MaAsLin User Guide v3.0

February 2012

Timothy Tickle and Curtis Huttenhower

**Table of Contents**

A. Introduction to MaAsLin

B. Related Projects and Scripts

C. Installing MaAsLin

D. MaAsLin Inputs

E. Process Flow Overview

D. Process Flow Detail

G. Expected Output Files

H. Troubleshooting

I. Installation as an Automated Pipeline

J. Commandline Options (Modifying Process and Figures)

**A. Introduction to MaAsLin**

MaAsLin is a multivariate statistical framework that finds associations between clinical metadata and potentially high-dimensional experimental data. MaAsLin performs boosted additive general linear models between one group of data (metadata/the predictors) and another group (in our case relative taxonomic abundances/the response). In our context we use it to discover associations between clinical metadata and microbial community relative abundance or function; however, it is applicable to other data types.

Metagenomic data are sparse, and boosting is used to select metadata that show some potential to be useful in a linear model between the metadata and abundances. In the context of metadata and community abundance, a sample's metadata is boosted for one Operational Taxonomic Unit (OTU) (*yi*). The metadata that are selected by boosting are then used in a general linear model, with each combination of metadata (as predictors) and OTU abundance (as response variables). This occurs for every OTU and metadata combination. Given we work with proportional data, the Yi (abundances) are arcsin(sqrt(Yi)) transformed. A final formula is as follows:



For more information about maaslin please visit <http://huttenhower.sph.harvard.edu/maaslin> .

**B. Related Projects and Scripts**

Other projects exist at www.bitbucket.com that may help in your analysis.

**GraPhlAn** is a visualization tool focused on annotated dendrograms. If installing MaAsLin in the SflE framework, install graphlan in sfle/input to allow MaAsLin to produce automated GraPhlAn figures. This is optional and does not affect MaAsLin analysis. For more information on GraPlAn please visit <http://huttenhower.sph.harvard.edu/graphlan> .

**MetaCheck** is a utility project, still underdevelopment, which is targeted at diagnosis and visualizing the metadata in a study. The intent is to create visualizations to comment on the balance, correlation structure, and other properties of your metadata to better use MaAsLin. To download please visit <https://bitbucket.org/timothyltickle/metacheck> .

**QiimeToMaAsLin** is a project that reformats abundance files from Qiime for MaAsLin. Several formats of Qiime consensus lineages are supported for this project. To download please visit <https://bitbucket.org/timothyltickle/qiimetomaaslin> .  
  
**merge\_metadata.py** is a script included in the MaAsLin project to generically merge a metadata file with a table of microbial (or other) measurements. This script is located in maaslin/src and is documented in maaslin/doc/ Merge\_Metadata\_Read\_Me.txt .

**C. Installing MaAsLin**R Libraries: Several libraries need to be installed in R these are the following:  
  
agricolae, FactoMineR, gam, gbm, inlinedocs, lme4, logging, MASS, nlme, optparse, outliers, penalized, pscl, robustbase, testhat  
  
You can install them by typing R in a terminal and using the install.packages command  
  
install.packages(c('agricolae', 'FactoMineR', 'gam', 'gbm', 'inlinedocs', 'lme4', 'logging', 'MASS', 'nlme', 'optparse', 'outliers', 'penalized', 'pscl', 'robustbase', 'testthat'))

**D. MaAsLin Inputs**

There are 3 input files for each project, the "\*.read.config" file, the "\*.pcl" file, and the "\*.R" script. Although the "\*" in the file names can be anything, it needs to be identical for all three files. All three files need to be in the ../sfle/input/maasalin/input/ folder. Details of each file follow:

1. "\*.pcl"

Required input file. A PCL file is the file that contains all the data and metadata. This file is formatted so that metadata/data (otus or bugs) are rows and samples are columns. All metadata rows should come first before any abundance data. The file should be a tab delimited text file with the extension ".pcl" .

2. "\*.read.config"

Required input file. A read config file allows one to indicate what data is read from a PCL file without having to change the pcl file or change code. This means one can have a pcl file which is a superset of metadata and abundances which includes data you are not interested in for the run. This file is a text file with ".read.config" as an extension. This file is later described in detail in section “F. Process Flow Overview” subsection “4. Create your read.config file”.

3. "\*.R"

Optional input file. The R script file is using a call back programming pattern that allows one to add/modify specific code to customize analysis without touching the main MaAsLin engine. A generic R script is provided “maaslin\_demo2.R” and can be renamed and used for any study. The R script can be modified to add quality control or formatting of data, add ecological measurements, affect the MFA visualization, or other changes.

**E. Process Flow Overview**

1. Obtain your abundance table.

2. Obtain your metadata.

3. Format and combine your abundance table and metadata as a pcl file for MaAsLin.

4. Create your read.config file.

5. Create your R script or use the default.

6. Place .pcl, .read.config, .R files in ../sfle/input/maasalin/input/

7. Run.

8. Discover amazing associations in your results!

**F. Process Flow Detail**

1. Obtain your abundance table.

Abundance tables are normally derived from sequence data using Mothur, Qiime, or MetaPhlAn. Please refer to their documentation for further details.

