Phylogenetic Inference using `RevBayes`

Historical biogeography for many areas

Michael Landis

1 Large numbers of areas

1.1 Data augmentation

For small rate matrices, transition probabilities of beginning in state i and ending in state j equal the matrix exponential of the underlying rate matrix, scaled by the elapsed time of the process. This integrates over all unobserved transition events during the time interval t. Unfortunately, computing the matrix exponential scales poorly as the state space increases, *i.e.,* $O(n^4)$ for n states.

Alternatively, the probability of beginning in state i and ending in state j can be computed easily when the explicit series of event types and times are known. While we will never know the exact history of events, we can use stochastic mapping in conjunction with Markov chain Monte Carlo (MCMC) to repeatedly sample range evolution histories that are consistent with the ranges observed in the study taxa at the tips of the phylogeny. This technique is called data augmentation. It was first applied in phylogenetics to tertiary structure-dependent evolution of protein-coding nucleotide sequences (**?**), and later extended to range evolution in **?**.

Using data augmentation, we will approximate $\text{Prob}(\mathbf{X}_{aug}, \theta \mid \mathbf{X}_{obs}, T, M)$, where $\mathbf{X}_{obs}$ is the range data observed at the tips, $\mathbf{X}_{aug}$ is the distribution of ancestral range reconstructions over the phylogeny, T, where $\mathbf{X}_{aug}$ is inferred jointly with the parameters, θ, assuming the range evolution model, M, that describes $\mathbf{Q}$ above. From a Bayesian perspective, $\mathbf{X}_{aug}$ is effectively treated as part of the parameter space in the posterior distribution. Just as MCMC evaluates the posterior distribution for sampled parameter values, the same is done for sampled character histories. Thus, ancestral range reconstructions are a by-product of data augmentation, which are often of primary interest to biogeographers.

You may wonder why matrix exponentiation works fine for molecular substitution models and large multiple sequence alignments. It is the non-independence of character evolution that induces large state spaces, along with cladogenic processes that affect the entire set of characters (e.g. non-identical range inheritance following speciation). Molecular substitution models typically assume each character in an alignment evolves independently, which may be justified by the ability of recombination to degrade linkage disequilibrium over geological time scales. Conveniently, this keeps $\mathbf{Q}$ small even for datasets with many sites.

1.2 Large rate matrices and distance-dependent dispersal

A rate matrix may be represented compactly as the rate function

$$q^{(a)}_{\mathbf{y},\mathbf{z}} = \begin{cases} \lambda_0 & \text{if } z_a = 0 \\ \lambda_1 & \text{if } z_a = 1 \\ 0 & \mathbf{y} \text{ and } \mathbf{z} \text{ differ in more than one area} \end{cases} .$$

where $\mathbf{y}$ and $\mathbf{z}$ are the "from" and "to" ranges and a is the area that changes, λ_0 is the per-area rate of loss, and λ_1 is the per-area rate of gain. For example, $q^{(1)}_{011,111}$ is the rate of range expansion for $011 \rightarrow 111$ to gain area 1, which equals λ_1. Note the rate of more than one event occurring simultaneously is zero, so a range must expand twice by one area in order to expand by two areas. This model is analogous to the Jukes-Cantor model for three independent characters with binary states, except the all-zero "null range" is forbidden.

Extending this idea, we may reasonably expect that a range expansion event into an area depends on which nearby areas are currently inhabited, which imposes non-independence between characters. The transition rate might then appear as

$$q^{(a)}_{\mathbf{y},\mathbf{z}} = \begin{cases} \lambda_0 & \text{if } z_a = 0 \\ \lambda_1 \eta(\mathbf{y}, \mathbf{z}, a, \beta) & \text{if } z_a = 1 \\ 0 & \mathbf{y} = 00...0 \\ 0 & \mathbf{y} \text{ and } \mathbf{z} \text{ differ in more than one area} \end{cases} .$$

For this tutorial, you can take $\eta(\cdot)$ to adjust the rate of range expansion into area a by considering how close it is to the current range, $\mathbf{y}$ relative to the closeness of all other areas unoccupied by the taxon. The β parameter rescales the importance of geographic distance between two areas by a power law. Importantly, $\eta(\cdot) = 1$ when $\beta = 0$, meaning geographic distance between areas is irrelevant. Moreover, when $\beta > 0$, $\eta(\cdot) < 1$ when area a is relatively distant and $\eta(\cdot) > 1$ when area a is relatively close. See **?** for a full description of the model.

