Comparative Genomics 2018
Practical 5: Gene order analysis

Group 11
Fuqi Xu, Milda Valiukonyte, Shuhan Xu

Summary
In this practical we learned about how gene order analysis can help us understand the evolution of genomes. We used otholog clusters identified in exercise 4 to infer the order they appear in the genomes and analyzed the differences between the genomes by visualizing genes against each other in a dotplot using Dotter program. We later used GRIMM to determine the genomic rearrangement distance between genomes and construct a phylogenetic tree using Belvu. It showed that the biggest similarities are between *E.coli* and *S. curvatus,* and between *S.coelicolor* with *R.xylanophilus.*

Exercise 1

Ortholog clusters from practical 4 is used for this practical: see attachment *cluster*

Exercise 2

1.
Gene order output for 09.fa.txt: *09_gene_order*
Gene order output for 17.fa.txt: *17_gene_order*
Gene order output for 49.fa.txt: *49_gene_order*
Gene order output for 51.fa.txt: *51_gene_order*

2.
a.
There can be ambiguities in clustering.

For genome 09.fa.txt, no genes will appear in more than one clusters since this genome is the reference genome and is used as the query to BLAST search against other genomes. Hence each cluster contain a unique 09.fa.txt gene.

For 17.fa.txt, 49.fa.txt and 51.fa.txt, there may be genes appearing in more than one cluster since each of these genomes are used as reference database in the BLAST search. For instance, two different genes from 09.fa.txt may have the same best-scoring BLAST hit in 17.fa.txt. As a result, two clusters may have different genes from 09.fa.txt but the same gene from 17.fa.txt.

If a gene appears in more than one cluster, the script will assign to it the cluster number of the first cluster in the list in which the gene appears. Hence, if the gene appears in subsequent clusters, these clusters will not have that gene after the script is run.

b.
This script does not handle forward and reverse strandedness in the gene order list. To implement this functionality in the script, the script can add a positive sign ‘+’ in front of the cluster number if the gene appears in the forward strand and a negative sign ‘-’ if the gene appears in the reverse strand. This can be done in the last loop of the script:

```
for aGene in geneOrderList:
        if partOfCluster.has_key (aGene):
                if aGene.endswith('rev'):
                        print '-' + str(partOfCluster [aGene]),
                else:
                        print '+' + str(partOfCluster [aGene]),
```

Downstream scripts can check the sign and deal with the situation accordingly, e.g. generate reverse complement if the sign is negative.

Exercise 3

We wrote a script to perform the tasks in question 3, 4, 5 and 6. This script incorporates rndseq.py

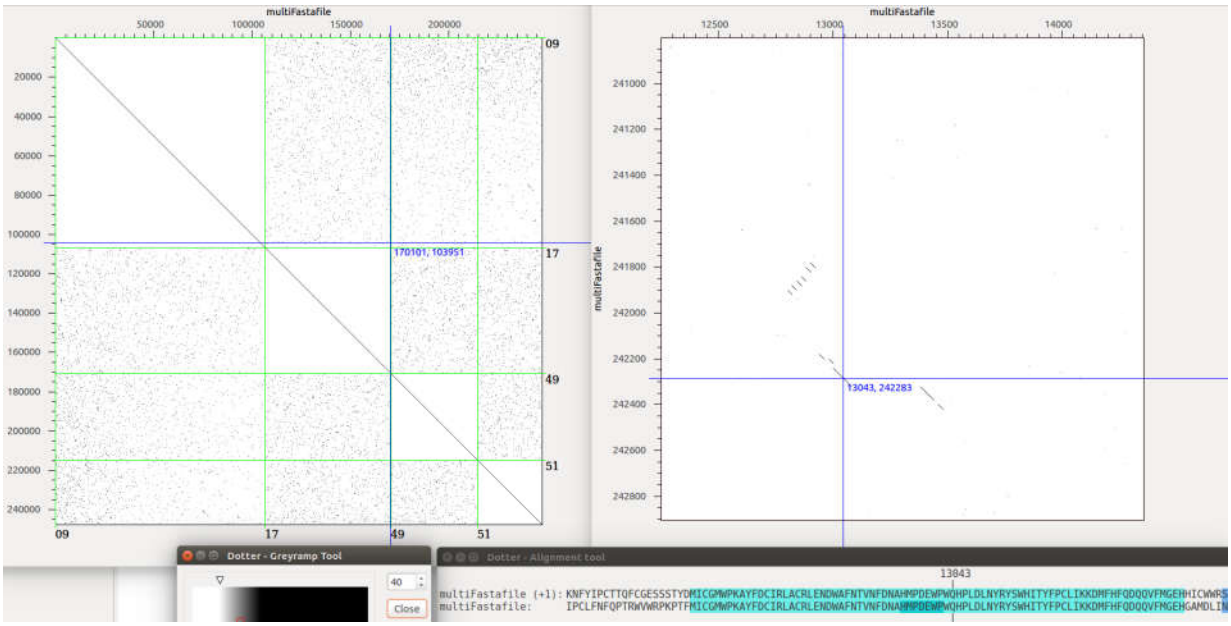
Script for generating pseudo-genomes from gene order files: see attachment *generate_pseudogenomes.py*

