CompareM User's Guide
A Toolbox for Comparative Genomics

Donovan Parks

July 13, 2016

Introduction

CompareM is a software toolkit for performing large-scale comparative genomic analyses. It provides pairwise genome statistics (e.g., amino acid identity) and statistics for individual genomes (e.g., codon usage). Parallelized implementations are provided for computationally intensive tasks in order to allow scalability to thousands of genomes. Common workflows are provided as single methods to support easy adoption by users, and a more granular interface provided to allow experienced users to exploit specific functionality. CompareM is open source and released under the GNU General Public License (Version 3).

Contact Information

CompareM is in active development and we are interested in discussing all potential applications of this software. Suggestions, comments, and bug reports can be sent to Donovan Parks (donovan.parks [at] gmail.com). If reporting a bug, please provide as much information as possible and a simplified version of the dataset which causes the bug. This will allow us to quickly resolve the issue.

Citing CompareM and Associated Software

If you use CompareM in your research, please cite the GitHub repository:

> https://github.com/dparks1134/CompareM

CompareM uses DIAMOND to perform sequence similarity searches:

> Buchfink B. *et al.* (2015) Fast and sensitive protein alignment using DIAMOND. *Nat Methods*, **12**, 59-60.

Gene calling is performed using Prodigal:

> Hyatt D. *et al.* (2010) Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC Bioinformatics*, **11**, 119.

Installation

CompareM makes use of the numpy, scipy, matplotlib, and biolib python packages, and assumes the following 3rd party dependencies are on your system path:

- DIAMOND ≥ 0.8.14 (https://github.com/bbuchfink/diamond)
- Prodigal ≥ 2.6.3 (https://github.com/hyattpd/Prodigal)

Most systems already contain the “SciPy Stack” of numpy, scipy, and matplotlib. However, if you need to install these on your system, instructions can be found at:

http://www.scipy.org/install.html

Once these are installed, CompareM can be installed using pip:

```
> sudo pip install comparem
```

The CompareM source code, including official code releases, is available through GitHub if you wish to run CompareM directly from source:

- https://github.com/dparks1134/CompareM

Quick Start

The functionality provided by CompareM is available from the help menu:

```
> comparem -h
```

Usage information about specific functions can also be accessed through the help menu, e.g.:

```
> comparem aa_usage –h
```

Amino Acid Identity Workflow

The most common task performed with CompareM is the calculation of pairwise amino acid identity (AAI) values between a set of genomes. This can be performed using the *aai_wf* command:

```
> comparem aai_wf <input_files> <output_dir>
```

Positional Arguments. The <input_file> argument indicates the set of genomes to compare and can either be i) a text file where each line indicates the location of a genome, or ii) a directory containing all genomes to be compared. Each genome should be represented as a single FASTA file containing genomic nucleotide sequences. The <output_dir> indicates the desired directory for all output files. A typical use of this command would be:

```
> comparem --cpus 32 aai_wf my_genomes aai_output
```

where the directory *my_genomes* contains a set of genomes in FASTA format, the results are to be written to a directory called *aai_output*, and 32 processors should be used to calculate the results.

Optional Arguments. A number of optional arguments can also be specified. This includes the sequence similarity parameters used to define reciprocal best hits between genomes (i.e., othologs). By default, the e-value (*--evalue*), percent sequence identity (*--per_identity*), and percent alignment length (*--per_aln_len*) parameters are set to 1e-5, 30%, and 70%. When specifying a directory of genomes to process, CompareM only processes files with a *fna* extension. This can be changes with the *--file_ext* argument. In addition, if the input FASTA files already contain identified proteins (as opposed to genomic nucleotide sequences), this must be specified with the *--proteins* flag. Otherwise, genes will be identified *de novo* using the Prodigal gene caller. The time to compute all pairwise AAI values can be substantially reduced by using multiple processors as specified with the *--cpus* argument. Other arguments are for specialized uses and are discussed below.

