**Group 7:**

**Ryno Lawson**

**Vasileios-Evripidis Kyriakidis**

**Comparative Genomics 2017**

**Practical 5 Report : Gene Order Analysis**

**Summary**

The main aim for this exercise was to create pseudogenes and then create dot plots by using Dotter and learn how to interpret the results. Then the next step was to reconstruct phylogenetic trees with the use of GRIMM which performs multiple genome rearrangements and search for differences between the tree it provides and the tree we got from practical 4.

**Step 1: Creating Gene Order dataset**

For this task we used an already existing python script (getGeneOrder.py) which we needed to edit in order to make it compatible to our data and our output requirements. After running that code we got 4 different output files with gene order datasets, one for each of our prokaryotic query genomes. The code we used was the following and **only works in python2** :

**####### working\_getGeneOrder.py #######**

import sys, re  
  
# acquire first needed data  
  
geneOrderList = []  
  
aHandle = open (sys.argv [1]) # input ortholog cluster file  
  
lines = aHandle.readlines ()  
 #reading lines of orthologs file

for aLine in lines:  
   
 aLine = aLine.replace ("\n", "")  
  
 if aLine.startswith (">"):  
   
 print aLine [0:len (aLine)],  
 geneOrderList.append (aLine [0:len (aLine)])  
  
# acquire second needed data  
  
partOfCluster = {}  
  
bHandle = open (sys.argv [2])   
  
lines = bHandle.readlines ()  
  
id = 0  
  
for aLine in lines:   
 #print aLine  
 aLine = aLine.replace ("\n", "")  
 words = aLine.split (" ")  
 for aWord in words:  
 #print aWord,  
 if not partOfCluster.has\_key (aWord): #lookings the gene in the proteosomes  
 partOfCluster [aWord] = id  
   
 id = id + 1  
  
# put together  
output = open ('geneOrder12.txt', 'w+')  
  
for aGene in geneOrderList:  
   
 if partOfCluster.has\_key (aGene):  
 print partOfCluster [aGene],  
 else:  
 continue   
 print >> output, partOfCluster [aGene],  
  
output.close()

#####################

So, when we input the following command we get the the gene order files mentioned above :

*$ python2 working\_getGeneOrder.py ortholog \*.fa.txt.pfa*

If the gene is clustered more than once it could be repeats in the gene order. This would be ambiguous due to a gene being homologous to multiple genes. This script does not take the direction into account so it can’t handle the forward and the reverse strand. Changes would need to be made to the script to run it backwards.

**Step 2: Generating dot plots with Dotter**

For this task first we had to create a list of random 20aa sequences that would at least match the number of our ortholog clusters that we created in the previous practical. For that we used the following script which again runs only in **python2**:

########### **rndseq.py​** #############

#prints first argument aa sequences of length second argument to screen"  
import random  
import sys  
number=int(sys.argv[1]) #input1  
length=int(sys.argv[2]) #input2  
aas="ARNDCEQGHILKMFPSTWYV"  
def r20():  
 return random.randrange(20) #generates random number 1- 20  
for i in range(number):  
 s=""  
 for j in range(length):  
 s+=aas[r20()] #adding on the random aa residue to the sequence  
 print s

After creating the 20aa sequence list we had to assign this sequences with a corresponding number. For that purpose we wrote the following python script and also combine all the sequences in on long single sequence:

######### **pseudo\_gen.py** #########

import glob   
import sys  
  
dict\_seq = {}  
list1 = []  
list2 = []  
dict\_both = {}  
  
  
  
f2 = open(sys.argv[1], 'r')  
f2 = f2.read().split()  
for counter, seq in enumerate(f2): #enumerating the sequence dictionary  
 dict\_seq[counter] = seq  
  
with open(('pseudo\_all.txt'), 'w+') as allwrite:  
 for name in glob.glob('geneOrder\*.txt'):   
 print(name)  
 f1 = open(name, 'r')  
  
   
 list\_ele = []  
 seq\_list = []  
 join\_list = []  
   
 for line in f1:  
 line = line.split(' ')  
   
 for ele in line:  
 ele = int(ele) #geneorder element list creation  
 list\_ele.append(ele)  
 for i\_ele in list\_ele:  
 if i\_ele in dict\_seq.keys():  
 dict\_both[i\_ele] = dict\_seq[i\_ele] # combining the gene order num with the sequence in a dictionary  
 seq\_list.append(dict\_both[i\_ele])   
 join\_list = ''.join(seq\_list) #joining the random sequences into a genome  
 print(join\_list)   
 with open(('pseudo\_'+name), 'w+') as pseudowrite:  
 pseudowrite.write('>pseudo' + '%s\n%s' % (name, join\_list)) #writing to single file  
  
 allwrite.write('>pseudo' + '%s\n%s\n' % (name, join\_list)) #writing to a group file  
pseudowrite.close()  
allwrite.close()

