Analysing target capture data for phylogenomics

Paul Bailey and Alexandre Zuntini

Session 3 - Making Species Trees

Objectives

- Take HybPiper gene recovery files for each sample and make gene trees
- Make species trees with ASTRAL III and raxml-ng
- Visualise species trees

Step 1 - Create gene alignment files with MAFFT

- MAFFT home page

- The starting point for this session is a set of fasta files, each one containing the same gene from all the samples being investigated i.e. the step after running retrieve_sequences.py on your sample directories.

- The aim is to run MAFFT on each gene fasta file, submitting each gene to align into a Slurm job.

- To do this task there needs to be a way to loop through a list of gene names. All the gene names should be listed in the targets fasta file, at least they are for the HybPiper test dataset used here. They can be extracted from the fasta header line using four bash commands as follows

```
cat <path_to>/test_targets.fasta | grep '>' | awk -F '-' '{print $2}' | sort -u >
test_targets_geneIds_ONLY.txt
```

- cat prints out the file

- grep prints out only lines with '>', i.e. the fasta header

- awk prints the second column of the row

- sort -u prints out a single copy of each gene name

- Now run MAFFT on each gene file

```
cat test_targets_geneIds_ONLY.txt | \
while read gene; do
  sbatch -J ${gene}_mafft -p fast -t 0-2:00 -c 1 --mem=1000 -o ${gene}_mafft.log -
e ${gene}_mafft.log_err --wrap "
  mafft --thread 1 \
  --retree 2 \
  --reorder \
  --preservecase \
  ${gene}.fasta \
  > ${gene}_mafft_dna_aln.fasta
  "
done
```

- MAFFT alignment algorithms:

1. progressive method (fast - for up to 5k seqs) --retree 1 FFT-NS-1/NW-NS-1
2. progressive method (fast - for up to 5k seqs) --retree 2 FFT-NS-2/NW-NS-2 (better than retree 1)
3. iterative refinement method --maxiterate 1000 - refinement repeated until no more improvement in the WSP score is made or the number of cycles reaches 1,000 (slower, more accurate)
4. L-INS-i, E-INS-i, G-INS-i - Iterative refinement methods using WSP and consistency scores - (slowest, most accurate)

Step 2 - Create gene trees with FastTree and RAxML-NG

- FastTree home page and the RAxML-NG home page

- In this step the alignment file from step 1 is used as the input to a phylogeny program that uses a maximum likelihood method to calculate the tree topology and branch lengths. We will run fastTree which is very fast and RAxML-NG which is slower but more accurate when using real bootstrap datasets

- What other types of phylogeny methods are there and what are there strengths and weaknesses?

- The phylogeny program can be run for each gene alignment in a similar way as in part 1, in a loop and via Slurm. First run FastTree as follows

```
cat test_targets_geneIds_ONLY.txt | \
while read gene; do
  sbatch -J ${gene}_fasttree -p fast -t 0-2:00 -c 1 --mem=1000 -o
${gene}_fasttree.log -e ${gene}_fasttree.log_err --wrap "
  fasttree -nt -gtr \
  ${gene}_mafft_dna_aln.fasta \
  > ${gene}_mafft_dna_aln_fasttree.nwk
  "
done
```

-nt flag specifies that input sequences are nucleotides
-gtr flag specifies the model of evolution (general time reversible, gtr)

- Now repeat the above loop but substitute FastTree with this RAxML-NG command

```
raxml-ng --threads 1 \
--all \
--msa ${gene}_mafft_dna_aln.fasta \
--model GTR+G \

--prefix ${gene}_mafft_dna_aln \
--seed 2 \
--bs-metric fbp,tbe \
--bs-trees 100
```

--all flag specifies to create bootstrap datasets, perform a maximum likelihood search on each bootstrap dataset and draws support values onto the best-scoring tree --bs-trees flag specifies the number of bootstrap datasets to use, 100 or 1000 for publication are usual

- Note that you could also add all the above steps to a single loop to perform the alignment and tree steps together, one after the other

Step 3 - Coalescent-based species tree estimation with ASTRAL

- ASTRLAL III home page

- The ASTRAL program takes gene trees as input and finds the species tree that has the maximum number of shared induced quartet trees with the set of gene trees
- It doesn't appear to be on the cluster so download it from the GitHub into your home directory, unpack it and test it works

```
git clone https://github.com/smirarab/ASTRAL.git
cd ASTRAL
unzip Astral.5.6.3.zip
# Test the program:
java -jar Astral/astral.5.6.3.jar -i Astral/test_data/song_primates.424.gene.tre
```

- First, all the Newick gene trees from one of the phylogeny programs used in step 2 need to be added into a single file e.g.

```
cat *_mafft_dna_aln.raxml.supportFBP > raxml_trees_for_coelescence_phylo.nwk
```

- Now run Astral with the gene trees file just created (via Slurm):

```
java -jar <your_path_to_astral>/astral.5.6.3.jar \
-i <your_concatenated_newick_trees> \
-o phylo_astral.nwk
```

Information on the run of the program are output onto the command line. You could also redirect the output to a file for examining later by adding to the end of the command

```
> phylo_astral.log 2>&1
```

The output 2>&1 sends any errors printed to stderr into the .log file as well.

