Chapter 17

Robust Analysis of Time Series in Virome Metagenomics

Jose Manuel Martí

Abstract

Metagenomics is a powerful tool for assessing the functional and taxonomic contents in biological samples as it makes feasible to study, simultaneously, the whole living community related to a host organism or medium: all the microbes, including virus, bacteria, archaea, fungi, and protists. New DNA and RNA sequencing technologies are dramatically decreasing the cost per sequenced base, so metagenomic sequencing is becoming more and more widespread in biomedical and environmental research. This is opening the possibility of complete longitudinal metagenomic studies, which could unravel the dynamics of microbial communities including intra-microbiome and host-microbiome interactions through in-depth analysis of time series. For viruses, this is particularly interesting because it allows broad interaction studies of viruses and hosts in different time scales, as in bacteria–phages coevolution studies.

This chapter presents computational methods for an automatic and robust analysis of metagenomic time series in virome metagenomics (RATSVM). The same theoretical frame and computational protocol is also suitable for longitudinal studies of spatial series to uncover the dynamics of a microbial community with viruses along a selected dimension in the space. In order to conveniently illustrate the procedure, real data from a published virome study is used. The computational protocol presented here requires only basic computational knowledge. Several scripts have been prepared to ease and automate the most complicate steps, they are available in the RATSVM public repository. For some of the methods a mid-range computing server is advisable, and for some others, it is required. A fat-node with large memory and fast I/O would be the best choice for optimum results.

Key words Robust-analysis, Time-series, Longitudinal metagenomics, RATSVM, Virome, Host-microbiome coevolution, Bacteria-phages coevolution

1 Introduction

Studies of microbial communities by metagenomics are becoming more and more widespread in different biological fields, like in clinical, food, and environmental research [1, 2]. These works are encouraged by new DNA and RNA sequencing technologies that are dramatically decreasing the cost per sequenced base. Thanks to these technological advances, sets of sequences belonging to microbial communities from different sources and times can be retrieved and analyzed to unravel spatial and temporal patterns in the involved microbiota [3].

Generally speaking, biology has recently acquired new conceptual and technological tools to model, investigate, and finally, understand living organisms at a systems level. This has been possible, ultimately, thanks to joint experimental and computational approaches, progress in quantitative techniques, and large-scale measurement methods. Particularly, systems biology strives to unravel the general laws governing the complex behavior of microbial communities [4–6] which includes an attempt for a universal dynamics formulation [7]. Complex microbial communities with viruses and their hosts can be approached in light of ecological theory, which includes general principles like Taylor's law [8, 9] relating the temporal or spatial variability of the population with its statistical average. This law is known to be ubiquitous in the natural world and has been called "fluctuation scale law." It can be found in the core of very diverse systems that could seem unrelated at first sight, such as stock markets [10, 11], random walks [12], animal populations [9, 13, 14], spatial and temporal dynamics in tree forests [15], gene expression in yeasts [16], and even the human genome [17].

Taylor's law has been successfully applied to microbiota spatially [18], with results showing that this population tends to be a concentrated one, far from exhibiting a random or uniform distribution. This ubiquitous law has been tested in experimental settings [19, 20], beyond several efforts to infer the community structure from a dynamic perspective [21–23].

Any rigorous dynamic analysis of the composition of the microbiome requires accurate and reliable tools for assessing the taxonomic contents in the biological samples involved in the study. The best laboratory protocols should be followed by the best computing procedures to minimize any bias from the real composition of the studied microbial community [24]. Post-sampling and crossover contamination, negative and positive errors, abundance rate distortion, are all examples of subtle and persistent problems that often arise in the practice of metagenomics [25–27]. The challenge of precision in metagenomics can only be tackled by a joint experimental and computational effort in such a direction [24, 28, 29]. On the computational side and in metagenomics, that means, at least, robust procedures able to give a reliable taxonomic composition for each and every sampling point belonging to the longitudinal study. In this way, the last step of the protocol, the time series analysis, is done in the right conditions concerning data quality.

So, one clear goal is to efficiently assign taxonomic labels to the reads down to the species level for reads with reference representation in the used databases and, at the same time, to maintain accuracy in the presence of novel organisms by avoiding overly specific (e.g., species and strain) taxonomic assignments. One important collateral effect of this approach is to alleviate the computational bottleneck by limiting the number of unlabeled reads subjected to additional computational interrogation. The software chosen for accomplishing this central objective has been LMAT [30] (Livermore Metagenomics Analysis Toolkit) from LLNL, with some improvements and the better database available, LMAT-Grand [31].

Nevertheless, before the taxonomic classification, it is recommended to remove all the reads that map against the host, in case it is a single organism not to be included in the study, as happens typically with humans. Even if that is not the case, this step eliminates a very significant amount of human contamination. The classifier could manage these issues to some extent, but from a robust approach, it is quite better to use a specialized method to effectively detect and remove the reads belonging to the host before the step of taxonomic classification. The software selected between several alternatives for an efficient and reliable host removal has been Bowtie2 [32].

