Table of contents

Expected learning outcomes

The objectivity of this exercise is to help you understand how to infer maximum-likelihood phylogenetic trees using RAxML, PhyML, and IQ-TREE. You will learn how to conduct both basic and more thorough tree searches on DNA and protein sequence alignments, obtain support values using various approaches, and compare the results from multiple independent tree searches and/or different programs.

Getting started

The data sets we will use in this exercise are taken from a large-scale phylogenomic study of 48 modern birds (Jarvis et al, 2014), including a DNA alignment of the exons of a protein-coding gene and the corresponding protein alignment.

RAxML, PhyML, and IQ-TREE differ in their tree search algorithms, functions, command line parameters, and other aspects. You can read the help message of each program as a quick way to learn about its usage (invoke the program with the "-h" option to print its help message). For more details about these programs, the best resources are their own manuals:

RAxML: https://github.com/stamatak/standard-RAxML/blob/master/manual/NewManual.pdf,
PhyML: https://github.com/stephaneguindon/phyml/blob/master/doc/phyml-manual.pdf
IQ-TREE: http://www.iqtree.org/doc/

Along the way, we will explain some important parameters for these programs, but we still recommend you to read the full description in their manuals.

In this exercise, we will use Phylo.io (http://phylo.io/) to visualize and compare trees. Phylo.io is a web-based and quite easy to use. On the web page, simply select the desired mode ("View" or "Compare"), paste your tree(s) in Newick format into the box(es), and then click "Render". You can click on branches to reroot the tree or click on internal nodes to swap subtrees. In the "Compare" mode, you can also use the "reroot" and "reorganize" functions find the best corresponding visualization of two trees.

RAxML

Exercise R1: Basic tree search

To conduct a basic RAxML tree search using parsimony starting tree, run:

```
raxmlHPC -p $RANDOM -m GTRGAMMA -# 1 -s exon_10112.dna.phy -n R1
```

When it finishes, take a look at the output files.
Note: The "-p" option specifies the random number seed (picked by "$RANDOM") required for generating the parsimony starting tree. By using different random number seeds, you can start tree searches from (hopefully) distinct parsimony trees which would allow you to explore different parts of the tree space. Also, by specifying the same random number seed, you will always start from the same tree and get exactly the same result. The "-m" option specifies the models; "GTR" is the only nucleotide substitution model that can be specified via "-m" (check the help message to find out how to use other models if interested), while for models of rate heterogeneity, you can choose between the classic "GAMMA" model and the newer "CAT" model.

Exercise R2: More thorough tree search

We will have to tune some RAxML parameters before we conduct a thorough ML analysis.

First, we will choose between the "GAMMA" and "CAT" models for rate heterogeneity. We will run 10 tree searches under each model (remember to use the same random number seed for both analyses; use "12345" as example here, you can pick your own with "echo $RANDOM"):

```
raxmlHPC -p 12345 -m GTRGAMMA -# 10 -s exon_10112.dna.phy -n R2.a
```

```
raxmlHPC -p 12345 -m GTRCAT -# 10 -s exon_10112.dna.phy -n R2.b
```

Check the final GAMMA-based likelihood score of the best tree returned by each analysis. We will choose the model that gives a better score (use "GAMMA" as example here).

Second, we will test if a better tree can be found using the slower but more thorough search algorithm (again, remember to use the same random number seed):

```
raxmlHPC -f o -p 12345 -m GTRGAMMA -# 10 -s exon_10112.dna.phy -n R2.c
```

Note: The slower algorithm is turned on using the option "-f o".

Check the final GAMMA-based likelihood score of the best tree found by the slower search algorithm. We will use the algorithm that gives a better score.

Third, we will test if a better tree can be found using randomly generated tree as starting trees:

```
raxmlHPC -d -p $RANDOM -m GTRGAMMA -# 10 -s exon_10112.dna.phy -n R2.d
```

Note: The “-d” option instructs RAxML to start from random trees.

Check the final GAMMA-based likelihood score of the best tree found by the analysis using random starting trees. We will choose the type of starting trees that gives a better score.

