Latest version of this document can be obtained at https://github.com/JeremyCoxBMI/IMSA-A/blob/master/IMSA%2BA_Detailed_Directions.pdf

## December 2016 Update

With NCBI switching from GI numbers to Accession.Version numbers, the program must be updated when a new database is adopted. On a functional level, this change means the NCBI uses different codes to name reference sequences. This difference is cosmetic, but is a pain for IMSA+A, because we have to be compatible with both.

The new version includes GI number compatible script, as well as an Accession.Version compatible script.

When using a database with Accession.Version names, use the **postprocesscount4acc.py** script in place of **postprocesscount4.py** script. They are identical, except they reference the different databased for each number type. Please note that both versions of the script require downloading a look-up file (an NCBI database dump (.dmp) file to convert numbers to taxon ID.

Obviously, if you want to run only one, you need not download the other. The directions include both for completeness.

All the example databases distributed with the paper use GI numbers, so you may want to install both databases.

## Installation

IMSA+A protocol uses the following software, which must be installed prior to use

*Programs*

IMSA http://sourceforge.net/projects/arron-imsa/

IMSA+A https://github.com/JeremyCoxBMI/IMSA-A

IMSA+A files are placed in the imsa/ directory, where imsa was installed

Also, you may wish to download and evaluate low.bacteria.coverage.fa as an example.

BLAST+ https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download

(Hereafter referred to as simply BLAST)

BLAT https://genome.ucsc.edu/FAQ/FAQblat.html

Instead of BLAT, try using pBLAT (for multiple cores, see below) http://icebert.github.io/pblat/

BOWTIE2 http://bowtie-bio.sourceforge.net/bowtie2/index.shtml

(IMSA v2 uses BOWTIE2, original IMSA uses BOWTIE)

Oases https://github.com/dzerbino/oases

velvet https://github.com/dzerbino/velvet

(Installs automatically when oases installs)

Trinity https://github.com/trinityrnaseq/trinityrnaseq/wiki

*Databases*

| | | |
|---|---|---|
| NCBI taxa dump | taxdump.tar.gz | ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz<br><br>https://figshare.com/s/9e62995bf520e1ac0fc8#/articles/3482813 |
| GI taxa list | gi_taxid_nucl.dmp.gz | ftp://ftp.ncbi.nih.gov/pub/taxonomy/gi_taxid_nucl.dmp.gz |
| Accession Version taxa list | nucl_gb.accession2taxid.gz | ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/accession2taxid/nucl_gb.accession2taxid.gz |
| Custom BLAST Database | NCBI.fungiDBselect.genome.tar | https://drive.google.com/file/d/0B53eFJKst-koSzNtSVlGSllzcTQ/view?usp=sharing<br><br>https://figshare.com/s/9e62995bf520e1ac0fc8#/articles/3482825 |
| Provided premade HUMAN Database GRCh37 | human.DB.tar | https://drive.google.com/file/d/0B53eFJKst-koOFBYSEZPbWdWY0E/view?usp=sharing<br><br>https://figshare.com/s/9e62995bf520e1ac0fc8#/articles/3482822 |

You need a BOWTIE, BLAT, and BLAST version of your host genome and BLAST version of your Microbiome metagenome. We provide a precompiled Microbiome metagenome, which we used in our paper. We provide compiled version of human genome (GRch37) in BLAT, BOWTIE, and BLAST formats.

To make your own Microbiome Metagenome, you must collect FastA sequence files for the genomes you wish to include and then compile the database using makeblastdb.

```
makeblastdb -in *.fa  -dbtype 'nucl' -out metagenome -title
"My Microbiome metagenome"
```

When adding reference sequences without Genbank id (GI) number in the sequence name, e.g. “gi|158333233|”, you must provide IMSA a list of corresponding taxon IDs. (2016-12 update: Accession Version numbers are acceptable using the modified script, and need not be added to this file.). The **fungiDB.seqNames.2.speciesID** tab delimited file is used for sequence names for the fungiDB.org genomes. Append your entries as needed to this file; any reference sequence name of non-standard format can be used. Note that the sequence name is between “>” and first whitespace of the line. All other text is ignored.

Example FastA lines, sequence name shown in **dark red**:

```
>gi|158333233|ref|NC_009925.1| Acaryochloris marina MBIC11017 chromosome, complete genome
```

```
>ASPAC_10177 | organism=Aspergillus_aculeatus_ATCC_16872 | product=Actin regulatory ...
```

While BLAST databases can be found pre-compiled by NCBI, you will have to compile your own BOWTIE and BLAT databases. We have included these for Human genome so you can get a quick-start.

