**Multi-gene incongruence consistent with hybridisation in *Cladocopium* (Symbiodiniaceae), an ecologically important genus of coral reef symbionts**

Joshua I. Brian, Simon K. Davy, Shaun P. Wilkinson

*Exemplar code*

What follows is examples of the commands used the program PAUP\* to run the analyses executed.

1. Example (using site BBR) of the Incongruence Length Difference test

In this example, the three gene regions for samples from site BBR have been concatenated and saved as a nexus file ‘ildBBR.nex.’

paup> execute 'ildBBR.nex';

#read in data

paup> cstatus;

#ensure data has been read correctly

paup> PSet gapMode=newState;

#set gaps as a fifth character state

paup> set maxtrees=10000;

#set maximum number of trees to be stored

paup> charset cob=1-787; charset its2=788-1156; charset

psba=1157-1687;

#differentiate the gene regions

paup> charpartition ildBHB = 1:cob, 2:its2, 3:psba;

#set each gene region as a character partition

paup> hompart partition=ildBBR nreps=1000;

#run the Incongruence Length Difference test

1. Example (using site BBR) of the Approximately Unbiased test

In this example, the three gene regions for site BBR are in three separate nexus files (‘cobBBR.nex’, ‘ITS2BBR.nex’, psbABBR.nex’)

paup> execute 'cobBBR.nex';

#read in data

paup> cstatus;

#ensure data has been read correctly

paup> Set criterion=likelihood;

#set PAUP\* to use Maximum Likelihood criterion (note the default setting is Maximum Parsimony)

paup> outgroup D1 /only;

#set the outgroup

paup> dset distance=jc;

paup> nj bionj=y;

#create a basic Neighbour-Joining tree (necessary for calculation of evolutionary model)

paup> automodel;

#calculate appropriate evolutionary model for data

paup> LSet nst=6 rclass=(abcdbe) rmatrix=(0.5364643 2.3108171 0.7265159 25.137558 2.3108171) basefreq=(0.27927308 0.14717115 0.12010736) pinv=0.92122966;

#set the appropriate evolutionary model for the data (note PAUP\* runs this command #automatically when ‘automodel’ is used to set the chosen model)

paup> hs;

#generate a tree using a heuristic search

paup> bootstrap nreps=1000 search=heuristic /addseq=random;

#bootstrap the selected tree

paup> savetrees from=1 to=1 savebootp=nodelabels file=cobBBR.tre;

#save the bootstrapped tree

paup> generate random model=equiprobable ntrees=100;

paup> savetrees file=cobBBRrandom.tre;

#generate 100 random trees from the same dataset (see Methods)

#Follow the same procedure using files ‘ITS2BBR.nex’ and psbABBR.nex’

#This leaves the following output files, to be used in the testing procedure:

## - ‘cobBBR.tre’

## - ‘cobBBRrandom.tre’

## - ‘ITS2BBR.tre’

## - ‘ITS2BBRrandom.tre’

## - ‘psbABBR.tre’

## - ‘psbABBRrandom.tre’

#Begin with using the *cob* dataset as a base

paup> execute 'cobBBR.nex';

#read in data

paup> outgroup D1 /only;

#remind PAUP\* of outgroup

paup> gettrees file=cobBBR.tre; gettrees file=ITS2BBR.tre; gettrees file=psbABBR.tre; gettrees file=cobBBRrandom.tre;

#read in the appropriate saved trees

paup> LScores;

#gives the output for Shimodaira-Hasegawa, Weighted Shimodaira-Hasegawa and Approximately Unbiased tests

#Repeat the above steps, executing ‘ITS2BBR.nex’ and ‘psbABBR.nex’ instead of #‘cobBBR.nex’, and substituting in ‘ITS2BBRrandom.tre’ and ‘psbABBRrandom.tre’ #respectively in place of ‘cobBBRrandom.tre’