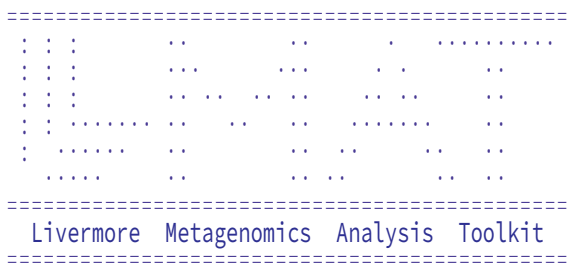
1 LMAT and its pythonic tools



1.1 LMAT Overview

LMAT (Ames, Hysom, Gardner, et al. 2013; Ames, Gardner, Martí, et al. 2015) is an all-in-1 metagenomic analysis toolkit written in C++ and parallelized using OpenMP: taxonomic classification (`read_label` with front-end `run_rl.sh`), content summarization (`content_summ` with front-end `run_cs.sh`), and gene identification (`gene_label` with `run_gl.sh`).

The primary goal of the LMAT taxonomic classification engine is to efficiently assign taxonomic labels to the reads with reference representation down to the species level while maintaining accuracy in the presence of novel organisms. Scalable performance is demonstrated on real and simulated data to show accurate classification even with novel genomes on samples that include viruses, prokaryotes, fungi, and protists. The three related subcomponents of the Livermore Metagenomics Analysis Toolkit (taxonomic profiling, content summarization, and gene annotation) can run separately.

This section contributions to the LMAT package and its software environment focuses on the following tasks:

1. Porting the LMAT software to diverse architectures.
2. Upgrade the C++ code of LMAT to modern standards suitable to current compilers families.

3. Implementing a straightforward way to install the software and the dependencies through the use of `CMake`.
4. Developing post-analysis utilities to take profit of the LMAT rich output for different purposes.
5. Providing support for LMAT to `Recentrifuge` not only for the LMAT parallel output but also for LMAT plasmids detection.

1.2 LMAT Details

LMAT is a parallel shared-memory code that enables faster search using larger seeds (*k-mers*). LMAT seeks taxonomic identifiers (*taxids*) connected to the k-mers located in the reference genome database in a non-redundant fashion (Ames, Hysom, et al. 2013). The LMAT databases promote powerful data mining of taxonomic information while supporting exhaustive comparison between taxids competing for the assignments by using a different classification method adaptable to the taxonomic level of interest (Ames, Gardner, et al. 2015). The new classification technique of LMAT is a particular implementation of the Lowest Common Ancestor (LCA) algorithm with the best-match choice depending on the specific taxonomic exploration results to assign the most taxonomic-level-specific taxid available to each sequence (Ames, Hysom, et al. 2013).

1.3 LMAT improved installation with CMake

We developed a quick installation procedure that uses `CMake` to ease the process, by downloading, building and installing all the required packages.

1.3.1 Required software

The software that should be pre-installed in the system is:

- CMake3
- C/C++ compiler with OpenMP support (like gcc, clang, icc, xlc)
- Recommended: python, for some tools
- Optional: MPI, for use in building a Reference Database

1.3.2 Using redoall to build LMAT easily

The shell script `redoall` is a convenient wrapper that directs the installation through CMake for typical compilers (GNU gcc, clang/LLVM, Intel C/C++ compilers and IBM XL compilers for Power 8 and 9):

```
usage: redoall [profile] [compiler]
```

The first optional parameter chooses the build profile of CMake:

- **D** for `Debug`
- **R** for `Release` (this is the current default)
- **I** for release with debug info (`RelWithDebInfo`)
- **M** for release with minimum size (`MinSizeRel`)
- **clean** for just cleaning everything

The second optional parameter selects the compiler family:

- **gnu** for using GCC
- **intel** for using Intel compilers
- **clang** for using clang compilers
- **ibmpwr9** for compiling in Power 9 with IBM compilers
- **ibmpwr8** for compiling in Power 8 with IBM compilers

1.3.3 Example for GNU gcc (release profile)

```
git clone https://github.com/LivGen/LMAT.git
cd LMAT
./redoall
```

1.3.4 Example for Intel compilers (debug profile)

```
git clone https://github.com/LivGen/LMAT.git
cd LMAT
./redoall D intel
```

