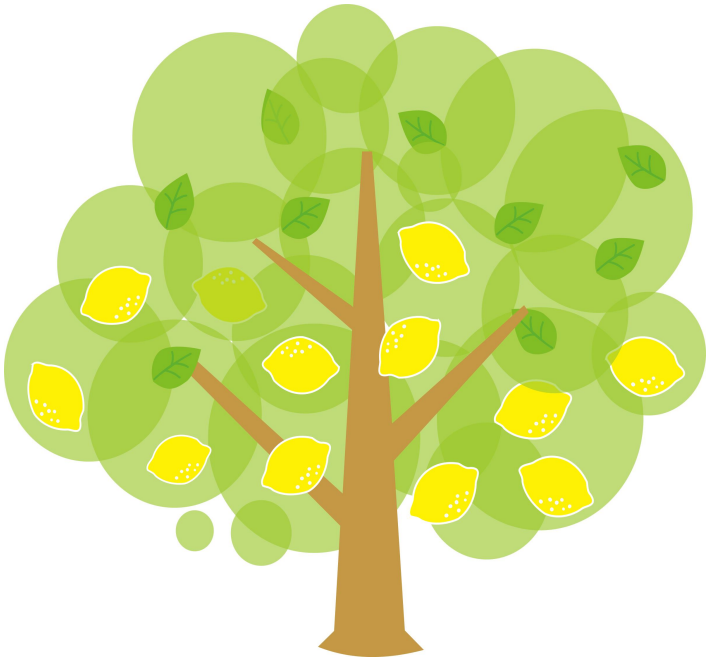
SqueezeM, a fully automatic metagenomic analysis pipeline, all the way from raw reads to bins

Javier Tamames, Fernando Puente
June 2018



1. What is squeezeM?

SqueezeM is a full automatic pipeline for metagenomics/metatranscriptomics, covering all steps of the analysis, In contrast to other available tools, SqueezeM includes multi-metagenome support allowing the co-assembly of related metagenomes and the retrieval of individual genomes via binning procedures. Thus, Squeeze M features several unique characteristics:
1) Coassembly procedure with read mapping for estimation of the abundances of genes in each metagenome
2) Coassembly of unlimited number of metagenomes via merging of individual metagenomes
3) Includes binning and bin checking, for retrieving individual genomes
4) the results are stored in a database, where they can be easily exported and shared, and can be inspected anywhere using a web interface.
5) Internal checks for the assembly and binning steps inform about the consistency of contigs and bins, allowing to spot potential chimeras.
6) Metatranscriptomic support via mapping of cDNA reads against reference metagenomes

SqueezeM can be run in three different modes, depending of the type of multi-metagenome support. These modes are:
-Sequential mode: All samples are treated individually and analysed sequentially. This mode does not include binning.
-Coassembly mode: Reads from all samples are pooled and a single assembly is performed. Then reads from individual samples are mapped to the coassembly to obtain gene abundances in each sample. Binning methods allow to obtain genome bins.
-Combined mode: if many samples are available, co-assembly will probably crash because of memory requirements. This mode allows the co-assembly of an unlimited number of samples, using a procedure similar to the one used by (ref protocol). Briefly, samples are assembled individually and the resulting contigs are merged in a single co-assembly (see below). Then the analysis proceeds as in the coassembly mode.

SqueezeM uses a combination of custom scripts and external software packages for the different steps of the analysis:

1 Assembly
2 RNA prediction
3 ORF (CDS) prediction
4 Homology searching against taxonomic and functional databases
5 Hmmer searching against Pfam database
6 Taxonomic assignment of genes
7 Functional assignment of genes
8 Taxonomic assignment of contigs, and chimera checking
9 Coverage and abundance estimation for genes and contigs
10 Estimation of taxa abundances

11 Estimation of function abundances
12 Merging of previous results to obtain the ORF table
13 Binning with Maxbin
14 Binning with metabat2
15 Taxonomic assignment of bins, and chimera checking
16 Checking of bins
17 Merging of previous results to obtain the bin table
18 Merging of previous results to obtain the contig table

2. Installation

For installing squeezeM, download the latest release from the GitHub repository and uncompress the tarball in a suitable directory. The tarball includes the squeezeM scripts as well as the third-party software redistributed with squeezeM (see section 6). The INSTALL file contains detailed installation instructions, including all the external libraries required to make squeezeM run in a vanilla Ubuntu 14.04 installation.

