ShotgunUnifrac

Charlie Bushman

Aug 24, 2022

CONTENTS:

ShotgunUnifrac is a snakemake pipeline for generating a phylogeny per core gene over a selection of genomes and then combining them together to create a consensus phylogeny. The pipeline is written in snakemake and includes a python library for data curation and prep. View the *Quickstart Guide* page to see a worked example of installing and using the pipeline.

CHAPTER

ONE

QUICKSTART GUIDE

Contents

1.1 Installation

The first steps are to clone this repo and then install conda and snakemake. We can then install the Core Genes Extraction (CorGE) library and run tests to make sure everything is properly installed.

```
git clone git@github.com:Ulthran/ShotgunUnifrac.git

cd ShotgunUnifrac/CorGE
pip install .

cd ../
pytest CorGE/tests/
pytest .tests/
```

Tip: If you've never installed Conda before, you'll need to add it to your shell's path. If you're running Bash (the most common terminal shell), the following command will add it to your path: `echo 'export PATH=$PATH:$HOME/miniconda3/bin' > ~/.bashrc`

If you see "Tests failed", file an issue on GitHub.

1.2 Setup

We'll start by creating some dummy inputs to work with using Escherichia coli, Buchnera aphidicola, Cellulomonas gilvus, Dictyoglomus thermophilum, and Methanobrevibacter smithii (a randomly selected group of gut bacteria plus Methanobrevibacter smithii as an outgroup for tree rooting).

```
echo $'7\n9\n2173' > EX_TXIDS.txt
echo $'GCF_000218545.1\nGCF_000020965.1' > EX_ACCS.txt
```

This creates two dummy input files

- `EX_TXIDS.txt` contains species-level taxon ids which CorGE will fetch genes and proteins for from NCBI.
- `EX_ACCS.txt` contains genome accessions which CorGE will fetch from NCBI.

Tip: You can curate genomes/proteins from NCBI using taxon ids and genome accessions but you can also (in the same command) gather local genomes/proteins as well using the `--local` flag.

1.2.1 Data curation

To gather the genes and proteins we need for tree building using the files we just created we use `CorGE collect_genomes`. Then to filter single copy core genes (SCCGs) from each file and merge nucleotide/amino acid sequences by SCCG we use `CorGE extract_genes`.

```
CorGE collect_genomes --ncbi_species EX_TXIDS.txt --ncbi_accessions EX_ACCS.txt ./
CorGE extract_genes ./
```

This should create the following directories and files from root

- `assembly_summary.txt` is downloaded from NCBI to find the best genome accessions for each taxon id.
- `config.yml` is provided to the snakemake pipeline to specify what it should look for and where.
- `nucleotide` is a directory containing all gathered nucleotide-encoded genomes (saved as .fna).
- `protein` is a directory containing all gathered protein-encoded genomes (saved as .faa).
- `outgroup` is a directory containing the nucleotide and protein files for the outgroup (if there is an outgroup).
- `filtered-sequences` is a directory containing each SCCG from each genome (protein-encoded) in their own files.
- `merged-sequences` is a directory containing each SCCG from each genome this time in per-SCCG files.

1.2.2 Tree building

To build the per-SCCG phylogenies and then merge them together we use the snakemake pipeline. Everything should be properly set up from running CorGE so we can just go ahead and run the pipeline.

```
snakemake all -c --use-conda --conda-prefix .snakemake/
```

This should create the following directories and files from root

- `RAxML_outgroupRootedTree.final` is the final consensus tree.
- `aligned-sequences` is a directory containing alignments for the merged-sequences.

- `trees` is a directory containing phylogenies built from each SCCG alignment as well as some intermediates in the merging process.

Tip: `--use-conda` causes snakemake to use per-rule defined conda environments while it runs the pipeline. `--conda-prefix .snakemake/` tells conda where to put/look for these environments.

1.3 Viewing results

The output is `RAxML_outgroupRootedTree.final` which can be viewed using any newick-format tree viewer (like ETE Toolkit).

1.4 tl;dr

Follow instructions to install anaconda / miniconda and snakemake then

```
git clone git@github.com:Ulthran/ShotgunUnifrac.git
cd ShotgunUnifrac
echo $'7\n9\n2173' > EX_TXIDS.txt
echo $'GCF_000218545.1\nGCF_000020965.1' > EX_ACCS.txt
cd CorGE
pip install .
cd ..
CorGE collect_genomes --ncbi_species EX_TXIDS.txt --ncbi_accessions EX_ACCS.txt ./
CorGE extract_genes ./
snakemake all -c --use-conda --conda-prefix .snakemake/
```

You should now have an output called `RAxML_outgroupRootedTree.final`.