1.4 LMAT post-processing with Recentrifuge

When analyzing datasets with LMAT, the score-oriented approach of `Recentrifuge` offers an easy way to visualize the data, compare between samples, and assess the validity of the LMAT classifications. Please see the Section 1.4.2 for details on running Recentrifuge for LMAT datasets. A distinctive hallmark of LMAT is its ability to classify thousands of plasmids properly. LMAT assigns a taxonomic id (`taxid`) to these plasmids beyond the NCBI system. Recentrifuge offers support for this extended classification (see Section 1.4.1 and Section 1.4.2.4 for details).

1.4.1 Support for LMAT plasmid classification

One of the most interesting but still quite unknown features of the LMAT software is its ability to properly classify over 4000 plasmids. These plasmids are assigned a taxonomical id (`taxid`) beyond the NCBI system. Recentrifuge offers support for this extended classification, but requires the LMAT provided file `plasmid.names.txt` located in the same directory as the NCBI nodes information files. This location is controlled by the flag **-n/--nodespath**.

If Recentrifuge finds the `plasmid.names.txt` file, it will parse the plasmids taxid and name using ad hoc regular expressions in order to present the user with a meaningful name for the very diverse plasmids. Recentrifuge is also doing a check to assure that every plasmid in the file is compatible with the NCBI taxonomy used, so that only those passing are added. Please see further details in Section 1.4.2.4.

1.4.2 Running Recentrifuge for LMAT

1.4.2.1 Quick start

Let's suppose you have installed Recentrifuge with `pip` or cloned the repo in your home directory under `~/recentrifuge`, and you would like to analyze and compare the LMAT output from several samples. Provided you have used `retaxdump` to populate `./taxdump` and you have each sample in one different subdirectory under the current one, you can put Recentrifuge to work with just the following line:

```
~/recentrifuge/rcf -l .
```

where you should exclude the `~/recentrifuge/` part for `pip` installations in this and future command-lines.

If you would like to include LMAT plasmids classifications in your results, please add the LMAT file `plasmid.names.txt` to the `taxdump` directory as detailed in Section 1.4.1.

1.4.2.2 Automatic detection of LMAT multiple output

The above-stated solution is Recentrifuge LMAT automatic mode: it searches for directories not starting with a dot under the current one and, in each of them, it will suppose that a set of LMAT output files resides (typically `Long.file.name.ending.in_output??.out`, where the `??` goes from 0 to the number of cores used by LMAT minus one). Then, Recentrifuge will load each sample in parallel, parsing sequentially the output files under each of those subdirectories. Obviously, this is the most convenient mode of operation of Recentrifuge for large sets of samples analyzed by LMAT, to avoid specifying them one by one.

In case you want to be more specific about the place of the samples to be processed, you can specify the directory that contains the output files for each sample. So, if the samples *S1*, *S2* and *S3*, for instance, are in the directories `./setA/S1`, `./setA/S2`, `./setB/S3`, respectively, the command would be:

```
~/recentrifuge/rcf -l ./setA/S1 -l ./setA/S2 -l ./setB/S3
```

Finally, if you happen to have several samples analyzed by LMAT in the same directory, you can select them one by one. For example, if the samples *S10*, *S11* and *S12* are together in the directory `./all_samples`, the command would be:

```
~/recentrifuge/rcf -l ./all_samples/S10 -l ./all_samples/S11 -l ./all_samples/S12
```

You can mix the two latter approaches, as Recentrifuge will detect if, for the LMAT output files to be parsed, you are selecting a whole directory or just a file prefix to be used inside a directory.