2. Obtain your metadata.

Metadata would be information about the samples in the study. For instance, one may analyze a case / control study. In this study, you may have a disease and healthy group (disease state), the sex of the patents (patient demographics), medication use (chemical treatment), smoking (patient lifestyle) or other types of data. All aforementioned data would be study metadata. This section can have any type of data (factor, ordered factor, continuous, integer, or logical variables). If a particular data is missing for a sample for a metadata please write NA. It is preferable to write NA so that, when looking at the data, it is understood the metadata is missing and it's absence is intentional and not a mistake. Often investigators are interested in genetic measurements that may also be placed in the metadata section to associate to bugs.

If you are not wanting to manually add metadata to your abundance table, you may be interested in associated tools or scripts to help combine your abundance table and metadata to create your pcl file. Both require a specific format for your metadata file. Please see the documentation for QiimeToMaaslin or merge\_metadata.py (for more details see section B).

3. Format and combine your abundance table and metadata as a pcl file for MaAsLin.

Please note two tools have been developed to help you! If you are working from a Qiime output and have a metadata text file try using QiimeToMaaslin found at bitbucket. If you have a tab delimited file which matches the below .pcl description (for instance MetaPhlAn output) use the merge\_metadata.py script provided in this project (maaslin/src/merge\_metadata.py) and documented in maaslin/doc/Merge\_Metadata\_Read\_Me.txt .

PCL format description:

i. Row 1 is expected to be #ID\_indicator and then sample ids in each following column separated by tabs.

ii. Rows of metadata. Each row is one metadata, the first column entry being the name of the metadata and each following column being the metadata value for that each sample.

iii. Row of taxa/otu abundance. Each row is one taxa/otu, the first column entry being the name of the taxa/otu followed by abundances of the taxa/otu per sample.

iv. Abundances should be normalized by dividing each abundance by the sum of the column (sample) abundances.

v. Here is an example of the contents of an extremely small pcl file; another example can be found in this project at maaslin/input/maaslin\_demo.pcl .

#SampleID Sample1 Sample2 Sample3 Sample4

metadata1 True True False False

metadata2 1.23 2.34 3.22 3.44

metadata3 Male Female Male Female

taxa1 0.022 0.014 0.333 0.125

taxa2 0.406 0.029 0.166 0.300

taxa3 0.571 0.955 0.500 0.575

4. Create your read.config file.

A \*.read.config file is a structured text file used to indicate which data in a \*.pcl file should be read into MaAsLin and used for analysis. This allows one to keep their \*.pcl file intact while varying analysis. Hopefully, this avoids errors that may be introduced while manipulating the pcl files.

Here is an example of the contents of a \*.read.config file.

Matrix: Metadata

Read\_PCL\_Columns: Sample2-Sample15

Read\_PCL\_Rows: Age-Height,Weight,Sex,Cohort-Profession

Matrix: Abundance

Read\_PCL\_Columns: Sample2-Sample15

Read\_PCL\_Rows: Bacteria-Bug100

The minimal requirement for a MaAsLin .read.config file is as follows. The Matrix: should be specified. Metadata needs to be named "Matadata" for the metadata section and "Abundance" for the abundance section. “Read\_PCL\_Rows:” is used to indicate which rows are data or metadata to be analyzed. Rows can be identified by their metadata/data id. Separate ids by commas. If there is a consecutive group of metadata/data a range of rows can be defined by indicating the first and last id separated by a “-“. If the beginning or ending id is missing surrounding an “–“, the rows are read from the beginning or to the end of the pcl file, respectively.

A minimal example is shown here:

Matrix: Metadata

Read\_PCL\_Rows: -Weight

Matrix: Abundance

Read\_PCL\_Rows: Bacteria-

With this minimal example, the delimiter of the file is assumed to be a tab, all columns are read (since they are not indicated here). Metadata is read as all rows from the beginning of the pcl file (skipping the first Sample ID row) to Weight; all data are read as all rows from Bacteria to the end of the pcl file. This example refers to the default input files given in the MaAsLin download as maaslin\_demo2.\* .

5. Optionally, create your R script.

The R script is used to add code that manipulates your data before analysis, and for manipulating the multifactoral analysis figure. A default “\*.R” script is available with the default MaAsLin project at maaslin/input/maaslin\_demo2.R . This is an expert option and should only be used by someone very comfortable with the R language.

6. Place .pcl, .read.config, and optional .R files in ../sfle/input/maasalin/input/

7. Run.

Go to ../sfle and type the following: scons output/the\_Name\_of\_your\_pcl\_file\_without\_the\_extension

8. Discover amazing associations in your results!

**G. Expected Output Files**

The following files will be generated per MaAsLin run. In the following listing the term *projectname* refers to what you named your “\*.pcl” file without the extension.

