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

Please contact jsbowman@ucsd.edu with any questions.

Changes from v0.3

Version 0.4.0 is a major upgrade from v0.3.1. The major changes are summarized below.

- The domain Eukarya has been added. In place of completed genomes the domain Eukarya relies on the transcriptomes produced by the MMETSP. Thus this domain is limited to a very small subset of marine microbial eukaryotes. As for the domain Archaea, consider this domain experimental and use with caution.
- Taxonomic information (according to the NCBI taxonomy) is now included with terminal and internal nodes. More details can be found in the manual.

Caveat and Request

Paprica started as the 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 pipeline. Version 0.11 reflects this origin. Version 0.20 was my first attempt at making paprica user friendly. Versions 0.3 and 0.4 are continued evolutions adding additional features. I'm really excited to keep improving the method and scripts. If you try to use paprica and run into difficulty please don't give up! Create an issue on Github and I'll do my best to fix it. Similarly, if you have a suggestion to improve paprica either create an issue or shoot me an email.

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 how 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/releases/tag/paprica_v0.4.0a

tar –xzvf paprica_v0.4.0a.tar.gz

mv paprica-paprica_v0.4.0a paprica
```

You now have a directory titled paprica. 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 against the database for the domain Bacteria ("bacteria"). For your own analysis replace test with your-file-without-extension. While you should not specify the file extension, the extension must be .fasta.

You can test the archaeal or eukarya databases in the same way:

```
./paprica-run.sh test.archaea archaea

./paprica-run.sh test.eukarya eukarya
```

For your own analysis I recommend adding the paprica directory to your PATH (so that you system knows where to look for the Python scripts), and copying the **paprica-run.sh** script to your working directory. You can modify all your run parameters as needed within the paprica-run.sh script. This provides a convenient record of how you ran paprica for different analyses. When you execute paprica-run.sh you will need to be explicit about its location (e.g. ./paprica-run.sh…) or your system will try to execute the original located in the paprica directory.

Introduction

Paprica is a pipeline to determine community structure and 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 the Output Files section for a complete list). For bacteria and archaea, the abundance of each genome, EC number, and 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 use the guidelines in the **paprica-build.sh** script and follow this tutorial. Building the database from scratch isn't recommended unless you have a specific reason for wanting to do so. If you would like to skip the (time and space intensive) database build, you can ignore this step and simply use **paprica-run.sh**. Reasons you might want to build the database from scratch include:

1. You can update it from Genbank as often as you like.
2. You have access to your own collection of PGDBs which you can explore with the pathway-tools software.
3. You can add custom genomes to it that are no included in the pool of completed Genbank genomes.
4. You can assign EC numbers to genomes if they are missing.

Requirements

Paprica requires a bash equipped, Linux-like operating environment. It was developed on Ubuntu. I do not regularly test it on OSX, but I hear that it works fine. Paprica will also work in Windows 10 with the bash shell installed. For all operating systems, a virtual box appliance running Ubuntu with paprica and all dependencies pre-installed is available here. Last, paprica is also available as an Amazon Web Service machine instance. If you are interested in using the paprica AWS machine instance follow the instructions here.

Most users will be want to install paprica and the **paprica-run.sh** dependencies on their local system or server. I'd recommend running Ubuntu in your virtual box. The **paprica-run.sh** script relies on several open-source tools to function. All of these tools are useful on their own if you're working with 16S rRNA gene data. Required tools for **paprica-run.sh** that should be located in your path are:

1. Python 2.7, including these modules:
    a. pandas
    b. joblib
    c. Biopython
2. Infernal, (including the easel/miniapps subdirectory*)
3. pplacer, must be v1.1alpha19 or later!

4. Seqmagik

In addition this software is recommended for viewing the trees produced by pplacer:

Archaeopteryx (optional but highly recommended, it will allow you to view the phyloxml "fat" trees produced by **paprica-place_it**.

If you want to build the database from scratch you need all of the dependencies for paprica-run.sh plus:

1. RAxML (pplacer will assume the combiled binary is standard-RAxML/raxmlHPC-PTHREADS-AVX2)
2. Pathway-Tools

Last, if you use the paprica-mg or paprica-mt extensions for metagenomic or metatranscriptomic analysis you will need:

1. DIAMOND
2. BWA

*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 OSX users easel might be in a slightly different location, refer to this tutorial.

Directory Structure

The paprica directory structure and a brief description of the contents are as follows, directories are underlined and scripts are in bold. Files and directories in gray will only be present if you build the database from scratch.

1. models
    a. bacteria_ssu.cm: This is the covariance model for the domain Bacteria from the Rfam database here.
    b. archaea_ssu.cm: Same as above, but for the domain Archaea.
    c. eukarya_ssu.cm: Same as above, but for the domain Eukarya.
    d. 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/bacteria/eukarya directories
        i. refseq: This directory contains subdirectories for each genome downloaded from Genbank.
        ii. combined_[16S or 18S].[domain].tax.refpkg: This is a reference package for phylogenetic placement. Refer to the pplacer documentation for more details.
        iii. 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.

iv. 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.

v. 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

vi. 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.

vii. 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.

viii. terminal_paths.csv: This is a matrix of all pathways contained in the current database and their occurrence in each completed genome.

ix. terminal_ec.csv: This is a matrix of the number of times each enzyme with an EC number appears in each genome.

x. combined_[16S or 18S].tax.[domain].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. user: This directory contains information that you would like to use to supplement the database completed from draft genomes. It is only relevant if you plan to build the database from scratch using **paprica-build.sh**.

i. user_ec.csv: Genome annotations are always, at best, imperfect. If you know that a genome should have an enzyme with an EC number, but that gene product isn't represented in the genome's Genbank file, you can add it here. Enzymes added here will propagate to internal nodes on the reference tree but will not be used in pathway prediction (sorry).

ii. bacteria: If you are building the database from scratch, and you have custom bacterial genomes paprica will look for them here.

iii. archaea: If you are building the database from scratch, and have custom archaeal genomes paprica will look for them here.

c. paprica-mgt.database: This directory will only be present if you've run the paprica-build_mgt.py script, or downloaded (and untarred) the pre-built paprica-mgt.database here. It contains the BWA and DIAMOND Blastx databases, and some custom files for metatranscriptomic or metagenomic analysis.

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.
4. test.bacteria.fasta: A small file of arbitrary reads from the domain Bacteria.
5. test.archaea.fasta: A small file of arbitrary reads from the domain Archaea.
6. test.eukarya.fasta: A small file of arbitrary reads from the domain Archaea.
7. **paprica-build.sh**: The core script for building the database.
8. **paprica-run.sh**: The core script for running an analysis.
9. **paprica-make_ref.py**: Executed by **paprica-build.sh**, this script downloads genomes from either Genbank or MMETSP, extracts 16S rRNA sequences, calculates other information on the genomes, and prepares everything for building a reference tree.
10. **paprica-build_core_genomes.py**: Executed by **paprica-build.sh**, 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.
11. **paprica-place_it.py**: Executed by **paprica-run.sh** and **paprica-build.sh**, this script either builds a reference package for phylogenetic placement or uses an existing reference package to conduct a phylogenetic placement.
12. **paprica-tally_pathways.py**: Executed by **paprica-run.sh**, 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.
13. **paprica-mgt_build.py**: This script uses the paprica database to build an annotation database for metagenomic or metatranscriptomic analysis.
14. **paprica-mg_run.py**: This script uses DIAMOND blastx to annotate a metagenome against the paprica database, then predicts pathways on the annotation (optional).
15. **paprica-mt_run.py**: This script uses BWA to annotate a metatranscriptome against the paprica database.