Outputs. Pairwise AAI statistics are provided in the file *./<output_dir>/aai/aai_summary.tsv*. This file consists of 8 columns indicating:

1. Identifier of the first genome
2. Number of genes in the first genome
3. Identifier of the second genome
4. Number of genes in the second genome
5. Number of orthologs identified between the two genomes
6. The mean amino acid identity (AAI) across all orthologs
7. The standard deviation of the AAI across all orthologs
8. The orthologous fraction (OF) between the two genomes, defined as the number of orthologs genes divided the minimum number of genes in either genome

Other output files produced by this command are described below.

Program Usage

Common Workflows

CompareM contains two common workflows. The first is for calculating the pairwise AAI between genomes (*--aai_wf*), and the second if for classifying query genomes by calculating AAI values against a set of reference genomes (*--classify_wf*). These workflows are provided for convenience and call several of the more granular commands provided by CompareM. Specifically, the *aai_wf* runs the *call_genes*, *similarity*, and *aai* commands, while the *classify_wf* runs the *call_genes*, *similarity*, and *classify* commands. These are described below.

Gene Prediction

Predicted genes are required for many of the functions provided by CompareM including the computation of AAI values. CompareM can call genes over a set of genomes using Prodigal.

Calling Genes

Genes can be called with the --*call_genes* function:

```
> comparem call_genes <input_files> <output_dir>
```

Positional Arguments. The <input_file> argument indicates the set of genomes to process and can either be i) a text file where each line indicates the location of a genome, or ii) a directory containing all genomes to be compared. Each genome should be represented as a single FASTA file containing genomic nucleotide sequences. The <output_dir> indicates the desired directory for all output files.

Optional Arguments. A number of optional arguments can also be specified. When specifying a directory of genomes to process, the *call_genes* command only processes files with a *fna* extension. This can be changed with the --*file_ext* argument. By default, CompareM will attempt to determine the best translation table for each genome. If a specific translation table is desired for all genomes, this can be indicated using the --*force_table* argument. The time to predict genes can be substantially reduced by using multiple processors as specified with the --*cpus* argument.

Output. The output directory contains three files for each genome: i) genes in nucleotide space, ii) genes in amino acid space, and iii) a generic feature file describing each called gene. In addition, a summary file is provided (*call_genes.summary.tsv*) indicating the translation table used for each genome along with the coding density of the genome under different translation tables.

Gene Homology and Genome Similarity

CompareM provides a granular interface for establishing reciprocal best hits (i.e. putative homologs) between genomes in order to allow experienced users to build customized workflows.

Reciprocal Protein Sequence Similarity

Reciprocal sequence similarity search between pairs of genomes can be performed using the *similarity* command:

```
> comparem similarity <query_proteins> <target_proteins> <output_dir>
```

Positional Arguments. The <query_proteins> and <target_proteins> arguments indicates the set of genomes to compare. Each genome must be represented by a single FASTA file containing proteins as amino acids. Similar protein sequences are first identified for each query genome against all target genomes. Proteins in target genomes that are similar to a query protein are than compared against all query proteins. This reciprocal search strategy allows large numbers of target genomes to be considered. The set of query and target genomes can be identical (e.g., when calculating pairwise AAI values). The query and target files to be process can be specified using either i) a text file where each line indicates the location of a genome, or ii) a directory containing all genomes to be compared. The <output_dir> indicates the desired directory for all output files.

Optional Arguments. The optional arguments specify the criteria used to identify similar proteins (i.e., putative homologs). By default, the e-value (--*evalue*), percent sequence identity (--*per_identity*), and percent alignment length (--*per_aln_len*) parameters are set to 1e-5, 30%, and 70%. The --*blastp* argument can be used if BLASTP+ should be used for the sequence similarity search instead of DIAMOND. This is orders of magnitude slower than using DIAMOND and requires *blastp* to be on your system path. When specifying a directory of genomes to process, the *similarity* command only processes files with a *faa* extension. This can be changes with the --*file_ext* argument. For systems with large amounts of memory (e.g., >64GB), the --*high_mem* argument can be used to reduce computation time. The temporary directory used to store results can be specified with the --*tmp_dir* argument. For the best performance, this should be set to a local directory with 100+ GB of space.