CHAPTER

TWO

CORGE USAGE GUIDE

Contents

2.1 Options

2.1.1 CorGE

usage: CorGE [-h] {collect_genomes,extract_genes} ...

positional arguments:

{collect_genomes,extract_genes}

Subcommands

collect_genomes Collect nucleotide- and protein-encoded genomes of interest extract_genes Extract SCCGs from all collected genomes and curate data for tree building

2.1.2 CorGE collect_genomes

usage: CorGE collect_genomes [-h] [–all] [–ncbi_species NCBI_SPECIES] [–ncbi_accessions NCBI_ACCESSIONS] [–local LOCAL] [–outgroup OUTGROUP] [-n] output_dir

positional arguments:

output_dir Directory to collect genomes in

options:

-h, --help	show this help message and exit
--all	Collect one representative genome from each species listed in NCBI's RefSeq database. Don't use this with –ncbi_species, –ncbi_accessions, or –local

--ncbi_species NCBI_SPECIES File listing species level taxon ids to be collected from NCBI

--ncbi_accessions NCBI_ACCESSIONS File listing genome accessions to be collected from NCBI

--local LOCAL Directory containing nucleotide- and protein-encoded pairs of genome files. Any unpaired files will be ignored

--outgroup OUTGROUP Specify the outgroup for tree rooting. Integers will be parsed as species level taxon ids and retrieved from NCBI. Otherwise will search for a matching nucleotide-encoded file in ouput_dir or local (Default: 2173, enter None to not use outgroup rooting)

-n Dry run, show what would be gathered but don't do it

2.1.3 CorGE extract_genes

usage: CorGE extract_genes [-h] [-o OUTPUT] [-t {prot,nucl}] [-n {acc,txid,strain,species}] genomes

positional arguments:

genomes Directory with collected genomes (curated with collect_genomes)

options:

-h, --help show this help message and exit

-o OUTPUT, --output OUTPUT Directory to write output to (Default: ./)

-t {prot,nucl}, –type {prot,nucl}
Output in merged-sequences can be nucleotide- or protein-encoded (Default: prot)

-n {acc,txid,strain,species}, –name {acc,txid,strain,species}
Names to show on final tree (Default: txid)

2.2 collect_genomes

This is the command for retrieving genomes for all of the bacteria you want in your tree, from NCBI, a local directory, or both. *CorGE* does all of its work with pairs of files: a protein-encoded genome (usually saved with a *.faa* extension) and a nucleotide-encoded genome (usually saved with a *.fna* extension). These files have the same name but different extensions with the name for any NCBI files being the genome accession (e.g. "GCF_000218545") and the name for any local files being preserved.

Tip: Especially when downloading lots of genomes, it's always a good idea to run your command with the dryrun option (*-n*) first. This will print out what it's planning to do without doing it, so you can verify that it will do the right thing.

2.2.1 Common Use Cases

To collect a set of representative/reference genomes for a list of species, create a file of species-level taxon ids, one id per line, and pass that file to *CorGE*:

```
CorGE collect_genomes . --ncbi_species species_list.txt
```

Tip: If you don't care about rooting the final tree, you can specify *–outgroup None*. The pipeline will still use a midpoint algorithm to root the final tree, but the input to that step will be the unrooted tree.

To collect one genome for each species NCBI has, use the *–all* option:

```
CorGE collect_genomes /path/to/db --all
```

Suppose you want to create a strain level tree from some existing NCBI E. coli genomes and some that you have locally and then root that tree with a reference Clostridium botulinum genome. You create a list of the genome accessions you want to collect (same as species taxa, one accession per line in a text file) and run:

```
CorGE collect_genomes ecoli-db/ --ncbi_accessions accession_list.txt
```

This will collect each of those genomes and put them in *ecoli-db/* (as well as grabbing Methanobrevibacter smithii in the *outgroup* dir, this will be overwritten by the next step). Next we need to get all the local files in there, but we need them to follow a couple rules: 1) only pairs of files will be collected so every nucleotide file should have a paired protein file with the same name (e.g. *example.fna* and *example.faa*) and 2) the annotations for the protein files should have a unique name as the first space-separated piece of the annotation and then corresponding sequences in the nucleotide files should have annotations that contain that name (e.g. first protein sequence is annotated with > *example00001 description and so on* and corresponding nucleotide sequence is annotated with > *WXK40_example00001_DEADBEEF*). Provided they are in compliance with the above and all in one directory run:

```
CorGE collect_genomes ecoli-db/ --local assembled-genomes/ --outgroup 1491
```

This should leave you with all your files organized into *ecoli-db/protein*, *ecoli-db/nucleotide*, and *ecoli-db/outgroup* directories.

Tip: You could also do this all in one command *CorGE collect_genomes ecoli-db/ –ncbi_accessions accession_list.txt –local assembled-genomes/ –outgroup 1491*

2.3 extract_genes

To curate genes for a multi-species bacteria tree run:

```
CorGE extract_genes /path/to/db
```

Continuing the example from the last section, to curate genes for this strain level E. coli tree, run:

```
CorGE extract_genes ecoli-db/ --type nucl --name strain
```

This will prepare you to build a nucleotide based tree of E. coli strains where the leaf names will be either the strain name (if it came from NCBI) or the file name (if it was local).

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search