###########################

The result was a multifasta file which contains all of our new pseudogene sequences for each individual query genome. This file was used to create a dot plot in dotter by inputting the following command:

*$ dotter pseudo\_all.txt pseudo\_all.txt #using the multifasta file against itself to create the dot plot*

Then we got the following plot:

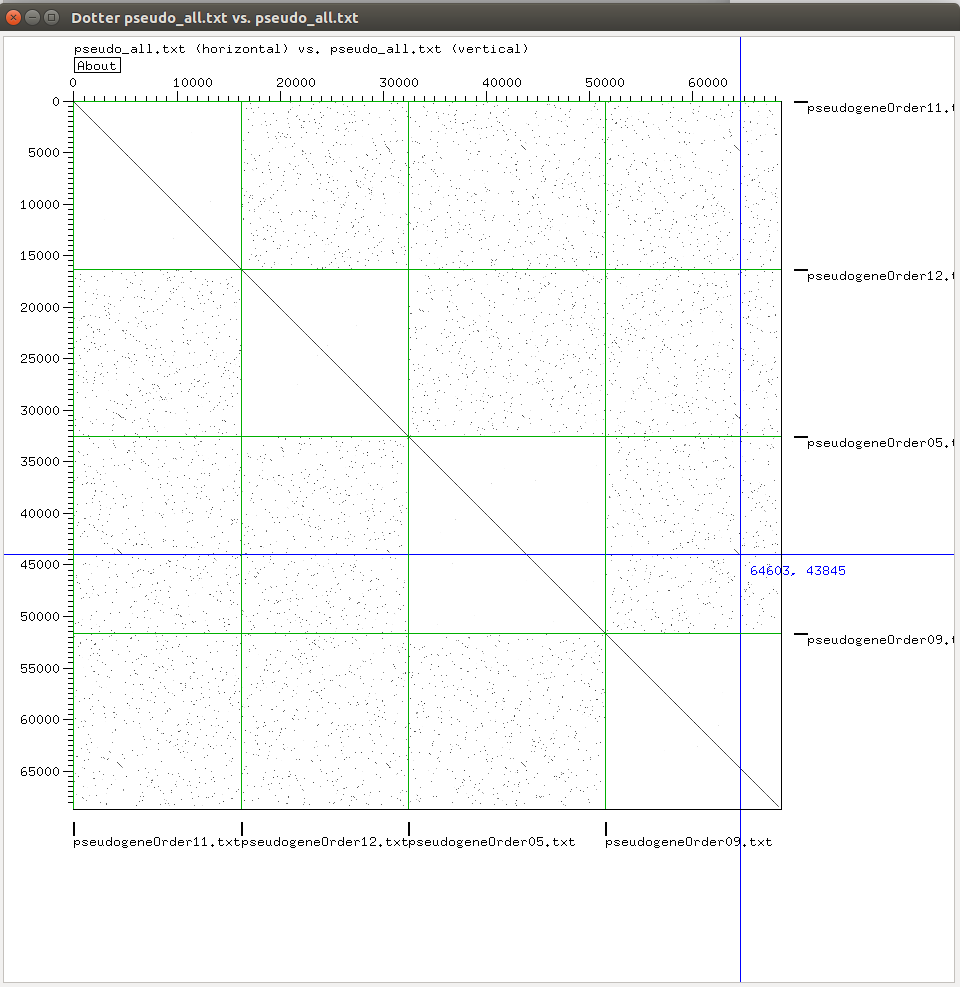


Image 1. Dot plot for all query genomes

While observing it we clicked on a specific area which appeared to have a distinctive black mark and that way it zoomed in and gave us a more clear look of this pattern (Image 2) and also provided us with the sequences that match between those two orthologs (Image 3).

