Practical 4: Gene Prediction
Group number: 2
Group members: Maximilian Senftleben, Zhong Hao Daryl Boey

Summary: In this practical the end purpose was to reconstruct phylogenic trees from orthologous clusters, via the creation of a metagene. The first step to determining orthologues for these trees was by BLASTing the prokaryotic genomes against each other, followed by multiple parsing steps to obtain 2 main outputs, a parsed text file from BLAST's xml output, and cluster files which chained the respective orthologous hits from the 3 other datasets into one. These cluster files were then used to generate metagenome sequences, prior to feeding to K-align for sequence alignment. Phylogenic trees were then generate from these alignments using Belvu. Several technicalities about understanding a phylogenic tree and it's reliability were also covered in this practical.

Exercise 1 - Orthology Search
Select one of your prokaryotic genomes as the reference genome. For this prokaryotic reference genome, you will search for the best hit orthologs in the remaining prokaryotic genomes. Use BLAST for this purpose.

1. For all of your prokaryotic genomes (including the reference genome), find multi FASTA files with proteins from one of the previous practicals.
The multi FASTA file contains all protein sequences inferred from your genomes and will be called as the proteome file.

Proteome files uploaded for the genomes 16, 20, 44, 47 (7 each)

2. Configure BLAST
Copy the configuration file to be able to use the BLAST commands.
cp /afs/pdc.kth.se/home/a/arnee/.ncbirc ~/

3. Create a BLAST database for your proteomes.
In the previous practicals, you used makeblastdb to create a database for blastn. Repeat the same procedure for all of your selected proteomes individually (amino acid type).

Files included in the proteome – folder.

4. Perform a blastp search against your reference proteome.
blastp -outfmt 5 -query <ref-proteome.fa> -db <query-proteome.fa> -out <output file>
(Added -m 5 parameter returns an XML output which is easy to parse in biopython)
4.1. Repeat this process for all of your query proteomes and create the output XML files.

Dbs created for the 4 genomes, 12 files in total (example: ref16db20)

5. Modify your script from Practical 3 to parse the XML output. The input will be like: <xml output> <tag for reference genome> <tag for target genome>

5.1. Reference and target genome tag parameters are used to assign the genome name for the query and target sequences.
<tag for reference genome> <query protein id in reference genome>
<tag for target genome> <best hit protein id in target genome>

5.2. The output for the script should be in one line with the following four columns:

Output:
./16.fa.txt_orf12634_rev
EGEREDVV---RPRVRQKSSDGSS-----HEVDLATYRVAKDPGLLQAQIVQAIVS
./20.fa.txt_orf02832
EGRKYDLVVLDPPSFAKSSSNLESARRGYKEINLRAMRILKKPGVLVTSSCTQIVS

python code: parser_ex1_5.py
output files: example r16_20

6. Combine the best hits into one cluster file
6.1. Use the output of your BLAST parser as the input. Each query can have the best hits in different species.
6.2. Group them in one line as in the example below, in which each line represents a cluster of orthologs:
human_proteinl chicken_proteinXV mouse_proteinXX ...

Output file: orth_parsed_all
code: orth_parser_2.py

7. Select 10 different ortholog clusters such that all of your prokaryotic genomes are included and acquire the corresponding protein sequences from the multi FASTA files.
7.1. The result should be a number of multi FASTA files, one for each ortholog cluster.
7.2. The multi FASTA file should include all sequences appearing in the cluster of orthologs (including the query sequence).

Output files: FASTA_cluster*0-9
code FASTA_finder.py

Exercise 2 - Metagene Approach

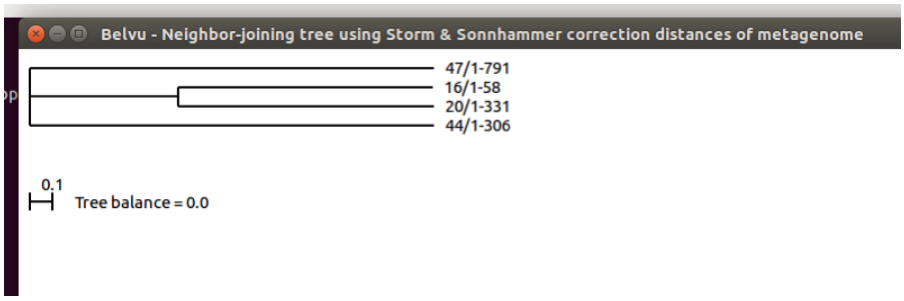
1. Make a multiple alignment of each of your multi FASTA cluster files in order to have one alignment for each cluster of orthologous genes.
1.1. Concatenate the alignments into a single, long metagene. The end result should be a single FASTA file with a sequence for each bacterial genome consisting of the aligned genes from each genome.