Now we are ready to run the thorough search for a ML tree by conducting 20 independent searches (assume that random starting tree won the battle above):

```
raxmlHPC -d -p $RANDOM -m GTRGAMMA -# 20 -s exon_10112.dna.phy -n R2.e
```

What is the final GAMMA-based likelihood score of the best tree found by this analysis? Is it the best score we have seen so far? Among the 20 searches, how many times have this best tree being found? In order to answer the last question, we can calculate the distance between the trees returned by the 20 searches:

```
cat RAxML_result.R2.e.RUN.{0..19} > R2.e.trees
raxmlHPC -f r -z R2.e.trees -m GTRGAMMA -n R2.f
```

Note: The “-f r” option computes pairwise RF distances between all pairs of trees in a tree file passed via "-z". The “-m” option has to be provided even though the “GTRGAMMA” model has nothing to do with this analysis (in fact, “-m” and “-n” are always required by RAxML).

Check the output file (“RAxML_info.R2.f”) for the RF distances between the best tree and other trees. If the best tree was found multiple times, it is a good indication that we have conducted sufficient number of tree searches. Otherwise, we might have to perform additional tree searches.

Note: This exercise is adapted from Stamatakis, A. 2015. Using RAxML to infer phylogenies. Curr. Protoc. Bioinform. 51:6.14.1-6.14.14.

Exercise R3: Bootstrap analysis

To perform a standard bootstrap analysis, run:

```
raxmlHPC -b $RANDOM -p $RANDOM -m GTRGAMMA -# 100 -s exon_10112.phy -n R3.a
```

Note: The “-b” option specified a random number seed for standard bootstrap analysis. The “-#” option specified the number of bootstrap replicated to be performed. Bootstrap analysis can be very time consuming, so reduce the number of replicates if you feel the analysis is too slow.

To perform a rapid bootstrap analysis, run:

```
raxmlHPC -x $RANDOM -p $RANDOM -m GTRGAMMA -# autoMRE -s exon_10112.phy -n R3.b
```

Note: The command for rapid bootstrap is largely the same as standard bootstrap, with the only difference being the option used to specify the random number seed (“-x” instead of “-b”). Here we specified “autoMRE” for the “-#” option; it indicates that, instead of running a fixed number of replicates, the “autoMRE” automatic bootstopping criterion will be used to decide when to stop the bootstrap analysis (up to 1000 replicates).

Once these analyses are finished, you can compare the speed of the two types of bootstrap analyses. Also you can map the bootstrap replicates onto the best tree we obtained in exercise R2 and obtain the support values:

```
raxmlHPC -f b -t RAxML_bestTree.R2.e -z RAxML_bootstrap.R3.a -m GTRGAMMA -n R3.c
raxmlHPC -f b -t RAxML_bestTree.R2.e -z RAxML_bootstrap.R3.b -m GTRGAMMA -n R3.d
```

You can then visualize and compare the two trees on Phylo.io. Are the two types of bootstrap support values similar to each other?

In addition, you can combine rapid bootstrap and ML tree search into one single analysis with the “-f a” option:

```
raxmlHPC -f a -x $RANDOM -p $RANDOM -m GTRGAMMA -# autoMRE -s exon_10112.dna.phy -n R3.e
```

Note: In this way, RAxML will first perform the rapid bootstrap analysis, and take the every 5th bootstrap replicate tree as starting tree to conduct ML searches.
Compare the results with the thorough ML search we did in exercise 2 in terms of run time, best tree score, etc.

Exercise R4: Partition model and protein data

In phylogenomic analysis, the data set (e.g. supermatrix containing multiple genes) are often divided into partitions (e.g. by gene) to accommodate the differences in their evolutionary rates and substitution models. Here, we will run a partitioned analysis on the exon sequence alignment. We will divide the alignment into two partitions based on codon positions by creating the following partition model file:

```
DNA, c1 = 1-xxx\2,
DNA, c2 = 2-xxx\2
```

Note: The first line defines the “c1” partition as every other column starting from position 1 to xxx. Similarly, the second line defines the “c2” partition as every other column starting from position 2 to xxx.

We can then run a ML search as we have done in exercise R1, now with the partition model (use the same random number seed so the results can be compared):

```
raxmlHPC -p $RANDOM -m GTRGAMMA -# 1 -s exon_10112.dna.phy -q exon_10112.dna.part
-n R4.a
```

Note: The partition model is provided via “-q”.