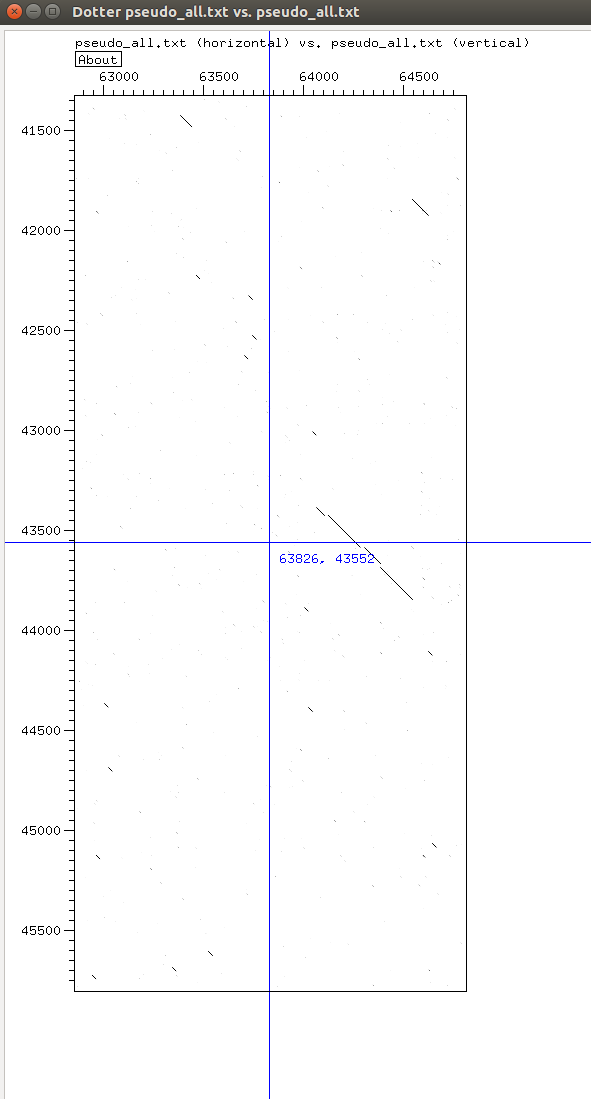


Image 2. Detailed visualisation of one set of long matching sequences between pseudogene order for *E.coli* and *C. trachomatis*

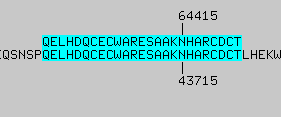


Image 3. Matching pseudogene order sequences corresponding to the genomes for *E.coli and* and *C. trachomatis*

**Step 3: Reconstructing phylogeny from gene order**

*$ python2 getGeneOrderGrimm.py 1000 geneOrder05.txt geneOrder09.txt geneOrder11.txt geneOrder12.txt > output.txt*

*Chlamydia trachomatis*  > G1

*Escherichia coli* > G2

*Geobacter sulfurreducens* > G3

*Gloeobacter violaceus* > G4

After editing the distance.grim file so that the Genome labels were in the first row, we use the resulting distance matrix to reconstruct the species tree in Belvu.

*$ grimm -f labelsNew.txt -o distance.grim -C -m*

*$ ./belvu -T R belvu\_distance.txt*

The final output was a phylogenetic tree which uses the Neighbor-Joining method:

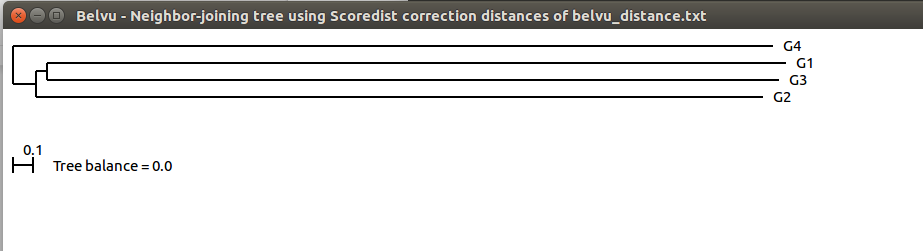


Image 4. New belvu distance tree of the gene order of pseudogenes

The tree in Image 4 appears to have similarities with the previous tree that we reconstructed. The genome that goes further back in the evolutionary path is the genome of *Gloeobacter violaceus* in both of them. Also, although the branches have similar construction, in tree 4 the genomes that are more closely related are *Chlamydia trachomatis* and *Geobacter sulfurreducens* although in tree 5 themost closely related organisms are *Chlamydia trachomatis* and *Escherichia coli.* That difference might appear due to the fact that for the metagene tree reconstruction we only used 10 orthologous genes and for the pseudogenes tree reconstruction we used the all the orthologous genes between all four query genomes so we have a more complete amount of information than the metagene has. So the metagene construction might give more random results compared to pseudogenes.

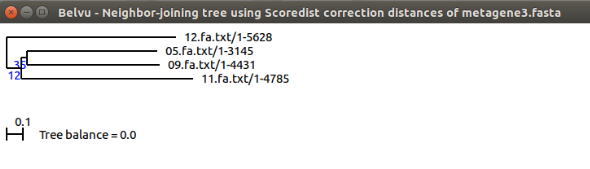


Image 5. Metagene tree with bootstrapping

**References**

http://grimm.ucsd.edu/MGR/

http://sonnhammer.sbc.su.se/Dotter.html

https://en.wikipedia.org/wiki/Pseudogene