I recommend using subdirectories ncbi_gi_taxid/ blast/ blat/ bowtie/ in the imsa directory.

You will need to know the absolute path (starting with / root) to your databases for the configuration process.

IMSA+A postprocessing script uses local NCBI taxa dump database to convert gi numbers to taxonomies, because internet access of the database is unreliable (in our experience). Path must be configured in **imsa/systemSettings.py**.

## IMSA+A

Download all the files listed under Databases above, as well as the IMSA+A tar file to your home directory.

Extract the imsa-a.tar file. Detailed bash example for a default installation is included below.

### IMSA install & configuration

Download the database files and install programs if not yet complete. Here, we assume you are installing to your home directory (~/), but anything path is acceptable.

Once these databases are created, IMSA must be configured to locate the programs and databases by editing **imsa-a/config.py**.

```
First, download files to the install directory:
gi_taxid_nucl.dmp.gz  (for GI number database)
nucl_gb.accession2taxid.gz (for Accession number database)
human.DB.tar
imsa-a.tar
NCBI.fungiDBselect.genomeonly.tar
taxdump.tar.gz

COMMANDS:
gunzip taxdump.tar.gz
gunzip gi_taxid_nucl.dmp.gz
tar -xvf imsa-a.tar
cd imsa-a
mkdir human
mkdir microbiome
mkdir gi_taxid
mkdir accession
cd human
tar -xvf ../../human.DB.tar
cd ../microbiome
tar -xvf ../../NCBI.fungiDBselect.genomeonly.tar
cd ../gi_taxid
tar -xvf ../../taxdump.tar
mv ../../gi_taxid_nucl.dmp ./
touch extra_gi_taxid_nucl.dmp
cd ../accession
mv ../../ nucl_gb.accession2taxid.gz ./
gunzip nucl_gb.accession2taxid.gz
touch extra_acc_taxid_nucl.dmp
cd ..
kwrite config.py &

This opens an editor to change settings in the config.py file.
Changes are reflected in blue.  Note that your username must be changed to your actual username in
the paths below.


# BOWTIE_DATABASES specifies where the ebwt index lives for each bowtie database you want to use in
# IMSA.  The path should be the full path that you would use on a bowtie command line when referencing
# the ebwt file.  BOWTIE_DB_ABBREV links the name of the database to the short code used internally.
BOWTIE_DATABASES = {"human":"/home/username/imsa-a/human/bowtie/GRCh37"}
BOWTIE_DB_ABBREV = {"human":"hg"}
```

```
# BLAT_DATABASES specifies where the 2bit blat index lives for each blat database you want to use in
# IMSA.  The BLAT_OOC_FILES list the ooc file for the database, if it exists.
BLAT_DATABASES = {"human":"/home/username/imsa-a/human/bowtie/GRCh37.2bit"}

BLAT_OOC_FILES = {"human":"/home/username/imsa-a/human/bowtie/GRCh37.11.ooc"}
BLAT_DB_ABBREV = {"human":"hg"}

# BLAST_DATABASES list where each blast database lives for the databases you want to use within imsa.
BLAST_DATABASES = {"nt":"/home/username/imsa-a/microbiome/NCBI.fungiDBselect.genomeonly",
                   "human":"/home/username/imsa-a/human/blast/GRCh37"}
BLAST_DB_ABBREV = {"nt":"nt", "human":"hg",
                   "humanRNA":"hr", "humanRNA-1":"hr1", "humanRNA-2":"hr2", "humanRNA-3":"hr3",
                   "humanRepeat":"rep"}
BLAST_TAX_DB = "/home/username/imsa-a/gi_taxid/gi_taxid_nucl.dmp"


If these programs are not in your PATH, you will have to include the path here.
PATH_TO_BOWTIE = "bowtie"
PATH_TO_BOWTIE2 = "bowtie2"
PATH_TO_BLASTN = "blastn"
PATH_TO_BLAT = "blat"
```

