Tutorial: Substitution Models and Model Testing

Seraina Klopfstein & Fredrik Ronquist
RevBayes workshop, ACEBB, The University of Adelaide. November 17-21, 2014

Exercise 1: Substitution models

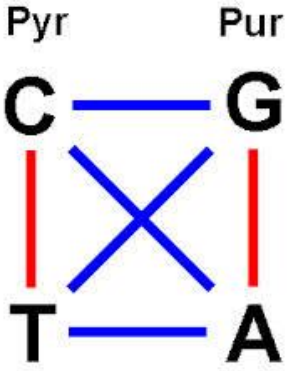
1.a: GTR model script

Have a look at the script of the GTR phylogenetic models from Monday or Tuesday. Make sure you understand every line, and solve the following questions:

- Which DAG nodes (= variables) in your model graph are stochastic, which are deterministic?
- Which DAG nodes are part of a prior? Which type of dag nodes have priors?
- Which DAG nodes have moves attached to them?
- Which DAG nodes are clamped? What type of DAG nodes are those?
- If you draw the DAG, where on the graph are those nodes that are clamped?

2.b: Jukes-Cantor and HKY85 models

Based on the GTR model script, write one for the Jukes-Cantor model. To do so, set the priors on the exchangeability rates and base frequency according to JC, and use the function `fnGTR` to set up the Q matrix, as we did previously for GTR.



Then try the same for the HKY85 model, which has unequal state frequencies and two different exchangeability rates, one each for transitions and transversions. Think carefully of how many free parameters you have in that model, and add adequate priors to the free parameters.

Pass your parameters for the state frequencies and exchangeability rates to the function fnGTR in the following sequence:

A, C, G, T // r(AC), r(AG), r(AT), r(CG), r(CT), r(GT)

Exercise 2: Partitioned GTR + Gamma + Invar model

Use the provided skeletal Rev script "Hymenoptera_Partitioning_skeletal.Rev" to include your blocks from exercises 2.a and 2.b.

2.a: Creating data partitions

Based on previous Rev scripts, create a Rev block that reads in the Hymenoptera data in the following partitions:

- 28S and 16S genes as two separate data partitions, each unpartitioned
- EF1α and CO1, both protein-coding genes, should be partitioned into combined 1st and 2nd versus the 3rd codon positions.

Use a vector of data objects to store the data. Functions that you will need:

```
data[1] <- readDiscreteCharacterData("data_16S.nex")
data[2] <- readDiscreteCharacterData("data_EF1a.nex")
data[2].setCodonPartition(v(1,2))
```

2.b: Set-up a partitioned GTR substitution model

For each of your partitions, now set up a separate GTR substitution model and a separate among-site variation model (gamma distribution and a proportion of invariant sites). You will need to use a 'for'-loop over the partitions to set up parameters for each partition. You can get the number of partitions using the function `data.size()`.

```
for (i in 1:n_parts){
... (GTR model setup here)
}
```

As we have seen before for the data objects, use vectors for the model parameters as well. The stationary state frequencies could f.i. be defined as follows, with the square brackets defining a vector object and containing the element index:

```
pi[i] ~ dnDirichlet(pi_prior[i])
```

When partitioning the data, we also want to allow each gene to evolve at a different rate. To achieve this, define a stochastic node 'PartMult' which is a Simplex with as many elements as there are partitions (use a Dirichlet distribution prior). Because the actual rate multipliers per partition have to be weighted by the number of sites they include, then add the following lines to create the final parameter 'part_rate_mult' vector:

```
for (i in 1:n_parts) {
     part_rate_mult [i] := PartMult[i] * tot_nSites / n_sites[i]
}
```

Do you understand why these lines are necessary, and what they are doing? Where do you have to add a move?

2.c: Combine with tree model and run MCMC

Now scroll over the tree model set-up in the skeletal file, up to the block called "PhyloCTMC Models for each partition" ('Phylogenetic Continuous-Time Markov Model'). Add the parameter vectors that you created previously to the `'dnPhyloCTMC'` call.

We create one stochastic node for `each` partition, each distributed according to its own PhyloCTMC distribution. They now each get clamped with the respective data partition (add the correct name to the `.clamp` function).

Check the remainder of the file, and run the mcmc analysis.

Exercise 3: Model testing using power posteriors

3.a: Estimate the marginal likelihood of your partitioned model

Use stepping-stone sampling to estimate the marginal likelihood of your partitioned model. To do so, create a copy of your file and replace the mcmc part with the estimation of a power posterior and subsequent stepping-stone sampling. You will need the following lines:

```
pow_p = powerPosterior(mymodel, moves, "file.out", cats=20)
pow_p.burnin(generations=5000,tuningInterval=100)
pow_p.run(generations=200)

ss = steppingStoneSampler(file="file.out", powerColumnName="power",
likelihoodColumnName="likelihood")
ss.marginal()
```

This will take a while - nevertheless, run it again after modifying the numbers of generations and/or categories and check the impact on the estimate of the marginal likelihood (the number of categories is pretty low with 20, a value of 50 is usually recommended). As in MCMC, it is good practise to run several independent runs and compare the results, to make sure that the samples are taken from the posterior / power posterior.

Record the estimated marginal likelihoods.

3.b: Model testing

Now test one or several of the following models against the partitioned GTR model, by running another stepping-stone analysis under a different model setup:

- Jukes-Cantor instead of GTR model for all partitions
- HKY85 model for 18S
- unpartitioned model

- not separating 18S and 28S
- using a SYM model for 28S (6 exchangeability rates, but equal base frequencies), and GTR models for the rest
- remove the proportion of invariant sites for the 3rd codon positions
- a directional model of evolution: the dnPhyloCTMC distribution accepts root frequencies as a separate argument, which leads to a non-stationary model of evolution. Add a directional component to your model for the first and second codon positions of the protein-coding genes, and test it against the stationary model.
- ...

Compare your models using Bayes factors; the test statistic 2*(lnMarginalLikelihood[model1] - lnMarginalLikelihood[model2]) should be interpreted according to the following table:

2 ln BF	**Strength of evidence**
0 to 2	Not worth more than a bare mention
2 to 6	Positive
6 to 10	Strong
>10	Very strong