- In the log file look for the line containing 'Final normalized quartet score'. This score is the proportion of input gene tree quartet trees satisfied by the species tree and is a number between zero and one, the higher the number, the less discordant your gene trees are. For further explanations of the output, read the section under 'EXECUTION' in the ASTRAL documentation

- Branch support at each node is measured as local posterior probabilities. Further values can be added at each node using the -t flag as described in the Astral tutorial

- Branch lengths are in coalescent units and are a direct measure of the amount of discordance in the gene trees

Step 4 - Tree visualisation

- Newick format definition

- Toy example: ((l1:3,(l2:2,(l3:1,l4:1)n1[c]:1)n2:1)n3:1,l5:2,l6:2)n4;
  l = leaf node
  n = internal node
  [c] = comment - free text (NB - comments are allowed but downstream applications have to understand what to do with them)

- The easiest way to view Newick file for the species tree is to use a good online viewer e.g. iTol

- Other viewers exist: Figtree, EvolView

Step 5 - Other considerations for steps 1 -3

- **Point 1** - It is good to assess coverage of each gene across the alignment and consider removing sequences that are not long enough. The first command finds the longest gene in the alignment, then the second command outputs the number of sequences that pass a certain length threshold e.g. 60 % . Each gene would need to be processed in a loop via Slurm

```
lenLongestGene=`fastalength  <gene_name>_mafft_dna_aln.fasta | sort -n | tail -n 1
| awk '{print $1}' `
echo lenLongestGene: $lenLongestGene

fastalength <gene_name>_mafft_dna_aln.fasta \
| awk -v LENG=$lenLongestGene -v FRACTN=0.60 '{if( ($1/LENG) >= FRACTN ) {print
$2} }' \
> ${gene}_mafft_dna_aln_ovr60pc_aln_covrg.txt
```

Note that the above steps could be done just after running the gene alignments in the same loop!

- Next step is to retrieve just those seqs from the above output file with seqtk subseq command - give it a go! - then those sequences ONLY would be input to the phylogeny stage. Also consider that the gene alignments might contain less than four sequences before or after filtering - a phylogenetic tree needs to have at least four sequences

- **Point 2** - Before running ASTRAL, it is recommended to remove clades from the gene trees that have low bootstrap values (less than 10 %). This can be done with Newick Utilities using the file containing all the gene trees

```
nw_ed  fasttree_trees_for_coelescence_phylo.nwk 'i & (b <= 0.10)' o >
fasttree_trees_for_coelescence_phylo_bs_less10pc_rmed.nwk
```

This output file would become the input to ASTRAL.

Step 6 - Creation of a species tree with RAxML-NG from an alignment supermatrix

- It is easy to estimate a species tree from an alignment supermatrix by concatenating the gene tree alignments and then running a phylogeny program e.g. RAxML.

- Use the AMAS program to concatenate all the columns of each gene tree alignment

```
AMAS concat -c 1 \
-i *_mafft_dna_aln.fasta \
--in-format fasta \
-d dna \
--out-format fasta \
-t mafft_dna_alns_concatenated.out
# NB- also creates a partitions file with coords for each gene which could be used
to set different models.
```

- AMAS also creates a partitions file with coordinates for each gene which could be used to set different models of evolution for each gene for use in the phylogeny program

- The concatenated alignment can be viewed if you install JalView on your computer

- Now run RAxML with the concatenated alignment (via Slurm)

```
raxmlHPC-PTHREADS-SSE3 -T 4 \
-f a \
-x 12345 \
-p 12345 \
-# 100 \
-m GTRGAMMA \
-s mafft_dna_alns_concatenated.out \
-n raxmlHPC-PTHREADS-SSE
```

"-f a" option is for a rapid Bootstrap analysis and search for best-scoring ML tree in one program run - NB - this command uses the previous version of RAxML prior to the release of RAxML-NG.

- This step generates a species tree from exactly the same data set as used in Step 3 for ASTRAL but via a different theoretical approach. It may be useful to compare trees from these two approaches to verify the species topology

Other Programs and tools to investigate

- PRANK - a phylogeny aware aligner that deals with insertions more effectively; slow but suitable for small data sets
- PAL2NAL - aligns DNA based on the corresponding protein alignment which should give better accuracy for exon sequences
- FluentDNA - creates small alignment pictures for many gene alignments which can be viewed together; becomes easy to spot alignment issues by eye e.g. missing data, paralog anomalies
- PhyDesign - tool for profiling phylogenetic informativeness across gene loci
- Estimate trees with supercontigs (introns + exons) from HybPiper intronerate output
- IQTree - another popular phylogeny program which could be used in place of FastTree or RAxML-NG.
- TreeShrink Finding and removing spuriously long branches
- PhyParts - allows visualisation of conflicts between gene trees for a given species tree in the form of a pie chart at each tree node
- RogueNaRok - detects rogue taxa in tree data sets