## IMSA+A configuration

Next, edit the file systemSettings.py

Changes are reflected in blue:

```
# LOCAL DATABASES
PATH_TO_OASES="/path/to/oases/"
OASES_SCRIPT=PATH_TO_OASES+"/oases/scripts/oases_pipeline.py -m 19 -M 33 -o singleEnd -d "

# NCBI database

#GID based alignment database (old / reverse compatible)
PATH="/home/username/imsa-a/gi_taxid/"
BLAST_TAX_DB = PATH+"gi_taxid_nucl.dmp"
NAMES_BLAST_TAX_DB = PATH+"names.dmp"
NODES_BLAST_TAX_DB = PATH+"nodes.dmp"

# This file allows you to manually define Accesion numbers to Taxon ID conversion, if it somehow was left out of
# the database dump, and ISMA then throws an error.
# Add to this tab-delimited file lines matching the format of the BLAST_TAX_DB file:
# 2 columns: GI number, taxonID
# ACTION ITEM: you need to create an empty file until there is data to put in this file
EXTRA_BLAST_TAX_DB = PATH+"extra_gi_taxid_nucl.dmp"             #note, you may need to create an empty file

#ACCESSION number based alignments database (new)
ACC_PATH="/home/username/imsa-a/gi_taxid/accession"
ACC_BLAST_TAX_DB = ACC_PATH+"nucl_gb.accession2taxid"
ACC_NAMES_BLAST_TAX_DB = PATH+"names.dmp"
ACC_NODES_BLAST_TAX_DB = PATH+"nodes.dmp"

...

fungiLookupFile = "/home/username/imsa-a/fungiDB.seqNames.2.speciesID"
```

`fungiDB.seqNames.2.speciesID` contains mappings for Reference Sequence names to taxon ID representing species for sequence names that do not contain gi numbers or accession numbers. The file is tab delimited. If using a database of your own construction, you may add entries to this file and IMSA+A will recognize the sequence names. Note that in a FastA file, the sequence name is between “>” and the first whitespace. Anything else on the line is not part of the sequence name.

# Running IMSA+A

To run IMSA with Alignment (IMSA+A), the protocol contains 4 steps. Recommended resources are for 70 million reads.

IMSA+A takes single end RNA sequencing shotgun reads, typically rRNA depleted, converted to cDNA format. IMSA+A cannot handle paired end reads without modification. However, IMSA+A can take paired end reads as single end reads and process them without utilizing the paired end information.

**Step 1.**

**Run IMSA without the BLAST metagenome alignment. In your action file, remove the last step (see Fig 2)**

Recommended settings:

100 hour wall time, 8 CPU, 25 GB RAM. Data with high mutation rate requires more RAM..

```
python /path/to/imsa/master.py -i myFile.fa -p action_script.txt
python pipelinescript.py
```

(Step 1 does data preprocessing and multiple subtraction steps by your design. If you choose, you do not need to use IMSA. You can use an alternate subtraction pipeline or do the subtraction manually. Perhaps you or your sequencing core already did a subtraction! Note that the input to next step must be either fastq or fasta.)

```
quality                                          # for fastq files only, remove for fasta
bowtie human
removeN
blat human | -fastMap
blat human
blast human maxEval=1e-15| -word_size 24
blast human maxEval=1e-8
blast nt maxEval=1e-15 | -max_target_seqs 200    # remove this step
```

**Figure 2**. Example IMSA script (action_script.txt). To run IMSA+A, remove the metagenome alignment step and run as normal.

**Step 2.**

**Using the default settings, run assembly of shotgun reads using the oases pipeline and inchworm.**

*Run these at the same time if you want to compare results! Alternatively, just run Oases.*

Oases:

Recommended settings:

12 hour wall time, 1 CPU, 20 GB RAM. Junk reads inflate the memory usage.

```
export PATH=$PATH:/path/to/oases/velvet:/path/to/oases/oases
python /path/to/oases/oases/scripts/oases_pipeline.py -m 19 -M 31 -o singleEnd \
      -d " myFile_human_n5_lhg_lhg_shg_shg.fa "
```

/\ | Note the spaces inside quotes are required | /\

Note that the IMSA final filename may be different if IMSA ran different steps.

Inchworm:

Recommended settings:

1 hour wall time, 6 CPU, 20 GB RAM.

```
/path/to/trinity/Trinity --seqType fa --no_run_chrysalis --JM 9G --single \
        myFile_human_n5_lhg_lhg_shg_shg.fa --CPU 8
```

**Step 3.**

BLAST the assembled contigs against your metagenome database.

You can use a one line action file in IMSA or you can do it manually from BASH.

*Run these at the same time!*

For Oases assemblies:

Recommended settings:

2 hour wall time, 8 CPU, 15 GB RAM.