An unavoidable set of software and data is provided by NCBI. Particularly useful here is the SRA toolkit software, which takes two subsections in this chapter, and the NCBI SRA databases [33]. In other part of the protocol, they are also indirectly accessed when using entrez [34]. Another essential tool when working with SAM files is the SAMtools [35]. Git is in the center of the management for all the software repositories used along the protocol [36].

Figure 1 summarizes the main stages of RATSVM protocol, distributed in an imaginary axis from pure preparation steps to pure computational ones. With the aim of illustrating the procedure conveniently, real data from a published time series study about the human gut virome is used along the chapter [37].

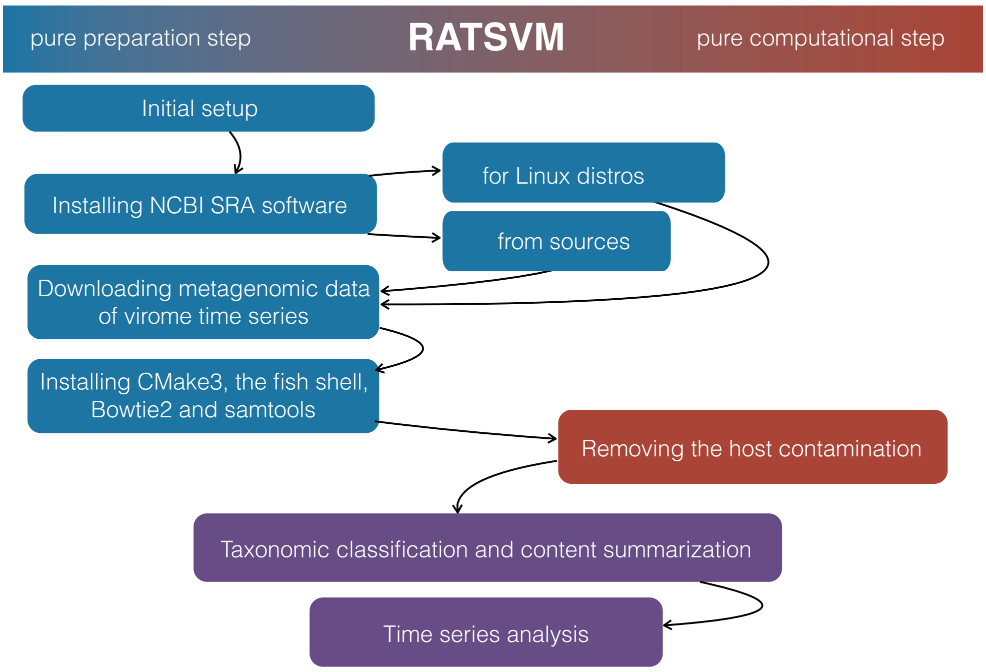
pure preparation step
RATSVM
pure computational step
Initial setup
Installing NCBI SRA software
for Linux distros
from sources
Downloading metagenomic data of virome time series
Installing CMake3, the fish shell, Bowtie2 and samtools
Removing the host contamination
Taxonomic classification and content summarization
Time series analysis

Fig. 1 Main stages of RATSVM protocol, distributed in an imaginary axis from pure preparation steps to pure computational ones

2 Materials

2.1 Overview

This biocomputing protocol, as well as state-of-the-art laboratory protocols concerning viruses, requires advanced equipment. For some of the methods of the chapter a mid-range computing server is advisable, and for some others, it is required. A fat-node with large memory and fast I/O would be the best choice for optimum results.

2.2 Details

Hardware: For best results, it is recommended a fat-node, that is, a SMP (Symmetric MultiProcessing) system with a large quantity of memory per core and a wide-bandwidth and low-latency access to memory and scratch I/O. A memory of ~0.5 TB is required. Intel processors or compatible are required, as some software will be using SSE4 (Streaming SIMD Extensions 4) for increased performance. The total disk space needed to perform the entire protocol is about 1.5 TB in a fast disk. For at least 0.5 TB of them, a local SSD (Solid-State Drive) is advised.

Software: A GNU/Linux distribution is recommended. If the other conditions are met, other flavors of UNIX could be used too, including OS X. A bash 4.2 shell could be used, though the fish shell 2 is recommended (installation instructions included in the protocol). Basic C/C++ development environment, including GNU GCC compilers and essential development libraries, are usually available by default in any GNU/Linux system. Python 3.5 or later is needed. Additionally, for the last step of the protocol the following python packages are required: pandas >1.16.0 with openpyxl, numpy, scipy, matplotlib >1.5.0 and xlrd >0.9. Detailed installation instructions are included in the protocol for any other software.

Network: Stable internet connection with medium to large bandwidth (>10 Mbps is preferable). Normal to high latency levels are acceptable.