3. License

SqueezeM is distributed with a GPL-3 license.
Additionally, squeezeM redistributes the following third-party software:

* [Megahit] (https://github.com/voutcn/megahit)
* [Spades] (http://cab.spbu.ru/software/spades/)
* [prinseq] (http://prinseq.sourceforge.net/)
* [prodigal] (https://github.com/hyattpd/Prodigal)
* [cd-hit] (https://github.com/weizhongli/cdhit)
* [Amos] (http://www.cs.jhu.edu/~genomics/AMOS/)
* [hmmer] (http://hmmer.org/)
* [Diamond] (https://github.com/bbuchfink/diamond)
* [Bedtools] (https://github.com/arq5x/bedtools2)
* [bowtie2] (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml)
* [barrnap] (https://github.com/tseemann/barrnap)
* [maxbin] (https://downloads.jbei.org/data/microbial_communities/MaxBin/MaxBin.html)
* [metabat] (https://bitbucket.org/berkeleylab/metabat)
* [Checkm] (http://ecogenomics.github.io/CheckM/)

4. Execution, restart and running scripts

Execution

The command for running squeezeM has the following syntax:

squeezeM.pl -m <mode> -p <projectname> -s <equivfile> -f <raw fastq dir> <options>

Arguments:
-m: Mode (sequential, coassembly, merged) (REQUIRED)

-p: Project name (REQUIRED in coassembly and merged modes)
-s|-samples: Samples file (REQUIRED)
-f|-seq: Fastq read files' directory (REQUIRED)
-t: Number of threads (Default:12)
-a: assembler [megahit,spades] (Default:megahit)
-c|-contiglen: Minimum length of contigs (Default:1200)
--nocog: Skip COG assignment (Default: no)
--nokegg: Skip KEGG assignment (Default: no)
--nopfam: Skip Pfam assignment (Default: no)
--nobins: Skip binning (Default: no)
-e|-evalue: max evalue for diamond run (Default: 1e-03)
-miniden: minimum identity perc for diamond run (Default: 50)
--megahit_options: Options for megahit assembler
--spades_options: Options for spades assembler

Example: squeezeM.pl -m coassembly -p test -s test.samples -f mydir --nopfam -miniden 60

This will create a project "test" for co-assembling the samples specified in the file "test.samples", using a minimum identity of 60% for taxonomic and functional assignment, and skipping Pfam annotation.

-m: SqueezeM can run in three different modes: sequential, co-assembly and merged. The sequential mode will process each metagenome independently. Co-assembly mode will join all reads from all samples and co-assemble them. Merged mode will first assemble each metagenome independently, and then merge the contigs to produce a co-assembly. Binning is only available for co-assembly and merged modes

-p: The name under all results and data file will be saved. This is not required for sequential mode, where the name will be taken from the samples file.

-s: A file specifying the samples, the corresponding reads and their location. It has the format:

<Sample> <filename> <pair1|pair2>

For example:

Sample1	readfile1	pair1	
Sample1	readfile2	pair2	
Sample1	readfile3	pair1	
Sample1	readfile4	pair2	
Sample2	readfile5	pair1	
Sample2	readfile6	pair2	
Sample3	readfile7	pair1	noassembly
Sample3	readfile8	pair2	noassembly

The first column indicates the sample id (this will be the project name in sequential mode), the second contains the file names of the sequences, and

the third specifies the pair number of the reads. A fourth optional column can take the "noassembly" value, indicating that these sample must not be assembled with the rest (but will be mapped against the assembly to get abundances). This is the case for RNAseq reads that can hamper the assembly but we want them mapped to get transcript abundance of the genes in the assembly. Notice also that paired reads are expected, and that a sample can have more than one set of reads.

-f: Directory where the read files specified in the sample file are stored.

Restart

Any interrupted squeezeM run can be restarted using the program restart.pl. It has the syntax:

restart.pl <projectname>

This command must be issued in the upper directory to the project <projectname>, and will restart the run of that project by reading the progress.txt file to find out the pint where the run stopped.

Running scripts

Also, any individual script of the pipeline can be run in the upper directory to the project using the same syntax:

script <projectname> (for instance, rundiamond.pl <projectname> to repeat the Diamond run for the project)

5. Building databases

SqueezeM uses several databases. GenBank nr for taxonomic assignment, and eggnog, KEGG and Pfam for functional assignment. The script make_databases.pl must be run to get and format all these databases.

