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

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

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. The abundance of each metabolic pathway will be corrected for 16S rRNA gene copy number. PAPRICA as 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.

Requirements

PAPRICA requires a Linux-like operating environment. It was developed on Ubuntu. PAPRICA uses several open source tools in addition to some novel components. Most of these are tools that you should probably have if you're working with 16S rRNA gene data anyway, and all come with precompiled executables for Linux/OSX. Required tools that should be located in your path are:

1. Python 2.7
2. Biopython
3. The Python joblib module
4. Blast+ (and the 16SMicrobial database)
5. Mothur (and one of the Mothur formatted alignment databases, preferably the Silva seed)
6. FastTreeMP
7. pplacer, Guppy, and taxit from the Matsen group
8. Pathway-Tools

Components

Seven Python scripts are provided to create the database and execute the metabolic inference. The **paprica.sh** script is a template to guide database construction and analysis. An addition text file, mean_phi_values.txt, provides data on the expected genomic plasticity of each genome.

Basic operation

The first time **paprica.sh** executes it must construct the database necessary to conduct the metabolic inference. Counterintuitively it needs to run part of an analysis before it can construct the database. Database construction is very time intensive and can take several days, however, subsequent executions are fairly fast.

Near the top of each script is a set of variables that should be modified by the user to reflect directory pathways and other aspects of the operating environment. A brief description of each script follows.

paprica_make_ref.py (only necessary to construct database)

1. Downloads all completed genomes from Genbank using wget
2. Finds the 16S rRNA genes in these genomes using blastn and the 16SMicrobial database
3. Compiles 16S rRNA gene copy number with the provided phi values
4. Outputs:
    a. Creates, in the reference directory you identify, a directory structure mimicking the relevant portion of the Genbank FTP site.
    b. combined_16S.fasta – a fasta file of the first 16S rRNA gene identified in each genome.
    c. genome_data.txt – a tab-delimited file giving each genome, the phi score of genomic plasticity, and the number of 16S rRNA gene copies for that genome.

paprica_parse_blast_xml.py (necessary to construct database)

1. A dependency of **paprica_make_ref.py**.

paprica_place_it.py [query without extension] [reference package name] (necessary to construct database and for analysis)

1. Uses the 16S rRNA genes from the completed genomes to build a reference tree with FastTreeMP after alignment with Mothur
2. Uses taxit to convert the alignment and tree into a reference package
3. Executes pplacer to place query reads on the reference tree
4. Outputs:
    a. combined_16S.refpkg (directory) - When run to build a reference package (see paprica.sh) produces a directory containing the reference package. This directory will be located in the identified reference directory. Also produces some additional Mothur outputs.
    b. [query]_combined_16S.pplacer.filter.jplace - When run to execute a phylogenetic placement produces a jplace file that can be passed to Guppy.

paprica_build_core_genomes.py [query.fat – a phyloxml tree] (necessary to construct database)

1. Uses a phyloxml format "fat" tree and blastn to build a consensus genome for each node on the reference tree.
2. Generates statistics for each consensus genome, including size, expected degree of genomic plasticity, and expected 16S rRNA gene copy number.
3. Outputs:
    a. combined_16S.core_genomes (directory) – A directory tree located in the identified reference directory containing the consensus genome for each internal node on the reference tree and some summary statistics.

paprica_get_genomes.py [query.csv – q guppy csv file of placements] [query – a prefix for output] (necessary for analysis)

1. Uses a csv file of edge placements generated by pplacer and Guppy (you must generate these files manually, see the paprica.sh script) to identify which consensus genomes should be included in the final metabolic inference.
2. Uses gene copy number information to correct the number of times that each consensus genome is represented in the final metabolic inference.
3. Checks to see if a Pathway Genome Database (PGDB) already exists for each included genome.
4. If a PGDB does not exist, it sets up the necessary files for Pathway-Tools and prepares a script to execute the Pathway-Tools pathologic module. The user should execute the generate_pgdbs.sh script after this script completes. Depending on the number of PGDBs that need to be created generate_pgdbs.sh can take some time to finish. Generally, the more analyses you run, the faster this gets, because fewer new PGDBs need to be created each time.
5. Calculates a confidence score for each sample, based on the expected genomic plasticity of the consensus genome and the relative size of the consensus genome compared to the size of all daughter genomes.
6. Outputs:
    a. **generate_pgdbs.sh** – a shell script in the working directory that must be executed manually to generate the PGDBs.
    b. Other files in the reference directory necessary for Pathologic execution.
    c. [query].edge_tally.txt – a tab-delimited file containing the placement edge number, number of placements, number of placements corrected for 16S rRNA gene copy number, size of the core genome, mean clade genome size, standard deviation of the clade genome size, 1 – phi, and the 16S rRNA gene copy number.

paprica_tally_pathways.py [query – the input csv file without extension] (necessary for analysis)

1. Uses the information in [query].edge_tally.txt and in the pathway_report.txt file created for each PGDB to generate a list of all the metabolic pathways that are inferred for the sample.
2. Outputs:
    a. [query].pathway_summary.txt – A tab-delimited file of each pathway and the number of times it is inferred for the sample.
    b. [query].pathway_detail.txt – A tab-delimited file of each consensus genome, the number of times it occurs (corrected for 16S rRNA gene copy number), and the pathways that were predicted for that genome.

A quick note on directories and workflow

You can run the scripts in whatever directories you choose, provided you set the variables appropriately in each script. I find it easiest, however, to create one directory for my database in my home directory. The make_ref, blast_xml, build_core_genomes, and a copy of place_it should be located in this directory. The reference directory is constructed inside this directory. This directory serves as the location of the reference directory in the remaining scripts (including the copy of place_it used for analysis). I create a new copy of the remaining scripts (including place_it) in each working directory. Of course it would be possible to keep them in one place and simply call to that location.

Because some initial analysis is required before build_core_genomes can construct the database (it needs a reference tree with node numbers to work from – I can only figure out how to get this from a guppy derived phyloxml tree) I suggest using any small set of fasta sequences located in your reference directory to conduct this initial analysis. There is no minimum number of reads that need to be used in this step; 100 reads will take very little time to complete.

Related Methods

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

Development Objectives

- Standardize inputs and outputs – some scripts require extension, some do not. All input files should require extension, output name arguments should not.

IN PROGRESS