Comparative Genomics 2018

Practical 3: Phylogenetic Reconstruction

Group 9
Martínez Hernández, Marina
Pérez Gómez, Fernando

Summary
In this practical we used bioinformatics tools to determine how closely related in evolution our genomes are. In order to do this, we looked at the 16S rRNA sequences in the different genomes and compared them against each other. After creating databases for each of our genomes, used BLAST to identify the best hit to the 16S rRNA sequence in E. coli. Once we had these sequences, we performed a multiple sequence alignment with the KALIGN tool and built a tree afterwards to obtain a visualization of the distribution of the genomes according to their evolutionary similarities. Finally, we applied bootstrapping for a higher reliability of the results represented by the tree.

Key questions to answer
1. Finding homologs
2. Parsing
3. KALIGN
4. Tree building
5. Sequence Bootstrapping

Attachments
None.

Exercise 1 – Finding homologs

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?

-in is the flag used to denote the input file.
-dbtype is the flag used to indicate the if the input is going to be nucleotides (nucl) or proteins (prot).
-outfmt is the flag used to denote the alignment view options. In this case, -outfmt 5 will display an XML BLAST output.
-query is the flag used to indicate the name of the query file.
-db is the flag used to indicate the name of the BLAST database.
-out is the flag that can be used to give a name to the output database that will be created.

1.2. Run the program for each nucleotide dataset.

This was done with the following command:

Makeblastdb -in /afs/pdc.kth.se/misc/pdc/volumes/sbc/prj.sbc.dmessina.5/Comparative_Genomics/data/genomes2018/Grp9/04.fa.txt -dbtype nucl -out db04

And so on with the rest of the files… (16.fa.txt, 18.fa.txt, 49.fa.txt). 34.fa.txt is a eukaryotic genome, so it was not necessary to do it with that one.

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

This was done with the following command:

cat /afs/pdc.kth.se/misc/pdc/volumes/sbc/prj.sbc.dmessina.5/Comparative_Genomics/data/genomes2018/Grp9/04.fa.txt /afs/pdc.kth.se/misc/pdc/volumes/sbc/prj.sbc.dmessina.5/Comparative_Genomics/data/genomes2018/Grp9/16.fa.txt /afs/pdc.kth.se/misc/pdc/volumes/sbc/prj.sbc.dmessina.5/Comparative_Genomics/data/genomes2018/Grp9/18.fa.txt /afs/pdc.kth.se/misc/pdc/volumes/sbc/prj.sbc.dmessina.5/Comparative_Genomics/data/genomes2018/Grp9/49.fa.txt > genomes_all

makeblastdb -in genomes_all -dbtype nucl -out dbALL

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.

This was done with the following command:

blastn -outfmt 5 -query /afs/pdc.kth.se/misc/pdc/volumes/sbc/prj.sbc.dmessina.5/Comparative_Genomics/data/16S_ecoli.fasta -db db04 -out blast04result

And so on with the rest of the files… (16.fa.txt, 18.fa.txt, 49.fa.txt).

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

This was done with the following command:

blastn -outfmt 5 -query /afs/pdc.kth.se/misc/pdc/volumes/sbc/prj.sbc.dmessina.5/Comparative_Genomics/data/16S_ecoli.fasta -db dbALL -out blastALLresult

3.2.1. What other parameters do you need? Explain their meanings.

We also need the additional parameters -outfmt, -query, -db and -out. All these parameters are explained in exercise 1.1.

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

In my home directory (as I did not specify any other route to save it) and with an XML BLAST output format because of the -outfmt flag that we used. I expect to see in it a summary with the best hits. The number of hits expected to be obtained are the sum of the individual hits done in the 4 genomes in exercise 3.1. I compared them and, as expected, all previously obtained individual hits appear in the BLAST results run against the whole genome. Also, the score of each hit appears, which is very useful for comparison between the different hits. Other relevant information is also contained in the report of the BLAST output: gaps, length of the alignment, etc.

