Comparative Genomics 2018

Practical 5: Gene order analysis

Assistants: Miguel Castresana, Deniz Seçilmiş, Stefanie Friedrich

All forms of plagiarism are forbidden, and if detected it will result in a lower grade.

Studies of gene order conservation are important to understand the evolution of the genomes. In this practical you will learn how to perform a basic gene-order analysis.

The report should be formatted in one PDF file and sent in by the end of the week. It should cover all points listed below. Missing or failed items will result in a reduced grade for this practical.

Material needed:

1. Your prokaryotic proteomes and the full ortholog cluster from Practical 4

Suggestions:

1. Dotter (https://www.sanger.ac.uk/science/tools/seqtools)
2. Dotter (http://nebc.nox.ac.uk/bioinformatics/docs/dotter.html)
3. GRIMM (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.620.7652&rep=rep1&type=pdf)

Exercise 1 - Ortholog groups

For this exercise you will use the ortholog groups (clusters) you found last week in Practical 4.

Alternatively, you may instead infer ortholog groups using a package, e.g. by running InParanoid for your 4 prokaryotic proteomes, and re-using (modifying) a provided script.

1. Copy InParanoid to your own directory.

```
cp -R /common/courses/comparative_genomics/scripts/InParanoid/ ./
```

2. You will need to run the following commands to make Inparanoid work. Do this in your Inparanoid directory:

```
export
PATH=/afs/pdc.kth.se/misc/pdc/volumes/sbc/prj.sbc.dmessina.5/Comparative_Genomics
/scripts/InParanoid/blast-2.2.18/bin:$PATH
```

3. Your proteomes must be copied into Inparanoid folder.

4. Run InParanoid one by one (not in parallel)

```
perl ./inparanoid.pl reference_proteome.pfa target_proteome1.pfa
perl ./inparanoid.pl reference_proteome.pfa target_proteome2.pfa
```

5. Adapt **getClustersInParanoid.py** to get clusters of orthologs.

Exercise 2 - Gene order

1. We provide the script **getGeneOrder.py** which reads a proteome multi FASTA file (i.e., 01.fa.txt.pfa) and a cluster file from Practical 4 (IF POSSIBLE, include all the clusters, not just the clusters where all the organisms tested are presented), and outputs the gene order list for the corresponding genome. This script is just an example, it will not directly work for all of you since you have different formats in your input data.
2. 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?
    b. Can this script handle forward and reverse strandedness in gene order lists? If not, how should this be done?

Exercise 3 - Dotter

Compare the gene order that you obtained between your proteomes to detect changes in location. *Check synteny definition and visualizations.*

3. Generate a numbered list (dictionary) of random 20aa sequences, at least as long as the highest number of lines (proteome with the highest number of ORFs) in any of the ortholog groups files from Practical 4 (use: **rndseq.py**).
4. For each file obtained from **getGeneOrder** script, assign random sequences (one for each gene). Combine all these 20aa sequences for each proteome and gene order file into a long single sequence.
5. Alternatively, 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 a short pseudo-code in your report
6. Combine all sequences into one multi FASTA file and use it as dotter input against itself into dotter.
    a. Initialize dotter with “module add dotter”. Play with the parameters to obtain a visual output that better explains the synteny between these organisms (read the suggested documentation to understand it properly). Explain your dotplot visualization.

Exercise 4 - Reconstructing phylogeny from gene order

1. Use a tool (such as GRIMM) to determine the genomic rearrangement distances between the species. GRIMM wants some lists of numbers corresponding to genes as input; so in that case, translate your gene order lists to the 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 one upwards. To convert your gene lists into this format, do it for all of your prokaryotic 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 with the format GRIMM/MGR wants it.

```
python getGeneOrderGrimm.py <max number of genes in output> <input gene order file 1> <input gene order file 2> ... <input gene order file N>
```

2. Calculate the distance matrix between these genomes

```
grimm -f grim_genomes.txt -o distance.grim -C -m
```

3. Use the resulting distance matrix to reconstruct the species tree in Belvu.

```
module add belvu
belvu -T R distance.grim
```

Distance matrix must be in space delimited format, where the columns are the genome names.

4. Compare the tree obtained in Practical 5 against the one obtained in Practical 4. How do they differ and why?