Comparative Genomics Practical 5

Gene Order Analysis

Group6 Tianlin He Xueqing Wang

SUMMARY

Gene order conversion, in other words synteny, refers to the gene order similarity or identity in the genomes between different species, indicating the evolutionary relationship among them, since that related organisms may possess the same gene order as they share the regulatory segments which need proximity. Thus, the study of gene order can be used to explore the evolution relevance; prokaryotes conserve their gene order even more than eukaryotes. In this practical we probed a little into the gene order of the four bacteria by utilizing the order of orthologs (assigned in previous practicals as clusters) in genomes, determined the rearrangement distance among them and built a phylogenetic tree based on the distance matrix. In the dotting section we also plotted artificially generated pseudogenes against itself.

ACTIVITY

We performed the following steps in this order.

Ortholog groups

InParanoid is a graph-based program to find orthologs in given genomes. It makes all-against-all sequence comparisons, detect mutually best hits and can resolve overlaooing groups. In this practical we learnt the operation procedures of InParanoid; however, in the end we used directly the ortholog groups of four prokaryotes identified in practical 4. We made modification of the ortholog file to contain only the gene name, removing the name of the organisms and preserved only the gene names, see Annex I.

Dataset

1. For the bacteria you have (the rest are probably too far distant to have retained gene order), use the information you have on which genes are homologous to create gene order lists for each genome.
2. There is a script getGeneOrder.py which reads in a proteome multifasta file and a cluster file from Lab 4, then outputs the gene order list for the corresponding genome.
3. If you use this script, please answer the following questions:

a. Can there be ambiguities in clustering, so that one gene appears in several clusters in the cluster file? If so, why is this? What does this script do in that case?

-Ambiguities in the cluster file do exist: for example, gene orf03432_rev in B.thetaiotaomicron (species03) is in ortholog group of both orf01575 and orf00007_rev in E.coli (species09). There are also many other genes that consist in two or more clusters.

-This phenomenon is related to the complexity of evolutionary events happening during the process. Because of the huge amount of genes on these genomes, duplication and

speciation occur at different times and various of combination of these events exist. For instance, if a duplication of one gene on genome 03 happens after its speciation with another gene on genome 09, then both the two gene acquired after duplication (in other words in-paralogs) are orthologs for the gene on genome 09. Since in the cluster file we follow the gene order on genome 09 (meaning that the same gene on genome 09 would not appear twice), the genes on other genomes may show up in different ortholog groups leading by different genes on 09.

-In this section we tried to assign each gene a number to simplify the position on the genome, based on the gene order of genome 09. To avoid that one gene ends up with different indexes, the script used the command "if not" in line 37 to attribute id for each gene name. In this way, if the gene name repeats, it won't be given a new id; at the same time the index itself keeps increasing, so that the subsequent position counting won't be erroneous.

b. Can this script handle forward and reverse strandedness in gene order lists? If not, how should this be done?

This script can handle forward and reverse strandedness, as the repeating gene in different groups won't be assumed several ids. However, when reading in different orientations (forward or reverse), the id of the same repeating gene does differ as it only counts for the first position index that is encountered, which would introduce some imparity of the gene position. But this is the built-in deviation of the ortholog data itself because of the ambiguity in clustering, which has nothing to do with the python script. We made a little modification for the getGeneOrder.py, as attached in Annex II.

Dotter

Comparing gene order between proteomes to detect change in location

4. Generate a numbered list (dictionary) of random 20aa sequences, at least as long as the highest number of lines in any of the ortholog groups files from exercise 4. (use: rndseq.py)

The script rndseq.py takes in the first argument in command line as number of aa sequences and the second as the length of each sequence (20). The highest number of lines in the four ortholog groups is 5312, thus we chose 5320 as the number of 20aa random sequences, and save these sequences into a file (see Annex III)

5. For each gene order file from getGeneOrder.py script, assign random sequence with corresponding number.

6. Combine all 20aa sequence for each proteome into long single sequence.

a. Result should be one long sequence assembled from all 20aa random sequences per proteome.