1.4.2.3 File format

Recentrifuge reads the direct outputs from LMAT and shows diverse descriptive statistics both about the reads and the LMAT classification process. Note that the output from LMAT is quite verbose and, in the case of large datasets, it could take some minutes to be processed by Recentrifuge. For example, for the LMAT analysis of the SRR829867 dataset using 32 cores and filtering reads classified with a LMAT score under -1, the Recentrifuge console output is:

```
Loading output file ~/SRR829867/SRR829867.fasta.lmat-4-14.20mer.db.lo.rl_output0.out...OK!
Loading output file ~/SRR829867/SRR829867.fasta.lmat-4-14.20mer.db.lo.rl_output1.out...OK!
(...)
Loading output file ~/SRR829867/SRR829867.fasta.lmat-4-14.20mer.db.lo.rl_output31.out...OK!
Seqs read: 23_025_264   [2.33 Gnt]
Seqs clas: 22_870_103   (0.67% unclassified)
Seqs pass: 7_245_243    (68.32% rejected)
DB Matching: Multi = 77.0%  Direct=22.3%  ReadTooShort=0.0%  LowScore=0.0%  NoDbHits=0.7%
Scores: min = -1.0, max = 2.8, avr = -0.5
Length: min = 101 nt, max = 101 nt, avr = 101 nt
4345 taxa with assigned reads
/Users/martijm/local/lmat-scripts/SRR829867 sample OK!
Load elapsed time: 347 sec
```

1.4.2.4 More about plasmids

If you would like to include LMAT plasmids classifications in your results, please add the LMAT plasmids file `plasmid.names.txt` to the `taxdump` directory as detailed in Section 1.4.1. Recentrifuge will detect it and proceed to parse the plasmid data using regular expressions. Recentrifuge will output a summary for this parsing and for a sanity check of the plasmids data found:

```
Loading LMAT plasmids... OK!
Plasmid sanity check:  rejected (taxid error) = 286  rejected (parent error) = 2
Plasmid pattern matching:  1st type = 4066   2nd type = 284   other = 0
```

If the debug flag is active (**-d**), Recentrifuge will give details:

```
Plasmid taxid ERROR! tid=294 already a NCBI tid. Declared parent = 294 but NCBI parent = 136843
Plasmid details:  Pseudomonas fluorescens strain PC20 plasmid pNAH20,
                  complete sequence[sequence_id 814379][seq_data_id 7244447][tax_node_id 294]
Plasmid taxid ERROR! Taxid=1318 already a NCBI taxid.
                  Declared parent is 1318 but NCBI parent is 1301.
Plasmid details:  Streptococcus parasanguinis plasmid pFW213,
                  complete sequence[sequence_id 806688][seq_data_id 7037937][tax_node_id 1318]
(...)
Plasmid parent taxid ERROR! Taxid=2521 and parent=2521.
Plasmid details:  Plasmid pADB201 (from Mycoplasma mycoides),
                  complete genome[sequence_id 829559][seq_data_id 8131462][tax_node_id 2521]
(...)
```

1.4.2.5 Scoring scheme

Recentrifuge supports the specific scoring scheme of LMAT, which is the default and only choice for LMAT outputs. The **--minscore** parameter works for the final LMAT score of the read. LMAT classification scores between 0 and -1 are considered not very reliable, and under -1 the classifications should be used with caution.

1.4.2.6 Advanced example

As a more complete example, to analyse the LMAT output:

- with the taxonomy files downloaded to `/my/tax/dir`,
- from samples X1 (in directory `./A/X1`), X2 (in directory `./B/X1`), X3 (in directory `./C/X3`) and X4 (in directory `./C/X4`),
- with ONE negative control (in directory `./CTRL`),
- excluding reads with a LMAT classification score under -1.0,
- excluding taxa (un)assigned to Cellular organisms (taxid 131567) and assigned to Chordata (taxid 7711),
- with the output Excel in `cmplexCruncher` format,
- and saving the output to `Xsamples.rcf.html` file,

the command would be:

```
~/recentrifuge/rcf -n /my/tax/dir -l ./CTRL -l ./A/X1 -l ./B/X2 -l ./C/X3 -l ./D/X4 -c 1
        -y "-1.0" -x 131567 -x 7711 -e CMPLXCRUNCHER -o Xsamples.rcf.html
```

1.5 LMAT and PERM: tuning the kernel

LMAT uses PERM, a `C` library for persistent heap management used with a dynamic-memory allocator, also developed at LLNL. For PERM (so LMAT) to work in the right conditions, some kernel tuning is advisable:

- Turn off periodic flush to file and dirty ratio flush:

```
echo 0 > /proc/sys/vm/dirty_writeback_centisecs
echo 100 > /proc/sys/vm/dirty_background_ratio
echo 100 > /proc/sys/vm/dirty_ratio
```

- Turn off address space randomization:

```
echo 0 > /proc/sys/kernel/randomize_va_space
```

1.6 LMAT front- and back-ends

With the aim of adapting LMAT software (Ames, Hysom, et al. 2013) for its easy use in longitudinal metagenomics analysis we developed Python front- and back-end scripts for the LMAT pipeline in order to manage the added complexity of longitudinal series analysis. Some of them are supplied as `lmat-tools` of the `cmplxCruncher` package (*in preparation*), which have been used in some publications (Martí et al. 2017; Martí 2018).

1.6.1 LMAT front-ends

They are: `pyLMAT_rl.py`, the launcher for LMAT `run_rl.sh`, that is, the first step in the LMAT pipeline or `read labeling`; `pyLMAT_cs.py`, the launcher for LMAT `run_cs.sh`, the `content summarization` step in the LMAT pipeline; and, `pyLMAT_gl.py`, the launcher for LMAT `run_gl.sh`, the `gene labeling` step in the LMAT pipeline. Basically, all three scripts permit to manage the execution of parallelized LMAT against several fasta files coming from different sampling points in longitudinal metagenomics studies, typically metagenomic time series.

1.6.2 LMAT back-ends

`pyLMAT_rescore.py` is the most significant LMAT back-end that we have developed. It enables not only to quick recalculate the LMAT results for different cutoff levels, but also to pull out reads with specific NCBI taxId (*human* reads by default). For example, issuing **pyLMAT_rescore.py -p path/to/results -f file.fna -m -1 0 1 -t=32** will regenerate the LMAT results for cutoffs -1, 0 and 1, and will pull human reads for each cutoff level, parallelizing in 32 threads.

1.7 `cmplxCruncher` interface with LMAT

Currently, the bridges between LMAT and `cmplxCruncher` are `LMATgl2cmplx.py` and the pipeline composed by `rawlmat2lmat.py` and `lmat2cmplx.py`. This pipeline enables to post-process the several `fastsummary` output files from LMAT first and second steps launched by the LMAT front-ends described above through transforming the LMAT results so that they achieve a format and content ready for `cC` computations. Such data arrangements are done independently for two different taxonomic levels: genus and species. Analogously, the script `LMATgl2cmplx.py` parses the LMAT gene labeling output to a format ready for `cC` to enable longitudinal analysis of functional results in SMS studies.

A detailed step-by-step procedure for a clinical longitudinal metagenomic application —human gut virome— (Minot et al. 2013) that includes this interface between LMAT and `cmplxCruncher` can be found in JM Martí. 2018. Robust Analysis of Time Series in Virome Metagenomics. In *The Human Virome: Methods and Protocols*. (Ed. by A Moya and V Pérez Brocal). Chap. 17, pp. 245–260. Springer Nature, New York, NY.

1.8 pydomain

Besides Recentrifuge, we have developed other post-processing tools that also parse the LMAT output, such as `pydomain`. This code searches for reads with assignment candidates exceeding a particular score and belonging simultaneously to different taxonomic domains and, then, reclassifies them in one of the eleven possible sets depending on their multi-domain nature: the intersections between Archaea, Bacteria, Eukarya, and Viruses&viroids. If all the candidates belong to the same taxonomic domain, `pydomain` classifies the read in such domain. The interest in multi-domain reads could be technical (chimeras), or biological, for example, viral DNA merged with the host DNA or CRISPR sequences.

As an example of the typical output, Figure 1.1 shows the graphics output of `pydomain` for a sample (*code day 882, subsample B*) of the study "Rapid evolution of the human gut virome," a longitudinal characterization of intestinal viral particles from SMS sequencing (Minot et al. 2013).