6. Results

A run of SqueezeM generates many files containing all the information relative to genes, contigs and bins. That information is summarized into three main tables:
1) The ORFs table, with all the information regarding ORFs (taxonomy, function, contig and bin origin, abundance in samples, aminoacid sequence).
2) The contig table, gathering all data from contigs (taxonomy, bin origin, abundance in samples, chimerism)
3) The bin table with everything related to the bins (taxonomy, completeness, contamination, abundance in samples, chimerism).

The ORF (gene) table (12.<projectname>.orftable)

ORF: ORF (gene) identifier
Contig ID: ID of the contig to which the ORF belongs
GC perc: GC percentage of the ORF
GenName: Gen name (if any)
Taxonomy ORF: Taxonomic assignment for the ORF, produced by the LCA algorithm (see 1.1)
KEGG ID: KEGG identifier for the ORF (if any)
KEGG Fun: KEGG functional annotation
KEGG Path: Pathway(s) to which the given KEGG ID belongs
COG ID: COG identifier for the ORF (if any)
COG Fun: COG functional annotation
COG Path: Pathway(s) to which the given COG ID belongs
Pfam: Pfam annotations for the ORF
Counts <samples>: RPKM abundances of the ORF in each of the samples
Raw Counts <samples>: Raw reads mapping to the ORF in each of the samples
AASEQ: Amino Acid sequence for the gene

The contig table (18.<projectname>.contigtable)

Contig ID: ID of the contig
Taxonomy: Taxonomic assignment for the contig, produced by the contig_consensus algorithm (see 1.2)
Chimerism: Maximum chimerism index (see 1.3)
GC perc: GC percentage of the contig
Length: Contig Length
Num Genes: Number of genes in the contig
Bin ID: Bin ID to which the contig belongs to (if any)
Coverage <samples>: Average coverage for the contig in each of the samples
RPKM <samples>: RPKM abundances of the contig in each of the samples
Raw <samples>: Raw reads mapping to the contig in each of the samples

The bin table (17.<projectname>.bintable)

Bin ID: ID of the bin
Method: Binning method
Taxonomy: Taxonomic assignment for the bin, produced by the contig_consensus algorithm
Size. Size of the bin (accumulated length of its sequences)
GC perc: Average GC percentage for the bin
Num contigs: Number of contigs in the bin
Chimerism: Maximum chimerism index (see 1.3)
Completeness: Percentage of completeness of the bin, provided by checkM (percentage of marker genes found)
Contamination: Percentage of contamination of the bin, provided by checkM (percentage of marker genes found in multiple copies)

Strain Het: Strain heteroigenity, provided by checkM (percentage of marker genes found in multiple copies that are highly similar)

Coverage <samples>: Average coverage (number of times each base was found) for the contigs in the bin, for each sample.
RPKM <samples>: RPKM value (Reads per Kilobase and Million reads) for the contigs in the bin, for each sample.

Other files

SqueezeM produces many other files:

01.<projectname>.fasta: Fasta sequences of the contigs in the assembly or co-assembly
01.<projectname>.lon: Length of the contigs
01.<projectname>.stats: Statistics for the assembly (size, N50, etc)
02.<projectname>.rnas.fasta: Predicted RNAs in the contigs, fasta format
02.<projectname>.maskedrna.fasta: Sequences of the contigs with RNAs masked for gene prediction, fasta format.
03.<projectname>.faa: Aminoacid sequences for predicted genes, fasta format
03.<projectname>.fna: Nucleotidic sequences for predicted genes, fasta format
03.<projectname>.gff: GFF file containing the position of each gene in the contigs.
04.<projectname>.eggnog.diamond: Result of the Diamond run against eggnog database, diamond format
04.<projectname>.kegg.diamond: Result of the Diamond run against KEGG database, diamond format
04.<projectname>.nr.diamond: Result of the Diamond run against nr database, diamond format
05.<projectname>.pfam.hmm: Result of the hmmer run against Pfam database, hmm format
06.<projectname>.fun3.tax: Taxonomic annotation for the genes, LCA algorithm (see below)
06.<projectname>.fun3.tax.wranks: Taxonomic annotation for the genes, with rank names.
07.<projectname>.fun3.cog: Annotation of eggnog functions for the genes
07.<projectname>.fun3.kegg: Annotation of KEGG functions for the genes
07.<projectname>.fun3.pfam: Annotation of Pfam functions for the genes
08.<projectname>.contiglog: Consensus taxonomic annotation for the contigs
09.<projectname>.coverage: RPKM and coverage values for all genes in all samples.
09.<projectname>.contigcov: RPKM and coverage values for contigs.
10.<projectname>.mcount: Counts for taxa abundance
11.<projectname>.cog.funcover: Counts for eggnog functions abundance
11.<projectname>.kegg.funcover: Counts for eggnog functions abundance
12.<projectname>.orftable: The ORF table
15.<projectname>.bintax: Consensus taxonomic annotation for the bins
16.<projectname>.<binmethod>.checkm: CheckM results for the bins created using binmethod (maxbin, metabat2)Q
17.<projectname>.bincov: RPKM and coverage values for bins
17.<projectname>.bintable: The bin table
18.<projectname>.contigsinbins: Contings in each bin, with taxonomic annotation

