

# Practical-4 : Phylogenomics

Stefanie Friedrich and Siddharth Tomar

May 15, 2017

## Summary

In Practical 4 we identified orthologs in our four complete genomes and created phylogenetic trees showing the evolution of our genomes (orthologs); the longest genome (15.fa) was set as reference genome for the ortholog search. After creating the BLAST database to predict the genes we linked these predicted genes to genes in our reference genome. Furthermore, we approached the problem of phylogenetic tree construction in two ways: for whole genomes and for individual orthologs. For the whole genome tree construction, we created one metagene file with the three aligned genomes containing 10 (and 30) orthologs and built the tree. For the tree construction for individual ortholog clusters, we choose all predicted and linked genes and created a tree for each cluster. Three ortholog cluster disagreed with the majority of trees. Creating the consensus tree of all trees built from each cluster, the consensus tree looks different from our metagene tree with bootstrapping. (Refer to Figure 3 for workflow)

## Sequences

Group 11

- 05.fa.txt
- 11.fa.txt
- 15.fa.txt
- 19.fa.txt
- 34.fa.txt

## Orthologous gene datasets

*Reference genome: 15.fa*

### Task 1

For all of your prokaryotic genomes (including reference) find multi-fasta files with proteins from one of previous exercises.

Done with given script `'parseGlimmery.py *.fa *glimmer.predict'` and an R script using package `seqinr` and considering reverse strands (**some DNA sequences contained errors and were skipped**), and finally, `cat *translatedProteins.fasta > proteomes.fasta`.

### Task 3

Create a BLAST database for you proteomes

- (a) In previous exercises you used `makeblastdb` to create a database for `blastn`, repeat that for all of your selected proteomes individually (this time indexed file is of protein type).

Done with

```
makeblastdb -in *.translatedProteins.fasta -dbtype prot -out *translatedProteins.fasta_BLAST.db
```

## Task 4

Do a blast p (for proteins) search against your reference proteome (against one selected proteome)

Done with

```
blastp -outfmt 5 -num_alignments 1 -query 15.proteins.fasta -db 05.proteins.fasta -out <output file>
```

We added `-num_alignments 1` to get only the strongest hit with the highest bit-score

## Task 5

Modify your script from the previous lab to parse the XML output.

Script (Instead of passing user defined variables, the script parses name according to file name defined inside) :

```
1 from Bio.Blast import NCBIXML
2
3 #Define file names which are to be parsed
4 filename = ["05.xml", "11.xml", "19.xml"]
5
6 #Driving loop
7 for i in filename:
8     bout = open(i)
9     b_records = NCBIXML.parse(bout) #Main parsing function
10    outfile = open("record_%s.record" % i, 'w') #Each parsed sequence is stored in a new file
11    for b_record in b_records:
12        for alig in b_record.alignments:
13            for hsp in alig.hsps:
14                print ("15"+"\t"+alig.hit_def+"\t"+i+"\t"+b_record.query+"\t"+str(hsp.bits)+"\n")
15                outfile.write ("15"+"\t"+alig.hit_def+"\t"+i+"\t"+b_record.query+"\t"+str(hsp.bits)+"\n")
16    outfile.close()
```

## Task 6

Combine the best hits into one cluster file.

