paprica - PAthway PRediction by phylogenetIC plAcement (formerly Genome Finder)

Please contact bowmanjs@ldeo.columbia.edu with any questions.

Changes from v0.2x

Version 0.3 is a major upgrade from v0.2. The major changes are summarized below.

- Enzymes are now inferred along with pathways. Right now only CDS with an EC number in terminal nodes are considered when making an inference for internal nodes.
- The domain Archaea has been added. The number of reference genomes available is very limited so use at your own risk.
- To accommodate both domains the structure of the database and the commands have changed. These changes are detailed below and at the top of each script. You can access a help message for each script with the flag -h.
- Extensions have been added to allow for enzyme annotation and pathway prediction on metagenomes.
- You should no longer migrate the Python scripts to your working directory. You can call them in whatever location you downloaded them to from the working directory and the output files will be created in the working directory. The intention is that you'll move the **paprica_run.sh** script around but not the Python scripts.

Caveat and Request

Paprica started as an analysis workflow for a paper I was writing. As it grew in complexity it became clear that it should be spun off as an independent entity. Version 0.11 reflects this origin. Version 0.20 is my first attempt at making paprica user friendly. Version 0.30 is a continued evolution adding additional features. I'm really excited to keep improving the method and scripts. If you try to use paprica and run into difficulty, or have a suggestion to improve it, please don't give up. Let me know about it by creating an issue on Github or contacting me by email. I'll do my best to make the necessary changes.

Please cite paprica as:

Bowman, J. S., & Ducklow, H. W. (2015). Microbial Communities Can Be Described by Metabolic Structure: A General Framework and Application to a Seasonally Variable, Depth-Stratified Microbial Community from the Coastal West Antarctic Peninsula. *PloS one*, *10*(8), e0135868.

What should you do if something goes wrong?

Bioinformatics involves a lot of troubleshooting. This can be frustrating, but is, in the end, a good thing because it teaches us to do stuff. To get paprica to work you'll have to install a few core bioinformatics programs. If you're a Linux whiz this is trivial. If you're not it can be a little intimidating. If something goes wrong with the installation of one of the core programs (e.g. pplacer, Infernal, Seqmagick, pathway-tools, RAxML) please refer to the documentation for those programs and refer any questions to those development teams. If something seems wrong with paprica itself (i.e. you've verified that

Infernal is installed and running, but paprica throws an error trying to use it) open an issue in Github. I'll get on it as quick as possible.

If you're having trouble with basic installation tasks that result from idiosyncrasies of your system, or lack of familiarity with Linux/Unix, I'm happy to help. I do ask however, that you take an active role in solving these problems. Very often the information you need is already out there on various forums.

Quick Start

Download, install, and test the installation of the dependencies for **paprica_run.sh** as noted under the requirements section. Dependencies will need to be added to your PATH (probably by editing and sourcing .profile, .bash_profile, or.bashrc). The dependencies have been selected in part for their dependability; all are mainstream software and, with the exception of RAxML, available as pre-compiled executables.

If you are installing on Mac OSX you can follow the installation tutorial here.

Once the dependencies are set, from your home directory (or wherever you like to install things) download the last stable release, untar, and change the directory name:

```
wget https://github.com/bowmanjeffs/paprica/archive/paprica_v0.3.0.tar.gz

tar –xzvf paprica_v0.3.0.tar.gz

mv paprica-paprica_v0.3.0 paprica
```

You now have a directory titled paprica. This directory contains the paprica database and scripts. **If you do not plan to build a new database from scratch** you can ignore the **paprica-build.sh**, **paprica-make_ref.py** and **paprica-build_core_genomes.py** scripts altogether. At this point you should add the paprica directory to your PATH.

Next make the scripts executable by navigating into the paprica directory and typing:

```
chmod a+x *sh

chmod a+x *py
```

Execute the **paprica-run.sh** script by typing:

```
./paprica_run.sh test.bacteria bacteria
```

This will execute an initial analysis on the included fasta file test.bacteria.fasta. For your own analysis replace test with yourfileprefix. While you should not specify the file extension the extension must be .fasta.

You can test the archaeal database in the same way:

```
./paprica_run.sh test.archaea archaea
```

If you need to modify the run parameters for any of the Python scripts you can do this within the **paprica-run.sh** script.