Exercise 2 – Parsing

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:

a. How does the script choose the single best BLAST hit in each genome in the database?

First, the sys module is imported for using the sys.argv tool, which enters to any command line arguments. Thus, sys.argv=list of command line arguments, being sys.argv[0] the name of the script. Looking at the length of sys.argv (len(sys.argv)) we can get the number of command line arguments.

Then, the script proceeds to open the BLAST report in XML format. NCBIXML.parse() returns an iterator. This tool let us go through the BLAST output, getting back individual BLAST records: it screens them one by one for each result given by BLAST. This is done by means of a for loop in the script (for

aSingleBlastRecord in listOfBlastRecords:). This program screens the BLAST records contained in the report one, thus for each record it shows (prints) the information that we are interested in.

At the end, it compiles the needed information and displays the alignment that is stored in the first position of the hsp argument, which corresponds to the single best BLAST hit.

b. What does a BLAST record correspond to?

It corresponds to a BLAST output search in which all the information that has been obtained is contained within different classes according to their nature. There are different classes, and depending on them different information will be stored. For example, the alignment class stores information about one alignment hit, meanwhile the HSP class contains information about one hsp in an alignment hit and so one with the different group of classes.

c. What is assumed about the BLAST XML output?

The BLAST XML outputs is assumed to contain several results and all the attributes that are required by BLAST files so that the program can work correctly. Also, the output is supposed to be ordered from best to lower-scoring hits. This is why the program takes the alignment that appears in position [0] within the hsp argument.

d. What does this script output?

It outputs a FASTA file that contains the best hit for each of the genome files introduced as queries. For example, for the 04-genome file, we can use the command below to parse it and get the FASTA file with the best hit:

python3 ./Desktop/Comparative_genomics/Practical3/blastResultParser.py blast04result > parsed04hit

>./04.fa.txt
GATTAGCTAGTTGGTGAGGTA-
ATGGCTCACCAAGGCGACGATCAGTAGCTGGTCTGAGAGGATGATCAGCCACATTGGGACTGAGACACGGCCCAAACTCCTACGGGAGGCA
GCAGTGGGGAATATTGGACAATGGGGGCAACCCTGATCCAGCCATGCCGCGTGAGTGATGAAGGCCCTAGGGTTGTAA-AGC-TCTTTT-
GTGCGG-G-AAG—A-TAATG--A---C----G------G---T-ACC-
GCAAGAATAAGCCCCGGCTAACTTCGTGCCAGCAGCCGCGGTAATACGAAGGGGGCTAGCGTTGCTCGGAATCACTGGGCGTAAAGGGTGC
GTAGGCGGGTCTTTAAGTCAGGGGTGAAATCCTGGAGCTCAACTCCAGAACTGCCTTTGATACTGAGGATCTTGAGT-
TCGGGAGAGGTGAGTGGAACTGCGAGTGTAGAGGTGAAATTCGTAGATATTCGCAAGAACACCAGTGGCGAAGGCGGCTCACTGGCCCGAT
ACTGACGCTGAGGCACGAAAGCGTGGGGAGCAAACAGGATTAGATACCCTGGTAGTCCACGCCGTAAACGATGAATGCCA-
GCCGTTAGTGGGTTT-ACTCACTAG-TGG-CGCAGCTAACGCTTTAAG-
CATTCCGCCTGGGGAGTACGGTCGCAAGATTAAAACTCAAAGGAAT-T-G-
ACGGGGGCCCGCACAAGCGGTGGAGCATGTGGTTTAATTCGACGCAACGCGCAGAACCTTACCAGCC-CTTG-ACATGTCCAG-
GACCGGTCGCAGAGATGTGACCCTCTCTTCGG-
AGCCTGGAACACAGGTGCTGCATGGCTGTCGTCAGCTCGTGTCGTGAGATGTTGGGTTAAGTCCCGCAACGAGCGCAACCCCCGTCCTTAGT
TGCTAC-CATTTAGTT-
GAGCACTCTAAGGAGACTGCCGGTGATAAGCCGCGAGGAAGGTGGGGATGACGTCAAGTCCTCATGGCCCTTACGGGCTGGGCTACACACG
TGCTACAATGGCGGTGACAATGGGATGCTAAGGGGCGACCCTTCGCAAATCTCAAAAAGC-
CGTCTCAGTTCGGATTGGGCTCTGCAACTCGAGCCCATGAAGTTGGAATCGCTAGT-AATCGTGGATCAGCACGC-
CACGGTGAATACGTTCCCGGGCCTTGTACACACCGCCCGTCACACCATGGGAGTTGGTTTTACCTGAAGACGGTGCGCT-
AACCCGCAAGGGAGGCAGCCGGCCACGGTAGGGT-CA-GCGACTGGGGTGAAGTCGTAACAAGGTAGCCGTAGGGGAACCTGCGGCTGGAT