Script:

```
1 import pandas as pd
2
3 df05 = pd.DataFrame.from_csv('record_05.xml.record', sep='\t', index_col=None, header = None)
4 df11 = pd.DataFrame.from_csv('record_11.xml.record', sep='\t', index_col=None, header = None)
5 df19 = pd.DataFrame.from_csv('record_19.xml.record', sep='\t', index_col=None, header = None)
6 df05.columns = ["Reference", "Reference_Name", "Target_05", "Target_Name_05", "Bit_Score"]
7 df11.columns = ["Reference", "Reference_Name", "Target_11", "Target_Name_11", "Bit_Score"]
8 df19.columns = ["Reference", "Reference_Name", "Target_19", "Target_Name_19", "Bit_Score"]
9 df05 = df05.sort('Bit_Score', ascending=False)
10 df11 = df11.sort('Bit_Score', ascending=False)
11 df19 = df19.sort('Bit_Score', ascending=False)
12 df05 = df05.head(300)
13 df11 = df11.head(300)
14 df19 = df19.head(300)
15
16
17
18 #s1 = pd.merge(df05, df11, df19, on=['Reference_Name'], how='inner')
19 s1 = pd.merge(pd.merge(df11, df19, on='Reference_Name', how='inner'), df05, on='Reference_Name',
20               how='inner')
21 #We need to remove the duplicates
22 s1 = s1.drop_duplicates(subset='Reference_Name', keep='first', inplace=False)
23 s1 = s1.drop_duplicates(subset='Target_Name_05', keep='first', inplace=False)
24 s1 = s1.drop_duplicates(subset='Target_Name_11', keep='first', inplace=False)
25 s1 = s1.drop_duplicates(subset='Target_Name_19', keep='first', inplace=False)
26 del s1['Reference_x']
27 del s1['Target_11']
28 del s1['Bit_Score_x']
29 del s1['Reference_y']
30 del s1['Target_19']
31 del s1['Bit_Score_y']
32 del s1['Reference']
33 del s1['Target_05']
34 del s1['Bit_Score']
35 s1 = s1.head(10)
36 s1.to_csv('condensed.csv')
```

## Task 7

Select 10 different ortholog clusters such that all your prokaryotic genomes are included and acquire the corresponding protein sequences from the multifasta files.

Script:

```
1 import pandas as pd
2 import numpy as np
3 from Bio import SeqIO
4
5 #File input operations
6 df = pd.read_csv('condensed.csv')
7 sep = ' '
8 #Iterate over rows in dataframe
9 for i, row in df.iterrows():
10     fasta05 = SeqIO.parse("05.fa", "fasta")
11     fasta11 = SeqIO.parse("11.fa", "fasta")
12     fasta19 = SeqIO.parse("19.fa", "fasta")
13     outfile = open("multi-%s.fa" % i, 'w')
14     protein05 = (row['Target_Name_05']).split(sep, 1)[0]
15     protein11 = (row['Target_Name_11']).split(sep, 1)[0]
16     protein19 = (row['Target_Name_19']).split(sep, 1)[0]
17     for record in fasta05:
18         if (protein05 == record.id):
19             print("i"+"\n")
20             outfile.write(">" + record.id + "\n" + (str(record.seq)).replace("*", " ") + "\n")
21     for record in fasta11:
22         if (protein11 == record.id):
23             outfile.write(">" + record.id + "\n" + (str(record.seq)).replace("*", " ") + "\n")
24     for record in fasta19:
25         if (protein19 == record.id):
26             outfile.write(">" + record.id + "\n" + (str(record.seq)).replace("*", " ") + "\n")
27     outfile.close()
```

## Metagene approach

### Task 8

Make a multiple alignment for each of your multifasta cluster files (so you have one alignment for each cluster of orthologous genes).

Done with:

```
for i in *.fa ; do kalign $i "${i%.*}_aligned.fa" ; done
```

- (a) Concatenate the alignments into a single, long metagene.

Script:

```
1 import pandas as pd
2 import numpy as np
3 from Bio import SeqIO
4
5 #counter
6 counter = 0
7 #number of files
8 maxFile = 10
9 #metagene variables
10 meta1 = ""
11 meta1_id = ""
12 meta2 = ""
13 meta2_id = ""
14 meta3 = ""
15 meta3_id = ""
16 #File output pointer
17 outfile = open("meta.fa", 'w')
18 #File input/concatination operations
19 for i in range(0, maxFile):
20     records = list(SeqIO.parse("%s.fa" % i, "fasta"))
21     meta1_id = meta1_id + "|" + records[0].id
22     meta1 = meta1 + str(records[0].seq)
23     meta2_id = meta2_id + "|" + records[1].id
24     meta2 = meta2 + str(records[1].seq)
```

```

26 meta3_id = meta3_id+"|"+records[2].id
27 meta3 = meta3+str(records[2].seq)
28
29 outfile.write(">"+meta1_id+"\n"+meta1+"\n">"+meta2_id+"\n"+meta2+"\n">"+meta3_id+"\n"
    "+meta3+"\n")
30 outfile.close()

```

- (c) Perform tree reconstruction using Belvu. What is the tree like?

