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, this consensus tree looks different from our metagene tree with bootstrapping. In general, we are more confident concerning a tree with bootstrapping than a consensus tree. (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) :

```python
from Bio.Blast import NCBIXML

#Define file names which are to be parsed
filename = ["05.xml","11.xml","19.xml"]

#Driving loop
for i in filename:
  bout = open(i)
  b_records = NCBIXML.parse(bout) #Main parsing function
  outfile = open("record_%s.record" % i,'w') #Each parsed sequence is stored in a new file
  for b_record in b_records:
    for alig in b_record.alignments:
      for hsp in alig.hsps:
        print ("15"+"\t"+alig.hit_def+"\t"+i+"\t"+b_record.query+"\t"+str(hsp.bits)+"\n")
        outfile.write ("15"+"\t"+alig.hit_def+"\t"+i+"\t"+b_record.query+"\t"+str(hsp.bits)+"\
    n")
  outfile.close()
```

Task 6

Combine the best hits into one cluster file.
Script:

```python
import pandas as pd

df05 = pd.DataFrame.from_csv('record_05.xml.record', sep='\t', index_col=None, header = None)
df11 = pd.DataFrame.from_csv('record_11.xml.record', sep='\t', index_col=None, header = None)
df19 = pd.DataFrame.from_csv('record_19.xml.record', sep='\t', index_col=None, header = None)
df05.columns = ["Refrence", "Refrence_Name", "Target_05", "Target_Name_05", "Bit_Score"]
df11.columns = ["Refrence", "Refrence_Name", "Target_11", "Target_Name_11", "Bit_Score"]
df19.columns = ["Refrence", "Refrence_Name", "Target_19", "Target_Name_19", "Bit_Score"]
df05 = df05.sort('Bit_Score', ascending=False)
df11 = df11.sort('Bit_Score', ascending=False)
df19 = df19.sort('Bit_Score', ascending=False)
df05 = df05.head(300)
df11 = df11.head(300)
df19 = df19.head(300)



#s1 = pd.merge(df05, df11, df19, on=['Refrence_Name'], how='inner')
s1 = pd.merge(pd.merge(df11,df19,on='Refrence_Name', how = 'inner'),df05,on='Refrence_Name',
    how = 'inner')
s1.to_csv('full.csv')
#We need to remove the duplicates
s1 = s1.drop_duplicates(subset='Refrence_Name', keep='first', inplace=False)
s1 = s1.drop_duplicates(subset='Target_Name_05', keep='first', inplace=False)
s1 = s1.drop_duplicates(subset='Target_Name_11', keep='first', inplace=False)
s1 = s1.drop_duplicates(subset='Target_Name_19', keep='first', inplace=False)
del s1['Refrence_x']
del s1['Target_11']
del s1['Bit_Score_x']
del s1['Refrence_y']
del s1['Target_19']
del s1['Bit_Score_y']
del s1['Refrence']
del s1['Target_05']
del s1['Bit_Score']
s1 = s1.head(10)
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:

```python
import pandas as pd
import numpy as np
from Bio import SeqIO

#File input operations
df = pd.read_csv('condensed.csv')
sep = ' '
#Iterate over rows in dataframe
for i, row in df.iterrows():
  fasta05 = SeqIO.parse("05.fa", "fasta")
  fasta11 = SeqIO.parse("11.fa", "fasta")
  fasta19 = SeqIO.parse("19.fa", "fasta")
  outfile = open("multi_%s.fa" % i,'w')
  protein05 = (row['Target_Name_05']).split(sep, 1)[0]
  protein11 = (row['Target_Name_11']).split(sep, 1)[0]
  protein19 = (row['Target_Name_19']).split(sep, 1)[0]
  for record in fasta05:
    if (protein05 == record.id):
      print("i"+"\n")
      outfile.write (">"+record.id+"\n"+(str(record.seq)).replace("*", "")+"\n")
  for record in fasta11:
    if (protein11 == record.id):
      outfile.write (">"+record.id+"\n"+(str(record.seq)).replace("*", "")+"\n")
  for record in fasta19:
    if (protein19 == record.id):
      outfile.write (">"+record.id+"\n"+(str(record.seq)).replace("*", "")+"\n")
  outfile.close()
```

Metagene approach

Task 8

Make a multiple alignment for each of your mulitfasta 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:

```python
import pandas as pd
import numpy as np
from Bio import SeqIO


#counter
counter = 0
#number of files
maxFile = 10
#metagene variables
meta1 = ""
meta1_id = ""
meta2 = ""
meta2_id = ""
meta3 = ""
meta3_id = ""
#File output pointer
outfile = open("meta.fa",'w')
#File input/concatination operations
for i in range (0,maxFile):
  records = list(SeqIO.parse("%s.fa" % i, "fasta"))
  meta1_id = meta1_id+"|"+records[0].id
  meta1 = meta1+str(records[0].seq)
  meta2_id = meta2_id+"|"+records[1].id
  meta2 = meta2+str(records[1].seq)
  meta3_id = meta3_id+"|"+records[2].id
  meta3 = meta3+str(records[2].seq)

outfile.write (">"+meta1_id+"\n"+meta1+"\n"+">"+meta2_id+"\n"+meta2+"\n"+">"+meta3_id+"\n
    "+meta3+"\n")
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.

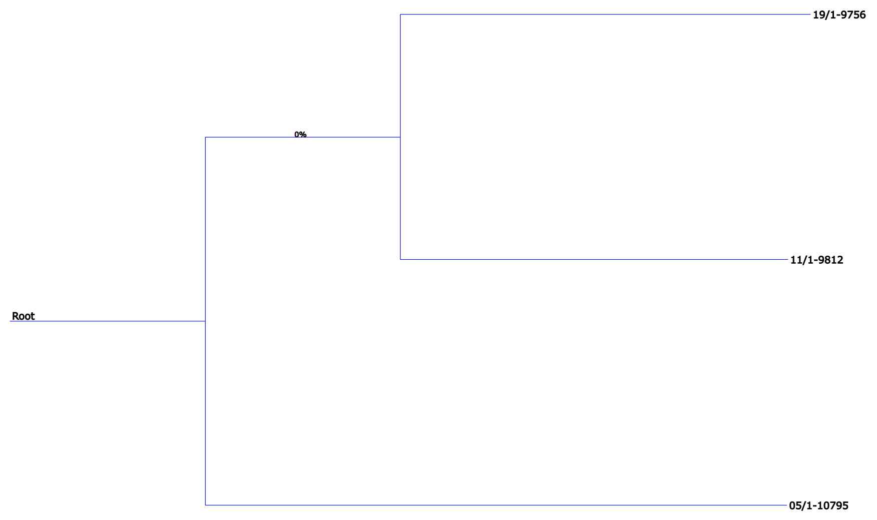

Figure 1: Tree for metagene

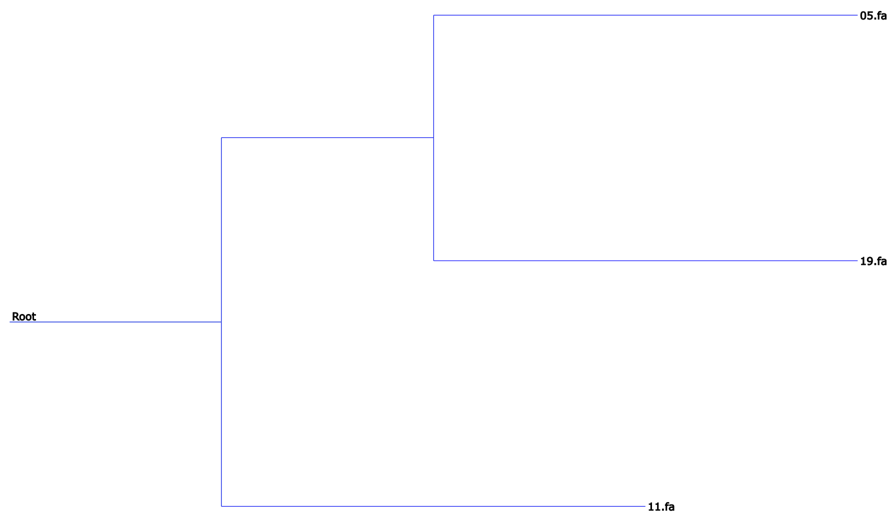

Figure 2: Tree from consensus

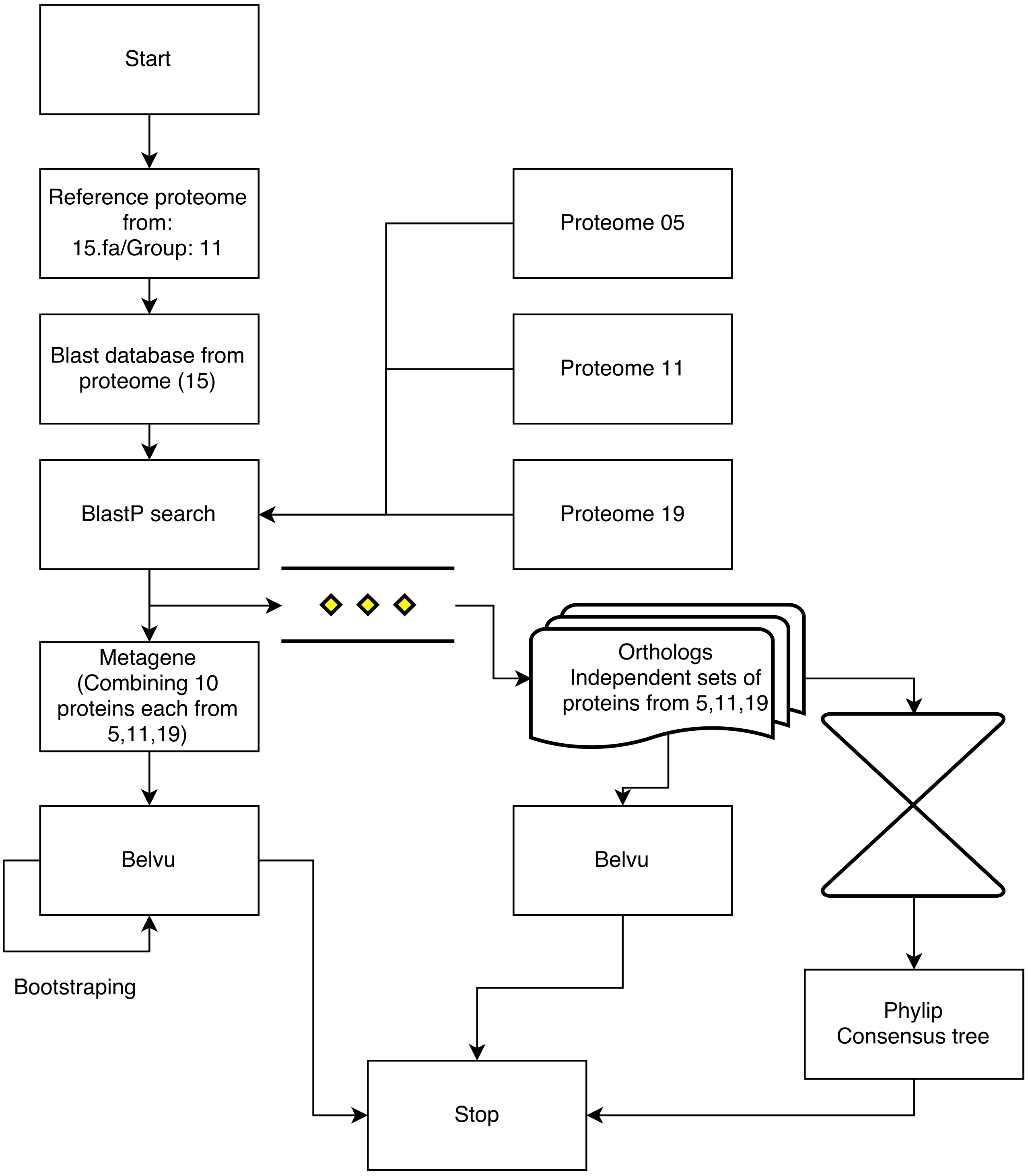
Start
Reference proteome from: 15.fa/Group: 11
Blast database from proteome (15)
BlastP search
Proteome 05
Proteome 11
Proteome 19
Metagene (Combining 10 proteins each from 5,11,19)
Orthologs Independent sets of proteins from 5,11,19
Belvu
Bootstraping
Belvu
Phylip Consensus tree
Stop

Figure 3: Workflow