Output. This command produces several output files. The file *hits_sorted.tsv* specifies all pairs of proteins meeting the specified sequence similarity criteria. The output of this file is nearly identical to the tabular output used by BLAST except that the first four columns indicate the query_file_id, query_gene_id, subject_file_id, and subject_gene_id instead of just specifying query and subject identifiers. This file is also sorted by the first and third columns in order to facilitate fast downstream processing. Other files are used by DIAMOND to perform the sequence similarity search and other CompareM commands:

- *query_genes.faa* contains all query proteins
- *target_genes.faa* contains all target proteins
- *query_genes.dmnd* is the DIAMOND database of all query proteins
- *target_genes.dmnd* is the DIAMOND database of all target proteins
- *target_genes_hit.faa* contain all target proteins identified as being similar to a query protein

If the query and target genomes are the same, the *target* files are not produced.

Amino Acid Identity

Pairwise AAI values between genomes can be calculated using the *aai* command and the set of putative homologs identified with the *similarity* command:

```
> comparem aai <query_gene_file> <sorted_hit_table> <output_dir>
```

Positional Arguments. The <query_gene_file> is a FASTA file containing all query proteins and the <sorted_hit_table> indicates pairs of similar query proteins. Both these files are produced by the *similarity* command (*query_genes.faa* and *hits_sorted.tsv*, respectively). The <output_dir> indicates the desired directory for all output files.

Optional Arguments. The optional arguments specify the criteria used to identify reciprocal best hits (RBHs). RBHs are general considered to be orthologs and are used to calculate AAI values. By default, the e-value (--*evalue*), percent sequence identity (--*per_identity*), and percent alignment length (--*per_aln_len*) parameters are set to 1e-5, 30%, and 70%. Setting the *aai* arguments to be less stringent than the criteria used by the *similarity* command will give misleading results. The --*keep_rbhs* argument can be used to generate a file with all identified RBHs. This file can be extremely large.

Outputs. Pairwise AAI statistics are provided in the file *aai_summary.tsv* as described in the *Amino Acid Identity Workflow* section.

Taxonomic Classification

Query genomes can be compared to a set of target genomes using the *classify* command and the set of putative homologs identified with the *similarity* command. This information can be used to taxonomically classify query genomes by identify highly similar target genomes. More generally, this command allows similar reference genomes to be identified based on AAI values:

```
> comparem classify <query_gene_file> <target_gene_file> <sorted_hit_table> <output_dir>
```

Positional Arguments. The <query_gene_file> and <target_gene_file> are FASTA files containing all query and target proteins, respectively. The <sorted_hit_table> indicates all reciprocal hits between query and target proteins in a sorted table. All three of these files are produced by the *similarity* command. The <output_dir> indicates the desired directory for all output files.

Optional Arguments. The number of most similar genomes reported for each query genomes is specified by the --*num_top_targets* argument. The taxonomic identification of all target genomes can be specified using the –*taxonomy_file* argument. This file should contain a single line for each target genome with the genome identifier and taxonomic identification separated by a tab character: <genome_id><\t><taxonomic_identifcation>. Similar to the *aai* command, the criteria used to identify putative homologs can also be set with the --*evalue*, --*per_identity*, and --*per_aln_len* arguments. The --*keep_rbhs* argument can be used to generate a file with all identified RBHs.

Outputs. Classification results are provided in the file *classify.tsv* and consist of four columns:

1. Identifier of query genome
2. Identifier of target genome

3. AAI value between genomes
4. Orthologous fraction (OF) between genomes

When the --*taxonomy_file* argument is given, a fifth column is appended indicating the taxonomy of the target genome.

Usage Profiles

CompareM allows genomic features to be determined across large sets of genomes. Individual commands are provided for calculating amino acid (*aa_usage*), codon (*codon_usage*), stop codon (*stop_usage*), and k-mer frequency (*kmer_usage*) profiles. Each of these commands has a similar set of required positional arguments:

```
> comparem aa_usage <protein_gene_files> <output_file>
```

```
> comparem codon_usage <nucleotide_gene_files> <output_file>
```

```
> comparem stop_usage <nucleotide_gene_files> <output_file>
```

```
> comparem kmer_profile <genome_files> <output_file>
```

Positional Arguments. These commands require sequence data for each genome to be in a separate FASTA file. These files should contain genes in amino acid (*protein_gene_files*) space, genes in nucleotide (*nucleotide_gene_files*) space, or genomic sequences in nucleotide space (*genome_files*). These argument can be either i) a text file where each line indicates a FASTA file to process, or ii) a directory containing all FASTA files to process. The resulting profiles are written to the <output_file>.

