WORKFLOW FOR GENERATING A QIIME-COMPATIBLE BLAST DATABASE FROM AN NCBI ENTREZ/GQUERY SEARCH

CHRIS BAKER*

7 OCTOBER 2016

This document describes how to take a FASTA file, plus the NCBI taxonomy database, and generate the id-to-taxonomy mapping file that you need to BLAST your metabarcode data against the FASTA file sequences using QIIME. The Python code described in Section 3 relies on being able to extract NCBI accession numbers (not GI numbers) from your input FASTA file to match against the taxonomy database, so the FASTA file needs to be correctly formatted. The code is designed to work with FASTA files downloaded from the NCBI database (e.g. through a gquery search), but should work fine as long as your FASTA file contains appropriate accession numbers formatted in the same way.

1 Obtain FASTA file of sequences to BLAST against

The first step is to obtain the sequences that will form your database. To download sequences from the NCBI, go to `http://www.ncbi.nlm.nih.gov/sites/gquery` and execute a search for the sequences you ultimately want to BLAST against. For example, you might search for

```
aftol AND "internal transcribed spacer 1"
```

which (as at 7 October 2016) finds 795 nucleotide results. Click on `Nucleotide` to show these results. Then, at the top of the screen, click `Send to:`. Select `File`, then format `FASTA`, then `Create File`. The file should download as something like `sequence.fasta.txt`.

The FASTA file that gquery creates for nucleotide sequences has deflines of the form

```
>XX######.# ...
```

where `XX######.#` is the NCBI accession.version number that uniquely identifies this sequence, and this is separated by a space from any other information on the defline.

Any FASTA file that fits this format can be used with the Python script `entrez_qiime.py` described in Section 3 below. i.e. the file need not be created by gquery, so feel free to obtain it from other sources, as long as each sequence has a suitable accession number. Any additional information on the defline is ignored by `entrez_qiime.py`, as is the sequence data itself – so you could, for example, download a file of sequences and annotate it or delete irrelevant portions of the file before running it through the rest of this workflow.

Note that, as an alternative to a FASTA file, `entrez_qiime.py` will also take a plain list of accession numbers using its `-L` option, one per line:

```
XX######.#
XX######.#
XX######.#
   ...
```

* Department of Ecology and Evolutionary Biology, Princeton University. Email: ccmbaker@princeton.edu. This PDF accompanies entrez_qiime.py v2.0. This update has been written to accession numbers instead of GI numbers in the input files. For older input files with GI numbers, see v1.0 of this workflow.

This might be useful if you already have a BLAST sequence database and have a list of the accession numbers for the sequences (perhaps because you used that list to filter or mask a larger database).

2 NCBI taxonomy database files

If you do not already have a local copy of the NCBI's taxonomy data, you will need to download it. Download links are available from `http://www.ncbi.nlm.nih.gov/guide/taxonomy/`. The files can also be downloaded directly from the command line, e.g. using Terminal on a Mac:

```
# approx 37MB
ftp ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/taxdump.tar.gz
tar -zxvf taxdump.tar.gz

# approx 900MB
ftp ftp://ftp.ncbi.nih.gov/pub/taxonomy/accession2taxid/nucl_gb.accession2taxid.gz
gunzip nucl_gb.accession2taxid.gz
```

The files you will need from `taxdump.tar.gz` are `nodes.dmp`, `names.dmp`, `merged.dmp` and `delnodes.dmp`. The other file, `nucl_gb.accession2taxid.gz`, should unzip to `nucl_gb.accession2taxid`.

Every sequence in GenBank is identified by a unique accession number. Each accession number is associated with a TaxonID number, indicating the organism that the sequence comes from. These associations are listed in `nucl_gb.accession2taxid` for nucleotide sequences (excluding EST, GSS, WGS or TSA nucleotide sequences – these are covered by the analogous files `nucl_est.accession2taxid.gz`, `nucl_gss.accession2taxid.gz`, `nucl_wgs.accession2taxid.gz` and `nucl_gb.accession2taxid.gz`, and you should download those files instead if appropriate for your gquery data). Every node in the taxonomy database's tree-of-life has a unique TaxonID, but several sequences may share the same TaxonID e.g. if they come from the same species. To describe the topology of the tree, `nodes.dmp` lists every TaxonID and, for each one, gives the TaxonID for the parent node. Finally, `names.dmp` provides the taxonomic name for each node in the tree.