Introduction

Paprica is a pipeline to conduct a *metabolic inference* on a collection of 16S rRNA gene sequences. Given a set of 16S rRNA gene sequences the pipeline returns a collection of metabolic pathways and enzymes with EC numbers that are expected for the observed taxa. It also returns some useful expected genome parameters, including genomic plasticity, size, number of coding sequences, 16S rRNA gene copy number, and number of genetic elements (see Output Files for a complete list). The abundance of each metabolic pathway will be corrected for 16S rRNA gene copy number. In this way paprica gives direct access to both community structure and metabolic structure data for your collection of 16S rRNA gene libraries. Paprica was designed with the analysis of large 16S rRNA gene sequence libraries in mind, such as those generated by 454 or Illumina sequencing, but is also appropriate for small datasets.

There are two options for running PAPRICA. If you would like to generate the database from scratch follow the guidelines in the **paprica-build.sh** script. If you would like to skip the (time and space intensive) database build, you can ignore this step and simply use **paprica-run.sh**. The output of your analysis will be the same in either case, but there are two advantages to building your own database. First, you can update it from Genbank as often as you like. Second, you have access to your own collection of PGDBs which you can explore with the pathway-tools software.

Requirements

Paprica requires a bash equipped, Linux-like operating environment. It was developed on Ubuntu. I have not tested it in OSX but it should work just fine. If you're a Windows user don't despair; paprica will work just fine in a Linux virtual box. I'd recommend running Ubuntu in your virtual box. Paprica relies on several open-source tools to function. Most of these tools are useful on their own if you're working with 16S rRNA gene data. If you do not plan to build your own database you can ignore all the dependencies labeled "build only". Required tools that should be located in your path are:

1. Python 2.7 (run, build, paprica-mg)
2. Infernal (including easel/miniapps) (run and build)*
3. pplacer (including Guppy and Taxtastic) (run and build)
4. Seqmagik (run and build)
5. RAxML (build only)**
6. Pathway-Tools (build, paprica-mg)
7. Archaeopteryx (optional but highly recommended, it will allow you to view the phyloxml "fat" trees produced by **paprica_place_it**.
8. DIAMOND (paprica-mg only)

*Easel is included as part of the Infernal package. You simply need to make sure that infernal/easel/miniapps is included in your path. For Mac users easel might be in a slightly different location, refer to this tutorial.

Currently **paprica-place_it.py assumes that you've been able to compile the AVX-2 version of RAxML and will attempt to call as raxmlHPC-PTHREADS-AVX2. If that version does not build, perhaps because you are on an older system, you will need to change that subprocess.Popen command in **paprica_place_it.py**.

Non-standard Python modules that you will need are listed below. I strongly recommend using a distribution such as Anacondas to acquire and manage these packages.

1. Biopython (run and build)
2. Pandas (run and build)
3. Numpy (run and build)
4. joblib (run and build)

The paprica directory structure and a brief description of the contents are as follows, directories are underlined and scripts are in bold:

1. models
    a. bacteria_ssu.cm (run and build): This is the covariance model for the domain Bacteria from the Rfam database here.
    b. archaea_ssu.cm (run and build): Same as above, but for the domain Archaea.
    c. kmer_top_1e5.txt (build): This is a list of the top 100,000 amino acid 5mers account for variability between proteomes (as determined by PCA). It is used to calculate the phi value (a measure of genomic plasticity) for each genome.
2. ref_genome_database
    a. archaea
        i. combined_16S.archaea.tax.refpkg: This is a reference package for phylogenetic placement. Refer to the pplacer documentation for more details.
        ii. genome_data.final.csv: This file holds data for each completed genome downloaded from Genbank. In addition to data provided by Genbank, it includes such things as clade number, a reasonable taxonomic description, genome size, number of 16S rRNA gene copies, etc.
        iii. internal_probs.csv: This is a matrix of all pathways contained in the current database and, for each internal node, the proportion of terminal daughter nodes the pathway appears in.
        iv. internal_data.csv: This file holds data for each internal node on the reference tree. Each value is typically the mean of the values for all the terminal daughter nodes for the clade
        v. internal_ec_probs.csv: This is a matrix of all EC numbered enzymes contained in the current database and, for each internal node, the proportion of terminal daughter nodes the enzyme appears in.
        vi. internal_ec_n.csv: This is a matrix of the mean number of occurrences of all EC numbered enzymes in the terminal daughter genomes for each internal node.