**Output files always created:**

*projectname*\_log.txt

Contains the detail for the statistical engine. This can be useful for detailed troubleshooting.

*projectname*-metadata.txt

Each metadata will have a file of associations. Any associations indicated to be performed after initial variable selection (boosting) is recorded here. Included are the information from the final general linear model (performed after the boosting) and the FDR corrected p-value (q-value). Can be opened as a text file or spreadsheet.

*projectname*-metadata.pdf

Any association that had a q-value less than or equal to the given significance threshold will be plotted here (default is 0.25; can be changed using the commandline argument -d). If this file does not exist, the projectname-metadata.txt should not have an entry that is less than or equal to the threshold. Factor data is plotted as knotched box plots; continuous data is plotted as a scatter plot with a line of best fit. Two plots are given for MaAslin Methodology; the left being a raw data plot, the right being a corresponding partial residual plot.

*projectname*.pdf

Contains the multifactoral analysis visualization. This visualization is presented as a build and can be affected by modifications in the R.script or by using commandline.

*projectname*.txt

A collection of all entries in the projectname-metadata.pdf. Can be opened as a text file or spreadsheet.

**Optional GraPhlAn output (if GraPhlAn is installed):**

*projectname*-ann.txt

Input file for GraPhlAn generated from the MaAsLin run summary file; contains annotation for figure.

*projectname*-core.txt

Input file for GraPhlAn generated from the MaAsLin run summary file; contains the elements of the dendrogram.

*projectname*-ann-core.txt

File for GraPhlAn, PhyloXML format.

*projectname*-graphlan.pdf

GraPhlAn representation of all associations in the summary file.

**Additional troubleshooting files when the commandline –v DEBUG is used:**

data.tsv

The data matrix that was read in (transposed). Useful for making sure the correct data was read in.

data.read.config

Can be used to read in the data.tsv .

metadata.tsv

The metadata that was read in (transposed). Useful for making sure the correct metadata was read in.

metadata.read.config

Can be used to read in the data.tsv .

read\_merged.tsv

The data and metadata merged (transposed). Useful for making sure the merging occurred correctly.

read\_merged.read.config

Can be used to read in the read\_merged.tsv .

read\_cleaned.tsv

The data read in, merged, and then cleaned. After this process the data is written to this file for reference if needed.

read\_cleaned.read.config

Can be used to read in read\_cleaned.tsv .

ProcessQC.txt

Contains quality control for the MaAsLin analysis. This includes information on the magnitude of outlier removal.

**H. Troubleshooting**

1. (Only valid if using Sfle) When using the command "scons output/maaslin/..." to run my projects I get the message:

ImportError: No module named sfle:

File "/home/user/sfle/SConstruct", line 2:

import sfle

Solution: You need to update your path. On a linux or MacOS terminal in the sfle directory type the following.

export PATH=/usr/local/bin:`pwd`/src:$PATH

export PYTHONPATH=$PATH

2. When trying to run a script I am told I do not have permission even though file permissions have been set for myself.

Solution: Most likely, you need to set the main MaAsLin script (Maaslin.R) to executable.

**I. Installation as an Automated Pipeline**

SflE (pronounced soufflé), is a framework for automation and parallelization on a multiprocessor machine. MaAsLin has been developed to be compatible with this framework. More information can be found at <http://huttenhower.sph.harvard.edu/sfle> . If interested in installing MaAsLin in a SflE environment. After installing SflE, download or move the complete maaslin directory into sfle/input. After setting up, one places all maaslin input files in sfle/input/maaslin/input . To run the automated pipeline and analyze all files in the sfle/input/maaslin/input directory type scons output/maaslin in a terminal in the sfle directory. This will produce output in the sfle/output/maaslin directory.

**J. Commandline Options (Modifying Process and Figures)**

Although we recommend the use of default options, commandline arguments exist to modify both MaAsLin methodology and figures. To see an up-to-date listing of argument usage, in a terminal in the maaslin/src directory type ./Maaslin.R -h .

An additional input file (the args file) can be used to apply commandline arguments to a MaAsLin run. This is especially useful when using MaAsLin as an automated pipeline (using SflE) and is a way to document what commandline are used for different projects. The args file should be named the same as the \*.pcl file except using the extension .args . This file should be placed in the maaslin/input directory with the other matching project input files. In this file please have one line of arguments and values (if needed; some arguments are logical flags and do not require a value), each separated by a space. The contents of this file will be directly added to the commandline call for Maaslin.R . An example of the contents of an args file is given here.

Example.args:

-v DEBUG –d 0.1 –C 5

In this example MaAsLin is modified to produce verbose output for debugging (-v DEBUG), to change the threshold for making pdfs to a q-value equal to or less than 0.1 (-d 0.1), and to plot 5 metadata and 5 data (bug) features in the MFA plot (-C 5).