Practical 3: Phylogenic Reconstruction
Group number: 2
Group members: Maximilian Senftleben, Zhong Hao Daryl Boey

Summary: This practical dealt with relations of genomes showing their degree of similarity. The tools being used were KALIGN for aligning the sequences and Belvu to reconstruct phylogenetic trees. First, the databases of the four genomes were created, which were further used by BLAST. Then, the 16S rRNA was used as a query and the databases of the genomes were searched. Second, a python code was used to parse the BLAST-output, which was in XML-format. There, the best hit was obtained. Third, KALIGN was used to align the four homologs, where the output was used to create distance trees with Belvu. In contrast, RaxML was used as another method using the maximum likelihood approach. Lastly, to apply bootstrapping, Belvu and the online tree-viewer tool were in use.

Exercise 1 - Finding homologs

For those of your genomes that are complete (i.e., bacteria and archaea), use BLAST to find homologs to the 16S rRNA *E. coli* gene. In case you get several hits, take the best one from each genome.

DNA Tree Reconstruction
1. In order to be able to search locally, format a BLAST database for your genomes.
    1.1. Explain the parameters. What do they mean?

    makeblastdb -in <inputfile.fa> -out <outputfile.fa> -dbtype nucl

    makeblastdb
    creates a blast database with the input file
    -in <inputfile.fa>
    specifies the input file
    -out <outputfile.fa>
    specifies the output files and their location
    -dbtype nucle
    specifies the content or the type of database, in this case, the database consists of nucleotides

    1.2. Run the program for each nucleotide dataset.

    makeblastdb -in <inputfile.fa> -dbtype nucl

    *for file in *.txt; do makeblastdb -in $file -out ../practical3/databases/$file -dbtype nucl; done*

    Output files: 16.fa.txt.nhr,nin,nsq for geneomes 16, 20, 44, 47

2. Gather all your genomes in one file in order to have a single database.

cat genome_0 genome_1 ... > genomes_all
Takes the content of the input files and adds them to a new file.

Output file: compl.nhr,nin,nsq

3. Use BLAST to query the database for the 16S rRNA file.

3.1. Find the best hit in each genome as “actual” 16S rRNA, and gather them as entries in a FASTA file.

blastn -outfmt 5 -query 16S_ecoli.fasta -db ../practical3/databases/47.fa.txt -out ../practical3/hit16.fasta -max_hsps 1

By using the flag *-max_hsps 1* the program blastn will only write the best n hits sequence into the output fasta-file. In this case, n=1, so only the best hit will be returned.

Outputfiles: hit16.fasta for 16, 20, 44, 47

3.2. Extract 16S sequence from BLAST results that you run against the whole genome.

3.2.1. What other parameters do you need? Explain their meanings.
-max_hsps 1.
Takes the n best hits, in this case n=1.
Outputfile: complete.txt

3.2.2. Where do you find the output? What do you expect to see in it?

In the output fasta-file, the first sequence is the query sequence (16S) and the second sequence is the best hit sequence. A XML-file is expected and given.

Exercise 2 - Parsing (Choose A or B)

B. A ready script (blastResultParser.py) is provided to you in order to parse the BLAST output in this option. You will discuss the following questions in detail:

I created an output file for each of the blast-output files with the given blastResultParser.py-script, which I modified in a way, so that the script creates a parsed outputfile:

Modification:

```
outputfile=open("out"+str(blastOutputXMLFile),"w")
for aSingleBlastRecord in listOfBlastRecords:

    for i in range (len (aSingleBlastRecord.alignments)):

        description = aSingleBlastRecord.descriptions [i]
        alignment = aSingleBlastRecord.alignments [i]
        title = re.compile ("gnl\IBL_ORD_ID\I\d* ").sub ("", description.title)

        print (">" + title          )
        print (alignment.hsps [0].sbjct)
        outputfile.write(">" + title)
        outputfile.write(alignment.hsps [0].sbjct)
outputfile.close()
```

Running script:

```
for file in hit*; do python3 ../blastResultParser.py $file ; done
```

Concenation:
cat outhit16.fasta outhit20.fasta outhit44.fasta outhit47.fasta > all_genomes.fasta
outputfile: all_genomes.fasta, code: blastResultParser.py

Exercise 3 - KALIGN

1. Use KALIGN on the resulting sequence file to make a multiple alignment of the homologs identified in the previous step.

kalign -gpo 60 -gpe 10 all_genomes.fasta kalign_out
outputfile: kalign_out

1.1. What are the gap penalties?

Gap penalties are score penalties assigned when a gap is introduced into the sequence alignment.