3 Run entrez_qiime.py

The Python script `entrez_qiime.py` takes your FASTA file (or accession number list) and the five taxonomy files described above as inputs. From these the script generates the accession-to-taxonomy file required by QIIME's `assign_taxonomy.py`, with each line containing an accession number separated by a tab from a semicolon-delimited list of taxonomic names for that accession:

```
XX######.#    phylum;class;order;family;genus;species
XX######.##   phylum;class;order;family;genus;species
XX######.#    phylum;class;order;family;genus;species
   ...            ...
```

The script also generates a log file that may include details of discrepancies that the script had to deal with in processing your input files.

On my Mac, you can run the script as follows[1]:

```
python entrez_qiime.py [options]
```

where the options are the following:

[1]My laptop has Python 2.7.10 installed, and this would commonly be installed by default on Macs. The PyCogent library and its dependency NumPy are also installed. These are less common: install them using `pip` or similar if required.

`-i`, `--inputfasta` The path, including filename, of an input FASTA file with accession number deflines as described in Section 1. [Required, unless an accession number list is supplied with `-L` or `--List`. If both are supplied, the script fails.]

`-L`, `--List` The path, including filename, of an input accession number list, one accession number per line as described in Section 1, as an alternative to supplying a FASTA file. [Required, unless a FASTA file is supplied with `-i` or `--inputfasta`. If both are supplied, the script fails.]

`-o`, `--outputfile` The path, including filename, for the accession-to-taxonomy output file as required by QIIME. Will be created if it does not exist. Defaults to same directory as input FASTA or list file, with file name derived from the input file's name, if not supplied here.

`-g`, `--outputlog` The path, including filename, for the output log file. Will be created if it does not exist. Defaults to same directory as input FASTA or list file, with file name derived from the input file's name, if not supplied here.

`-f`, `--force` If `-f` or `--force` option are passed, output files will overwrite any existing files with the same names. Without this option, output filenames will be modified by the addition of a date/time string in the event that a file with the same name already exists, so that no files will be overwritten. [Default: False]

`-n`, `--nodes` The directory where the files `nodes.dmp`, `names.dmp`, `merged.dmp` and `delnodes.dmp` are located. The files are typically downloaded from the NCBI in the compressed archive `taxdump.tar.gz`. [Default: `./`]

`-a`, `--acc2taxid` The path, including filename, of the (uncompressed) input accession number - taxid file, typically downloaded from the NCBI as `nucl_gb.accession2taxid.gz` and uncompressed to `nucl_gb.accession2taxid`. [Default: `./nucl_gb.accession2taxid`]

`-r`, `--ranks` A comma-separated list (no spaces) of the taxonomic ranks you want to keep in the taxon names output. Ranks can be provided in any order, but names output will be generated in that order, so should typically be in descending order. Any taxonomic names assigned to one of the ranks in the list will be retained; other names will be discarded. If a taxon has no name for one of the ranks in the list, then NA will be inserted in the output. No sequences or taxa will be discarded – this option only affects the names that are output for each taxon or sequence.The following ranks are available in the NCBI taxonomy database as at 7 October 2016 (in alphabetical order): class, family, forma, genus, infraclass, infraorder, kingdom, no_rank, order, parvorder, phylum, species, species_group, species_subgroup, subclass, subfamily, subgenus, subkingdom, suborder, subphylum, subspecies, subtribe, superclass, superfamily, superkingdom, superorder, superphylum, tribe, varietas. Note the use of underscores in three of the names. You should include these underscores on the command line but they will be replaced by spaces when the script runs so that the names match the NCBI ranks. [Default: phylum,class,order,family,genus,species]

`-h`, `--help` Print this help information.

The two output files will be saved wherever you specify with arguments `-o` and `-g`. If these arguments are not supplied, files will be saved in the same directory as the input FASTA or list file supplied with `-i` or `-L`. In this case, given the input file `somepath/filename.extn`, output files `somepath/filename_accession_taxonomy.txt` and `somepath/filename.log` will be saved. If these filenames conflict with any existing files, then the output filenames will be modified by the addition of a date-time string, unless `-f` is passed in which case those existing files will be overwritten.

Sometimes the script may encounter small discrepancies between the various input files. For example, your input FASTA file may contain accession numbers with outdated TaxonIDs in `nucl_gb.accession2taxid` if two taxa have been merged. Or your FASTA file may contain sequences that have only recently been uploaded to GenBank and have not yet made it into `nucl_gb.accession2taxid`. Check the log file for details of any changes made to deal with such discrepancies.