7. Combine all sequences into one multi-fasta file and use it as dotter input against itself into dotter.

a. Another script called makeIconicGenome.py reads in a list of random sequences (there must be more of these than there are cluster IDs) and a gene order file, and prints out the resulting pseudogenome. If you use this script explain what is it doing as short pseudo-code in your report.

The first part of this script takes in the random sequence file and built a dictionary, with the sequences themselves as values and the corresponding number (0-5319) as the keys

used to call them. The second part reads the gene order file (in command line) generated in previous steps, turning each position number into the corresponding 20aa sequence that connecting them together to build one pseudogene (one sequence). After running this script four times with the second argument being four different gene order files, we get four pseudogene sequences, and combined them in one multifasta file.

b. Using these "iconic genomes", look at pairs of genomes using the program dotter. Can you see how blocks of genes have been shuffled around at once?

By running dotter with these "iconic genomes" itself against itself locally we got a dotplot. We can see from the picture that the alignment above cutoff fits almost perfectly with the diagonal, since that we are comparing exactly the same sequence against itself. Besides, if we zoom in we can still see alignments in other areas, even short line fragments on the anti-diagonal, meaning the shuffling (turn-over) of gene blocks. Since we generated and spliced the pseudogenes randomly, this phenomenon just means that similarity of forward and reverse sequences.