Optional Arguments. By default, the abundance of each feature in a profile is given as a percentage. Raw counts can be obtained by using the --*counts* argument. For the *kmer_usage* command, the --*k* argument allows the desired *k*-mer size to be set. The --*keep_ambiguous* argument determines if codons containing ambiguous bases (e.g., N's) should be retained when calculating codon usage profiles with the *codon_usage* command. When specifying a directory of files to process, the --*file_ext* argument can be used to specify the desired extension of files to be process. The --*cpus* argument can be used to calculate profiles in parallel.

Output. All these command produce a single tab-separated output file with the same format. The first row is a header line indicating each of the features in the profile. Each subsequent row specifies the frequency or count of these features across a specific genome.

Lateral Gene Transfer

CompareM can calculate di-nucleotide and codon usage patterns across individual genes in order to help identify recent lateral gene transfers (Hooper and Berg, 2002):

```
> comparem lgt_di <nucleotide_gene_files> <output_dir>
```

```
>comparem lgt_codon <nucleotide_gene_files> <output_dir>
```

Positional Arguments. Both commands require individual FASTA files for each genome which specify genes in nucleotide space. The files to process can be specified as either i) a text file where each line indicates a FASTA file to process, or ii) a directory containing all FASTA files to process. The <output_dir> indicates the desired directory for all output files.

Optional Arguments. For the *lgt_di* command, the critical value used to identify putative LGT events can be set with the --*crit_value* argument. When specifying a directory of files to process, the --*file_ext* argument can be used to specify the desired extension of files to be process. The --*cpus* argument can be used to perform calculations in parallel.

Output. Individual files for each genome are written to the output directory. Each of these files is a tab-separate values file containing a header row followed by individual rows for each gene in the genome.

Visualization and Exploration

Dissimilarity tables and hierarchical cluster trees can be produced to aid in data exploration and communication.

Dissimilarity Tables

The *diss* command can be used to calculate the dissimilarity between genomes based on genomic usage profiles:

```
> comparem diss <profile_file> <output_file>
```

Positional Arguments. Any of the output profile files produced by the Usage Profile commands can be processed by the *diss* command. The resulting dissimilarity information is written to <output_file>.

Optional Arguments. The dissimilarity or distance between profiles can be calculated using a wide range of metrics as specified using the --*metric* argument. By default, Euclidean distances are calculated.

Output. By default, the output file specifies the dissimilarity between each pair of genomes on a separate line. Alternatively, the --*full_matrix* argument can be used to produce a matrix showing the dissimilarity between genomes. This output can be useful for direct visualization in programs such as Excel or for use in downstream programs expecting this format.

Hierarchical Cluster Trees

The *hclust* command can be used to create hierarchical cluster trees based on a measure of similarity or dissimilarity between genomes:

```
> comparem hclust <pairwise_value_file> <output_tree>
```

Positional Arguments. The <pairwise_value_file> must indicate the similarity or dissimilarity between all pairs of genomes with one pair given per line. This is the default output format of the *diss* command. The resulting tree is written to <output_tree> in Newick format.

Optional Arguments. Hierarchical cluster trees can be generated using a range of clustering criteria as specified using the --*method* argument. By default, trees are generated using "average" or UPGMA clustering. If the <pairwise_value_file> indicates similarity values the --*similarity* argument must be specified and the maximum similarity value specified with the --*max_sim_value* flag. The *hclust* command can process any tab-separated values files so long as each row indicates the two genomes compared and their similarity/dissimilarity. The columns containing this information can be indicated using the --*name_col1*, --*name_col2*, and --*value_col* arguments.

Output. Hierarchical cluster tree in Newick format.

References

Hooper S.D. and Berg O.G. (2002) Detection of genes with atypical nucleotide sequence in microbial genomes. *J Mol Evol*, **54**, 365-75.