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

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

Changes from v0.11

Version 0.2x is a major overhaul of paprica. The major changes are summarized below.

- Pathways are pre-calculated for each edge on the reference tree. This allows for a light-weight version of paprica that does not require the (very large) full database. The only downside to this is that you do not have access to the PGDBs if you want to do something more advanced than collect the pathways and data for your collection of 16S rRNA reads.
- If you are maintaining your own database the PGDBs are now re-used between database builds. This substantially reduces the time and overhead involved in updating the database. Simply running the **paprica_build.sh** script will generate a new database based on the current selection of completed genomes in Genbank.
- The paprica.sh script has been split into two scripts; **paprica_build.sh** and **paprica_run.sh**. If you would like to build your own version of the database you will need to run **paprica_build.sh**. If not you only need to run **paprica_run.sh**.
- Infernal is now used for alignment instead of Mothur. I'm a big fan of Mothur for 16S analysis, but I received guidance that the Silva alignment is not suitable for phylogenetic analysis given the large number of gaps in the alignment. Infernal is a little slower but does produce a nice (and not gappy) alignment. The alignment methods still need some improvement, however. The current method does not apply any filter to the alignment. This seems to produce an okay tree but is not optimal.
- RAxML is now used for tree building instead of Fasttree. Fasttree is a wonderful program but getting the tree topology correct is really important to paprica and RAxML does a better job at this. It is also much slower (but is only run once, to build the database).
- The confidence score for internal nodes is now based in part on the number of pathways in a given internal node, relative to the mean number of pathways in its terminal daughter nodes. Previously the number of coding sequences was used.
- The Archaeal genomes have been excluded from the database. It simply isn't possible as yet to make a good alignment with both domains, and the archaeal confidence scores are always very low due to the very few genomes available.

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. 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. To get paprica to work you'll have to install a few core bioinformatics programs. If you're good at Unix/Linux this is trivial. If you're not this 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 that are not Python modules 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_v0.21.tar.gz

tar –xzvf paprica_v0.21.tar.gz

mv paprica-paprica_v0.21 paprica
```

You now have a directory titled paprica. Open the file paprica_profile.txt and modify the variables as appropriate. It is recommended that you use full pathways for all variables even though this isn't strictly necessary in all cases. For a typical installation to your home directory on a Linux/Unix/OSX system **paprica_run.sh** should work out of the box without modifying the profile, but you should probably do this anyway. Please be sure to use full directory paths (i.e. /home/user/paprica/). The use of the profile file allows you to migrate the profile, paprica_run.sh, and the two Python scripts it calls around wherever you like. You don't need to migrate the database and build scripts around or maintain more than one copy of the database. **If you do not plan to build a new database from scratch** you can ignore the **paprica_build.sh**, **paprica_make_ref** and **paprica_build_core_genomes** scripts altogether.

Next make the **paprica_run.sh** script executable by navigating into the paprica directory and typing:

```
chmod a+x paprica_run.sh
```

Execute the script by typing:

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

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

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 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 and build)
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 only)
7. Gnu parallel (build only)
8. Archaeopteryx (optional but highly recommended, it will allow you to view the phyloxml "fat" trees produced by **paprica_place_it**.

*Easel is included as part of the Infernal package. You simply need to make sure that infernal/easel/miniapps is included in your path.

Currently **paprica_place_it 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**.

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)

You will find the following required files in the paprica directory:

1. bacterial_ssu.cm (run and build): This is the covariance model for the domain Bacteria from the Rfam database here.
2. 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.
3. test.fasta (build): A small file of arbitrary, identical reads.
4. paprica_profile.txt (run and build): A text file that contains some variables, such as directory paths, that paprica needs. You should modify this file according to the instructions included at the top of the file.

Script Specific Instructions

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.

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

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

-splits: 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.

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

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

-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.

Database Files

Paprica comes with a pre-built database described by four csv files. If you regenerate the database without changing the ref_dir variable in paprica_profile.txt these files will be overwritten. This is fine, but if you would like to save the old database you should rename ref_genome_database to prevent overwriting (or change the ref_dir variable).

- genome_data.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.
- terminal_paths.csv: This is a matrix of all pathways contained in the current database and their occurrence in each completed genome.
- 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.
- 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.
- combined_16S.tax.refpkg: This directory is the reference package for pplacer. It contains some files that might be of interest to you, including the reference alignment and the reference tree. Don't make modifications to these files however, and expect pplacer to work. You will need to rebuild using **paprica_place_it.py** and the correct flags.
- combined_16S.tax.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.

Output Files

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.combined_16S.tax.fasta: The query file combined with the reference alignment, as required by pplacer.
- test.combined_16S.tax.clean.fasta: The combined file with the names cleaned of punctuation that might break pplacer.
- test.combined_16S.tax.clean.align.sto: A Pfam format alignment returned by Infernal.
- test.combined_16S.tax.clean.align.fasta: Conversion from Pfam format to fasta.
- test.combined_16S.tax.clean.align.jplace: Output from the pplacer phylogenetic placement.
- test.combined_16S.tax.clean.align.csv: Output from the pplacer phylogenetic placement, in a human readable format.
- test.combined_16S.tax.clean.align.phyloxml: A "fat" style tree of the phylogenetic placements.
- test.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.
    - 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.pathways.csv: A matrix of the pathways found in each edge in the sample.

- test.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.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

Utilities

The utilities directory contains scripts that you may find useful in your analysis.

- **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.
- **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.

A quick note on directories and workflow

The **paprica_build.sh** script and associated Python scripts should be left in the paprica directory. Although you can have paprica build the database anywhere you like, it probably simplifies things to leave it in the default location inside the paprica directory. Note that you will still have to tell paprica where this is, by modifying the ref_dir variable in paprica_profile.txt.

When conducting an analysis I find it easiest to make new copies of the scripts called for in **paprica_run.sh** in my working directory for that analysis. Since the location of the database is specified in paprica_profile.txt by the ref_dir variable (which you need to modify) you should be able to run these scripts wherever you like.

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 alignment with Infernal. Because Infernal automatically uses as many cpus as are available I find it easiest to run multiple samples in sequence using a simple for or 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 and archaeal 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.

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 >= 90 % have it (i.e. cutoff = 0.90), 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.

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”)
- Archaeal version