3 Methods

3.1 Initial Setup

1. To begin with, create the ratsv directory where software and data will be allocated for Robust Analysis of Time Series in Virome Metagenomics and change directory there (Important: hereafter, the user prompt is marked with the initial > sign which is not part of the command; *see* **Note 1**):
   ```
   > mkdir ~/ratsvm; cd ~/ratsvm
   ```
2. Clone the ratsv git repository in GitHub (*see* **Note 2**) which contains tools and data:
   ```
   > git clone https://github.com/khyox/ratsvm.git
   ```

3.2 Installing NCBI SRA Software for 64 bits CentOS, Ubuntu or MacOS

As of March 2018, NCBI SRA Toolkit (*see* **Note 3**) is available pre-compiled for some operating systems and architectures, like:

- CentOS Linux 64 bits https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-centos_linux64.tar.gz
- Ubuntu Linux 64 bits: https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-ubuntu64.tar.gz
- MacOS 64 bits: https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-mac64.tar.gz

The updated list is available in https://www.ncbi.nlm.nih.gov/sra/docs/toolkitsoft/. If your system is not there, please proceed to the next section. We will continue this section supposing that your system is a 64 bits CentOS Linux or equivalent since the steps are just the same for the mentioned architectures with the difference of the download link and file name:

1. Download and expand the pre-compiled package:
```
> wget https://ftp-trace.ncbi.nlm.nih.gov/
sra/sdk/current/sratoolkit.current-centos_
linux64.tar.gz
> tar xvzf sratoolkit.current-centos_
linux64.tar.gz
```
2. Rename the resulting inflated folder, whose name could depend on the specific release, for instance, alternatively:
```
> mv sratoolkit.current-centos_linux64 sra-tools
> mv sratoolkit.2.9.0-centos_linux64 sra-tools
```
3. Check the fastdump SRA Toolkit binary:
```
> ~/ratsvm/sra-tools/bin/fastq-dump –version
```

The resulting version number should be 2.8.2 or higher. If so, please skip the next section.

3.3 Installing NCBI SRA Software from Sources

1. Create a directory where NCBI software and data will be allocated and move there:
```
> mkdir ~/ratsvm/ncbi_src; cd ~/ratsvm/ncbi_src
```
2. Check that your system has as many of the following optional prerequisites as possible: libraries and development headers for XML2, Magic and HDF5 (*see* **Note 4**).
3. Clone the required NCBI SRA software git repositories from GitHub (*see* **Note 2**):
```
> git clone https://github.com/ncbi/ncbi-vdb.git
> git clone https://github.com/ncbi/ngs.git
> git clone https://github.com/ncbi/sra-tools.git
```
4. Configure the NCBI VDB software (*see* **Note 5**):
```
> cd ncbi-vdb;configure -prefix = ~/ratsvm/ncbi-vdb
```

5. Build the NCBI VDB software:
   ```
   > make
   ```
6. Install the NCBI VDB software in ~/ratsvm/ncbi-vdb:
   ```
   > make install
   ```
7. Analogously, configure, build, and install the NGS software (*see* **Note 5**):
   ```
   > cd ~/ratsvm/ncbi_src/ngs; ./configure
   -prefix = ~/ratsvm/ngs; make; make install
   ```
8. Configure, build, and install the SRA Toolkit software (*see* **Note 5**):
   ```
   > cd ~/ratsvm/ncbi_src/sra-tools; ./configure
   -prefix = ~/ratsvm/sra-tools; make; make install
   ```
9. Check the fastq-dump SRA Toolkit binary and confirm that the resulting version number is 2.8.2 or higher:
   ```
   > ~/ratsvm/sra-tools/bin/fastq-dump -version
   ```

3.4 Downloading Metagenomic Data of Virome Time Series

1. Clone the entrez (*see* **Note 6**) git repository in GitHub (*see* **Note 2**):
   ```
   > git clone https://github.com/jordibc/entrez
   ```
2. As an example, get the paired-ends set of FASTQ files from all the runs in the study "Rapid evolution of the human gut virome," a longitudinal characterization of intestinal viral particles from WGS (whole genome shotgun) sequencing. The public sequences of this study are already trimmed and quality-checked. The associated NCBI SRA code is SRP021107, so we run the following python scripts (*see* **Note 7**):
   ```
   > entrez/sra2runacc.py -s SRP021107 | rats-
   vm/runs2fastq.py -s sra-tools
   ```
3. Rename files with the script in the ratsvm repo in order to have meaningful information about the sampling (time and replica) in the filename:
   ```
   > ratsvm/rename_fastq.sh
   ```

3.5 Installing CMake3, the Fish Shell, Bowtie2, and Samtools

1. If CMake3 is not installed in your system (*see* **Note 8**) you can install it locally for your user by cloning the git repository:
   ```
   > git clone https://gitlab.kitware.com/
   cmake/cmake.git; cd cmake/
   ```
2. Configure the CMake3 build for a local installation under ~/ratsvm/CMake directory:
   ```
   > ./bootstrap -prefix =  ~/ratsvm/CMake
   ```
3. Build and install CMake3 in ~/ratsvm/CMake directory:
   ```
   > make -j 16; make install
   ```
4. Check the cmake binary and confirm that the installed version is 3.11 or higher, though any release over 3.2 would be acceptable for this protocol:
   ```
   > cd ..; CMake/bin/cmake --version
   ```