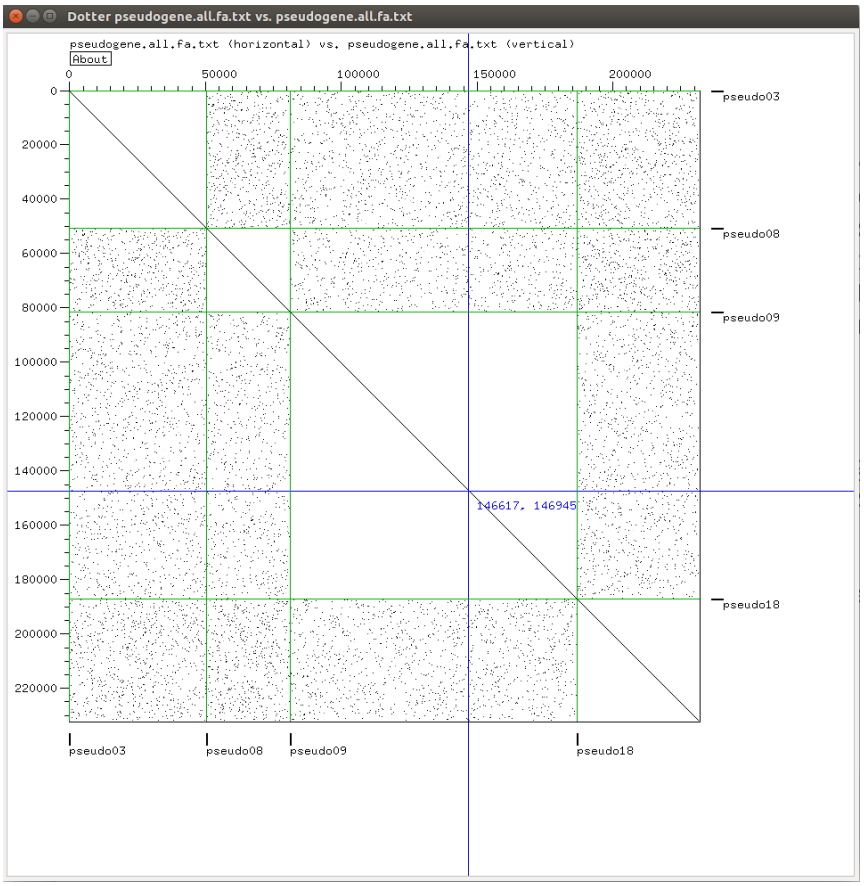

Fig.1 The Overall Dotplot

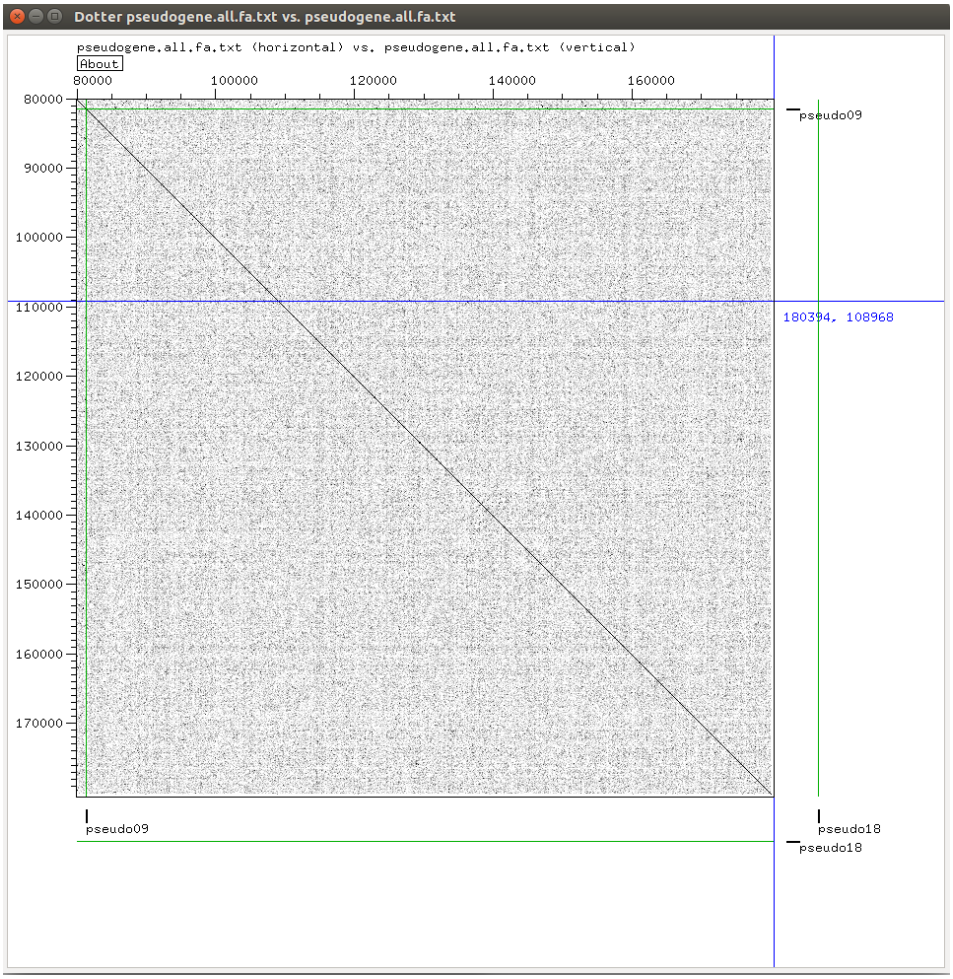

Fig.2 Zoomed-in Dotplot

Reconstructing phylogeny from gene order

8. Use a tool (such as GRIMM) to determine the genomic rearrangement distances between the species. GRIMM wants as input lists of numbers corresponding to genes, so in that case, translate your gene order lists to lists of corresponding numbers. This should give you a distance matrix.

9. It turns out that you need to change the format of the gene order lists from the previous part to get GRIMM/MGR to accept it. It only accepts input data where both sequences have exactly the same set of orthologous genes, and they must be listed from 1 upwards.