vii. terminal_paths.csv: This is a matrix of all pathways contained in the current database and their occurrence in each completed genome.
viii. terminal_ec.csv: This is a matrix of the number of times each enzyme with an EC number appears in each genome.
ix. combined_16S.tax.archaea.database_info.txt: This file contains some information about the database, including when the database was built and the number of genomes it contains. The primary reason for including this file was to provide some means of versioning different builds of the database. The date of the database build will appear in the sample_data.txt file that paprica produces with each run.

b. bacteria

i. combined_16S.archaea.tax.refpkg: This is a reference package for phylogenetic placement. Refer to the pplacer documentation for more details.
ii. genome_data.final.csv: This file holds data for each completed genome downloaded from Genbank. In addition to data provided by Genbank, it includes such things as clade number, a reasonable taxonomic description, genome size, number of 16S rRNA gene copies, etc.
iii. internal_probs.csv: This is a matrix of all pathways contained in the current database and, for each internal node, the proportion of terminal daughter nodes the pathway appears in.
iv. internal_data.csv: This file holds data for each internal node on the reference tree. Each value is typically the mean of the values for all the terminal daughter nodes for the clade
v. internal_ec_probs.csv: This is a matrix of all EC numbered enzymes contained in the current database and, for each internal node, the proportion of terminal daughter nodes the enzyme appears in.
vi. internal_ec_n.csv: This is a matrix of the mean number of occurrences of all EC numbered enzymes in the terminal daughter genomes for each internal node.
vii. terminal_paths.csv: This is a matrix of all pathways contained in the current database and their occurrence in each completed genome.
viii. terminal_ec.csv: This is a matrix of the number of times each enzyme with an EC number appears in each genome.
ix. combined_16S.tax.bacteria.database_info.txt: This file contains some information about the database, including when the database was built and the number of genomes it contains. The primary reason for including this file was to provide some means of versioning different builds of the database. The date of the database build will appear in the sample_data.txt file that paprica produces with each run.

3. utilities

a. **make_edge_fasta.py**: creates a fasta file containing the reads associated with a given edge or range of edges. Instructions for running can be found at the top of the script.

    b. **combine_edge_results.py**: aggregates the data created by running paprica_run.sh on multiple samples. Instructions for running can be found at the top of the script.
    c. **paprica_ref_tree_quick_parse.py**: parses a phyloxml format tree to obtain the reference sequence name (taxa) associated with edge edge. Instructions for running can be found at the top of the script.
    d. **read_qc.py**: given a fastq or fastq.gz file performs a reasonably fast quality control of the reads by converting low scoring bases to ambiguous bases (n). Further details and instructions for running can be found at the top of the script.

4. test.bacteria.fasta (build): A small file of arbitrary reads from the domain Bacteria.
5. test.archaea.fasta (build): A small file of arbitrary reads from the domain Archaea.
6. **paprica-build.sh**: This is a convenient wrapper for the Python scripts required to build the paprica database. You don't need to use this script.
7. **paprica-run.sh**: This is a convenient wrapper for the Python scripts required to conduct an analysis with paprica. You don't need to use this script to conduct an analysis.
8. **paprica-make_ref.py**: This script downloads genomes from Genbank, extracts 16S rRNA sequences, calculates other information on the genomes, and prepares everything for building a a reference tree.
9. **paprica-build_core_genomes.py**: This script uses a reference tree to determine what pathways and enzymes are likely to be shared downstream of each branch point (internal node) on the tree.
10. **paprica-place_it.py**: This script either builds a reference package for phylogenetic placement or uses an existing reference package to conduct a phylogenetic placement.
11. **paprica-tally_pathways.py**: This script compares the output of a phylogenetic placement with the paprica database to determine what enzymes and pathways are likely to be associated with each placement.
12. **paprica-mg_build.py**: This script uses the paprica database to build an annotation database for metagenomic analysis.
13. **paprica-mg_run.py**: This script uses DIAMOND blastx to annotate a metagenome against the paprica database, then predicts pathways on the annotation (optional).

<u>Script Specific Instructions</u>

paprica_make_ref.py:

-cpus: The number of cpus for RAxML to use. I don't recommend more than 8, see RAxML manual for details.

-domain: Which domain are you building the database for? Either "archaea" or "bacteria".

-download: Do you want to initiate a fresh download of the genomes in Genbank? Either "T" or "F".

-ref_dir: The name of the reference directory (ref_genome_database unless you are building you own and want a different name).

paprica_build_core_genomes.py:

-domain: Which domain are you building for? Either “archaea” or “bacteria”.

-pgdb_dir: The location where pathway-tools stores PGDBs.

