Practical 5: Gene Order Analysis
Group number: 2
Group members: Maximilian Senftleben, Zhong Hao Daryl Boey

Summary: In this practical, the change of arrangement of the genes is observed. For that, the in practical 4 found gene cluster is taken and the gene order is then obtained with the script getGeneOrder.py. Then, the gene order between the different proteomes is compared with the program DOTTER, where changes in location can be detected. Lastly, given the obtained gene order, a phylogenetic tree can be detected.

Exercise 1 - Ortholog groups

For this exercise you will use the ortholog groups (clusters) you found last week in Practical 4.

Used clusters from Practical 4 instead of using InParanoid (tried once but it was immensely slow).
file: orth_clusters

Exercise 2 - Gene order

1. We provide the script getGeneOrder.py which reads a proteome multi FASTA file (i.e., 01.fa.txt.pfa) and a cluster file from Practical 4 (IF POSSIBLE, include all the clusters, not just the clusters where all the organisms tested are presented), and outputs the gene order list for the corresponding genes (ORFs) that appear in the clustering output. This script is just an example, it will not directly work for all of you since you have different formats in your input data. PLEASE SAVE THE GENE ORDER OUTPUT, which should be a vector of genes (ORFs) positions.

Ran the script getGeneOrder.py for 16, 20, 44, and 47.fa.txt.pfa against orth_clusters file. Saved outputs as output16, output20, output44, and output47 respectively.

2. If you use this script, please answer the following questions:
a. Can there be ambiguities in clustering, so that one gene appears in several clusters in the cluster file?If so, why is this? What does this script do in that case?

Yes, a single gene can appear in several clusters. This is because we ran a series of BLAST runs between and against 4 different organism's sequences, and therefore a single gene, if highly conserved, may appear multiple times between various clusters.

b. Can this script handle forward and reverse strandedness in gene order lists? If not, how should this be done?

This script is so far only capable of handling a single strand (coding strand in this case). Therefore in order for the script to be capable of parsing gene order for both strands, an additional module should be used to generate complementary sequences for the coding strand prior to running the original script.

EXERCISE 3 – DOTTER

Compare the gene order that you obtained between your proteomes to detect changes in location. Check synteny definition and visualizations.

3. Generate a numbered list (i.e., dictionary) of random 20aa sequences, at least as long as the highest number of ORFs in the biggest proteome(use: rndseq.py).

I used this code to count the size of the different proteomes:
grep -o '>' Proteomes/16.fa.txt.pfa | wc -l

Prot.	Length	pseudogenome
16	6414	6500
20	1877	2000
44	1821	2000
47	2498	2500

Random sequences were created, (randomseq16_6500.txt) in the rnd-folder.

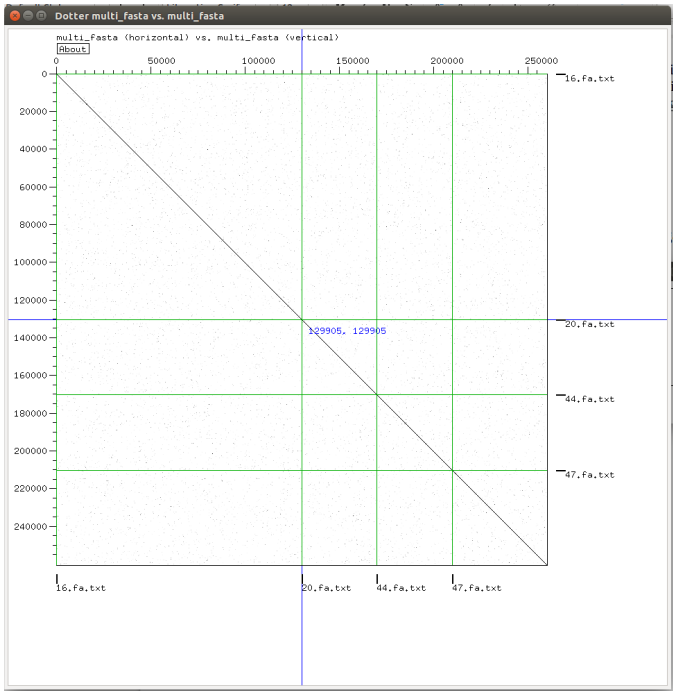
4. For each file obtained from getGeneOrder script, assign random sequences (one for each gene). Combine all these 20aa sequences for each proteome and gene order file into a long single sequence.