Exercise 3 - KALIGN

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

In order to do this, the output files from the previous step were first merged into a single file. Then, this file was input to kalign and this generated a multiple alignment of the homologs that we had previously identified and created a file where this alignment is stored (KALIGNresult).

1.1. What are the gap penalties?

As displayed in the terminal window, the gap penalties applied to the alignment are:

217.00000000 *gap open penalty*
39.40000153 *gap extension*
292.60000610 *terminal gap penalty*

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

Since insertions and deletions are uncommon events in nature, it makes sense that gaps are counted as penalties when scoring the alignment between sequences. When these insertions and deletions do happen, they usually involve several adjacent nucleotides rather than many single ones. Thus, it also makes sense that gap extensions are penalized, although not as harsh as the gap openings. Regarding the terminal gap penalty, it does not make sense to apply them here since we are performing local alignments instead of global ones, where it would make more sense to apply these penalties.

Exercise 4 – Tree building

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

1.1. Which distance correction methods does Belvu use?

There are different correction methods in Belvu. The default distance correction method is Scoredist. On the other hand, it is possible to use 3 different distance correction methods: Jukes-Cantor, Kimura and Storm & Sonnhammer.

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

NOTE: First, the Kalign output file was modified and the headers containing the name of each alignment were modified: ./ removed, otherwise it did not work! Eg: >./04.fa.txt → >04.fa.txt

First tree building method: UPGMA

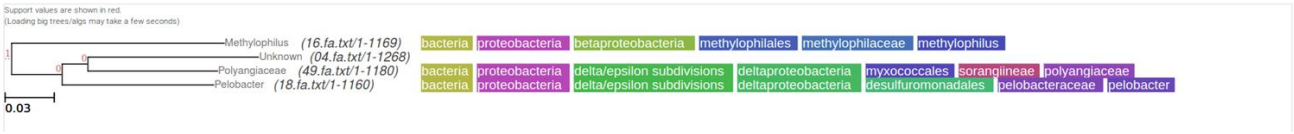
- With Scoredist distance correction (default):

- With Storm and Sonnhammer distance correction:

<u>Second tree building method: Neighbor-joining</u>

- With Scoredist distance correction (default):

- With Storm and Sonnhammer distance correction:



It seems that the distance correction does not change the tree at all. We have also tried other distance corrections apart from the default and the Sonnhammer ones, such as Kimura and Jukes-Cantor distance corrections, but the tree is not affected by these changes.

2. Build a maximum likelihood tree with RaxML

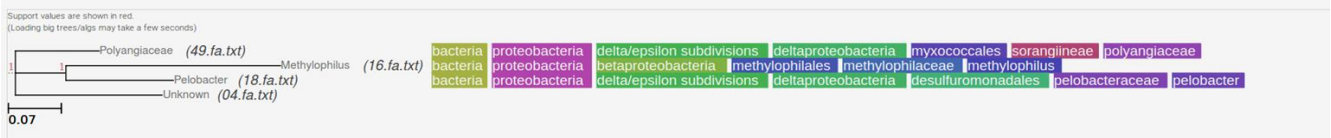
2.1. First make sure all gaps in the alignment are denoted with "-"

All the gaps in the alignment are already denoted with "-".