Refer to Figure 1 for tree.

- (d) Perform sequence bootstrapping on the metagene. Can you say anything on the quality of the reconstruction ?

Done with belvu:

```

-b -1000 -B -o tree metagene.fa > metagene_tree.csv
( ( 11.fa:0,303,
19.fa:0,313)
0:0,132,
05.fa:0,440) ;

```

In general, bootstrapping improves tree constructing immensely. To test if we have coincidentally chosen 10 wrong proteins, we re-run the alignment with 30 proteins and the tree looks the same:

```

( ( 11.fa:0,345,
19.fa:0,316)
0:0,114,
05.fa:0,445) ;

```

## Consensus reconstruction

### Task 9

Perform tree reconstruction using belvu for each individual alignment

- (a) You should get ten different trees (one for each ortholog cluster) in Newick format.

Done with

```
for f in multifasta_aligned.fa ; do belvu -b -1000 -B -out tree $f > ${f%.*}_tree.csv ;
```

- (b) How precise are those trees ?

Each of the ten tree presents the relation of our genomes concerning one cluster (i.e. query gene). The majority of the trees do agree concerning topology.

- (c) Can you point a few specific genes or classes of genes that cause disagreement ?

Three of the 37 trees have a divergent tree structure but each of these three trees contains the same genes from the query genomes:

05.fa: chaperone protein ClpB

11.fa: ClpV1 family T6SS ATPase

19.fa: ATP-dependent chaperone ClpB

- (d) How do you think that happens ?

We think that different reason might cause this. One possibility is the different origin and evolution of these genes compared to the other orthologs. We also can think of shared domains that were aligned to genes in our reference genome 15.fa but the complete genes are not related.

## Task 10

Construct a consensus tree from the gene trees using Phylip example  
Refer to Figure 2 for tree.