18.<projectname>.contigstable: The contig table
19.<projectname>.stats: Full statistics for the project (genes, contigs, bins)

The number in the filename corresponds to the step of the pipeline creating these files.

7. Restarting and re-running parts of the analysis

It is possible to restart an interrupted run by means of the restart.pl script. Move to the parent directory (the one in which squeezeM has been run, that is, the parent directory of <projectname> directory) and run:

perl restart.pl <projectname>

SqueezeM will read the progress file and continue the analysis from the last checkpoint.

It is also possible to re-run a single part of the analysis. All result files are identified with a number that correspond to the number of the script creating it. To re-generate them, move to the parent directory and run:

perl <script> <projectname>

The current list of scripts is the following:

squeezeM.pl: main script
01.run_assembly.pl: Runs assemblies in sequential and coassembly modes
01.run_assembly_merged.pl: Runs assemblies in merged mode
01.merge_assemblies.pl: Merges individual assemblies in merged mode
02.run_barrnap.pl: Runs RNAs prediction
03.run_prodigal.pl: Runs ORF prediction
04.rundiamond.pl: Runs Diamond for homology searching
05.run_hmmer.pl: Runs Hmmer for Pfam assignment
06.lca.pl: Runs LCA algorithm for taxonomic assignment
07.fun3assign.pl: Eggnog, KEGG and Pfam annotation
08.summarycontigs3.pl: Produces taxonomic annotations for contigs
09.mapbamsamples.pl: Maps reads to contigs and counts ORF abundances
10.mcount.pl: Counts abundances for taxa
11.funcover.pl: Counts abundances for functions
12.mergeannot2.pl: Produces the ORF table
13.bin_maxbin.pl: Runs Maxbin binning
14.bin_metabat2.pl: Runs metabat2 binning
15.addtax2.pl: Produces taxonomic annotations for bins
16.checkM_batch.pl: Runs CheckM for bin evaluation
17.getbins.pl: Makes the bin table
18.getcontigs.pl: Makes the contig table
19.stats.pl: Generates final statistics for the full project

8. Algorithms

The LCA algorithm

We use a Last Common Ancestor (LCA) algorithm to assign taxa to genes. For the aminoacid sequence of each gene, diamond (blastp) homology searches are done against the GenBank nr database (updated weekly). A e-value cutoff of 1e-03 is set by default. The best hit is obtained, and then we select a range of hits (valid hits) having at least 80% of the bitscore of the best hit and differing in less than 10% identity also with the best hit (these values can be set). The LCA of all these hits is obtained, that is, the taxon common to all hits. This LCA can be found at diverse taxonomic ranks (from phylum to species). We allow some flexibility in the definition of LCA: a small number of hits belonging to other taxa than the LCA can be allowed. In this way we deal with putative transfer events, or incorrect annotations in the database. This value is by default 10% of the total number of valid hits, but can be set by the user. Also, the minimum number of hits to the LCA taxa can be set.

An example is shown in the next table

GenID	Hit ID	Hit tax	Identity	e-value
Gen1	Hit1	Genus:Polaribacter Family: Flavobacteriaceae Order:Flavobacteriales	75.2	1e-94
Gen1	Hit2	Genus:Polaribacter Family: Flavobacteriaceae Order:Flavobacteriales	71.3	6e-88
Gen1	Hit3	Family: Flavobacteriaceae Order:Flavobacteriales	70.4	2e-87
Gen1	Hit4	Genus:Algibacter Family: Flavobacteriaceae Order:Flavobacteriales	68.0	2e-83
Gen1	Hit5	Genus:Rhodospirillum Family: Rhodospirillaceae Order:Rhodospirillales	60.2	6e-68

In this case, the four first hits are the valid ones. Hit 5 does not make the identity and e-value thresholds. The LCA for the four valid hits is Family: Flavobacteriaceae, that would be the reported result.