**one line action file:**

```
blast nt maxEval=1e-15 | max_target_seqs 200 -num_threads 8
```

commands:

```
python /path/to/imsa/master.py -i singleEndMerged/transripts.fa -p action_script2.txt
python pipelineScript.py
```

**manual commands:**

```
DATABASE_FILE=/home/username/imsa-a/microbiome/NCBI.fungiDBselect.genomeonly
/path/to/blast/blastn -max_target_seqs 200 -num_threads 8 -evalue 1e-15 -outfmt 6 \
       -db $DATABASE_FILE -query singleEndMerged/transripts.fa -out oases_snt.bln
```

For Inchworm assemblies:

Recommended settings:

2 hour wall time, 8 CPU, 15 GB RAM.

**one line action file:**

```
blast nt maxEval=1e-15 | max_target_seqs 200 -num_threads 8
```

commands:

```
python /path/to/imsa/master.py -i trinity_out_dir/inchworm.K25.L25.DS.fa \
      -p action_script2.txt
python pipelineScript.py
```

**manual commands:**

```
DATABASE_FILE=/path/to/databases/NCBI.fungiDBselect.genomeonly
/path/to/blast/blastn -max_target_seqs 200 -num_threads 8 -evalue 1e-15 -outfmt 6 \
        -db $DATABASE_FILE -query trinity_out_dir/inchworm.K25.L25.DS.fa  \
       -out inchworm_snt.bln
```

**Step 4.** **Run the upgraded IMSA+A postprocessing script.**

Recommended settings:

2 hour wall time, 1 CPU, 15 GB RAM.

```
python /path/to/imsa/postprocesscount4.py -b oases_snt.bln

python /path/to/imsa/postprocesscount4.py -b inchworm_snt.bln
```

Open results as spreadsheet: inchworm_snt.tax_genus.IMSA+A_4count.txt
oases_snt.tax_genus.IMSA+A_4count.txt

See "Output files" section below to understand the output files.

For virus detection, use instead: inchworm_snt.tax_firstTaxon.IMSA+A_4count.txt
oases_snt.tax_firstTaxon.IMSA+A_4count.txt

# Using and Understanding Output

### *Impact of Reference Sequence Database*

NCBI sequence database contains several versions, from most liberal and most inclusive to most conservative and least inclusive: NCBI Refseq database is the largest, next nucleotide (NT) database, and finally the unofficial genomes-only database. The first principle of reference databases is that if the matching sequencing is not in the database, then the search will not report alignment to that sequence. Therefore, one would be biased towards using the most liberal database. However, results indicate that quality of reference sequence data is more important than quantity. We recommend building and using a Metagenome database using genomes only. Augment the database with reference sequences for organisms and viruses expected to be found, which are not in this database. Note that IMSA only works with reference sequences which contain Genbank id number in the sequence name, e.g. "gi|158333233|". (2016-12 update: Accession Version numbers are acceptable using the modified script). (See **Installation** for instructions to use databases without Genbank id, such as the custom database.)

Note that if plasmids are included in the database, they will be reported as sequences belonging to the corresponding bacterium they were found in, not as plasmids separately. One might desire for plasmids to be reported separately, as plasmids can be shared between organisms or differentiate between disease-causing bacteria or between levels of anti-biotic resistance. IMSA does not report these differences.

### *Output Files*

There are four primary output files, which end in "IMSA+A_4count.txt". Each of four reports summarize the counts at the four clade levels: firstTaxon (the lowest taxon corresponding to the reference sequence), species, genus, and family.

The first four columns of these files are the taxon ID, taxon Name, IMSA+A count, and kingdom. This count is the number of assembled sequences that align to this taxon as the single best hit. The next four columns break the full IMSA counting score into its components. Typically, you ignore these last 4 columns. If using EXCEL, I recommend sorting largest to smallest on the IMSA+A count column.

Columns

| | |
|---|---|
| Taxa ID | NCBI taxon database primary key. |
| Scientific Name | The scientific name for the clade as reported by NCBI. (There are multiple name forms in the database, so using the Taxa ID is imperative to prevent ambiguity.) |
| IMSA+A count | This is the unique hits calculated at the file's clade level. Identical to subsequent column. |
| Kingdom | Kingdom reported by NCBI Taxonomy tree. If kingdom is not available, the super kingdom. Note that in rare cases, this will be "root", meaning that neither could be found. |
| Total | The original IMSA counting method. The sum of unique clade hits and partial clade sum. |
| Unique clade hits | Number of queries which have best hits only to this taxon. |
| Partial clade hits | Number of sequences with ambiguous alignments. |
| Partial clade sum | The sum of partial scores attributed by ambiguous alignments. |

Files ending in "IMSA_count.txt" are the original IMSA output files. Note that the information here is redundant with the information in the IMSA+A output files.

Additional intermediate files are generated, which may contain useful information. See "Additional Output Files" below for more information.

### *Role and Comparison of Two Assemblers*

Short sequences (35 – 75bp) are difficult to use for taxa classification, because a short sequence gives a lower confidence score in alignment, which ultimately hurts classification performance by leading to increased false positives or decreased true positives. Furthermore, BLAST search time increases. BLAST search diverges and then short circuits on unproductive search paths. With short sequences, more of the search paths go to completion. BLAST searches are computationally intensive, therefore minimizing number of BLAST alignments is the single most important thing to reduce computation time for IMSA+A.

In this work, two de novo assembler programs were used to assemble microbiome RNA shotgun reads with the aim of reconstructing putative genes. This reduces the data from many sequences to few contigs. This reduces computational complexity in the bottleneck as well as improving the information content of the data by increasing length of sequences and removing noise.

Inchworm assembler is conservative and does not use an intelligent algorithm. Inchworm decomposes all reads into a set of subsequence k-mers, default k = 25. The most frequent k-mers are extended by comparison to the other k-mers; this iterates over all k-mers (from most to least frequent) until the k-mers are used up. Inchworm is conservative, because only when there is an exact k-mer match will the assembly extend. Oases applies additional steps of merging multiple assemblies and using an intelligent algorithm. Oases will build assemblies based on different k-mer lengths by extension similar to Inchworm, and then works to combine the assemblies. Oases applies a set of transcriptome-specific error corrections, a modified version of the TourBus algorithm from Oases' predecessor, Velvet. Inchworm will faithfully report all sequences found (within parameters specified), whereas Oases will eliminate unassembled sequences and some sequences due to error correction.

Due to the differences in calculation, IMSA+A with Oases is the more conservative classifier, whereas IMSA+A with Inchworm is the more liberal classifier. When the two methods agree, this is strong evidence that the organism is present. Organisms identified by IMSA+A with Inchworm but not IMSA+A with Oases will be a mixture of true positives and false positives; together a clearer picture of what is uncertain is established. Therefore, it is recommended to do both analyses. The two assemblies and subsequent analysis can operate with minimal cost: they can be executed in parallel and consist of small portion of total CPU time for pipeline. Disparate results serve as a warning sign that there is low coverage within a data set.

In most conditions, performance between two assemblers is similar. However, Inchworm is an order of magnitude faster. Oases, however, gives better overall classification performance. Therefore, a quick look can be made using Inchworm. If quality is desired, use Oases or both.

### *Interpreting Results*

Because the final counts are based on assembled contigs, and not shotgun sequences, the numbers will be greatly decreased, especially in contrast to the sequencing depth. Thousands of sequences could be represented by a single assembly, thus a single contig is much stronger evidence than a single shotgun read. This raises the question of how many unique alignments should be considered as acceptable evidence for the presence of the genus or species. Because any criterion for a unique count threshold would depend on experimental conditions, our analyses use the default at least 1 unique count. If counts are commonly low or suspicious, use the results utilizing the second assembler to put the results in context.

Another consideration is that due to conservation, determination at the species clade level may not be possible and has higher error. Therefore, the genera counts will often be a much less noisy and more useful report for bacteria and fungi detection. The first taxon report should be used for virus detection, as the viruses may fall out of species and genus reports due to incomplete taxonomic trees.

When the source organism is not contained in the database, alignments to a related organism in the database will likely be made. Therefore, IMSA+A will report a false positive, but only because the true match is not in its database. The unique count may also be deflated, because the sequences align to cousins equally well, resulting in ambiguous hits. Results with multiple related organisms achieving either only partial hits, or disproportionately many partial hits, is an indicator that the source organism is not in the database and cannot be identified. If misidentification is suspected, one may add the genome of organism to the BLAST database and repeat the metagenome alignment and final analysis. (Adding sequences to a BLAST database does not require re-computing the entire database, so these steps can be done relatively quickly.)

If low coverage causes signal to be too weak to be detected, the number of organisms reported by analyzing Oases assembly will be much less than those reported by Inchworm assemblies. When this occurs, organisms found by both methods should be considered true positives and those otherwise detected could be true positives or false positives.

**Upgrading Pipeline to use Parallelized BLAT**

BOWTIE and BLAST allow for using multiple threads to reduce time to completion. However, BLAT does not.

Parallelize blat (pBLAT) can be downloaded from http://icebert.github.io/pblat/

pBLAT allows the use of multiple threads to accelerate alignment time. This can significantly reduce the total time to completion for the analysis. pBLAT is identical in its syntax to BLAT (including the executable name is "blat"), so it can be substituted simply by designating the PATH or explicitly calling the pBLAT executable.

(Config.py provides options to enable parallel processing for BOWTIE and BLAST. However, if these are left at their default values, you can control this directly through the action script, which is helpful because you may wish to use a different number of processors as your needs or resources change.)

In imsa/config.py

```
PATH_TO_BLAT = "blat"
```

Set path for blat to point to the pblat directory and executable.

For example,

```
PATH_TO_BLAT = "/usr/local/pblat/1.1/bin/blat"
```

The following is an upgraded version of the action script to include parallelization to 8 threads. The commands for parallelization are shown in bold. Note that if parallelization is set up inside IMSA's config.py file, it does not hurt to be redundant here. Underlined command switches are to prevent long sequences (100bp +) from finding multitude of hits per sequence, increasing alignment time dramatically.

```
bowtie human | -p 8
removeN
blat human | -fastMap -threads=8
blat human | -threads=8
blast human maxEval=1e-15 | -word_size 24 -num_threads 8 -max_target_seqs 20
blast human maxEval=1e-8 | -num_threads 8 -max_target_seqs 20
```

## Additional Input Files

These files, which are configured in systemSettings.py, contain additional mappings of sequence names to taxon IDs. These files are used for two purposes:

1) To define alternate sequence names not using NCBI GI or Accession numbers
2) To define NCBI GI or Accession numbers omitted in NCBI download

| | |
|---|---|
| fungiDB.seqNames.2.speciesID | This file was created because we augmented our database with fungal genomes from FungiDB. This allows defining any reference sequence name to map to a Taxon ID.<br>Tab-delimited, 2 columns: sequenceName, taxon ID |
| extra_gi_taxid_nucl.dmp | On occasion, NCBI will remove a GI or Accession number from their database. In such cases, both the database dump files download and a efetch request to database will come up with no hits. If you get an error thrown that a key is not recognized, you can define GI number here.<br>Tab-delimited, 2 columns: GI number, taxonID |

| | |
|---|---|
| extra_acc_taxid_nucl.dmp | On occasion, NCBI will remove a GI or Accession number from their database. In such cases, both the database dump files download and a efetch request to database will come up with no hits. If you get an error thrown that a key is not recognized, you can define Accession number here.<br>Tab-delimited, 4 columns: Accession Number, Accession.Version, taxonID, GI number<br>Note you may leave the last column blank – but you need the correct number of tab characters on the line. |

## Additional Output Files

| | |
|---|---|
| BASENAME.tax_family.IMSA_count.txt<br>BASENAME.tax_genus.IMSA_count.txt<br>BASENAME.tax_species.IMSA_count.txt | These are the original IMSA output files, based on assembled sequences, included for completeness for reverse compatibility. |
| gi.counts.txt<br>or<br>acc.counts.txt | This file can be generated with a utility script (see below).<br>TSV file stores the counts by the reference sequence name. Typically this is the GI number, but recall IMSA+A allows using non-GI numbers for taxonomy conversion by defining the relationships in an extra file. These codes would be used here. Note that either "gi.counts.txt" or "acc.counts.txt" is generated depending on whether the database is defined by GI numbers or Accession numbers. Columns are labelled. These are the breakdown of counts in 4 column format, same as presented in the IMSA+A_4count.txt files. |
| tax_uniqueAlignmentsJWC.txt | TSVs file stores the relationship of the queryName to each of the four clades, if there is a unique hit to that clade.<br>3 columns: queryName, clade level, taxon ID |
| tax_unresolved_giJWC.txt<br>OR<br>tax_unresolved_accJWC.txt | If everything is working, this file should be empty (size 0).<br>If a taxon ID cannot be found for a reference sequence, because it is not in the local NCBI dmp files and cannot be found by querying the database by eUtils, this error will be logged here. This usually happens because of internet/proxy difficulties. NCBI usually does not remove old keys from the database. However, in the case of merged records they do remove the keys which were merged from database, which can cause this error. You can look up the sequence at https://www.ncbi.nlm.nih.gov/Taxonomy/. Then, you can add them to input data files (as described under unidentifiedTaxaAlignmentsJWC.txt) |
| tax_NAMES_JWC.txt | List of names of the values looked up via eUtils based on taxon IDs found corresponding to the alignments. Primarily for debugging. |
| unidentifiedTaxaAlignmentsJWC.txt | If everything is working, this file should be empty (size 0).<br>The program outputs blast alignments, which it could not process and obtain a corresponding taxon ID. This is rare, but provides a convenient way to see what is going on.<br>You can add these entries, with a converted taxon ID, to file defined by **systemSettings.EXTRA_BLAST_TAX_DB**, file defined by **systemSettings.ACC_EXTRA_BLAST_TAX_DB**, or **fungiDB.seqNames.2.speciesID**, as appropriate. Note that the first files is for GI numbers, the second is for Accession numbers, and the third is any other sequence name format. |
| tax_GIS_JWC.txt<br>or<br>tax_ACCS_JWC.txt | List of the GI numbers (or Accession numbers) which had a corresponding taxon ID identified. Note that gi.counts.txt contains this information as well, but this file is created first. The codes saved here can also be those defined in **fungiDB.seqNames.2.speciesID** file. |

| | |
|---|---|
| taxonomy.pickle[1] | A pickle is a python binary file. Because parsing the dmp files can take minutes, the program saves these look-ups in case the program is run again, which might happen if there is unexpected crash. |
| bestHits.bln[1]<br>bestHits.bln.pickle[1] | IMSA+A, like IMSA, parses blast output to form |

[1] These files are loaded instead of overwritten. These are intended to support resuming and save time from intensive calculations, when the program crashes. If you are restarting an analysis, for example by adding new alignments, you must delete these 3 files.

## Utility Scripts

We have written several helper scripts, which may be useful to you when looking through the data.

### *utilUniqueGiCount.py and utilUniqueAccCount.py*

These two scripts perform the same function: create a four column count by reference sequence name. This can give some guidance whether a specific reference sequence is getting a lot of unique hits, or if they are more spread out among sequences. We expect the latter, because assembly should include all sequences to make one putative gene.

```
python utilUniqueGiCount.py BASENAME.bestHits.bln > BASENAME.gi.counts.txt
```

### *utilUniqueHitsFastaFilter.py*

This script allows you to prepare a FastA list of query names that map to unique hits at a clade level. You can choose firstTaxa, species, genus, or family.

```
python utilUniqueHitsFastaFilter.py firstTaxa \
BASENAME_sub_assemble.fa BASENAME_sub_assemble.fa.tax_uniqueAlignments.txt \
BASENAME_sub_assemble.fa.firstTaxa.unique.fa
```

Where **BASENAME_sub_assemble.fa** is the name of the assembled sequences file.

### *Isolating hits to a taxon ID*

The NCBI Taxon ID numbers serve as a primary key for each taxon, so we can search these numbers in the unique alignment report to see which queries were uniquely aligned to that taxon.

```
grep -e "470" BASENAME.fa.tax_uniqueAlignments.txt | cut -f1
Locus_100_Transcript_10/42_Confidence_0.097_Length_480
Locus_139_Transcript_12/54_Confidence_0.013_Length_549
Locus_100_Transcript_29/42_Confidence_0.000_Length_222
Locus_139_Transcript_11/54_Confidence_0.067_Length_643
Locus_100_Transcript_25/42_Confidence_0.145_Length_322
Locus_100_Transcript_28/42_Confidence_0.065_Length_261
Locus_100_Transcript_8/42_Confidence_0.032_Length_241
Locus_100_Transcript_26/42_Confidence_0.065_Length_265
Locus_100_Transcript_9/42_Confidence_0.065_Length_262
Locus_139_Transcript_22/54_Confidence_0.080_Length_507
```

