Reassortment Detection Manual

The purpose of this note is to document, in a first draft, how to re-run the code for the Reassortment Detection analysis.

Input files:

A CSV file called "[handle] Sequences.csv", which contains the following information downloaded from the IRD:

- Segment
- Protein Name
- Sequence Accession
- Complete Genome
- Sequence Length
- Collection Date
- Host Species
- Country
- State/Province
- Flu Season
- Strain Name
- A FASTA file containing all of the sequences downloaded, with only the Accession Number in the header.

Place both of these files under a single directory.

Make the following sub-directories:

- alignments
- distmats
- edges
- reassortant_edges
- shell_scripts
- split_fasta
- sge_outputs

There is also a list of scripts present:

- `align_sh.py`
- `align.py`
- `clean_affmats_sh.py`
- `clean_affmats.py`
- `compile_affmats_sh.py`
- `compile_affmats.py`

- `full_affmat.py`
- `graph_combiner.py`
- `graph_initializer.py`
- `max_edge_finder_sh.py`
- `max_edge_finder.py`
- `node_data_imputer.py`
- `preprocessing.py`
- `second_search.py`
- `sequence_splitter_sh.py`
- `sequence_splitter.py`
- `source_pair_combiner.py`
- `source_pair_manual_sh.py`
- `source_pair_sh.py`
- `tables_functions.py`

Step 1: Pre-processing CSV file

- On the Rous server (or any system that has the SunGrid Engine installed, write a .sh file with the following bash command:
- `python preprocessing.py [handle]`, where `[handle]` refers to the standard name that is used across all of your scripts.
- Wait a while for the preprocessing script to complete.
- If everything completes successfully, you will see a CSV file by the name `[handle] Full Isolates.csv`.
- Clean up the directory by running the following command in the terminal:
  - `mv \*.sh.\* sge_outputs/.`
  - `cd ..`

Step 2: Splitting FASTA file by segment

- To enable parallel running of the splitting script, which might take a while, I have provided a Python script (sequence_splitter_sh.py) which writes a series of shell scripts to submit.
- Modify this file, by replacing the handle string at the top with your handle. An example modification can be: ‘“20141103 All IRD”’
- Run the following bash commands:
  - `python sequence_splitter_sh.py`
  - `cd shell_scripts`
  - `qsub sequence_splitter.sh`
- Wait until all of the jobs have been run.

- You should see all 8 segments being processed and written to disk. They will be in the directory `/split_fasta/[Handle] Segment [Number].fasta`
- While in the `shell_scripts` directory, clean up the directory by running the following commands:
- `mv \*.sh.\* ../sge_outputs/.`
- `cd ..`

Step 3: Alignment and Distance Matrix Generation

- To enable parallel running of the alignment script, I have provided align_sh.py.
- Modify this file by replacing the handle string at the top with your handle.
- Run the following bash commands:
  - `python align_sh.py`
  - `cd shell_scripts`
  - `qsub align.sh`
- Wait until all of the jobs have been run. On a large dataset, it'll take a while - on the order of hours, depending on CPU usage and data size.
- While in the `shell_scripts` directory, clean up the directory by running the following commands:
  - `mv \*.sh.\* ../sge_outputs/.`
  - `cd ..`

Step 4: Convert Distance Matrix into Thresholded Similarity Matrix, and Rename the Index/Columns.

- To enable parallel running of the compilation scripts, I have provided the following scripts:
  - `clean_affmats_sh.py` <— corresponding with `clean_affmats.py`
  - `compile_affmats_sh.py` <— corresponding with `compile_affmats.py`
- Run the following bash commands:
  - `python clean_affmats_sh.py`
  - `cd shell_scripts`
  - `qsub clean_affmats.sh`
- Wait until all of the jobs have been run.
- When all of the jobs are done, clean up the `shell_scripts` directory as per before.
- Then, run the following bash commands:
  - `python compile_affmats_sh.py`

  - `cd shell_scripts`
  - `qsub compile_affmats.sh`
- Wait until the job is done.
- When the job is done, clean up the `shell_scripts` directory again.

Step 5: Make a summed affinity/similarity matrix

- Create a shell script in the `shell_scripts` directory, that includes the following commands:
  - `cd ..`
  - `python full_affmat.py [handle]`
- Run the following bash commands:
  - `cd shell_scripts`
  - `qsub full_affmat.sh`
- Wait until the job is done.
- When the job is done, similarly, clean up the `shell_scripts` directory.

Step 6: Search for max edges for each virus - full 'transmissions' only.

- Run the graph initializer script: `python graph_initializer.py [handle]`. This one should run fast. The expected output is a `.pkl` file which houses the initialized network with only nodes present.
- Run the script: `python max_edge_finder_sh.py`
- Then, run the following commands:
  - `cd shell_scripts`
  - `qsub max_edge_finder.sh`
- Wait until all the jobs are done.
- When done, clean up the `shell_scripts` directory.

Step 7: Combine found edges into a condensed graph.

- Run the graph combiner script: `python graph_combiner.py [handle]`
- If your "handle" is different, i.e. you have a different prefix for all of the files, then replace the text inside the quotation marks with your handle.
- This one should be fast, i.e. within a dozen minutes.

Step 8: Compile a list of nodes to perform source pair searches on.

- Run the script using: `python second_search.py [handle] [percentile]`
  - `[handle]`: the common prefix to all of your files.
  - `[percentile]`: the cutoff percentile of whole genome edges to try source pair searches. Our analysis used the 10th percentile, so the value input was "10".
- This script should run fast.
- The expected output is a `.pkllist` file with a list of nodes to perform source pairs on.

Step 9: Perform source pair searches.

- Run the following bash commands:
  - `python source_pair_sh.py`
  - `cd shell_scripts`
  - `qsub source_pair.sh`
- Wait for the job to finish completing. This one should take a while, depending on CPU availability. May take up to a few hours.
- When done, clean up the `shell_scripts` directory.

Step 10: Combine source pairs with full graph.

- Run the following bash commands: `python source_pair_combiner.py [handle]`
- This one should run fast (~minutes).

Step 11: Annotate graph with edge and node metadata

- Run the following bash commands:
  - `python graph_pwi_finder.py [handle]` (this should take a few minutes) - this will annotate edge PWIs
  - `python graph_cleaner.py [handle]` (also should be fast)
- At this point, the graph construction steps are complete.