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

Tree results viewed through ETE tree viewer.

Command:
raxmlHPC-PTHREADS-AVX -f a -x 54321 -N 100 -T 4 -p 12345 -m PROTCATBLOSUM62 -s Kalign_nonames_UPGMA -n RaxmlHPC_UPGMA



2.3. Explain what the options do.

-f a: this flag is used to select the algorithm we want to be executed. In this case, -f a indicates that a Bootstrap analysis will be performed by which the maximum likelihood trees with the best scores will be chosen.
-x 54321: this flag is used to indicate the start of a rapid bootstrapping algorithm.
-T 4: this flag is used to indicate the number of threads that the user wants to run.
-p 12345: this flag is basically used to ease the debugging of the program and set a number for parsimony inferences that will be useful for the user to reproduce the results.
-s: this flag is used to indicate the name of our input file; in this case, the name of the file where the alignments are contained.
-n: this flag is used to indicate the name of the file that will be output by the program.

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

Distance-based methods such as UPGMA and Neighbor joining are computationally quick algorithms to generate phylogenetic trees, but they are not as reliable as maximum likelihood methods. Distance-based methods do not perform well when there is a large divergence time between sequences, and this is one of the main reason why they are rarely used for scientific publications. Instead, maximum likelihood methods may take longer than distance based-methods but since they are based on evolution models, the results they provide are reliable and suitable for scientific publications.

Exercise 5 - Sequence Bootstrapping

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

It is an analysis described in: Efron and Gong, 1983; Felsenstein, 1985; Swofford et al., 1996. It consists in sampling again your estimated data with replacements so that you can construct a series of bootstrap fragments that are the same size as the initial data. Here, the data that is sampled again is the amino acids contained in the sequence and the statistical significance corresponding to a cluster is derived from the number of portions in the tree that contain that encompass that cluster.

The reason to use bootstrapping is because it is like a statistical analysis that is used to examine the trustworthiness of specific branches that are included in an evolutionary tree under study. It gives us a measure of reliability!

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

2.1. What does the option N mean?

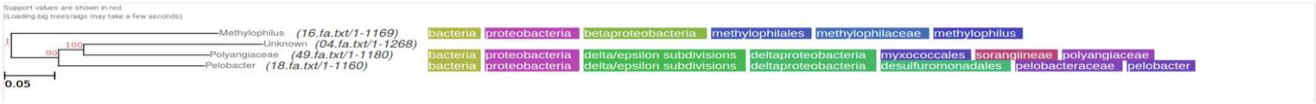
N is the number of bootstrap samples.

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

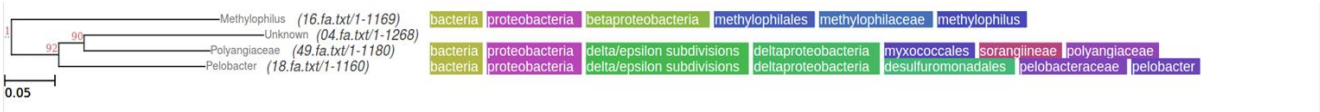
With N=10	*With N=100*	*With N=100000*
16.fa.txt/1-1169:0.206,	*16.fa.txt/1-1169:0.206,*	*(16.fa.txt/1-1169:0.206,*
04.fa.txt/1-1268:0.178,	*04.fa.txt/1-1268:0.178,*	*04.fa.txt/1-1268:0.178,*
49.fa.txt/1-1180:0.125)	*49.fa.txt/1-1180:0.125)*	*49.fa.txt/1-1180:0.125)*
100:0.024,	*90:0.024,*	*87:0.024,*
18.fa.txt/1-1160:0.145)	*18.fa.txt/1-1160:0.145)*	*18.fa.txt/1-1160:0.145)*
90:0.046)	*92:0.046)*	*93:0.046)*

We have tested several values for N. The higher the N the more accurate results, but slower process.

For N=10:



For N=100:



For N=1000: