Session 2: Orthology assignment

A.- Sequence based orthology assignment

Sequence similarity is often used as a substitute for orthology prediction. The assumption is that the closer two sequences are related based on sequence identity, the more likely it is that they are orthologs.
There are several approaches that allow us to predict orthology based on sequence similarity and that range from very simple to more complex. In this practical we will go through some of the most common ways: best bidirectional hits, inparanoid, mcl and orthoMCL.

1.- BRH (Best bidirectional hits or Best reciprocal hits):

This is a fairly simple method that assumes that if two sequences are each other's best hits then they are orthologs. All other proteins with a blast hit that is not reciprocal are considered paralogs.

Step 0: Download the practice folder and uncompress it:

tar -zxvf session2.tar.gz

Load the docker image we are going to use for some of the exercices:

docker load <phylogenyDocker.tar

Once the image has been loaded, enter it:

docker run -it -v /complete/path/to/folder/session2:/session2 phylogeny bash

Careful, anything you do within the docker will be reflected in the session2 folder!

Step 1: Go to the folder named exercise1. There you will see two fasta files called PENCH.fasta and ASPCL.fasta. Each of the files contains between 200 and 300 proteins and we will assume they represent a whole proteome. We will use a local blast to compare each proteome to the other.

Reminder: In order to run a local blast search you first need to build a blast database which can be done with formatdb. In this case we will use formatdb. Then you can run blastall. For detailed options you can consult the different manuals using -help.

```
formatdb -i ASPCL.fasta
formatdb -i PENCH.fasta
blastall -p blastp -i ASPCL.fasta -d PENCH.fasta -o ASPCL2PENCH.blast -m8 -e 0.01
blastall -p blastp -i PENCH.fasta -d ASPCL.fasta -o PENCH2ASPCL.blast -m8 -e 0.01
```

Step 2: You can use the get_BRH.py script to obtain the pairs of best reciprocal hits:

python get_BRH.py -s1 ASPCL.fasta -s2 PENCH.fasta -h1 ASPCL2PENCH.blast -h2 PENCH2ASPCL.blast -t TRY1

The script will output two files, one that will be called TRY1.BRH.txt and contains all pairs of best reciprocal hits and a second file called TRY1.unpaired.txt that provides a list of proteins for which we were not able to find an ortholog.

Questions:

1.1- How many orthologous pairs did you find?

1.2.- Which is the ortholog of ASPCL_0083_04882?

1.3.- Does ASPCL_0083_04882 have any other homologs in PENCH? What kind of relationship do they have regarding ASPCL_0083_04882?

1.4.- ASPCL_0080_03602 does not have a ortholog. Why?

1.5.- Which advantages and drawbacks can you see of the BRH method?

2.- Inparanoid (http://inparanoid.sbc.su.se/).

Inparanoid adds a layer of complexity to the best reciprocal hits approach. In BRH it is assumed that all proteins that have a blast hit against another protein but it is not reciprocated are paralogs. Inparanoid makes the distinction between inparalogs and outparalogs. Given one member of a pair of orthologs, if there is a sequence in the same species that has a higher blast score than the corresponding ortholog, inparanoid assumes that this sequence is an inparalog. Additionally, inparanoid allows the use of a outgroup that will help distinguish between orthologs, inparalogs and outparalogs. Details about inparanoid: http://inparanoid.sbc.su.se/cgi-bin/faq.cgi

Step 1: Move to the folder exercise2. Open inparanoid.pl and adjust the parameters if needed. Make sure the matrix you are going to use is in the folder you are going to execute the program as well as the two fasta files. Also make sure the lines for seqstat and blast_parser.pl are pointing to the folder where the programs are.

Step2: Execute inparanoid.

perl inparanoid.pl ASPCL.fasta PENCH.fasta

This will result in numerous files. The main tables of interest to us are table.ASPCL.fasta-PENCH.fasta which is a list of the orthologous groups while the Output.ASPCL.fasta-PENCH.fasta gives a more detailed overview.

Questions:

2.1.- How many groups of orthologs do we have? Are the groups the same as before?

2.2.- Are there any groups with more than two members? Give the most robust example. What are the evolutionary relationships between the three proteins?

2.3.- Go back to the BRH results and check what it says regarding this family. Which of the two results do you think is more reliable?

2.4.- Which is the ortholog of ASPCL_0083_04882? Is it the same one as before? Do we have any further information regarding the other paralogs we found for this family?

Step 3: Move the main results to a different folder so that they are not overwritten:

```
mkdir no_outgroup
mv -t no_outgroup/ table.ASPCL.fasta-PENCH.fasta
orthologs.ASPCL.fasta-PENCH.fasta.html Output.ASPCL.fasta-PENCH.fasta
```

Step4: Run inparanoid with an outgroup (PENMQ.fasta), you will need to modify the parameters in the script first.