5. If the fish shell is not installed in your system (*see* **Note 9**) you can install it locally for your user by cloning the git repository in GitHub (*see* **Note 2**):
```
> git clone https://github.com/fish-shell/
fish-shell.git; cd fish-shell/
```
6. Use CMake3 to build and install fish locally under ~/ratsvm/fish directory:
```
> mkdir build
> ~/ratsvm/CMake/bin/cmake -DCMAKE_INSTALL_
PREFIX:PATH~/ratsvm/fish -DCMAKE_BUILD_
TYPE=Release..
> make install
```
7. Test the fish shell:
```
> cd ~/ratsvm/; fish/bin/fish
```
8. Exit from the fish shell session with "Ctrl+D" or exit.
9. Install Bowtie2 (*see* **Note 10**) locally for your user by cloning the git repository in GitHub (*see* **Note 2**):
```
> git clone https://github.com/BenLangmead/
bowtie2.git
> cd bowtie2; make -j 16; cd..
```
10. Install HTSlib (*see* **Note 11**) from the GitHub repository (*see* **Note 2**):
```
> git clone https://github.com/samtools/htslib.git
> cd htslib; autoheader; autoconf -Wno-syntax
> ./configure --prefix = ~/ratsvm/HTSlib
> make -j 16; make install; cd..
```
11. Install SAMtools (*see* **Note 11**) from the GitHub repository (*see* **Note 2**):
```
> git clone https://github.com/samtools/
samtools.git
> cd samtools; autoheader; autoconf -Wno-syntax
> ./configure --prefix = ~/ratsvm/SAMtools
> make -j 16; make install; cd..
```

3.6 Removing the Host (Human) Contamination

1. From NCBI RefSeq database, get human mRNA and ncRNA sequences to extend a human DNA database:
```
> mkdir HSS; cd HSS
> wget ftp://ftp.ncbi.nih.gov/refseq/H_
sapiens/H_sapiens/RNA/rna.fa.gz
```
2. Launch the make_hg19plusRNA.sh script to download the UCSC hg19 human genome reference and manage the building of a corresponding bowtie2 index (*see* **Note 12**):
```
> ~/ratsvm/ratsvm/make_hg19plusRNA.sh
```

3. Run the launch_bt2.fish script to manage the alignment of the reads against the extended hg19 human genome reference prepared in the previous step (*see* **Note 13**):

```
> cd ~/ratsvm; ratsvm/launch_bt2.fish --very-
sensitive --bt2_path ~/ratsvm/bowtie2 --db_
path ~/ratsvm/HSS --proc 32 hg19plusRNA
```

4. Execute the launch_samtools.fish script to obtain FASTQ files for the sequences that did not align against the extended hg19 human genome (*see* **Note 14**):

```
> ratsvm/launch_samtools.fish --samtls_path ~/
ratsvm/SAMtools --proc 32 hg19plusRNA
```

5. Check that we have the expected number of FASTQ files with the unmapped sequences (27 WGS runs, so 27 FASTQ files):

```
> ls -1 *_unmap.fastq | wc -l
```

6. Move the FASTQ files with the unmapped:

```
> mkdir unmap; mv *_unmap.fastq unmap/
```

3.7 Taxonomic Classification and Content Summarization

1. Install LMAT (Livermore Metagenomics Analysis Toolkit, *see* **Note 15**) starting by cloning the git repository in GitHub (*see* **Note 2**):

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

2. Use redoall wrapper around CMake3 to easily build LMAT in place using GNU compilers (*see* **Note 16**):

```
> ./redoall.sh R gnu
```

3. Download the LMAT runtime inputs:

```
> bin/get_db.sh --dtype=inputs
--name=04072014 –outdir=.
```

4. Set the environment variable LMAT_DIR to the runtime inputs directory downloaded:

```
> set -x LMAT_DIR ~/ratsvm/LMAT/runtime_inputs/
```

5. Download and move the LMAT Grand database (*see* **Note 17**):

```
> bin/get_db.sh --dtype=db --name=lmat-4-
14.20mer.db --outdir=.
> mv lmat-4-14.20mer.db /fastdisk; cd ~/ratsvm
```

6. Use pyLMAT_rl.py script to automate processing all the samples with the LMAT taxonomic classifier (*see* **Note 18**):

```
> ratsvm/pyLMAT_rl.py --path unmap --bdir ~/rats-
vm --fulldbdir /fastdisk --lmat ~/ratsvm/LMAT/
--threads 32 --minscore 0 --wgs
```

7. Run the pyLMAT_cs.py script to automate processing all the samples with the LMAT content summarization step (*see* **Note 19**):

```
> ratsvm/pyLMAT_cs.py
```