So far, we have only analyzed the nucleotide alignment, but protein alignment can be analyzed in the same way. The only difference is the that there are more substitution models available for amino acid. You can analyze the protein alignment as following:

```
raxmlHPC -p $RANDOM -m PROTGAMMAAUTO --auto-prot=bic -# 1 -s exon_10112.dna.phy -q
-n R4.a
```

Note: “-m PROTGAMMAAUTO –auto-prot=bic” instructs RAxML to perform an automatic selection for the best-fit protein model based on the BIC criterion, and use the “GAMMA” model for rate heterogeneity.

X

Exercise R5: Web server

If you have time, explore the CIPRES portal (http://www.phylo.org/index.php/) which is a valuable resource for analyzing large-scale data sets using RAxML.

PhyML

Exercise P1: Basic tree search

To conduct a basic PhyML tree search, run:

```
phyml -i exon_10112.dna.phy -d nt -q -b 0 -m GTR -c 4 -a e -f e -s SPR -o tlr --run_id P1
```

Here, the model is equivalent to the “GTRGAMMA” model we have used in the RAxML exercise. The same model is specified in different ways in the two programs. Check the help message (or manual) to find out the meanings of all the options in the command above.

Note: PhyML also uses the random seed number, but unlike RAxML, it generates the seed automatically. You can provide your own seed using the “--r_seed” option.

Exercise P2: More thorough tree search

To conduct more thorough tree search, we can simply run PhyML multiple times (assign a unique ID to each run so that the results are not overwritten):

```
phyml -i exon_10112.dna.phy -d nt -q -b 0 -m GTR -c 4 -a e -f e -s SPR -o tlr --run_id P2.a0
phyml -i exon_10112.dna.phy -d nt -q -b 0 -m GTR -c 4 -a e -f e -s SPR -o tlr --run_id P2.a1
……
```

In addition, we can add one or more searches using random starting trees to the analysis:

```
phyml -i exon_10112.dna.phy -d nt -q -b 0 -m GTR -c 4 -a e -f e -s SPR -o tlr --rand_start
--n_rand_starts 10 --run_id P2.b
```

Note: This analysis include five searches using random starting trees and one search using the default starting tree.

Another option is to provide your own starting trees (e.g. the starting trees from the RAxML exercise). This can be done via the “-u” option:

```
phyml -i exon_10112.dna.phy -d nt -q -b 0 -m GTR -c 4 -a e -f e -s SPR -o tlr --u
RAxML_parsimonyTree.R1 --run_id P2.c
```

Exercise P3: Compare PhyML and RAxML

Having analyzed the same data set using both PhyML and RAxML, we can now compare the likelihood scores of the best trees found by the two programs. It is important to keep in mind that the likelihood scores from different phylogenetic programs are not necessarily comparable. For a fair comparison, we will evaluate all the trees using the same program. To calculate the likelihood score for a given tree (e.g. the best tree found in exercise R1) using RAxML, run:

```
raxmlHPC -f e -t RAxML_bestTree.R1 -m GTRGAMMA -s exon_10112.dna.phy -n P3.a
```

In exercise P2, we have performed a PhyML tree search using the RAxML parsimony starting tree from exercise R1. We can evaluate the best tree found by PhyML in the same way:

```
raxmlHPC -f e -t exon_10112.dna.phy_phyml_tree_P2.c -m GTRGAMMA -s
exon_10112.dna.phy -n P3.b
```

Check the likelihood scores of the two trees, which one is better? Besides the scores, you should also calculate the RF distance between the trees; different Newick representations of the same topology may give rise to slightly different scores. Also, keep in mind that this is a single data set and a single tree search, so this one comparison cannot tell you which program performs better in general.

To do the same thing in PhyML:

```
phyml -i exon_10112.dna.phy -d nt -q -b 0 -m GTR -c 4 -a e -f e -o lr --u RAxML_bestTree.R1
--run_id P3.c
phyml -i exon_10112.dna.phy -d nt -q -b 0 -m GTR -c 4 -a e -f e -o lr --u
exon_10112.dna.phy_phyml_tree_P2.d --run_id P3.d
```

Exercise P4: Bootstrap analysis and alternative approaches

To perform a standard bootstrap analysis in PhyML, run:

```
phyml -i exon_10112.dna.phy -d nt -q -b 100 -m GTR -c 4 -a e -f e -o lr --u RAxML_bestTree.R1
--run_id P4.a
```

This one analysis includes a search for the ML tree and 100 bootstrap replicates, and the bootstrap support values will be mapped onto the tree at the end of the analysis. You can also map the PhyML bootstrap trees onto the RAxML best tree (or vice versa) and compare the bootstrap support values from the two programs.
Note: PhyML does not have the “bootstopping” function so you have to specify the number of bootstrap replicates. Here we run 100 replicates, but feel free to reduce the number of replicates.