-ref_dir: The name of the reference directory (probably ref_genome_database).

-tree: The phyloxml format tree that specifies the clade numbers within the reference tree.

paprica_place_it.py: This script has several flags. If only -ref is specified a new reference package will be built, taking the name passed with -ref and adding the extension .refpkg. The ref_genome_database directory must have a .fasta file of 16S rRNA sequences that you want to use to build the reference package of the same name as -ref. Specifying -query indicates that instead of building a reference package you want to place reads onto an existing reference package. If you specify -query you must also specify -splits. If you don’t want to split the query file simply specify -splits 1. If you want to use a random subsample of reads, for example to normalize the number of reads analyzed across samples (you should do this if you are making a comparative analysis across samples), specify the number of reads you would like to use with -n. The -n flag is not required.

-cpus: The number of cpus for RAxML to use during tree building. You only need to specify this if you are building a new reference package (i.e. in paprica_build.sh). Note that Infernal will always use as many cores as necessary. More is not always better with RAxML, please refer to the RAxML documentation.

-domain: The domain (bacteria or archaea) you are analyzing. This specifies the reference package that will be used.

-n: Optional. The number of reads that should be subsamples from the query fasta file.

-query: The name of the query fasta file, without .fasta

-ref: The name of the reference package, without .refpkg.

-ref_dir: The name of the reference directory (probably ref_genome_database).

-splits: Optional. The number of files you would like to split the query into so that phylogenetic placement can proceed in parallel. Be aware that there is significant memory overhead in making splits. Placing reads on the combined_16S.tax reference package requires about 8 Gb. Two splits will require 16 Gb.

paprica_tally.py: This script has the flags -o and -i to specify an output file name and the input file.

-cutoff: This specifies the fraction of genomes in a clade that need to contain a given pathway or enzyme for that pathway to be assigned to a read placed at the node ancestral to the clade.

-domain: The domain (bacteria or archaea) you are analyzing. This specifies the reference package that will be used.

-i: The name of the input csv file produced by paprica_place_it.py. You should use the complete file name, including the extension.

-o: The name that you would like to use as prefix for the output files. For example "-o test" will produce the files test.edge_data.csv, test.pathways.csv, test.sum_pathways.csv, and test.sample_data.txt.

-ref_dir: The name of the reference directory (probably ref_genome_database).

paprica-mg_build.py: This script builds a database that can be used to annotate a metagenome for pathway prediction on that metagenome. It contains all the proteins from the Bacteria and Archaea used in paprica that have an EC number. In order to build the paprica-mg database you need to have first built a paprica database using paprica_build.sh. To get around this you can get the relevant files paprica-mg.ec.csv.gz and paprica-mg.dmnd here (if you download these files you do not need to build the paprica database). You should download them to ref_genome_database and gunzip paprica-mg.ec.csv.gz.

Comment [j1]: do this

-ref_dir: The name of the directory containing the paprica database.

paprica-mg_run.py: This script uses DIAMOND blastx to annotate a metagenome against the paprica-mg database (all unique enzymes with an EC number).

-i: Input fasta or fasta.gz.

-o: Prefix for output files.

-ref_dir: Name of the directory containing the paprica database. The paprica-mg database files should be in the specified directory.

-pgdb_dir: The location where pathway-tools stores PGDBs.

-pathways: T or F, whether pathways should be predicted. Pathway prediction for a single deep metagenome can take a long time. If pathways are not predicted EC numbers are returned instead.

Output Files

Comment [j2]: check these against latest run

Running **paprica_run.sh** will produce a number of output files. Many will be useful in a downstream analysis. Running test.fasta will produce:

- test.[domain].combined_16S.[domain].tax.fasta: The query file combined with the reference alignment, as required by pplacer.

