SH & AU Tests

Ian G. Brennan

Shimodaira-Hasegawa and Approximately Unbiased Analyses

Prerequisites:

1. Alignment (Phylip format for RAxML)
2. Partition File (optional)
3. Constraint Trees (monophyletic groups for topology testing). Make these yourself, and format them similar to those presented below:

```
((X,Y),A,B,C,D)
#this constrains X&Y, but leaves ABCD free)
((A,B),X,Y,C,D)
#this constrains A&B, but leaves XYCD free)
(((A,B),(X,Y)),C,D)
#this constrains A&B, X&Y, but also AB&XY, CD are always free
```

4. Your preferred ML topology - likely your unconstrained RAxML tree

Necessary Programs:

1. RAxML: suggest downloading raxmlHPC-PTHREADS +compile our download from http://sco.h-its.org/exelixis/web/software/raxml/hands_on.html
2. CONSEL: get it from http://www.sigmath.es.osaka-u.ac.jp/shimo-lab/prog/consel/ +use instructions from: http://phylo.bio.ku.edu/slides/lab7-Testing/lab7Testing.html

+before each step you'll have to physically drop the raxmlHPC-PTHREADS-SSE3 into the terminal
+remember if you publish constrained trees, indicated constrained nodes with a unique symbol
+things within square brackets [. . .] represent objects you're dragging/dropping

Step by Step:

Getting the Best Unconstrained Tree

Basic Glossary, other can be found in the RAxML Manual:
-m [your molecular model of choice, I will use GTRGAMMA]
-q [your partition file, this is optional]
-s [your phylip alignment]
-n [your output file name]
+don't actually use the brackets

1. Run RAxML for 100 (or however many you want) searches for the best scoring unconstrained tree (command -N)

```
[drag/drop RAxML executable] -T 8 -f d -N 100 -m [...] -q [...] -s [...] -n [...]
#above is an out line it will actually look something like this:
raxmlHPC-PTHREADS-SSE3 -T 8 -f d -N 100 -m GTRGAMMA -q partfile.txt -s align.phy -n Tree.tre
```

2. Run RAxML for 1000 (or however many you want) bootstraps (command -# and -b)

```
[RAxML] -T 8 -f d -# 1000 -m [...] -q [...] -s [...] -n [...] -b 12345
# you can change -b to -x for rapid bootstrapping
```

3. Add your 1000 bootstraps to your best unconstrained tree (command -t, -z)

```
[RAxML] -T 8 -f b -m [...] -q [...] -s [...] -z [...] -t [best tree, step1] -n [...]
```

4. Step 3 should culminate in producing a single best tree with bootstrap support values, check it in figtree to make sure you constrained the correct nodes.

Getting the Best Tree for Each Constrained Scenario

1. We need to repeat the above steps (with one big change, constraints). We can constrain either using the -g or the -r command. If you use -g, you must include all the taxa in the tree in your constrained newick string, otherwise they're unconstrained and fall out wherever. If you use -r, use need only provide a backbone tree (not all the taxa), and the remainder are fit into the tree initially via maximum parsimony, and then optimized using maximum likelihood. Here, I'll use the command -r.
2. Run RAxML for 100 (or however many you want) searches for the best scoring constrained tree

```
[RAxML] -T 8 -f d -N 100 -m [...] -q [...] -s [...] -r [constrained newick tree] -n [...]
```

3. Run RAxML for 1000 (or however many you want) bootstraps

```
[RAxML] -T 8 -f d -# 1000 -m [...] -q [...] -s [...] -n [...] -b 12345
```

4. Add your 1000 bootstraps to your best constrained tree

```
[RAxML] -T 8 -f b -m [...] -q [...] -s [...] -z [boostraps, step2] -t [best tree, step1] -n [...]
# -f h will give you the likelihood of each bootstrap given your provided (constrained) tree
```

Compiling the Trees and Comparing Them

1. When you're done with all the constrained trees, open the files and paste each tree into a new text file which contains all of the topologies (constrained and unconstrained)
   +keep track of which tree is in which position (1 tests X,Y; 2 tests A,B; etc)

```
((X,Y),A,B,C,D)
((A,B),X,Y,C,D)
(((A,B),(X,Y)),C,D)
```

2. Run RAxML to get the log likelihood values between all sets of trees (command -f h), this is for your own use, or inclusion in a table

```
[RAxML] -T 8 -f d -N 100 -m [...] -q [...] -s [...] -z [new tree, step1] -t [unconstrained tree] -n [..
```

3. Run RAxML for the log likelihood output files in PUZZLE format, to be tested in CONSEL (command -f g)

```
[RAxML] -T 8 -f g -m GTRGAMMA -q [...] -s [...] -z [new tree, step1] -n [...]
```

4. Compile the CONSEL files:

```
#navigate to your CONSEL directory
cd src
make
make install
```

5. Navigate back to the folder encapsulating CONSEL (cd ..) and enter your newly named log likelihood file from step3

```
./consel/bin/makermt --puzzle [drag/drop the file from step3]
```

6. You've now got a [name.rmt] file which can be used by CONSEL, so run it:

```
./consel/bin/consel [drag/drop the file from step5]
```

7. You should now have two files, [name.pv] and [name.ci]. The .pv file is all we care about, so read your results in the program catpv:

```
./consel/bin/catpv [drag/drop the .pv file from step6]
```

8. BOOM. You should be sorted.