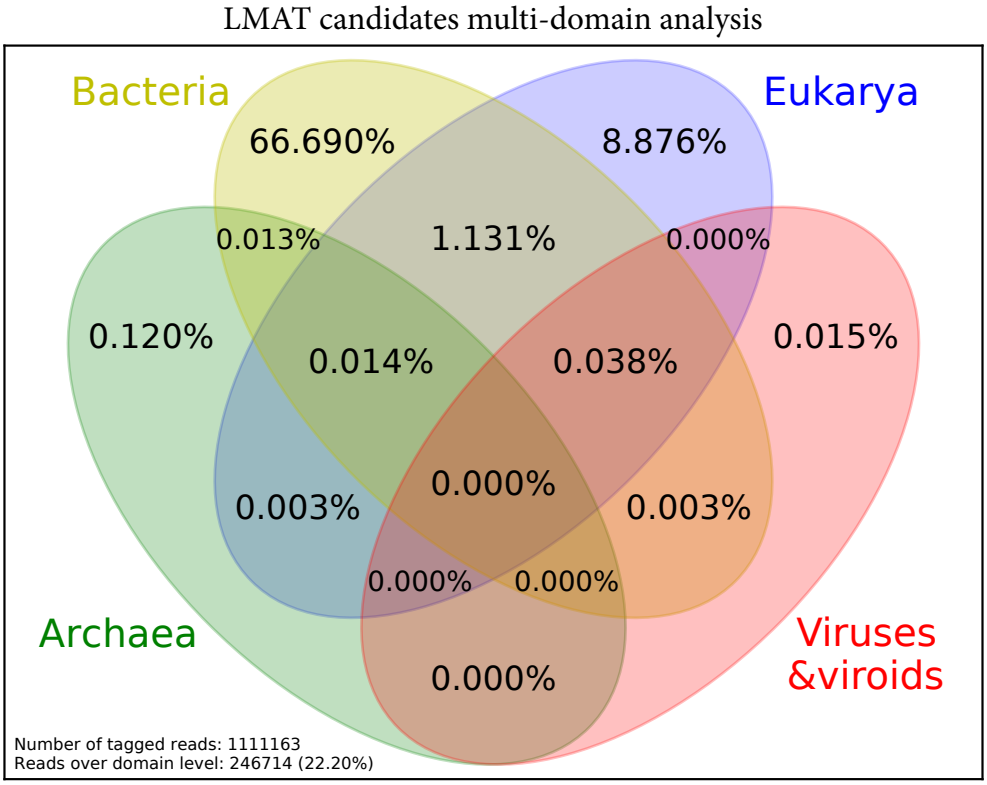


Figure 1.1. Example of the graphics output of `pydomain`.

1.9 pylasmid

`pylasmid` is yet another microbiological analysis tool and yet another parallel code that post-process the output of LMAT. For a given set of LMAT plasmids (or for all the plasmids in the LMAT database), it walks the reads of the LMAT output and those that are assigned to a plasmid pass to the next step. Then, the algorithm searches for the LCA of all the assignment candidates exceeding a particular score in all those reads and reassigns the read to such LCA. Finally, for each of the plasmids found, `pylasmid` writes the sequences to a file and plots a summary tree. Figure 1.2 shows an example of this tree for the plasmid pBWH2B of the bacterium *Bacteroides cellulosilyticus* WH2.