### *utilGetMissingNuccoreGID.py and utilGetMissingNuccoreAccession.py*

These scripts will take unidentified reference sequences, attempt to look the values up in the NCBI database, and update local database files. This is particularly helpful when running the script; if an error is detected, then it can be corrected and a second attempt made.

```
FA_FILE=your_filename_here.fa
python /home/username/imsa-a/postprocesscount4acc.py -b $FA_FILE.bln
missing=`wc -c $FA_FILE.tax_unidentifiedTaxaAlignments.txt | cut -d' ' -f1`
echo **********
echo "missing count = $missing   (additional steps if ($missing>0)"
if [ $missing -gt 0 ]
then
  program=/home/username/imsa-a/utilGetMissingNuccoreGID.py
  python $program $FA_FILE.fa.tax_unidentifiedTaxaAlignments.txt
  rm $FA_FILE.bestHits.bln.taxonomy.pickle
  python /data/aplab/cox1kb/imsa/postprocesscount4acc.py -b $FA_FILE.bln
fi
```

## Frequently Asked Questions

### *The taxon counting reports have a listed for a clade indicated by "none" or "not found in DB". What is this?*

If this happens in the firstTaxon report, this means that IMSA+A does not know what reference sequences matches too in the Taxonomy tree. NCBI updates its databases daily, so if you did not download all databases to work with IMSA+A on the same calendar day, then it is possible for there to be an error. IMSA+A strives to auto-correct, but experience teaches to prepare for the unexpected. Several of the additional output files can help you root out and solve this problem (see Additional Output Files section).

This is expected to happen in the species, genus, or family reports. The NCBI Taxonomy tree is incomplete, particularly for organisms that do not yet have a full classification. Viruses are almost always incomplete in this regard.

You can use grep to see what reference sequences generated this error and in what category by grepping for "-1":

```
[cox1kb@bmiclusterp2 1801.read1]$ grep -e "-1" BASENAME.fa.tax_uniqueAlignments.txt
Locus_1044_Transcript_2/3_Confidence_0.444_Length_153   genus   -1
Locus_1050_Transcript_4/4_Confidence_0.292_Length_191   family  -1
Locus_1366_Transcript_3/3_Confidence_0.316_Length_176   genus   -1
Locus_943_Transcript_2/3_Confidence_0.706_Length_239    genus   -1
```

If any of these sequences were unknown at the firstTaxon level, representing missing information in our databases, they would report a failure to find the reference sequence for all four clades.

```
$ grep -e "-1" BASENAME.fa.tax_uniqueAlignments.txt
Locus_1044_Transcript_2/3_Confidence_0.444_Length_153   firstTaxa   -1
Locus_1044_Transcript_2/3_Confidence_0.444_Length_153   species -1
Locus_1044_Transcript_2/3_Confidence_0.444_Length_153   genus   -1
Locus_1044_Transcript_2/3_Confidence_0.444_Length_153   family  -1
```

To see which organism is missing a family (from the first example), I do this command

```
$ grep "Locus_1050_Transcript_4/4_Confidence_0.292_Length_191" BASENAME.fa.tax_uniqueAlignments.txt
Locus_1050_Transcript_4/4_Confidence_0.292_Length_191   firstTaxa       525904
Locus_1050_Transcript_4/4_Confidence_0.292_Length_191   species 166501
Locus_1050_Transcript_4/4_Confidence_0.292_Length_191   genus   262406
Locus_1050_Transcript_4/4_Confidence_0.292_Length_191   family  -1
```

I look up 525904 at https://www.ncbi.nlm.nih.gov/guide/taxonomy/

The lineage is

*Lineage( full )*

> cellular organisms; Bacteria; Terrabacteria group; unclassified Terrabacteria group; Thermobaculum; Thermobaculum terrenum

This organism only has a super kingdom (Bacteria), and also a genus and a species (Thermobaculum terrenum). Therefore, it has -1 for unknown in the family clade.

### *The Kingdom for my organism is "root." What does this mean?*

This means IMSA+A could not determine the Kingdom.

### *I get an error in the Python script that says "Exception: The blat PSL file had no hits. Stopping IMSA execution."*

If preceding steps subtracted all the sequences, then BLAT finds no host sequences to subtract. The pipeline thinks the program did not run and quits.

In the error traceback, you will see a line like the one below. Take the highlighted file names and run the commands below.

```
pipelineUtils.filterUsingBlat("low.bacteria.coverage_chg_n5.fa",
"low.bacteria.coverage_chg_n5_chg.psl",  …
```

NOTE: depending on your steps, the file names WILL be different! You have to change the names accordingly

Run these commands

```
touch low.bacteria.coverage_chg_n5_chg.psl
```

```
ln -s low.bacteria.coverage_chg_n5.fa low.bacteria.coverage_chg_n5_chg.fa
```

Once `low.bacteria.coverage_chg_n5_chg.fa` is made, the pipeline will smoothly resume when you run it again.