Our LCA algorithm includes strict cut-off identity values for different taxonomic ranks, according to Luo et al, Nucleic Acids Research 2014, 42, e73. This means that hits must pass a minimum (aminoacid) identity level in order to be used for assigning particular taxonomic ranks. These thresholds are 85, 60, 55, 50, 46, 42 and 40% for species, genus, family, order, class, phylum and superkingdom ranks, respectively. Hits below these levels cannot be used to make assignments for the corresponding rank. For instance, a protein will not be assigned to species level if it has no hits above 85% identity. Also, a protein will remain unclassified if it has no hits above 40% identity. The inclusion of these thresholds guarantees that no assignments are done based on weak, inconclusive hits.

Contig (or bin) consensus algorithm

The consensus algorithm attempts to obtain a consensus taxonomic annotation for the contigs according to the annotations of each of its genes. The consensus taxon if the one fulfilling:
-50% of the genes of the contig belong to (are annotated to) this taxon, and
-70% of the annotated genes belong to (are annotated to) this taxon.
Notice that the first criterion refers to all genes in the contig, regardless if they have been annotated or nor, while the second refers exclusively to annotated genes.
As the assignment can be done at different taxonomic ranks, the consensus is the deepest taxon fulfilling the criteria above.

For instance, consider the following example for a contig with 6 genes:

Gen1: k_Bacteria;p_Proteobacteria;c_Gamma-Proteobacteria;o_Enterobacteriales;f_Enterobacteriaceae;g_Escherichia;s_Escherichia coli
Gen2: k_Bacteria;p_Proteobacteria;c_Gamma-Proteobacteria;o_Enterobacteriales;f_Enterobacteriaceae;g_Escherichia
Gen3: k_Bacteria;p_Proteobacteria;c_Gamma-Proteobacteria;o_Enterobacteriales;f_Enterobacteriaceae;g_Escherichia
Gen4: k_Bacteria;p_Proteobacteria;c_Gamma-Proteobacteria;o_Enterobacteriales;f_Enterobacteriaceae
Gen5: No hits
Gen6: k_Bacteria;p_Firmicutes

In this case, the contig will be assigned to k_Bacteria;p_Proteobacteria;c_Gamma-Proteobacteria;o_ Enterobacteriales;f_ Enterobacteriaceae, which is the deepest taxon fulfilling 50% of all the genes belonging to that taxon (4/6=66%), and having 70% of the annotated genes (4/5=80%). The assignment to genus Escherichia was not done since just 3/5=60% of the annotated genes belong to it.

For annotating the consensus of bins, the procedure is the same, just using the annotations for the contigs belonging to the bin instead.

Chimerism calculation

Notice that in the example above, the end part of the contig seems to depart from the common taxonomic origin of the rest. This is an indication of a potential chimerism produced in the assembly (although it can be die to other causes, such as a recent LCA transfer or a missannotation for the gene). The chimerism index attempts to measure this effect, so that the contigs can be flagged accordingly (for instance, we could decide not trusting highly chimeric contigs). Chimerism index is calculated for the taxonomic rank assigned by consensus (in the previous example, family). We compare the assignments at that level for every pair of genes in the contig, and count the number of agreements and disagreements. If one of the taxa has no annotation at that level, is not counted for agreement but it is counted for disagreements if previous ranks do not coincide. That is:

Gen1-Gen2: Agree
Gen1-Gen3: Agree
Gen1-Gen4: Agree
Gen1-Gen5: Unknown
Gen1-Gen6: Disagree (at phylum level)
Gen2-Gen3: Agree
Gen2-Gen4: Agree
Gen2-Gen5: Unknown
Gen2-Gen6: Disagree (at phylum level)
Gen3-Gen4: Agree
Gen3-Gen5: Unknown
Gen3-Gen6: Disagree (at phylum level)
Gen4-Gen5: Unknown
Gen4-Gen6: Disagree (at phylum level)
Gen5-Gen6: Unknown

Chimerism index is the ratio between the number of disagreements and the total number of comparisons, in this case 4/15=0.26

For calculating the chimerism of bins, the procedure is the same, just using the annotations for the contigs belonging to the bin instead.

9. Contact

For any suggestions, comments, bugs, please contact us at:

Javier Tamames: jtamames@cnb.csic.es

Fernando Puente: fpuente@cnb.csic.es