The script will take several minutes to run, even with a small input FASTA file, since it takes some time to load the NCBI's taxonomy data. Running time also increases with the size of the input FASTA file.

4 Generate BLAST database

The QIIME script `assign_taxonomy.py`, used with the option `-m blast`, needs a BLAST database to search in order to assign taxonomy to your metabarcode sequences. If you started with a FASTA file as described in Section 1, you may supply that directly as an input using the `-r` or `--reference_seqs_fp` option, and `assign_taxonomy.py` will convert this to a BLAST database for you as it runs. Alternatively, if you already have a formatted BLAST database, then you can supply this to `assign_taxonomy.py` using the `-b` or `--blast_db` options. In these cases, you can skip the remainder of this section and proceed directly to running your BLAST search as described in Section 5.

However, if you have a FASTA file and wish to reformat it as a BLAST database before running `assign_taxonomy.py` – e.g. because you plan to run `assign_taxonomy.py` more than once and would like to avoid having to convert it on the fly each time – then this conversion can be done using the NCBI's command-line `makeblastdb` utility something like this:

```
makeblastdb \
-in somepath/sequence.fasta \
-dbtype nucl \
-title 'a title for your database' \
-out yourblastdbpath/yourblastdbname \
-max_file_sz '1GB'
```

This will save a handful of files called `yourblastdbpath/yourblastdbname.???` that together comprise the sequences in your FASTA file, but formatted appropriately for BLAST. You could now BLAST against that database using any of the usual BLAST command line tools, such as `blastall`. You can give the database any title you want, and it may be useful to be fairly descriptive. The name `yourblastdbname` is used in naming the output files, and will also be used later to refer other programs (like QIIME or `blastall`) to your database, just like you might use `nt` or `nr` to refer to the entire nucleotide database, so it is probably useful to keep that name short.

5 Run BLAST search in QIIME

You should now have all the files you need to BLAST your metabarcode data against the sequences you obtained through your gquery search. If you are supplying a FASTA file for `assign_taxonomy.py` to convert to a BLAST database on the fly, then you will run it something like this, e.g. on our Linux cluster:

```
module load bio/qiime-1.9.1
assign_taxonomy.py \
-i yourHTSdata.fasta \
-t sequence_accession_taxonomy.txt \
-m blast \
-r sequence.fasta
```

where the file `yourHTSdata.fasta` is a FASTA file containing your high-throughput sequence data (i.e. your metabarcodes that you want to identify, most likely a file of representative sequences, one for each OTU), `sequence_accession_taxonomy.txt` is the taxonomy mapping file output by `entrez_qiime.py`, and `sequence.fasta` is the FASTA file used to generate the taxonomy mapping file.

If you are supplying a BLAST database directly to `assign_taxonomy.py` (as opposed to an input FASTA file) you should first ensure that your new BLAST database is in QIIME's BLAST database path as determined by the BLASTDB variable. One way to do this is to check QIIME's BLAST database path, e.g. like this on our Linux cluster

```
module load bio/qiime-1.9.1
echo $BLASTDB
```

and move your BLAST database so it is in an appropriate location. Alternatively, you could temporarily change the contents of BLASTDB to reflect where your BLAST database is, for example like this

```
module load bio/qiime-1.9.1
BLASTDB_OLD=$BLASTDB    % save old BLASTDB so you can change it back later
BLASTDB=yourblastdbpath/
```

However you choose to make your BLAST database available, you can then run `assign_taxonomy.py`:

```
assign_taxonomy.py \
-i yourHTSdata.fasta \
-t sequence_accession_taxonomy.txt \
-m blast \
-b yourblastdbname
```

Finally, if you altered BLASTDB prior to running `assign_taxonomy.py`, you can now change it back:

```
BLASTDB=$BLASTDB_OLD    % change BLASTDB back to original value
unset BLASTDB_OLD
```

6 Further information

This work may be cited as follows:

Baker, Christopher CM (2016). entrez_qiime: a utility for generating QIIME input files from the NCBI databases. github.com/bakerccm/entrez_qiime release v2.0. 7 October 2016 (DOI: xxxxxxxxxx).

Please contact Chris Baker at `ccmbaker@princeton.edu` for queries about this documentation or the accompanying Python code.