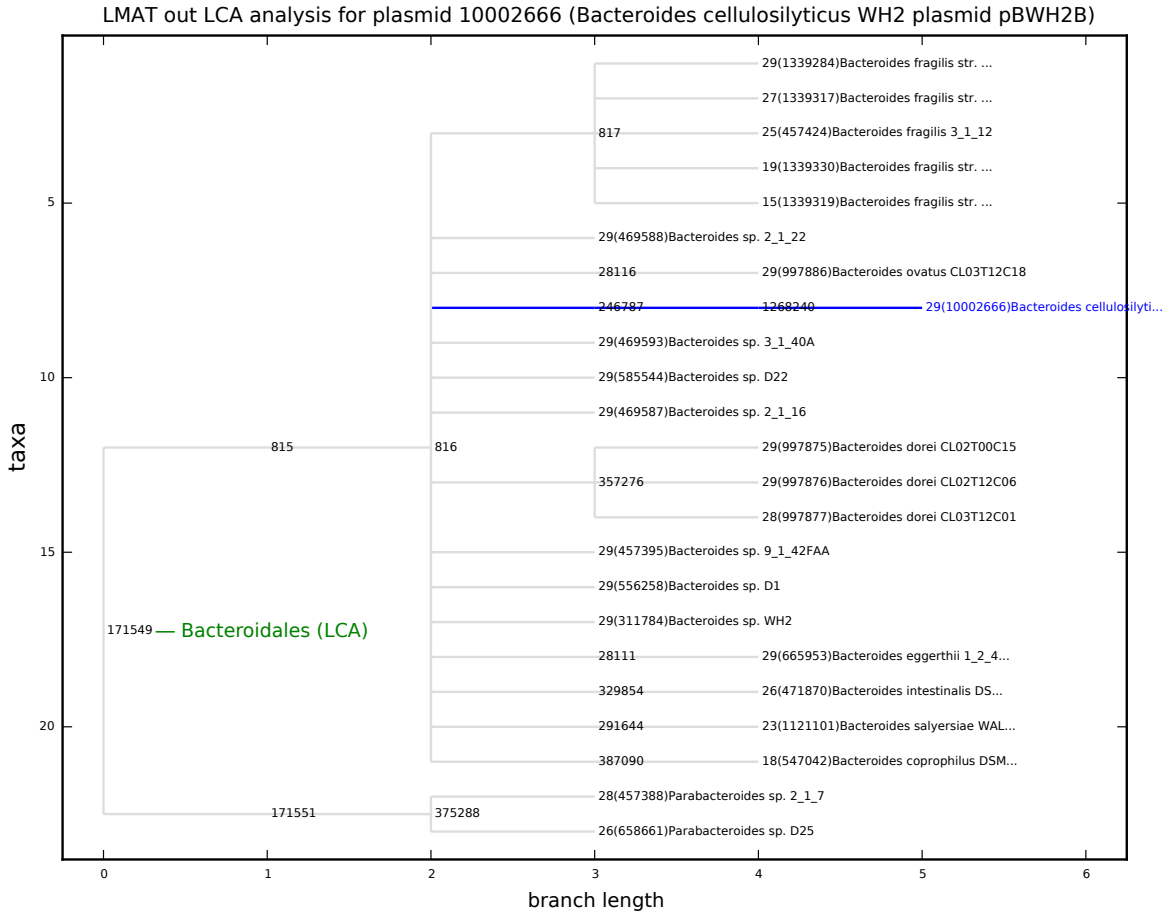


Figure 1.2. Example of tree plotted by `pylasmid` for a plasmid of *Bacteroides cellulosilyticus* WH2. The tree contain all the related candidates in the datasets to their LCA.

1.10 pyLCA

`pyLCA` is a tool with a very similar philosophy to `pylasmid` (see Section 1.9), but with a more general approach. `pyLCA` retrieves the LCA for all the assignment candidates exceeding a particular score in those reads assigned to an NCBI taxid entered by the user.

References

Ames SK, Gardner SN, Martí JM, Slezak TR, Gokhale MB, and Allen JE. 2015. Using populations of human and microbial genomes for organism detection in metagenomes. *Genome research*. **25**: 1056–1067.

Ames SK, Hysom DA, Gardner SN, Lloyd GS, Gokhale MB, and Allen JE. 2013. Scalable metagenomic taxonomy classification using a reference genome database. *Bioinformatics (Oxford, England)*. **29**: 2253–2260.

Martí JM, Martínez-Martínez D, Rubio T, Gracia C, Peña M, Latorre A, Moya A, and Garay CP. 2017. Health and Disease Imprinted in the Time Variability of the Human Microbiome. *mSystems*. **2**: e00144–16.

Martí JM. 2018. Robust Analysis of Time Series in Virome Metagenomics. In *The Human Virome: Methods and Protocols*. (Ed. by A Moya and V Pérez Brocal). Chap. 17, pp. 245–260. Springer Nature, New York, NY.

Minot S, Bryson A, Chehoud C, Wu GD, Lewis JD, and Bushman FD. 2013. Rapid evolution of the human gut virome. *Proceedings of the National Academy of Sciences of the United States of America*. **110**: 12450–12455.

Contents