Code parser.py
out_16 in the rnd-folder

5. Alternatively, script called makeIconicGenome.py reads in a list of random sequences (there must be more of these than there are cluster IDs) and a gene order file, and prints out the resulting pseudogenome. If you use this script, explain what is it doing as a short pseudo-code in your report.

No.

6. Combine all sequences into one multi FASTA file and use it as dotter input against itself into dotter.
a. Initialize dotter with "module add dotter". Play with the parameters to obtain a visual output that better explains the synteny between these organisms (read the suggested documentation to understand it properly). Explain your dotplot visualization.



The dotplot shows an obvious continuous correlation between the 2 sequences, given we are running the same file against itself. Adjusting the visual settings for the plot shows that the background dots are mostly noise. It is unlikely that we are expecting to visualize synteny between these 2 organisms, as we generated random sequences and concatenated them , and the random dots that appeared initially prior to adjustment are matches due to chance, despite having designed the pseudosequences based on gene order.'
file: multi-fasta

Exercise 4

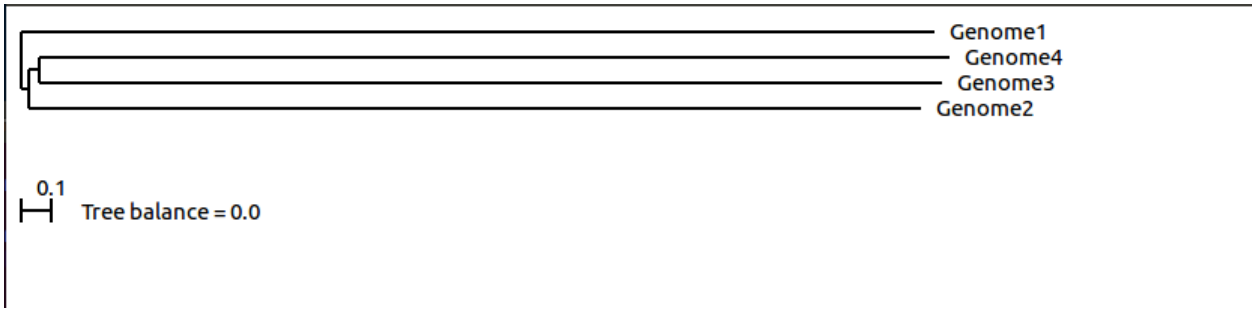
1. Use a tool (such as GRIMM) to determine the genomic rearrangement distances between the species. GRIMM wants some lists of numbers corresponding to genes as input; so in that case, translate your gene order lists to the lists of corresponding numbers. This should give you a distance matrix. It turns out that you need to change the format of the gene order lists from the previous part to get GRIMM/MGR to accept it.
It only accepts input data where both sequences have exactly the same set of orthologous genes, and they must be listed from one upwards. To convert your gene lists into this format, do it for all of your prokaryotic gene order lists at the same time. This will give you the output for all of them, consisting only of the genes present exactly
once in each of them, and with the format GRIMM/MGR wants it.
python getGeneOrderGrimm.py <max number of genes in output> <input gene order file 1> <input gene order file 2> ... <input gene order file N>

Done, File: outforgrimm

2. Calculate the distance matrix between these genomes
grimm -f grim_genomes.txt -o distance.grim -C -m

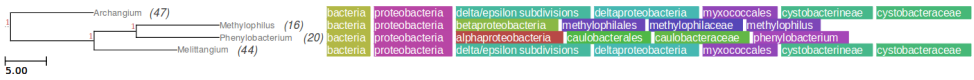
Genome1 Genome2 Genome3 Genome4
0 503 509 511
503 0 501 503
509 501 0 503
511 503 503 0
file: distance.grim

3. Use the resulting distance matrix to reconstruct the species tree in Belvu.
module add belvu
belvu -T R distance.grim
file: tree_grim



Distance matrix must be in space delimited format, where the columns are the genome names.

4. Compare the tree obtained in Practical 5 against the one obtained in Practical 4. How do they differ and why?



The tree branches are identical between the trees from Practical 4 and Practical 5, however there are some minor differences with regards to branch length. The distances may vary due to the difference in distance calculation methods; in practical 4 we relied on K-ALIGN's distance correction methods while in this practical we calculated the distances in GRIMM.