3.8 Time Series Analysis

1. The first and second steps in this section are for preparing the results of the LMAT taxonomic classifier and content summarization to the format needed by cmplxcruncher, the code that will perform the time series analysis. To begin with, use the rawlmat2lmat.py script, which will show a progress bar (*see* **Note 20**):
```
> mdir virome; ratsvm/rawlmat2lmat.py
```
2. Run lmat2cmplx.py script to convert the LMAT output (*see* **Note 20**):
```
> ratsvm/rawlmat2lmat.py
```
3. Check that the virome directory contains four Excel spreadsheets:
```
> ls virome/LMAT_*.xlsx
```
4. Run cmplxcruncher against the virome directory in automatic mode (*see* **Note 21**):
```
> ratsvm/cmplxcruncher.py --path virome/
--automatic pdf
```
5. The code generates detailed and exhaustive results for the time series analysis in the results subdirectory (*see* **Note 21**):
```
> cd virome/results; ls -R
```

Figure 2 shows an example of the output achieved in Taylor's Law plot, with a log-LR (Linear Regression) model for fitting the power law.

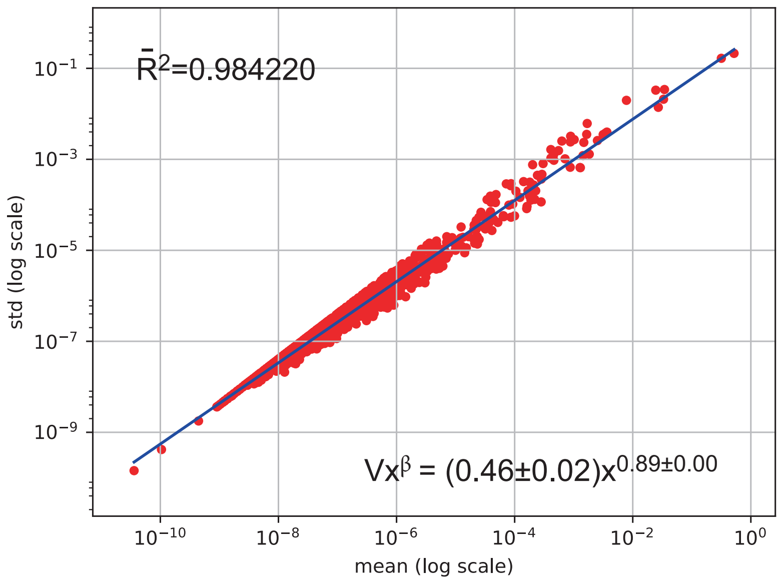

Fig. 2 Taylor's Law plot by fitting a log-L R (Linear Regression) model for the results obtained by LMAT "content summary" step at genus level along the time series

4 Notes

1. All computational methods presented here should work by inputting them directly in the command line. The prompt (initial >) should not be copied, it just marks where the command begins.
2. Git is a Distributed Version Control System (DVCS) that thinks of its data like a stream of snapshots of a mini filesystem, called the repository. Git was founded by Linus Tovals, the creator of the kernel of the GNU/Linux operating system, and it has quickly become the most used DVCS. A condensed course on Git titled "Git in a nutshell," from the author, is available here: http://som1.ific.uv.es/gitdoc. GitHub is the largest web-based hosting service for Git repositories. To interact with GitHub, it is recommended Git version 1.7.10 or higher. The latest version of Git is always available on the Git website: https://git-scm.com/downloads.
3. The Sequence Read Archive (SRA) of the US National Center for Biotechnology Information (NCBI) makes biological sequence data available to the world research community with the goal of enhancing reproducibility and allowing for new discoveries by comparing datasets. The NCBI SRA stores raw sequencing data and alignment information from high-throughput sequencing platforms. It can be manually accessed at https://www.ncbi.nlm.nih.gov/sra/ to search and retrieve sequences of interest. The SRA Toolkit is the recommended method by NCBI of retrieving multiple or large datasets from SRA database.
4. During the software building process, the NCBI SRA software will try to locate and use some third-party libraries and development headers from your system. They are not mandatory but it is quite advisable to have them installed in the system, especially the XML2 library. If you have administrative privileges in your computing box, you can install them using your preferred package-management utility, like APT or YUM for RPM-based distributions. For example, for installing XML2 and HDF5 prerequisites with YUM you can issue:
   ```
   > sudo yum install libxml2 libxml2-devel
   hdf5 hdf5-devel
   ```
5. The building order of the different NCBI SRA software packages is not arbitrary, as they have crossed dependencies.
6. Entrez is a Python interface to query the biological databases kept at the NCBI that uses the Entrez Programming Utilities (E-utilities), nine server-side programs that access the Entrez query and database system at NCBI.
7. The entrez sra2runacc.py script takes in his -s or --sra parameter a SRA database code or identifier (SRP021107 in our