- test.[domain].combined_16S.[domain].tax.clean.fasta: The combined file with the names cleaned of punctuation that might break pplacer.
- test.[domain].combined_16S.[domain].tax.clean.align.sto: A Pfam format alignment returned by Infernal.
- test.[domain].combined_16S.[domain].tax.clean.align.fasta: Conversion from Pfam format to fasta.
- test.[domain].combined_16S.[domain].tax.clean.align.jplace: Output from the pplacer phylogenetic placement.
- test.[domain].combined_16S.[domain].tax.clean.align.csv: Output from the pplacer phylogenetic placement, in a human readable format.
- test.[domain].combined_16S.[domain].tax.clean.align.phyloxml: A "fat" style tree of the phylogenetic placements.
- test.[domain].edge_data.csv: Real or predicted genome data for terminal and internal nodes respectively:
    - edge_num: the edge number assigned during database construction.
    - taxon: a reasonable taxonomic name, for terminal nodes only.
    - nedge: the number of placements made to that edge (i.e. edge abundance in your sample).
    - n16S: the 16S rRNA gene copy number.
    - nedge_corrected: the copy number corrected edge abundance.
    - nge: the number of genetic elements (chromosomes + plasmids).
    - ncds: the number of coding sequences.
    - genome_size: total size of the genome in bp.
    - phi: a measure of genomic plasticity, 1 is maximum (infinitely high genomic plasticity), 0 is minimum.
    - clade size: how many terminal daughter nodes the clade has. 1 indicates a terminal node.
    - npaths_terminal: the mean number of pathways found in terminal daughters (internal nodes only).
    - npaths_actual: the number of pathways inferred for internal nodes (falling above the cutoff set in **paprica_tally_pathways**) or predicted for terminal nodes.
    - nec_terminal: the mean number of enzymes with EC numbers found in terminal daughters (internal nodes only).
    - nec_actual: the number of enzymes with EC numbers inferred for internal nodes (i.e. falling above the cutoff set in **paprica_tally_pathways**) or present in the annotation for terminal nodes.
    - confidence: the confidence score calculated for that edge
    - branch_length: the branch length to the node in the reference tree
    - GC: the GC content of the genome
- test.[domain].pathways.csv: A matrix of the pathways predicted or inferred according to edge. Abundance is normalized to 16S rRNA gene copy number.

- test.[domain].sum_pathways.csv: A two column matrix, column 1 is pathways and column2 is abundance in the sample. All possible pathways for that version of the database are reported, and pathways are in lexicographic order, to make it easy to combine multiple samples into a single matrix, which is probably what you want to do.
- test.[domain].sum_ec.csv: A two column matrix, column 1 is pathways and column2 is abundance in the sample. All possible pathways for that version of the database are reported, and pathways are in lexicographic order, to make it easy to combine multiple samples into a single matrix, which is probably what you want to do.
- test.[domain].ec.csv: The number of enzymes with EC numbers reported or predicted for each edge. Abundance is normalized to 16S rRNA gene copy number.
- test.[domain].sample_data.txt: Some useful data for the sample, in a tab-delimited format.
    - npathways: total unique pathways in the sample "pathway richness"
    - ppathways: the total number of pathways found in the current database
    - nreads: the total number of reads analyzed
    - sample_confidence: the weighted mean confidence score
    - database_created_at: when the database used for sample analysis was created

Running paprica-mg.py on the file test-mg.fasta.gz with pathways -F would produce:

- test-mg.annotation.csv: This csv file gives you the metabolic structure of the metagenome in the following columns:
    - index column: The accession of a representative enzyme in the database for that EC number.
    - genome: The genome to which the representative enzyme belongs.
    - bacteria: The domain to which this genome belongs.
    - EC_number: The EC_number of this enzyme.
    - Product: The product name associated with the representative enzyme.
    - n_occurrences: The number of occurrences of the sequence associated with the representative enzyme across the entire paprica database.
    - nr_hits: The number of hits to the EC_number. It is important to realize that the EC_number might have many protein sequences associated with it in the database, the nr_hits are to all of these, not the representative enzyme. The EC_number and nr_hits columns together give the "annotation" of the metagenome.
- test-mg.paprica-mg.nr.txt: This is the hits table produced by DIAMOND. Only one hit is reported for each read.
- test-mg.paprica-mg.nr.daa: This is the binary hits table produced by DIAMOND.

Running paprica-mg.py with the pathways -T would produce a PGDB describing all the metabolic pathways predicted for the metagenome (wherever pathway-tools stores these for your system) and also:

- test_mg.pathologic: This is a directory containing all the files required by the pathway-tools program pathologic.

- test_mg.paprica-mg.txt: A massive redundant hit table produced by DIAMOND. As many hits are allowed as there are genomes in the paprica database.
- test_mg.paprica-mg.daa: The binary version of the massive redundant hit table produced by DIAMOND. As many hits are allowed as there are genomes in the paprica database.
- test_mg.pathways.txt: A list of pathways predicted for the metagenome.

A quick note on directories and workflow

The **paprica_build.sh** script and associated Python scripts should be left in the paprica directory. If you want to build new a new, custom database without overwriting ref_genome_database it is best to just give the new database a unique name and build it within the paprica directory.