The standard bootstrap analysis can be very time consuming for larger data sets, therefore PhyML offers several alternative, fast approaches to calculate branch supports. These can be specified using the “-b” option and we can calculate the support values for a user provided tree. For example, to calculate the approximate likelihood ratio test (aLRT) support values:

```
phyml -i exon_10112.dna.phy -d nt -q -b -2 -m GTR -c 4 -a e -f e -o lr --u RAxML_bestTree.R1
--run_id P4.b
```

To calculate SH-like aLRT support values:

```
phyml -i exon_10112.dna.phy -d nt -q -b -4 -m GTR -c 4 -a e -f e -o lr --u RAxML_bestTree.R1
--run_id P4.c
```

To calculate approximate Bayes support values:

```
phyml -i exon_10112.dna.phy -d nt -q -b -5 -m GTR -c 4 -a e -f e -o lr --u RAxML_bestTree.R1
--run_id P4.d
```

You can then use Phylo.io to visualize and compare these trees which have the same topology but different types of support values. Note that the alternative support values range between 0 and 1 while the bootstrap support values range between 0 and 100.

Exercise P5: Partition model and protein data

In PhyML, the partition model is specified using a XML configuration file, which is quite complicated but at the same time really powerful. You can use the provided partition model and alignment files to run the partitioned analysis:

```
phyml --xml=exon_10112.dna.xml
```

Protein alignments can be analyzed in the same way as DNA alignments. Unlike RAxML, PhyML does not offer automated model selection so you have to specify the amino acid substitution model for the analysis. Here we analyze the protein alignment with the “LG” model:

```
phyml -i exon_10112.pep.phy -d aa -q -b -0 -m LG -c 4 -a e -f m -o tlr --run_id P5.b
```

You can also run the analysis using the model selected by RAxML (in exercise R4), and compare the results of PhyML and RAxML.

Exercise R6: Web server

If you have time, explore the PhyML web server (http://www.atgc-montpellier.fr/phyml/). Note that it runs an earlier version of PhyML (version 3.0).

IQ-TREE

Exercise I1: Basic tree search

To conduct a basic tree search in IQ-TREE, run:

```
iqtree -s exon_10112.dna.phy -m GTR+G -pre I1
```

Note how the same model (GTR for nucleotide substitution model, and Gamma for rate heterogeneity) is specified in the three programs. Also, IQ-TREE prints out detailed progress information during its run. Check the information and try to match with the algorithm of IQ-TREE.

Note: Similar to PhyML, IQ-TREE generates its random seed number automatically. You can provide your own seed using the “--seed” option.

Exercise I2: More thorough tree search

There are multiple strategies to conduct more thorough tree searches in IQ-TREE. First, we can run IQ-TREE multiple times (assign a unique ID to each run, otherwise IQ-TREE will complain):

```
iqtree -s exon_10112.dna.phy -m GTR+G -pre I2.a0
iqtree -s exon_10112.dna.phy -m GTR+G -pre I2.a1
……
```

In addition, we can increase the length of the tree search:

```
iqtree -s exon_10112.dna.phy -m GTR+G -nstop 500 -pre I2.b
```

Note: The option “-nstop” specifies the number of unsuccessful iterations before stop. Here we change the value from 100 (default) to 500.

Another critical parameter to tune for difficult data set is the strength of randomized NNI perturbation. It is recommended to use smaller perturbation strength for data sets with many short sequences. Here we try to change to strength from 0.5 (default) to 0.2, but you can try larger values as well:

```
iqtree -s exon_10112.dna.phy -m GTR+G -pers 0.2 -pre I2.c
```

IQ-TREE by default only consider a subset of the NNI neighbors during its tree search. We can conduct a more thorough NNI search by turn on the “-allnni” option:

```
iqtree -s exon_10112.dna.phy -m GTR+G -allnni -pre I2.d
```

Compare the results and run time of all the four strategies. Which one has the best performance on this data set?