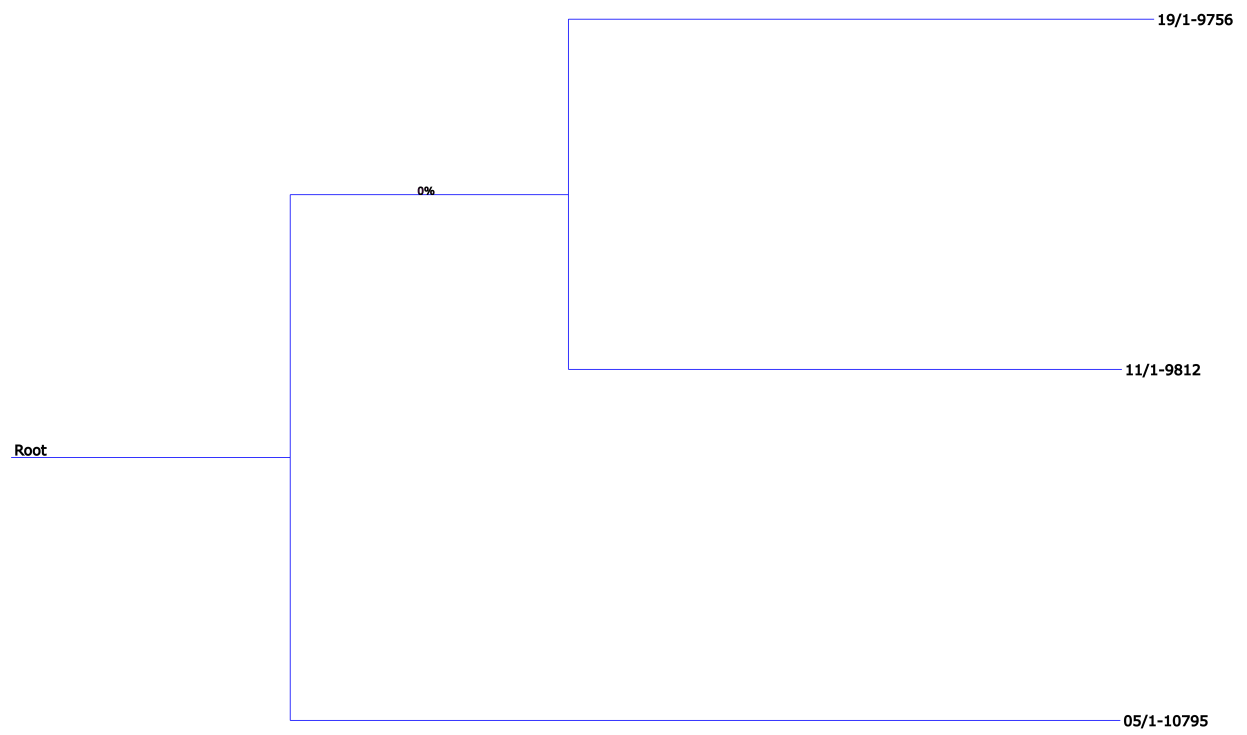


Figure 1: Tree for metagene

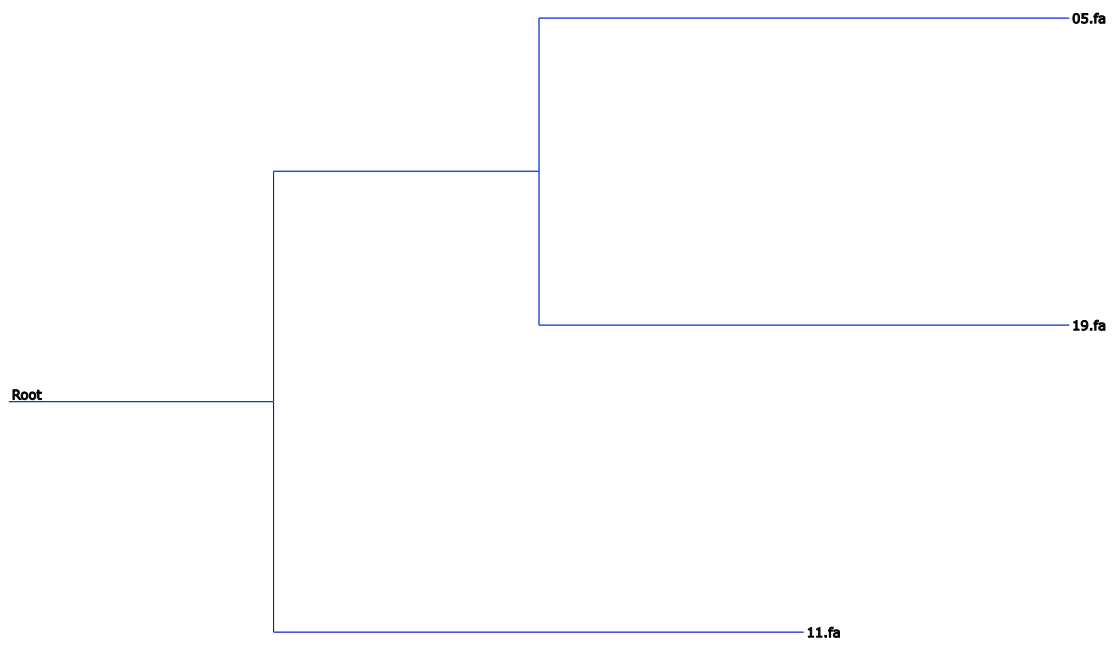


