Comparative Genomics course, HT07, 4: Phylogenomics

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

Part 1 - orthologous gene datasets

- For your species, you need several genes that are present in each - effectively, orthologs.
- Choose one of your prokaryotes genomes. For each protein in that, find the best blast hit in each of the other genomes. Each such collection of proteins forms an approximate ortholog cluster.
    - You should have proteome multifasta files since previously with each protein in your genomes.
    - In the previous exercise, you used formatdb to train a blast database. Do the same now, but do one for each prokaryote proteome individually.
    - Do a blastall search (like yesterday) with one bacterial proteome as query. Add the **-m 7** parameter to get XML output. Do this with each of the other proteomes as database. The result will be a list of outputs that show which genes in the other bacterial genomes are most closely related.
    - To parse this output, you can use the script **/afs/pdc.kth.se/home/k/krifo/Public/multiBlastResultParser.py**. This parses XML blast output for one proteome against another, and reports a list of the best matches in the format <query protein name> <target protein name>. Input is <xml output> <tag for reference genome> <tag for target genome>. Leave the tags as “” to supply none.
        - Questions that should be answered if you use this script:
        - What is the NCBIXML.parse function doing?
        - What data type is the object aSingleBlastRecord?
        - What is in the descriptions list?
    - Do this for all your proteomes (except the reference one), saving the proteome-proteome best hits in separate files.
    - Combine the best file hits into clusters with the script **/afs/pdc.kth.se/home/k/krifo/Public/gatherClusters.py**. It reads in a list of files and combines the pairwise information to give you the clusters of orthologs to the reference genome.
        - Questions that should be answered if you use this script:
        - What type of object is mappings?
        - What does the call mappings.keys () do?
        - What would happen if the replace function call was removed?
        - What would happen if the else: clause was removed?
    - Each line in the result is an ortholog cluster. Take several such either manually or by a

script and acquire the corresponding protein sequences to the protein names on each line. Gather them together from your proteome multifasta file. At least ten different ortholog clusters should be used, but it is even better if you write a script to use them all. The result should be a number of multifasta files, one for each row from the ortholog cluster file that you work with.

Part 2 - metagene approach

- Align all gene sequence files separately (so you have one alignment for each cluster of orthologous genes).
    - Run kalign as in the previous exercise on all the fasta files containing the orthologous protein clusters. If you use more than a few, you should write a script to do this.
- Concatenate the alignments into a single, long metagene.
    - If you use just a few, you could do this manually, otherwise you should write a script. Even if you do, it makes sense not to use too many genes, as the running time might increase. The end result should be a single fasta file with a file for each bacterial genome consisting of the aligned genes from each genome.
- Perform tree reconstruction using your method of choice for the metagene. What is the tree like?
    - Belvu just as before makes sense.
- Perform sequence bootstrapping on the metagene. Can you say anything on the quality of the reconstruction?
    - Again, you can use belvu as in last exercise.

Part 3 - consensus reconstruction

- Perform tree reconstruction using your method of choice for each alignment.
    - Once more, belvu can be used, just as in last exercise. If you use more than a handful of genes, you should write a script to do this for you. Doing this is better but not necessary for passing.
- Construct a consensus tree from the gene trees. How precise is it? Can you detect specific genes or classes of genes that cause disagreement? How do you think that happens?
    - To construct a consensus tree, put the tree files together into a file containing a list of trees. Name this file intree. Then use phylip consense (**/afs/pdc.kth.se/home/a/arnee/MODULES/seqtools/phylip/Phylip-3.6/exe/consense** or just consense if you can add the phylip module). This reads the file named intree in the present directory, asks you a few questions, and then gives you the consensus tree.
- How does the consensus tree agree with the metagene tree?

Cleaning up fasta files

Most software you will use will react strangely if there are characters in the sequence data that are not actually amino acids/nucleotides. Some gene prediction programs may include characters for stop positions or similar which might still be left in your sequence files. If you receive strange results at later stages, this may be the reason.

If so, or if you just want to be sure, you should clean your sequence files. You could use the program **/afs/pdc.kth.se/home/k/krifo/Public/cleanFasta.py**. You give it two arguments, the input fasta file, and either "protein" or "dna" to determine what type of file it will assume you gave it. If you use this program, rather than writing one of your own (in which case you must present that), you must answer the following questions for full credit:

- Why is there a "firstLine" variable in the program? What is that used for?
- Does the program do anything else except for just removing nonstandard sequence characters?
- Why are there 5/21 letters present in the alphabets?
- Why are the two imports necessary?

Getting sequences from a multifasta file

If you have seen to it that your fasta files contain no unnecessary newlines, that is, that every sequence occupies exactly one line, there is an easy way to get hold of a subset of a fasta file. Suppose you have a fasta file **proteome.fasta** and a file of protein names **proteins.txt**, exactly one per row. Then **grep -A 1 -h -F -f proteins.txt proteome.fasta | grep -P "^[^-]"** will yield a smaller fasta file with exactly those proteins listed in **proteins.txt**. If your proteins are in separate files, like **proteome0.fasta ... proteome9.fasta**, you can replace **proteome.fasta** above with **proteome?.fasta**.