Exercise I3: Compare IQ-TREE and other programs

Now we can compare the results of IQ-TREE, RAxML, and PhyML. For example, we will use IQ-TREE to evaluate the ML trees found by the three programs:

```
iqtree -s exon_10112.dna.phy -m GTR+G -te [RAxML best tree file] -pre I3.a
iqtree -s exon_10112.dna.phy -m GTR+G -te [PhyML best tree file] -pre I3.b
iqtree -s exon_10112.dna.phy -m GTR+G -te [IQ-TREE best tree file] -pre I3.c
```

Check the likelihood scores of these trees, which one is better? You can also use IQ-TREE to calculate the RF distances between the trees:

```
cat [RAxML best tree file] [PhyML best tree file] [IQ-TREE best tree file] > I3.trees
iqtree -rf_all I3.tree -pre I3.d
```

Again, keep in mind that this is a single data set and a single tree search, so this one comparison cannot tell you which program performs better in general.

Exercise R4: Bootstrap analysis

To perform a standard bootstrap analysis in IQ-TREE, run:

```
iqtree -s exon_10112.dna.phy -m GTR+G -b 100 -pre I3.a
```

This will include one ML tree search and 100 bootstrap replicates in one run, and the bootstrap support values will be mapped onto the ML tree at the end of the analysis.

Note: IQ-TREE does not offer a “bootstopping” function for the standard bootstrap analysis, so you have to specify the number of bootstrap replicates. Here we run 100 replicates, but feel free to reduce the number of replicates.

To perform an ultra-fast bootstrap (UFBS) analysis, run:

```
iqtree -s exon_10112.dna.phy -m GTR+G -bb 1000 -pre I3.b
```

Note: The number of UFBS replicates is specified via the “-bb” option while for standard bootstrap it is via the “-b” option.

Different from the standard bootstrap and the rapid bootstrap of RAxML, the ultra-fast bootstrap analysis is conducted during the search for ML tree. Check the run time of the UFBS analysis and compare with that of the ML tree search and the standard bootstrap analysis. Also, you can compare the support values from the standard and ultra-fast bootstrap analyses.

Similar to PhyML, IQ-TREE also implements the aLRT, SH-like aLRT, and aBayes analyses, and it can run these analyses on a given tree. To calculate the aLRT support values:

```
iqtree -s exon_10112.dna.phy -m GTR+G -alrt 0 -te RAxML_bestTree.R1 -pre I3.c
```

To calculate the SH-like aLRT support values:

```
iqtree -s exon_10112.dna.phy -m GTR+G -alrt 1000 -te RAxML_bestTree.R1 -pre I3.d
```

To calculate the aBayes support values:

```
iqtree -s exon_10112.dna.phy -m GTR+G -abayes -te RAxML_bestTree.R1 -pre I3.e
```

You can then visualize and compare these trees which have the same topology but different types of support values on Phylo.io. You can also compare the aLRT, SH-like, and aBayes support values calculated by IQ-TREE and PhyML. Do they match?

Exercise I5: Partition model and protein data

IQ-TREE accepts partition model files in the RAxML format. It also supports a more complicated but more powerful NEXUS format. You can use either of them to run the analysis. To use the RAxML partition model file, run:

```
iqtree -s exon_10112.dna.phy -m GTR+G -q exon_10112.dna.part -pre I5.a
```

To use the NEXUS partition model file, run:

```
iqtree -q exon_10112.dna.nxs -pre I5.b
```

Note: The NEXUS partition model file contains information on the alignment file, partition, and model, therefore no other options are needed.

Protein alignments can be analyzed in the same way as DNA alignments. Like RAxML, IQ-TREE also implements the model selection function (for both DNA and protein data):

```
iqtree -s exon_10112.dna.phy -m TEST -mset raxml -pre I5.c
```

Note: The “-m TEST” option instructs IQ-TREE to perform model selection before the ML tree search, while the “-mset raxml” option restricts the candidate models to the ones supported by RAxML.

Now you can compare the result of IQ-TREE with that of RAxML and PhyML.

Exercise R6: Web server

If you have time, explore the IQ-TREE web server (http://iqtree.cibiv.univie.ac.at/).