10. Compare the species tree to the bacterial part of your trees from practical 4. How do they differ and why?

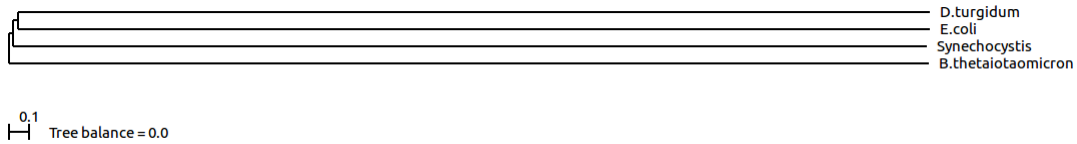

Fig.3 Phylogenetic Tree Got out of Distance Matrix

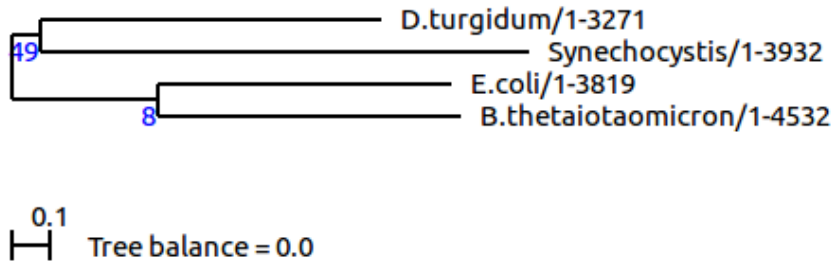

Fig.4 Phylogenetic Tree Got out of Metagene Alignment

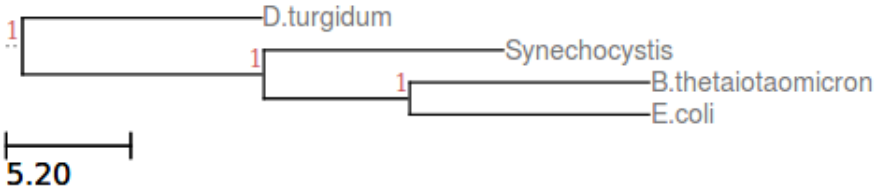

Fig.5 Consensus Phylogenetic Tree Got out of Ten Individual Trees

These phylogenetic trees are distinctive from each other. For example, from the tree we got in this practical, we can tell that B.thetaiotaomicron first diverge from the other three, Synechocystis follows, E.coli and D.turgidum diverge at last. In the other two trees the divergence order totally differs. Distinctive as they are, these are actually built-in features, as the tree-building principles are different from the beginning: the ones in practical 4 are built according to sequence alignment, with the comparing among amino acid orders, while the one in this practical are built on the order difference of the genes, totally discarding the original sequences. It is not easy to make a direct comparison between sequence-based or distance-based tree building methods to determine which is better, but in this case the sequence-based ones are even more inaccurate, as that neither the metagene file nor the ten gene groups are identified to be highly conserved—they are just random picked. In contrast, in this practical the trees are built on the synteny of the whole genome, which can be interpreted as more reliable.

DISCUSSION

In this practical we assigned number to the identified ortholog clusters to record their relative positions, getting the gene order information among the four prokaryote species in group 6. The gene order relationship can be visualized clearly from the dotplots generated by dotter. We also established distance matrix based on the rearrangement distances among our species, counting the smallest distance score (the least number of evolutionary events, e.g. inversions, transpositions), and built phylogenetic tree according to the distance matrix.

Through the procedures we assigned each gene a number to mark their relative positions. As some evolutionary reasons, the ortholog clusters overlap somehow, and we fixed this by giving one gene a number only once (a unique id). There may be some inaccuracy related to this but that is the essence of the evolutionary events themselves.

The dotplot got out of dotter is super noisy, with dots above threshold nearly everywhere outside the dialogue, even in the identical sequence part, identifying that there are similar areas within the sequences even when they are randomly generated.
The tree we acquired within this practical is super different from the ones we got in practical 3&4.
The phylogenetic trees in this practical might be more reliable than the ones got in the last practical, as the distance matrix established on the whole genomes could be thought to be more accurate than a few randomly picked orthologs. However, we cannot make further evaluation on the quality of this tree, as we cannot conduct bootstrapping on belvu when building trees merely based on distance matrix.