Orthology practicals on phylogenomics data.

BEST RECIPROCAL HITS (exercices_orthology/BRH)

We will search BRH between two closely related genomes, *A. nidulans* and *A. fumigatus*. The first step is to perform the blast search, then we will use a python script in order to filter the blast results and obtain the best reciprocal hits.

1.- Format the two files provided: ASPNI.fa and ASPFU.fa

Command: `formatdb -i ASPNI.fa`

2.- Now perform the two blast searches, ASPNI.fa to ASPFU.fa and ASPFU.fa to ASPNI.fa

Command: `blastall -p blastp -i ASPNI.fa -d ASPFU.fa -m8 -o ASPNI_2_ASPFU.blast`

Where:
-i indicates the query file
-d indicates the blast database you formated
-m8 defines that we will obtain the blast results in tabular format (you can use -m 9 to see what the columns mean)
-o is the output name

3.- Process the results with the script get_BRH.py.

python get_BRH.py -s1 ASPNI.fa -s2 ASPFU.fa -h1 ASPNI_2_ASPFU.blast -h2 ASPFU_2_ASPNI.blast -o BRH.ASPFU.txt

This script will first filter the blast results and remove those hits that do not fulfill a minimal requirement of e-value and sequence overlap. Then it will obtain that pairs of sequences that are each others best hits.

4.- Look at the results. How many BRH did you find?

ORTHOLOGY PREDICTIONS BASED ON A PHYLOGENETIC TREE

1- Species overlap algorithm.
In phylomeDB we saw that orthology and paralogy predictions are drawn onto the tree. But how can we obtain those predictions for our own trees. ETE implements the same algorithm we use in PhylomeDB. In order to use it we will have to do a bit of scripting.

A.- Open an terminal and write ipython, then press ENTER. This will open the interactive python window where one can run small scripts.

B.- The tree can be found in the file: tree_example.nw The codes codified in this tree are phylomeDB codes. They have a structure similar to the one found in UniProt where the first part of the code is the protein name, followed by an underscore and then a species tag. It is very important that all your sequences contain a reference to the species they come from and that this is always the same. The first thing we have to define is a small python function that will give us back the species tag from a code:

```
def get_species_tag(node):
      return node.split("_")[1]
```

In this case, since the node is the letters after the underscore we divide the code name by the underscore and take the second term. Remember python always starts counting from 0.

C.- Now we will use ETE to upload the tree and create a PhyloTree instance:

`Import ete3` #First we import the ETE library

`tree=ete3.PhyloTree("tree_example.nw",sp_naming_function=get_species_tag)` #Load the gene tree and use the function we created before to correctly define the species name

`tree.show()` # This will visualize your tree

Can you pinpoint the duplication nodes in this tree?

D.- The species overlap algorithm is implemented in the get_descendant_evol_events() function. Note that this will only work on rooted trees.

```
tree.get_descendant_evol_events()

tree.show()
```

How many duplication events do you observe?
Do you think there is any event that you don't agree with?

Run the command again and this time change the species overlap threshold. This threshold defines the amount of overlap between species on either side of the node that is needed to call a node a duplication.

```
tree.get_descendant_evol_events(sos_thr=0.5)
```

How will the use of this command change your orthology / paralogy prediction.

2- Reconciliation

A.- In order to use reconciliation we need a species tree. In order to properly load the species tree you need to tell the algorithm that the species is represented by the whole leaf name:

```
def get_whole_name(node):
      return node
```

B.- Now load the species tree:

```
spTree=ete3.PhyloTree("species_tree.nw",sp_naming_function=get_whole
_name)
```

C.- Now you can use the reconciliation algorithm implemented in ETE to obtain the duplication and speciation nodes:

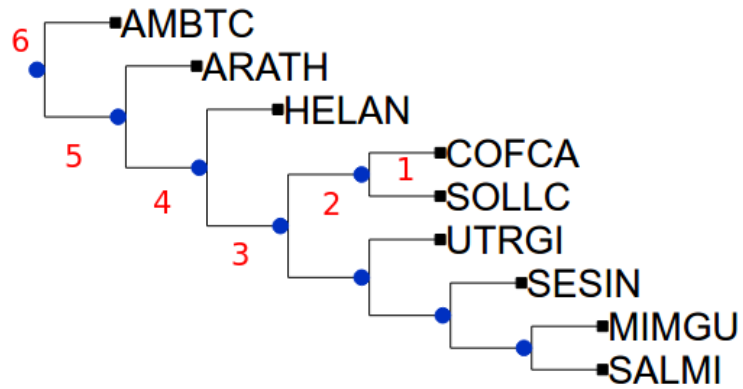
```
t.reconcile(spTree)

t.show()
```

Does the prediction change? Why do you think this is? Do you trust this prediction?

OBTAIN ONE TO ONE ORTHOLOGY RELATIONSHIPS FROM A LIST OF TREES

A.- Rooting the trees. An important aspect of automatically obtaining orthology and paralogy relationships is that you need to root the tree properly. ETE implements an automatic method that will take leaves by order of preference based on a list. So the outgroups will always have the highest value in this list and will be preferentially chosen to root the tree.



This is an example of how a species tree is used to create a preference list of rooting species. We assume that our species of interest is COFCA.

B.- Now run the script to obtain a list of one to one orthologs.

Python get_one_to_one_orthologs.py -t coffee_trees.txt -s species_tree.nw -n COFCA -o coffee.one2one.txt

Where:
-t: list of trees
-s: species tree
-n: species name of the species of interest
-o: output file

Have a look at the results.