<u>Script Specific Instructions</u>

The Python scripts called by **paprica-build.sh** and **paprica-run.sh** all have flags that you may want to modify in those shell scripts.

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", "F", or "test".

-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.

-unique: T or F, default is F. When set to true the number of unique reads are tracked at each edge, along with the abundance of each unique read.

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 Guppy in 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).

-omit: If you have edges that you wanted omitted from you analysis (e.g. cyanobacteria that might be chloroplasts) put as range here, for example "-omit 5:7" to omit edges 5, 6, and 7.

-override: If there are placements to the reference tree that you know to be incorrect, you can force paprica to use a user-defined edge instead. You can specify as many edges as you want as such "-override 1|3,16|14", which tells paprica to replace information for edge 1 with that for edge 3, and edge 16 with that for 14. Note that it is necessary to place this flag in quotes.

-unique: The name of the unique.seqs.csv file created by paprica-place_it.py. You should use the complete file names, including the extension. Requires that you ran paprica-place_it.py with the -unique flag set to "T". By default paprica assumes that you did not do this, and will not tally unique reads.

paprica-mgt_build.py: This script builds a database that can be used to annotate a metagenome or metatranscriptome. It contains all the proteins from the Bacteria and Archaea used in paprica that have an EC number, plus those proteins from the NCBI Refseq Viral database. In order to build the paprica-mgt database you need to have first built a paprica database using paprica-build.sh. To get around this you can get the relevant files here as a tarball (if you download these files you do not need to build the paprica database). You should download the tarball to ref_genome_database, and untar the directory with "tar –xzvf paprica-mtg.database.tgz".

-ref_dir: The name of the directory containing the paprica database. Default is ref_genome_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-mgt.database directory should be in the specified directory. Default is ref_genome_database.

-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. I don't really recommend setting this flag to "T".

paprica-mt_run.py: This script uses BWA to annotate a metatranscriptome against the paprica-mt database (genes coding all unique enzymes with an EC number).

-i: Input fasta or fasta.gz. If you have PE reads you would enter two files, like: paprica-mt_run.py -i file1.fasta.gz file2.fasta.gz...

-o: Prefix for output files.

-ref_dir: Name of the directory containing the paprica database.

-pgdb_dir: The location where pathway-tools stores PGDBs. Only necessary if pathways are predicted.

-pathways: T or F, whether pathways should be predicted. This can take a long time.

-t: The number of threads for bwa to use.

Output Files

Running **paprica-run.sh** will produce a number of output files. Many will be useful in a downstream analysis. Running a file names test.fasta would 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.
    - nedge: the number of placements made to that edge (i.e. edge abundance in your sample).
    - post_prob: mean posterior probabilities for placements to that edge, see pplacer documentation for details.
    - map_ratio: mean percent identity to edge for all placements at that edge.
    - map_overlap: mean shared bases with edge for all placements at that edge.
    - taxon: a reasonable taxonomic name from the NCBI taxonomy.
    - 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).
    - nec_terminal: the mean number of enzymes with EC numbers found in terminal daughters (internal nodes only).
    - branch_length: the branch length to the node in the reference tree.
    - GC: the GC content of the genome
    - 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_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
- 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
- test.[domain].unique.seqs.csv: An abundance table of the unique sequences. This is an intermediate file, use test.[domain].unique_seqs.csv for your analysis.
    - Index column: a hash of the sequence
    - abundance: the number of times the sequence appeared in the sample
    - edge_num: the edge number the sequence was assigned
    - rep: The name of a representative sequence
- test.[domain].unique_seqs.csv: A more sophisticated abundance table of the unique sequences.
    - identifier: the hash of the sequence concatenated with the edge number
    - abundance: the number of times the sequence appeared in the sample
    - edge_num: the edge number to which the sequence was assigned
    - rep: the name of a representative sequence
    - abundance_corrected: sequence abundance divided by the number of 16S rRNA gene copies for that edge
    - taxon: edge taxon name

Running the **paprica-mg_run.py** script will produce the following output for the specified output file prefix of "test", with the -pathways flag set to T:

- test.annotation.csv: The number of hits in the metagenome, by EC number. This is probably the most useful file to you. The columns are:
    - index: The accession of a representative protein from the database
    - genome: Genome the representative protein comes from
    - domain: Domain of this genome
    - EC_number: The EC number
    - product: A sensible name for the gene product
    - start: Start position of the gene in the genome
    - end: End position of the gene in the genome
    - n_occurences: The number of occurrences of this EC number in the database
    - nr_hits: The number of reads that matched this EC number. Each read is allowed only one hit.
    - n_cds: The number of occurrences of this gene sequence in the dataset
    - n_trans: The number of occurrences of this gene product in the dataset

- test.paprica-mg.nr.daa: The DIAMOND format results file. Only one hit per read is reported.
- test.paprica-mg.nr.txt: A text file of the DIAMOND results. Only one hit per read is reported.
- test_mg.pathologic (for -pathways T only): A directory containing .gbk files for each genome in the paprica database that received a hit, with each EC number that got a hit for that genome.
- test.pathways.txt: A simple list of all the pathways that were predicted for the metagenome.
- test.paprica-mg.txt (for -pathways T only): A text file of the DIAMOND results, as many hits were reported as there are genomes in the database. You probably want to get rid of this right away to save space.
- test.paprica-mg.daa (for -pathways T only): A DIAMOND format results file, as many hits were reported as there are genomes in the database. You probably want to get rid of this right away to save space.
- testcyc (for -pathways T only): A directory in ptools-local with the PGDB and reports for the pathway prediction.

Running the **paprica-mt_run.py** script will produce the following output for the specified output file prefix of "test":

- test.sam.gz: Mapping results, in SAM format.
- test.tally.ec.csv: CSV format file of each mapped gene. Columns are protein id, genome, domain, EC_number, product name, number of reads mapped.
- test.paprica-mt_report.txt: Some helpful stats including the query file name, number of reads, number of hits, number of genomes with a hit.

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** using the bacteria database on all the files listed, in a single column, in the text file samples.txt:

```
while read f;do
```

```
        ./paprica-run.sh $f bacteria
done < samples.txt
```

Then you aggregate the results by copying combine_edge_results.py to your working directory and executing:

```
combine_edge_results.py
```

You may wish or need to use flags, however the defaults will work in most cases. Refer to the instructions in the script for more details.

Adding custom genomes or enzymes to the database

Since version 3.0.0 it's been possible to add custom genomes to the database. These might include draft genomes from Genbank with features that are not present in the completed genomes, or custom genomes sequenced in the lab. Regardless of the source these genomes should be placed in a directory titled user/bacteria or user/archaea inside the database directory (so that the database directory contains directories titled user, bacteria, and archaea, and the user directory itself contains directories titled bacteria and archaea). Each genome should have its own directory, titled with its accession number or another short, unique identifier, and should contain a Genbank file (.gbff, not .gpff or .gbk for consistency with the naming format of completed genomes) with annotations and translations and a nucleotide fasta file (.fna) with the sequence of the contigs. No phi value is calculated on user genomes and they are not included in confidence scoring. **Important**: If your user genome(s) are not annotated with EC numbers you will get nothing out of them.

Sometimes you will encounter genomes that lack an enzyme that you are pretty sure should be there. This can result from incomplete or incorrect annotation of the genome. You can manually add these enzymes by modifying the paprica/ref_genome_database/user/user_ec.csv file. Follow the instructions and example at the top of the file. A proper entry follows the form index number, GI number, EC number, product name, like this:

1,GCF_000011305.1,1.7.2.1,nitrite reductase - NO forming

This file will be read every time you run **paprica-build.sh**, specifically by **paprica-build_core_genomes.py** (running only that script will update your database with new information in user_ec.txt without recalculating all the genome parameters, and without rebuilding your pplacer reference package (i.e. your edge numbers will remain the same).

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. The classification features of pplacer and guppy will provide an estimate of the taxonomy for internal nodes, but I recommend that you check these before reporting the results. 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 Seqmagick and Mothur for these tasks, but there are other ways. Make sure that the reads you feed into paprica have NOT been dereplicated.

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 in **paprica-tally_pathways** with the -cutoff flag.

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… as of writing the bacterial tree consisted of ~3,275 16S rRNA genes, the archaeal tree 187, and the eukaryote tree 355. These values reflect the unique 16S/18S rRNA genes that

remained after the alignment of representative genes from all completed genomes. Unfortunately it is not possible for paprica to distinguish between two strains with identical 16S rRNA genes.

Related Methods

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