Figure 2: Tree from consensus

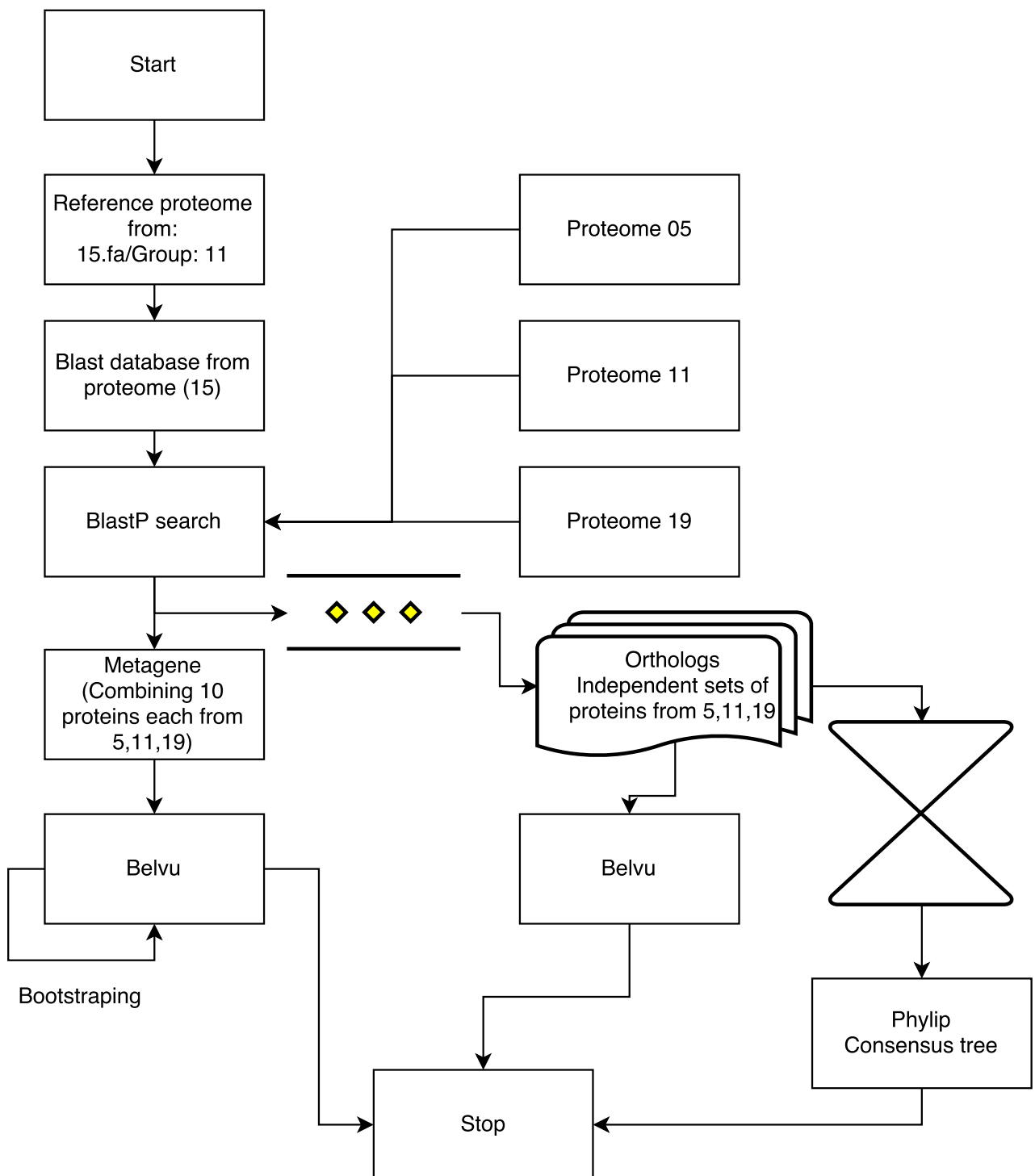


Figure 3: Workflow