General README for Compositae COS Bioinformatic and Phylogenetic Workflow

README generated by: Rishi R. Masalia, Burke Lab, Univ. of Georgia, August 2013
Corresponding email: jennifer.r.mandel@gmail.com or masalia@uga.edu

Goal: Perform the bioinformatic steps for processing and preparing data resulting from the targeted sequence capture and next-generation sequencing of Compositae COS loci for phylogenetic analyses.

Output of the Workflow: Some number of nexus files of aligned COS loci. Please also refer to workflow diagram in the Compositae-COS-workflow directory on GitHub.

Folder/Directory Preparation:

Create a main folder for this analysis to be run in. All raw sequence (fastq) read data must be in this main folder. In this main folder, create a ./programs and ./databases folder. Exactly named. In the ./programs folder put all downloaded scripts and outside programs, (except the optional wrappers). Programs listed: prinseq-lite.pl, blastall from blast v 2.2.26, TopHits.pl, Pairfq.pl, Shufflesequences.pl, Velvet Optimiser, Phyluce.

In the ./databases folder put all formatted databases. For Phyluce, you will need to make a ./contigs, ./lastz, ./fasta, In the ./contigs folder, you will place your assembled contig fasta files following velvet assembly.

If you are not using the Optional_Blast_Wrapper.pl and intend to use the Optional_Velvet_Wrapper.pl please create a ./Output and ./Archived_Files folder, as these will be used in the Optional_Velvet_Wrapper.pl script.

If you are not using either of the provided optional wrappers, please proceed to the workflow.

Workflow

Note: All programs are ready-made programs we call in this workflow. As such, any questions regarding their operation or usage should be directed to their manuals, forums and help pages, or follow the necessary channels in contacting the authors of these programs.

****Note: None of the program paths are given here. If you are running these programs yourself, please indicate the program executable path.****

1. Prinseq-lite.pl to clean your reads (http://prinseq.sourceforge.net/index.html)
    a. Code: perl prinseq-lite.pl -fastq <reads file.fq> -out_format 1 -out_good ./<reads file>.trimmed -out_bad null –log <reads file>.dirty.log -min_len 40 -noniupac -min_qual_mean 15 -lc_method entropy -lc_threshold 60 -trim_ns_right 10 -ns_max_p 20

    b. These parameters are hard coded in the Optional Wrappers, if you wish to adjust them please amend the code in the Wrappers or run this step separately.
2. Blast reads against a database (http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download):
    a. Choose the database
        i. COS_sunf_lett_saff_all.fasta [default; source ESTs for probe sequences]
        ii. COS_probes_blast.fasta
        iii. Create your own
    b. Code: blastall -i <Cleaned Sequence File.fa> -d <Database> -o <Outfile.blast> -p blastn -e <evalue> -v1 -b1 -m8
3. Top Hits.pl to pull out best hit from blast output and pair with sequence from original file (on Compositae-COS-workflow GitHub repository)
    a. Code: perl Top_Hits.pl <in_file> <out_file> <# of hits wanted> <Cleaned Sequence File.fa>
    b. This custom script can also be achieved by using any blast output parser followed by pairing the top hit information with the original cleaned fasta file.
4. Pairfq.pl to pair up reads between R1 and R2 and determine if there are singletons (https://github.com/sestaton/Pairfq):
    a. Code: pairfq.pl -f <forward reads.fa -r <reverse reads.fa -fp forward_paired.fasta -rp reverse.paired.fasta -fs forward.singletons.fasta -rs reverse.singletons.fasta
5. ShuffleSequences to order reads (https://github.com/dzerbino/velvet/tree/master/contrib/shuffleSequences_fasta):
    a. shuffleSequences_fasta.pl forward_paired.fasta reverse_paired.fasta <Read Name>_paired.fasta
6. Concatenate Singletons of R1 and R2:
    a. cat forward.singletons.fasta reverse.singletons.fasta > <Read Name>_singletons.fasta
7. Velvet Optimiser to assemble your reads (https://github.com/Victorian-Bioinformatics-Consortium/VelvetOptimiser; wrapper for velvet assembly https://github.com/dzerbino/velvet):
    a. Code: VelvetOptimiser.pl –s 51 –e 71 –o'-ins_length 350' –f '-fasta –shortPaired <Read Name>_paired.fasta –short <Read Name>_singletons.fasta'
8. Phyluce (https://github.com/faircloth-lab/phyluce; please read the Phyluce software overview at http://faircloth-lab.github.io/phyluce/installation.html to ensure all necessary programs/dependencies are installed, i.e., python 2.7, lastz, mafft, etc.; specific Phyluce programs used for this workflow include: match_contigs_to_probes.py, get_match_counts.py, get_fastas_from_match_counts.py, and seqcap_align_2.py):
    a. match_contigs_to_probes.py; matches velvet contigs to probes (cos_probes.fasta) using lastz
        i. Place assembled velvet contigs in ./contigs directory
        ii. Rename all contig file names as species-a.contigs.fa, species2-b.contigs.fa, etc.
        iii. Code: python match_contigs_to_probes.py \
        contigs/ \
        cos_probes.fasta \

lastz \
--regex "_p[0-9]+$" --repl "" \
--dupefile probes_toself.lastz

b. Make a text file with a list of species' names (see example: cos-taxa-groups.conf)
c. get_match_counts.py; runs a query against the seqlite database from above, and pulls out loci for those taxa in the cos-taxa-groups.conf file having COS contigs
    i. Code: python get_match_counts.py \
    lastz/probe.matches.sqlite \
    cos-taxa-groups.conf \
    'all' --output all-cos-taxa-incomplete-matrix.conf \
    -incomplete-matrix
d. Make a blank text file named "allcos-incomplete.notstrict"
e. get_fastas_from_match_counts.py; generates a dataset that includes any loci enriched across the taxa in the cos-taxa-groups.conf file
    i. Code: python get_fastas_from_match_counts.py \
    contigs \
    lastz/probe.matches.sqlite \
    all-cos-taxa-incomplete-matrix.conf \
    --incomplete-matrix allcos-incomplete.notstrict \
    --output ./fasta/cos-all-taxa-incomplete-matrix.fasta
f. seqcap_align_2.py; aligns data across COS loci; NOTE: do not mkdir "nexus"
    i. Code: python seqcap_align_2.py \
    fasta/cos-all-taxa-incomplete-matrix.fasta \
    nexus \
    3 \
    --aligner mafft \
    --incomplete-matrix \
    --cores 1
    [add --notrim, if you don't want MAFFT to removes ragged 5' and 3'edges from alignments]

Output is XXX number of .nex files with aligned loci that can be used for phylogenetic analyses. Cheers!

Optional Wrappers:

- Blast back ESTs:
    - Code: perl Optional_Blast_Wrapper.pl [parameters]
    - Incorporates workflow steps 1-3
- File Preparation for Velvet Assembly:
    - Code: perl Optional_Velvet_Prep_Wrapper.pl <namefile>
    - Incorporates workflow steps 4-6
    - Does not actually run velvet or velvet optimiser