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

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	ftp://ftp.ncbi.nih.gov/pub/taxonomy/gi_taxid_nucl.dmp.gz
	ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz
Custom BLAST Database	https://drive.google.com/file/d/0B53eFJKst-koSzNtSVlGSllzcTQ/view?usp=sharing
	NCBI.fungiDBselect.genome.tar
Provided premade HUMAN	https://drive.google.com/file/d/0B53eFJKst-koOFBYSEZPbWdWY0E/view?usp=sharing
Database GRCh37	human.DB.tar

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 number in the sequence name, e.g. "gi|158333233|", you must provide IMSA a list of corresponding taxon IDs. 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. 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

Download IMSA and the database files. 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/config.py**.

```
Files downloaded:
gi_taxid_nucl.dmp.gz  human.DB.tar  imsa_v2.tar NCBI.fungiDBselect.genomeonly.tar  taxdump.tar.gz

COMMANDS:
gunzip taxdump.tar.gz
gunzip gi_taxid_nucl.dmp.gz
tar -xvf imsa_v2.tar
cd imsa_v2
mkdir human
mkdir microbiome
mkdir gi_taxid
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_JWC_gi_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 is probably not "username" and this must be changed.

```
# 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_v2/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_v2/human/bowtie/GRCh37.2bit"}

BLAT_OOC_FILES = {"human":"/home/username/imsa_v2/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_v2/microbiome/NCBI.fungiDBselect.genomeonly",
                   "human":"/home/username/imsa_v2/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_v2/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_BLASTN = "blastn"
PATH_TO_BLAT = "blat"
```

IMSA+A

Download IMSA+A.tar to the imsa_v2/ directory

Extract the contents there and edit systemSettings.py

Change

```
PATH="/data/ncbi/gi_taxid/"
```

to

```
PATH="/home/username/imsa_v2/gi_taxid/"
```

```
fungiLookupFile = "/data/CAGE_clusterdata/Users/cox1kb/pIMSA.2015.11/fungiDB.seqNames.2.speciesID"
```

to

```
fungiLookupFile = "/home/username/imsa_v2/fungiDB.seqNames.2.speciesID"
```

`fungiDB.seqNames.2.speciesID` contains mappings for Reference Sequence names that do not contain gi numbers to taxon ID representing species. 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 to boost mRNA, 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 <u>your</u> 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. 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.

<u>Run these at the same time!</u>

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.

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=/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 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

Using and Understanding Output

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, then non-redundant (NR) 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|". (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.

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, therefore, if utilizing one assembler, Inchworm is preferred.

Interpreting Results

Open results file: inchworm_snt.tax_genus.IMSA+A_4count.txt and oases_snt.tax_genus.IMSA+A_4count.txt

This file is tab delimited, so it will import into EXCEL or LibreOffice Calc gracefully.

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
```