1.2. Does any of the gap penalties make any sense to apply? Discuss why.

Yes, the gap open and gap extension penalties are sensible to apply. This will ensure that the alignment will include the longest running sequences rather than short stretches, and avoid long gaps which may not be relevant.

Exercise 4 - Tree building

1. Investigate the various options of Belvu and how to create a tree from an alignment.

The main option to create a tree in Belvu is to use the flag -T. Multiple tree options are available, such as neighbour-joining and UPGMA. Therefore the input to create a flag in belvu is: belvu -T u <alignment file>.

1.1. Which distance correction methods does Belvu use?

There are 4 distance correction methods available in Belvu, with the various flags -T b, j, k, s, and r (uncorrected). B uses the scoredist distance correction method, which is default. J, k, and s select the Jukes-Cantor, Kimura, and Storm & Sonhammer distance correction methods respectively.

1.2. Use two different distance corrections in combination with two tree building methods. How does this affect the tree?

In this case, using both the Kimura and Storm & Sonhammer distance correction methods did not result in a differently shaped tree for the UPGMA.

Using the NJ method with scoredist and kimura distance correction also does not result in a differently shaped tree. However the NJ and UPGMA do result in different trees as UPGMA is rooted and NJ is unrooted.

2. Build a maximum likelihood tree with RaxML.
2.1. First make sure all gaps in the alignment are denoted with “-”

Yes.

2.2. Run e.g. raxmlHPC-PTHREADS-AVX -f a -x 54321 -N 100 -T 4 -p 12345 -m PROTCATBLOSUM62 -s input -n output.

The tree generated using this command is located at http://etetoolkit.org/treeview/?treeid=0754de6eb533ca12b8d67a78ad775de5&algid=ce443ed1e53858bf4e11d1e069c7a927".

2.3. Explain what the options do.

-PTHREADS selects the number of cores the script will run on, as the lab computers are multicore, using the -AVX option will allow the computer to run on multiple cores using the fastest vector instructions.

-f selects the algorithm used, and a in this case selects rapid bootstrap analysis, while searching for the best-scoring tree within a single program run.

-x is a random integer which is used as a seed number for the bootstrap.

-N represents the number of alternative runs, on different starting trees.

-T specifies the number of threads that will be ran.

-p represents a random seed number for the parsimony inferences, and can be utilised for debugging purposes.

-m specifies the substitution model, which in this case is BLOSUM62 for amino acids.

-s and -n specifies the input and output filepaths respectively.

2.4. Explain briefly what the main difference is between distance-based and maximum likelihood methods. What are their advantages/disadvantages?

Distance-based and maximum likelihood (ML) methods are critically different that while alignments are converted to distances in the former, ML does not. This results in a single tree with branches and distances for the former and possibly multiple trees for the latter. Therefore ML methods will consider evolutionary data through all steps of tree reconstruction while distance-based only uses the evolutionary data to compute distances. The main advantage of ML methods is that multiple trees are compared and all evolutionary data is used, as opposed to the more singular distance-based methods. The disadvantage of ML methods is the computational demands are much higher.

Exercise 5 - Sequence Bootstrapping

1. What is bootstrapping? What is the reason to apply bootstrapping?

Bootstrapping is a form of confidence testing, by sampling subsets from the dataset. The reason to apply it would be to verify the accuracy and reliability of the tree produced.

2. Construct a tree with bootstrap support values with Belvu from your alignment.

   Despite multiple attempts at updating and running Belvu 4.4.11 via SSH on various computers to multiple lab computers, Belvu failed to display an output when using the bootstrap flag. As a compromise we produced an output file (.tre) instead. The tree created with bootstrap value 4 is included as the attachment boot.tre.

   2.1. What does the option N mean?

   N refers to the number of samplings the program will go through to repeat the tree construction.

   2.2. What value of N do you choose and what consequences does that choice have?

   According to Zvelebil and Baum., 2008., the most appropriate N value to choose would be the number of data points N in the dataset. This will guarantee that the complete dataset is included in the selected data points.