example) and provides a list with the accession codes for all the runs related to the SRA identifier (27 runs in our case). The ratsvm runs2fastq.py script reads this list from the standard input (or from a file using the -f or --file parameter) and manages the retrieval of the FASTQ files using the NCBI SRA-tools and splitting them in different files for each paired-end. If the SRA-tools have been installed in a different location from the place indicated in this procedure, the -s or –sratoolkit parameter of runs2fastq.py should be updated accordingly. The download of any set of FASTQ files related to a NCBI bioproject, including this one, takes some time and requires some available disk space. In our example, the files use more than 310 GB in disk and could take more than 1 h in the download, depending on the network bandwidth. The runs2fastq.py progress indicator will help with this, but please note that it is showing complete downloads, so it will take some time to change from the starting 0% indication and this is totally normal.

8. CMake (https://cmake.org) is a cross-platform, open-source build system generator which eases deployment for software packages, especially for large or complex ones with several dependencies. If you have administrative privileges in your computing box, you can install CMake3 using your preferred package-management utility, like APT or YUM for RPM-based distributions. For example, for YUM you can issue:

```
> sudo yum install cmake3
```

9. The fish shell (https://fishshell.com/) is new-generation smart and user-friendly command line shell that has quite advantages over older shells not only for interactive use but also for programming, like a clean and easy to remember syntax. A good programming guide for the fish shell can be accessed here: http://som1.ific.uv.es/fishprog. In the fish shell website (https://fishshell.com/) there is a download section with links for different operating systems and GNU/Linux distributions. As this is a shell, the last installation step will require administrative privileges. An alternative option, covered in the protocol, is a local installation starting from the git repository and using CMake to build the software. Even with his choice, some basic system prerequisites are needed. If CMake reports that it could not find curses, try installing a curses development package and build again. Depending on if you have, on the one hand, Debian or Ubuntu or, on the other hand, RedHat, CentOS or Amazon EC2, you will issue, respectively:

```
> sudo apt-get install build-essential ncurs-
es-dev libncurses5-dev gettext autoconf
> sudo yum install ncurses-devel
```

10. Bowtie2 (http://bowtie-bio.sf.net/bowtie2) is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences developed at Johns Hopkins University, Baltimore, USA.

11. SAMtools is a set of tools for manipulating next-generation sequencing data, like SAM, BAM, and CRAM files. SAMtools is based on HTSlib, a C library for reading/writing high-throughput sequencing data, which should be installed first. The installation from git repositories described in the protocol is the preferred approach, but if in case of any trouble an alternate installation from packages could be used:

```
> wget https://github.com/samtools/sam-
tools/releases/download/1.7/samtools-
1.7.tar.bz2
> bunzip2 -c samtools-1.7.tar.bz2 | tar xvf -
> ./configure --prefix=~/ratsvm/SAMtools
> make -j 16; make install; cd ..
```

12. The make_hg19plusRNA.sh script downloads the human chromosomes 1 to 22, X, Y and the mtDNA. Those sequences join the mRNA and ncRNA sequences already downloaded in the previous step. Then, ~/ratsvm/bowtie2/bowtie2-build is called in order to build the bowtie2 reference database. This process is time-consuming and could take more than 3 h depending on the network speed and especially on the computing power.

13. The launch_bt2.fish script requires just one positional argument, the name of the bowtie2 alignment database. It also accepts various options whose default values have been chosen to be coherent with the previous steps in this protocol: -b/--bt2_path PATH for the Bowtie2 path, -d/--db_path PATH for the database path, -p/--proc PROC for selecting the number of processors used (of course, the time consumed will be highly dependent on this parameter) and -v/--very-sensitive to enable bowtie2 very sensitive mode (recommended though it will take some more time). The alignment of all the samples against the human reference is CPU-consuming so a computer with more computing cores available will mean a clear advantage. As a reference, using 32 cores, this step could take more than 1 h.

14. As the launch_bt2.fish script (*see* **Note 13**), the launch_samtools.fish script requires the name of the bowtie2 alignment database as positional argument. It also accepts various options with default values chosen to be coherent with the previous steps in this protocol: -s/--samtls_path PATH for the SAMtools package installation path, -p/--proc PROC for selecting the number of processors used in sorting the sequences. Internally, the script will pipe the data through different tools in the SAMtools package, starting with the SAM

files generated by bowtie2 in the previous step until obtaining the FASTQ files for the unmapped sequences. Again, this process takes some time, especially the intermediate stage of sorting the sequences, but not as much as the previous step.