2.5.- Does running inparanoid with an outgroup change the results? Why do you think that happens?

2.6.- Is it worth it then to run inparanoid with an outgroup?

3.- OrthoMCL.

OrthoMCL belongs to the clustering methods. These methods try to group elements based on a similarity measure. In this case we are going to see how they group sequences based on a blast-based similarities. Similarly to inparanoid, orthoMCL uses blast to detect orthologs between pairs of species, accepting the presence of inparalogs. But then uses a clustering algorithm (mcl) in order to join the different orthologous groups detected. This way it is able to extend an analysis limited to a pair of species, to multiple species.

Step0.- OrthoMCL is run in a separate docker container that is accessed by executing the run_orthomcl.sh file you have in your exercise3 folder. So, exit your current docker image by typing exit. Now move to the exercise3 folder and execute run_orthomcl.sh. Once inside the docker image, move to the folder called exercise3.

Step1.- In the file called how_to_run_orthomcl.txt you have the list of commands that you need to execute in the docker container to obtain the list of groups. Follow the commands until you get the groups.txt file.

3.1.- How many groups do you have in this case? Are they comparable to the previous results?

3.2.- Which proteins are orthologous to ASPCL_0083_04882? Is this result congruent with the previous results?

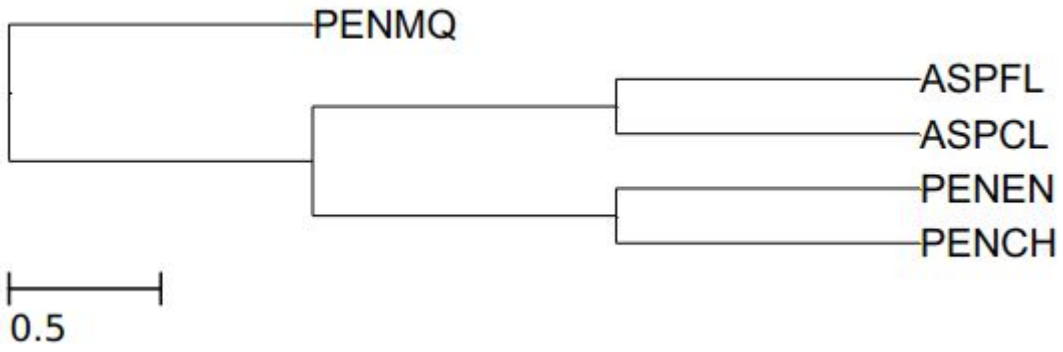
3.3.- Take group 40 (it contains the protein PENEN_0144_10558). In the folder called pairs, you will see the different relations between proteins. Can you figure out how the different proteins in this group are related to each other? Is there any incongruence? Can you solve this inconsistency?

B.- Phylogeny based orthology assignment

REMINDER: Gene tree reconstruction is divided in three steps: homology search, multiple sequence alignment and model selection plus tree reconstruction. There are many ways and many programs to perform each of the steps so it is often up to the person building the tree to decide how to build them. Here are some of the programs you can use to perform the different steps.
Homology search: blast, hmm, and similar programs as seen before
Alignment reconstruction: mafft, muscle, t-coffee
Tree reconstruction: phyml, raxml, fasttree, iqtree

In these exercises we are going to start with pre-build trees which you can find in the folder called exercise 4. To build this tree we have performed a blast search using protein ASPCL_0083_04882 as a starting point, then we have filtered the blast results to keep only those that have an evalue below 1e-05. The multiple sequence alignment was done with muscle and the resulting fasta alignment was converted to phylip format using readAl. Finally RAxML was used to build the phylogenetic tree using the PROTGAMMALG model and 100 rapid bootstrap repetitions.

Species tree:



4.- Obtain orthologs based on gene trees manually:

Open in a browser the following link: http://phylo.io/ Now insert the contents of the file ASPCL_0083_04882.tree.txt into the tree data part of the web and visualize the tree. Using this tree as example we are going to discuss how orthology inference is done using two kinds of algorithms: reconciliation and species overlap. We will also discuss how the previous predictions match the ones we have obtained in the tree.

4.1.- Based on what we have discussed for the tree ASPCL_0083_04882.tree.txt, now do the same for PENEN_0144_10558.tree.txt. And fill in the following table of orthology and paralogy relationships when referring to PENEN_0144_10558:

	Orthologs	Paralogs
BRH		
InParanoid		
orthoMCL		
Tree based (Species overlap)		
Tree based (Reconciliation)		

Discuss the differences between the different predictions and why they happened. Which method do you think is the most reliable?