temporal filtering options are applied to a copy of the dataset and a second comparison-methods crossvalidation is run. (Again, if log-transformation is specified in the 'preprocessing' section, this is applied to the data passed to the comparison methods.) The resulting report will be labeled `[tag]_benchmark_step4_comparison.txt`.

4. All later actions in the 'preprocessing' section are applied to the dataset normally, such as aggregation on the phylogenetic tree and log transformation. A third comparison-methods crossvalidation is run. The resulting report will be labeled `[tag]_benchmark_step5_comparison.txt`.

5. The dataset is used for MITRE crossvalidation, following the options specified in the 'model' and 'crossvalidation' or 'leave_one_out' section of the configuration file as usual.

The comparison method F1 scores reported in the manuscript are those from `...step3_comparison.txt` or `...step4_comparison.txt`, whichever are better, to provide the most favorable conditions for the comparison methods.

One option is allowed in this section (but its use is not recommended, as it is likely to lead to confusion.)

load_data_from_pickle String giving the location of a `dataset_object.pickle`, *preprocessed up to conversion to relative abundance, but not farther.* If given, this will be loaded and the first part of the preprocessing stage will be skipped; options governing later steps of the preprocessing operation must still be specified in this file.

Chapter 6

Example configuration file

This file is also available as `example.cfg` in the MITRE source distribution. It specifies that MITRE should load data from the dietary perturbation study of David et al (2014), reprocessed as described in the MITRE supplementary note, learn rules that will predict whether a subject ate the exclusively plant-based diet or not, and produce text and interactive summaries of the results.

This file cannot be used exactly as is: you will need to edit the paths to the data files, and change the outcome and filtering parameters to whatever is appropriate for your own study. (Alternatively, if you do want to rerun this calculation, you can simply replace the options in the 'data' section with `load_example = david` to use the copies of the relevant files included with the MITRE distribution.)

Listing 6.1: Template MITRE configuration file

```
# Note: comments placed on their own lines
# may start with ';' or '#', but
# inline comments should start only with ';'.

[description]
tag = david_diet

[data]
abundance_data = /path/to/your_data/abundance.csv
sequence_key = /path/to/your_data/sequence_key.fa
sample_metadata = /path/to/your_data/sample_metadata.csv
subject_data = /path/to/your_data/subject_data.csv
jplace_file = /path/to/your_data/placements.jplace
```

```
outcome_variable = diet
outcome_positive_value = Plant
taxonomy_source = hybrid
pplacer_taxa_table = /path/to/your_data/taxaTable.csv
pplacer_seq_info = /path/to/your_data/seq_info.csv
placement_table = /path/to/your_data/mothur_placements.csv

[preprocessing]
# Note that the order of these options in the configuration file is
# not important. Those filters and transformations selected will be
# applied in the following order:
# overall abundance filter, sample depth filter, window trimming,
# temporal sampling density filter, relative abundance
# transformation, phylogenetic aggregation, log transform,
# temporal abundance filter, surplus internal node filtering.

# Drop RSVs with less than a certain number of reads across
# all samples
min_overall_abundance = 10
# Drop samples with less than a certain total number of reads
min_sample_reads = 5000
# No need to trim the time window here, -5 to 10 is okay
# trim_start = 20
# trim_stop = 30
# Discard subjects with insufficiently dense temporal sampling:
# divide the window up into n_intervals equal pieces,
# then require at least n_samples within every n_consecutive
# consecutive such pieces
# We don't actually need to do this here, the sampling density
# is good for all 20 subjects.
# density_filter_n_samples = 1
# density_filter_n_intervals = 12
# density_filter_n_consecutive = 2
# Select which transformations should be applied to the data.
take_relative_abundance = True
aggregate_on_phylogeny = True
log_transform = False
# Temporal abundance filtration: keep only those taxa which
# exceed a threshold abundance in multiple consecutive observations
# at least once in a minimum number of subjects. Note that the
# threshold should be on a log scale if the log transform has been
# performed.
temporal_abundance_threshold = 0.001
temporal_abundance_consecutive_samples = 2
temporal_abundance_n_subjects = 4
# Discard taxa representing nodes in the phylogenetic tree not
# required to maintain the topological relationships among
# the other nodes of the tree still included in the model.
discard_surplus_internal_nodes = True
# Save this dataset as a pickle file?
pickle_dataset = True

[model]
# Divide the experiment into this many equal segments and use them
# as the atomic time windows
n_intervals = 10
# Allow rules to apply only to time windows longer than t_min
t_min = 1.0
# Allow rules to apply only to time windows shorter than t_max
t_max = 7.0
pickle_model = True

[sampling]
total_samples = 50000

[postprocessing]
quick_summary = True
full_summary = True
gui_output = True
burnin_fraction = 0.1
```