15. LMAT (Livermore Metagenomics Analysis Toolkit) is an all-in-1 metagenomic analysis toolkit including taxonomic classification, content summarization, and gene identification, by Lawrence Livermore National Laboratory. LLNL's history of always having one of the most power supercomputing systems in the world has left its mark in LMAT as it is a very powerful software but can be very demanding in computing resources. In fact, for using the LMAT Grand database, at least a "fat node" will be needed with at least half TB of main memory and a SSD for storing the database.. Fortunately, other LMAT databases have more modest requirements. LMAT uses PERM, a 'C' library for persistent heap management developed at LLNL too. The PERM memory allocator replaces the standard 'C' dynamic memory allocation functions with compatible versions that provide persistent memory to application programs. For PERM (so LMAT) to work in the right conditions, some kernel tuning is advisable (administrative privileges required), which consists in turning off periodic flush to file and dirty ratio flush and switching off address space randomization:

```
> sudo sh -c 'echo 0 > /proc/sys/vm/dirty_
writeback_centisecs'
> sudo sh -c 'echo 100 > /proc/sys/vm/
dirty_background_ratio'
> sudo sh -c 'echo 100 > /proc/sys/vm/
dirty_ratio'
> sudo sh -c 'echo 0 > /proc/sys/kernel/
randomize_va_space'
```

16. LMAT can be built with different compilers and with different building profiles. The redoall script accepts two different optional parameters. The first one chooses the build profile of CMake: "D" for debug, "R" for release (this is the default), "I" for release with debugging info, and "M" for minimum executable size release. The second parameter selects the compiler family to use: "gnu" for using GNU gcc compiler, "intel" for Intel compilers, "clang" for clang/LLVM compiler with OpenMP support, and "ibmpwr9" for compiling in an IBM supercomputer with Power9 processors using the IBM compilers optimized for this architecture.

17. The LMAT Grand database (lmat-4-14.20mer.db) requires 462 GB of disk space, so it takes some time to be downloaded and assembled by the get_db script. It is not mandatory but quite advisable to move this huge database to another filesystem, especially not the same that will suffer the I/O stress

related to the data being processed by LMAT. Ideally, this filesystem will be allocated in a storage device with a very broad reading bandwidth and very low latency, like a SSD (solid-state drive). In this protocol, we suppose such a filesystem is mounted under /fastdisk.

18. The LMAT taxonomic classifier core (read_label) is a C++ program that requires several parameters. LMAT provides with a convenient launcher (run_rl.sh) that not only simplifies the call to read_label, but also launches other tools (tolineage and fsreport) to get some additional results. The pyLMAT_rl.py script in the ratsvm repository eases even more the process by automating the call to run_rl.sh for several related samples, which is our case.
19. The LMAT content summarization core (content_summ) is a C++ program requiring several parameters, so LMAT provides with a convenient launcher (run_cs.sh), which simplifies the call to content_summ and does some additional work. The pyLMAT_cs.py script in the ratsvm repository goes one step beyond to facilitate even more the process by automating the call to run_cs.sh for several related samples, like the ours.
20. The scripts rawlmat2lmat.py and lmat2cmplx.py perform a two-step translation of the LMAT results to the input format of cmplxcruncher. The goal of the former is to centralize the LMAT output in a single directory, in this case, the virome one. The mission of the latter is parsing all the files in the directory to extract the right taxonomic information for every time sample and write it in a spreadsheet in the format expected by cmplxcruncher. Both scripts have different parameters, whose default values are the convenient ones for this protocol, but can be changed if the protocol is to be used for different data.
21. complexCruncher computing kernel is a Python tool to analyze the dynamics of ranking processes in metagenomics. Cmplxcruncher requires Python 3.5 or later is needed and following python packages: pandas >1.16.0 with openpyxl, numpy, scipy, matplotlib >1.5.0 and xlrd >0.9. The tool will let you play with the data and repeat the analysis. You can see the graphs interactively or save them to disk in different formats. Alternatively, you can run the code in automatic (non-interactive mode) with the flag --automatic, like in the protocol: it tries to process every text or Excel file it could find in its data file tree. The results will be saved in a subdirectory of the data one, typically "results," which some subdirectories with self-descriptive names: fits, hist, and corrank, for correlation and rank plots. You can choose different names for the data and results directories, and select different levels and depth of analysis. Finally, be aware that complexcruncher is still under development.