When using **paprica-run.sh** for analysis it is assumed that you're migrating paprica_run.sh to your working directory but leaving the Python scripts in their original location. This allows you to customize the flags for the Python scripts for each new analysis in a reproducible manner. Of course using **paprica-run.sh** isn't necessary at all; you can execute the Python scripts however you like (e.g. in ipython notebook or directly from the command line). Refer to the documentation and to **paprica-run.sh** to see what flags you should include as you structure your commands.

In all likelihood you will be running **paprica-run.sh** on multiple samples, possibly many samples. The bottleneck of the **paprica-run.sh** script is phylogenetic placement with pplacer during **paprica-place_it.py**. paprica now includes an option for running pplacer in parallel, but watch your memory useage! Because the whole pipeline is easily parallelized in this way it is most efficient to run multiple samples with a simple while loop. For example the following loop would execute **paprica-run.sh** on all the files listed, in a single column, in the text file samples.txt:

```
while read f;do
	./paprica_run.sh $f
done < samples.txt
```

Technical Details and Cautions

Edge numbers and database versions

The edge numbers assigned to the reference tree by pplacer during each new build of the database are NOT comparable. This means that while say, edge 1234 is one genome in one version of the database it will be something completely different in the next build. For terminal nodes this isn't really a problem, because paprica tells you the taxonomy of these nodes in edge_data.csv. You will need to figure out the taxonomy of internal nodes yourself for each build however, because there is no accurate way of doing this automatically. Use a tree viewer and the provided paprica_ref_tree_quick_parse.py to sort out the internal nodes of interest.

If you're updating paprica a lot to make use of new features or newer versions of the database, you can keep your output files and databases straight by paying attention to the database build date provided in the database and sample info files.

Read QC

Paprica assumes that you've gone through some standard QC measures with your reads. I suggest trimming the reads and getting rid of chimeras, chloroplasts, mitochondria, and errant eukaryotic reads. I like Mothur for these tasks, but there are other ways. Make sure that the reads you feed into paprica have NOT been dereplicated. The script utilities/read_qc.py can take care of some basic quality control for you but won't take care of chimeras or misamplified sequences.

Picking a cutoff

Paprica determines whether or not to assign a pathway to an internal node based on the proportion of terminal daughter nodes that have that pathway. By default if >= 50 % have it (i.e. cutoff = 0.50), it is assigned. This is a totally arbitrary (but probably reasonable) value, you should pick a value that you think works for your objectives. Obviously setting the cutoff to 1 would be most conservative. You can change this parameter under "user setable variables" in **paprica_tally_pathways**.

Phylogenetic placement

A major requirement for pplacer to work correctly is 1) a good reference tree and 2) a high quality alignment. In general the combination of Infernal plus RAxML produces a well-supported tree. There are problem areas, however. In initial testing the genus *Bdellevibrio*, for example, was collecting reads known to be Deltaproteobacteria. This could be a result of compositional bias or something else. Spot check the placements from your library by taking a few reads (using the make_edge_fasta.py utility) and blasting them or classifying them in RDP. If they approximate your placement you're good to go! Please let me know if you identify any troublesome taxa that don't seem to place to the right location.

Tree topology

It is well known that the topology of the bacterial 16S rRNA gene tree, constructed with standard substitution models, is not absolutely correct. In fact there can be large topological errors, with the placement of the Rickettsiales clade a common example. By necessity paprica reflects these errors and propagates them through the metabolic inference. I am confident in most sections of the tree, particularly in the relative locations of shallowly rooted clades, but you have been warned.

One final word on trees… the current tree consists of ~2,684 16S rRNA genes. That reflects the unique 16S rRNA genes that remained after the alignment of representative 16S rRNA genes from all completed genomes. Unfortunately it is not possible for paprica to distinguish between two strains with identical 16S rRNA genes.

paprica-mg

Executing paprica-mg.py with pathways -F is pretty straightforward. Executing with pathways -T is extremely time, space, and memory intensive. You've been warned.

Related Methods

PICRUSt - http://picrust.github.io/picrust/

Development Objectives for v0.30.

- Multigene alignment for the reference tree
- Expand the selection of genomes used (include categories other than “complete”)
- Inclusion of proteins/enzymes that don’t have EC numbers
- Better database versioning, including moving the refseq directories to a more permanent location and mirroring rather than overwriting.