- First, an alignment with k-align was performed
After alignment with k-align file names: koutFASTA_cluster*0-9
- long metagenome build with code: metagenome.py
- metagenome output file: metagenome

1.2. Perform the tree reconstruction using Belvu. What does the tree look like?

belvu -T s metagenome

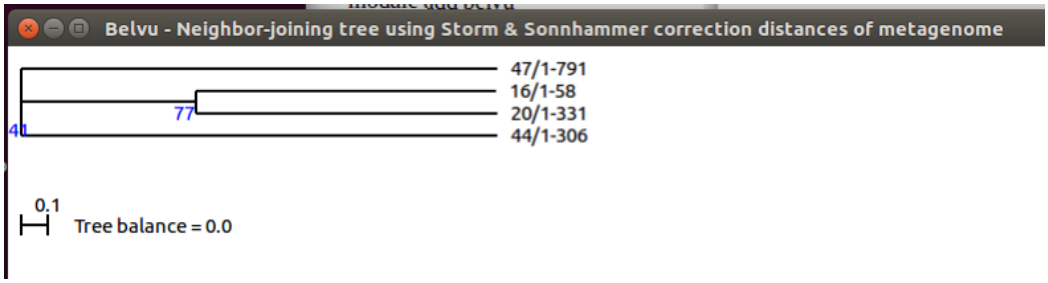
Tree reconstruction was performed with Neighbor-Joining amd the Storm & Sonnhammer distance correction. File name: tree_meta_noboot_dist_s



1.3. Perform sequence bootstrapping on the metagene. Evaluate the quality of the reconstruction.

Bootstrap-value of 100, Storm & Sonnhammer correction of distances
ssh -X uXXXX@studXX.dbb.su.se
module add belvu
belvu (4.)
belvu -b 100 -T n -T s filename

output tree-file: tree_meta_boot_dist_sf



The bootstrap values in the bootstrapped distance tree show us the reproducibility of the branch. The value of 77 on the second branch means, that this branch is reproducible for 77 percent. Both distance trees, non-bootstrapped and bootstrapped, look very similar, which means, that the non-bootstrapped tree is reliable, as it got verified via boostrapping.

Exercise 3 - Consensus Reconstruction
1. Perform a tree reconstruction using Belvu for each individual alignment.
1.1. Obtain 10 different trees, one for each ortholog cluster, in Newick format.

Tree files located in Trees subfolder.

ind_tree_dist_s*0-9.

1.2. How precise are those trees? Discuss.

It's not prudent to discuss the accuracy of these trees from a single alignment, a bootstrap has to be done to assess the precision.

1.3. Point a few specific genes or classes of the genes that cause disagreement.

Trees 3 and 7 differ from the other trees in general, with genes 20.fa.txt.orf2015 and 20.fa.txt.orf02887 causing some conflict compared with the other trees.

1.4. Discuss the reasons of this disagreement.

The ten different trees will definitely disagree because the orthologue clusters contain different genes to begin with, therefore potentially resulting in different trees, although some trees share similarities due to the genes coming overall from the same organisms.

2. Construct a consensus tree from the gene trees using Phylip example. For combining the tree, it is important that you use the species names as protein identifier in each tree.
2.1. Use Phylip consense to construct a consensus tree;
2.1.1. Put the tree files together into a file containing a list of the trees.
2.1.2. Name this file intree.
2.1.3. Phylip will read the file named intree in the present directory, ask you a few questions. Describe this process.

First the individual trees produced with Belvu were combined using the cat command, to produce a single tree file for all ten trees (cat_tree_clean, in Trees subfolder) . Some formatting was necessary to ensure that PHYLIP was able to read the file; ie., the species names were formatted from FASTA format to this format instead (44:0.556). The resulting file was called infile, and PHYLIP asked a few questions about the consensus options:

```
C        Consensus type (MRe, strict, MR, MI):  Majority rule (extended)
O                    Outgroup root:  No, use as outgroup species  1
R          Trees to be treated as Rooted:  No
T       Terminal type (IBM PC, ANSI, none):  ANSI
1          Print out the sets of species:  Yes
2       Print indications of progress of run:  Yes
3                    Print out tree:  Yes
4          Write out trees onto tree file:  Yes
```

The resulting tree file was viewed using the ETE toolkit.