References

1. Miller RR, Montoya V, Gardy JL et al (2013) Metagenomics for pathogen detection in public health. Genome Med 5(9):81
2. Ercolini D (2013) High-throughput sequencing and metagenomics: moving forward in the culture-independent analysis of food microbial ecology. Appl Environ Microbiol 79(10):3148–3155
3. Martí JM, Martínez-Martínez D, Rubio T et al (2017) Health and disease imprinted in the time variability of the human microbiome. mSystems 2(2):e00144-16
4. Wu H, Tremaroli V, Bäckhed F (2015) Linking microbiota to human diseases: a systems biology perspective. Trends Endocrinol Metab 26(12):758–770
5. Noecker C, Eng A, Srinivasan S et al (2016) Metabolic model-based integration of microbiome taxonomic and metabolomic profiles elucidates mechanistic links between ecological and metabolic variation. MSystems 1(1):13
6. Greenblum S, Turnbaugh PJ, Borenstein E (2012) Metagenomic systems biology of the human gut microbiome reveals topological shifts associated with obesity and inflammatory bowel disease. Proc Natl Acad Sci 109(2):594–599
7. Bashan A, Gibson TE, Friedman J et al (2016) Universality of human microbial dynamics. Nature 534(7606):259
8. Smith HF (1938) An empirical law describing heterogeneity in the yields of agricultural crops. J Agric Sci 28(1):1–23
9. Taylor LR (1961) Aggregation, variance and the mean. Nature 189(4766):732–735
10. Mantegna RN, Stanley HE (1995) Scaling behaviour in the dynamics of an economic index. Nature 376(6535):46
11. Eisler Z, Kertesz J, Yook S et al (2005) Multiscaling and non-universality in fluctuations of driven complex systems. Europhys Lett 69(4):664
12. De Menezes MA, Barabási A (2004) Fluctuations in network dynamics. Phys Rev Lett 92(2):028701
13. Reed DH, Hobbs GR (2004) The relationship between population size and temporal variability in population size. In: Anonymous animal conservation forum, vol 7. Cambridge University Press, p 1
14. Anderson RM, Gordon DM, Crawley MJ et al (1982) Variability in the abundance of animal and plant species. Nature 296(5854):245–248
15. Cohen JE, Xu M, Schuster WS (2013) Stochastic multiplicative population growth predicts and interprets Taylor's power law of fluctuation scaling. Proc R Soc Lond B Biol Sci 280(1757):20122955
16. Živković J, Tadić B, Wick N et al (2006) Statistical indicators of collective behavior and functional clusters in gene networks of yeast. Eur Phys J 50(1–2):255–258
17. Kendal WS (2003) An exponential dispersion model for the distribution of human single nucleotide polymorphisms. Mol Biol Evol 20(4):579–590
18. Zhang Z, Geng J, Tang X et al (2014) Spatial heterogeneity and co-occurrence patterns of human mucosal-associated intestinal microbiota. ISME J 8(4):881
19. Kaltz O, Escobar-Páramo P, Hochberg ME et al (2012) Bacterial microcosms obey Taylor's law: effects of abiotic and biotic stress and genetics on mean and variance of population density. Ecol Process 1(1):5
20. Ramsayer J, Fellous S, Cohen JE et al (2012) Taylor's law holds in experimental bacterial populations but competition does not influence the slope. Biol Lett 8(2):316–319
21. Pérez-Cobas AE, Artacho A, Ott SJ et al (2014) Structural and functional changes in the gut microbiota associated to Clostridium difficile infection. Front Microbiol 5:335
22. Ding T, Schloss PD (2014) Dynamics and associations of microbial community types across the human body. Nature 509(7500):357
23. Gajer P, Brotman RM, Bai G et al (2012) Temporal dynamics of the human vaginal microbiota. Sci Transl Med 4(132):132ra52
24. Nayfach S, Pollard KS (2016) Toward accurate and quantitative comparative metagenomics. Cell 166(5):1103–1116. https://doi.org/10.1016/j.cell.2016.08.007
25. Lusk RW (2014) Diverse and widespread contamination evident in the unmapped depths of high throughput sequencing data. PLoS One 9(10):e110808
26. Weiss S, Amir A, Hyde ER et al (2014) Tracking down the sources of experimental contamination in microbiome studies. Genome Biol 15(12):564. https://doi.org/10.1186/s13059-014-0564-2
27. Gruber K (2015) Here, there, and everywhere. EMBO reports:e201540822
28. Kircher M, Sawyer S, Meyer M (2012) Double indexing overcomes inaccuracies in multiplex sequencing on the Illumina platform. Nucleic Acids Res 40(1):e3
29. Thoendel M, Jeraldo P, Greenwood-Quaintance KE et al (2017) Impact of

contaminating DNA in whole-genome amplification kits used for metagenomic shotgun sequencing for infection diagnosis. J Clin Microbiol 55(6):1789–1801

30. Ames SK, Hysom DA, Gardner SN et al (2013) Scalable metagenomic taxonomy classification using a reference genome database. Bioinformatics 29(18):2253–2260
31. Ames SK, Gardner SN, Martí JM et al (2015) Using populations of human and microbial genomes for organism detection in metagenomes. Genome Res 25(7):1056–1067
32. Langmead B, Trapnell C, Pop M et al (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 10(3):R25
33. Anonymous (2016) Database resources of the National Center for biotechnology information. Nucleic Acids Res 44(D1):D19
34. Burguet-Castell J, Martí JM (2018) Entrez: A simple Python interface to the NCBI databases (https://github.com/jordibc/entrez)
35. Li H, Handsaker B, Wysoker A et al (2009) The sequence alignment/map format and SAMtools. Bioinformatics 25(16):2078–2079
36. Torvalds L, Hamano J (2010) Git: Fast version control system.
37. Minot S, Bryson A, Chehoud C et al (2013) Rapid evolution of the human gut virome. Proc Natl Acad Sci U S A 110(30):12450–12455