Usage: python3 generate_pseudogenomes.py <number of ORFs in largest proteome> <gene order file 1> <gene order file 2> <gene order file 3> …

Description: The script first generates a list of pseudo-genes (from rndseq.py code in the script) and converts it to a dictionary. Then, for every gene order file, it assigns a pseudo gene to each cluster number and concatenates the pseudo-genes into a single long sequence (pseudo-genome). Finally, it combines all the pseudo-genomes into one multi FASTA file called 'multiFastafile' in the working directory.

Multi FASTA file output from script: see attachment *multiFastafile*

6.



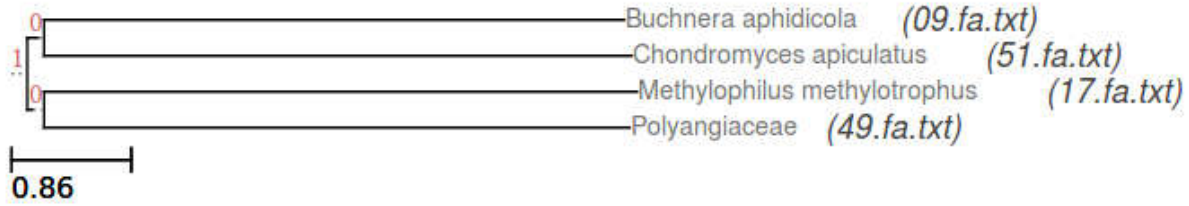
In the dot plot, each dot represents one matched residue in different genes, while the diagonal represents the sequences aligned to themselves (except for the diagonal). Short lines in the plots indicate synteny. Other separate dots are matches randomly happened, which are background noise. For example, the figure in the right is a segment in genome09 and genome17. An 80-residue match appears around residue (13043,242283), which indicates four 20-residue genes in genome09 and genome17 have the same gene order. Conservation in gene order suggests a closely related evolutionary relationship.

Exercise 4

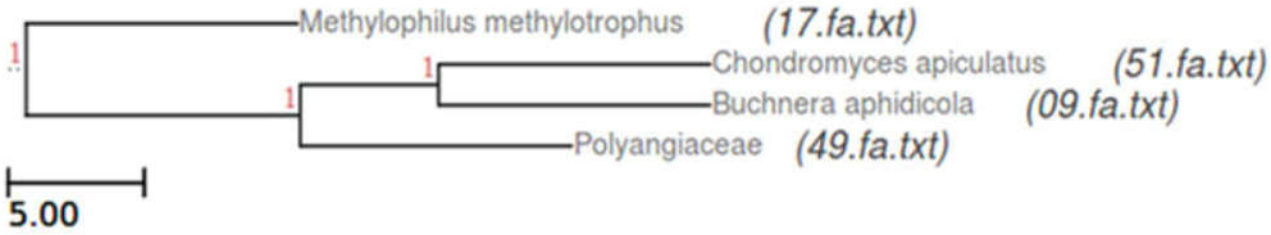
1.
Formatted gene order file for GRIMM: see attachment *grim_genomes.txt*

2.
Distance matrix from GRIMM: see attachment *distance.grim*

3.
Reconstructed Species Tree using distance matrix from GRIMM:



4.
Consensus Tree for 10 orthologs clusters from Practical 4:



The tree constructed using genomic rearrangement distances agrees with the tree built in exercise 3 which used just the sequences. Escherichia coli (09.fa.txt) genome seems to be mostly related to Spiribacter curvatus (51.fa.txt) while Streptomyces coelicolor (17.fa.txt) is mostly similar to Rubrobacter xylanophilus (49.fa.txt). The result agrees with the multiple sequence alignment based tree we made in practical 4. We got same evolutionary relationship based on gene order analyzation and sequence alignment method.

If the result didn't agree, one of the possible explanation is that the gene order depends on the evolutionary relationship. If the four organisms are distant in evolution, the genes will be shuffled, and the evolutionary relationship is not observable. Another explanation is that in practical four we only used ten genes to build the tree. We might have got a different tree if we had chosen other genes.