Comparative Genomics course, HT07, 5: Gene order analysis

Lab instructions here. Questions can be directed to Kristoffer Forslund, krifo at-sign sbc dot su dot se.

Part 1 - dataset

- 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.
- There is a script **/afs/pdc.kth.se/home/k/krifo/Public/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.
- If you use this script, please answer the following questions:
    - What are the two input arguments?
    - 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?
    - Are any of the imports unnecessary?
    - Can this script handle direction/sign in gene order lists? If not, how should this be done?

Part 2 - dotter

- Write a program that reads the lists of genes and, for each ortholog cluster, creates a random sequence of about 20 amino acids, and which replaces each gene with the corresponding sequence for its ortholog cluster. Effectively, the end result should be for each genome a single sequence with the corresponding "iconic sequence" in gene order.
    - A script called **/afs/pdc.kth.se/home/k/krifo/Public/rndseq.py** generates a number of random sequences and prints them to screen. If you use this script, answer the following questions:
        - What does the input arguments do?
        - What is the meaning of the random variable **s**?
    - Another script called **/afs/pdc.kth.se/home/k/krifo/Public/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, answer the following questions:
        - Why is the call int (aWord) there?
        - What do the split () and replace () calls do?
        - What do the different loops in the script traverse?
        - How is the name of the output sequence chosen?
- 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?

Part 3 - reconstructing phylogeny from gene order

- Use a tool of your choice (such as GRIMM, use version at http://nbcr.sdsc.edu/GRIMM/mgr.cgi) 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.
    - 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. To convert your gene lists to this format, use **python /afs/pdc.kth.se/home/k/krifo/Public/getGeneOrderGrimm.py <max number of genes in output> <input gene order file 1> <input gene order file 2> ... <input gene order file N>** for all your prokaryote gene order lists at the same time. This will give you the output for all of them, consisting only of the genes present exactly once in each of them, and renumbered the way GRIMM/MGR wants it. Paste the results into the GRIMM/MGR window, choose “unsigned” and “circular”.
- Use the resulting distance matrix to reconstruct the species tree.
- Compare the species tree to the bacterial part of your trees from practical 4. How do they differ and why?