? Write down a rate function where the per-area rates of gain and loss depend on the number of currently occupied areas.

1.3 Specifying a data augmented DEC model

Start to create helper variables for file handling,

```
area_fn = "data/earth25.still.atlas.txt"
data_fn = "data/primates_bg_n25.nex"
tree_fn = "data/primates.tree"
out_fn = "output/bg_large"
```

read in our tree,

```
psi  <- readTrees( tree_fn )[1]
```

populate our range observations,

```
data    = readDiscreteCharacterData( data_fn )
```

and read in our geographical information (for details, see Section XX).

```
atlas   = readAtlas( area_fn)
n_areas = atlas.nAreas()
```

Lastly, create index variables to populate our move and monitor vectors,

```
mi = 0
```

and assign the number of generations to run the MCMC analysis

```
ngen = 500000
```

To later interpret ancestral state monitors phylogenetically, save a copy of `tree` annotated with internal node indexes.

```
write(psi,filename=tree_fn+".index.tre")
```

Proceeding with the model configuration, we'll first create our rate matrix that determines the rate of per-area gain and loss given the current geographical layout of the range. In `RevBayes`, data-augmented CTMC analyses require `RateMap` functions to determine event rates, which differ from the familiar `RateMatrix` functions that include `fnJC` and `fnGTR` as members. In their simplest form, `RateMap` functions generate the rates of change over the full set of characters, where each character evolves according to a provided `RateMatrix` function. Additionally, a `RateMap` accepts a rate modifier function that induces some correlation structure to character change evolution. In this section, we'll be creating a biogeographic `RateMap` function for the dispersal-extinction process given in Section 1.2.

First, create a biogeographical clock to scale the rate of range evolution.

```
clock_bg ~ dnExponential(10)
moves[++mvi] = mvScale(clock_bg, lambda=0.5, weight=5.0)
```

Next, instantiate a simplex of gain and loss rates, distributed by a flat Dirichlet prior,

```
glr ~ dnDirichlet([1,1])
moves[++mvi] = mvSimplexElementScale(glr, alpha=30.0, weight=5.0)
```

and use deterministic nodes to assign the rates nicknames.

```
r_gain := glr[1]
r_loss := glr[2]
```

Insert the simplex into the rate matrix q_{area}, which gives the average rate of area gain and loss per area.

```
q_area := fnFreeBinary(glr)
```

Next, we will create `dp` to represent the β parameter, which determines the importance of geographical distance to dispersal. Remember that values of β far from zero means distance is important. So, if we we assign a prior that pulls β towards zero, then posterior values of β far from zero indicate the range data are informative of the importance of distance to dispersal. We'll use an exponential distribution with mean 1.0 as a prior for `dp`.

```
dp ~ dnExponential(10.0)
moves[++mvi] = mvScale(x=dp, lambda=0.5, tune=true, weight=5.0)
```

We will also create a deterministic node to modify the rate of dispersal between areas by evaluating `dp` and `atlas`. This node is determined by the function `fnBiogeoGRM`, where GRM stands for "geographical rate modifier", and plays the role of the $\eta(\cdot)$ rate-modifier function mentioned earlier. We will tell the `fnBiogeoGRM` function to modify dispersal rates based on distances and whether or not the area exists during an epoch.

```
grm := fnBiogeoGRM(atlas=atlas, distancePower=dp, useDistance=true)
```

Now we need a deterministic node to represent the rate matrix, **Q**. To determine the value of this node, we'll use the function `fnBiogeoDE` to assign our model parameters to transition rates as described in the

introduction. As input, we'll pass our gain and loss rates, `q_area`, our geographical rate modifier, `grm`, and the biogeographical clock `clock_bg`. In addition, we'll inform the function of the number of areas in our analysis and whether we will allow species to be absent in all areas (i.e. have the null range).

```
q_range := fnBiogeoDE(gainLossRates=q_area, branchRates=clock_bg, geoRateMod=grm,
   numAreas=n_areas, forbidExtinction=true)
```

As with the simple DEC model, we assign a flat Dirichlet prior over cladogenic event type probabilities

```
clado_prob ~ dnDirichlet( [1, 1, 1] )
widespread_sympatry := clado_prob[1]
subset_sympatry   := clado_prob[2]
allopatry         := clado_prob[3]
moves[++mvi] = mvSimplexElementScale(clado_prob, alpha=20.0, weight=5.0)
```

Finally, the data evolve according to a phylogenetic CTMC process. Here, we decalre a stochastic node using `dnPhyloDACTMC` where `DA` indicates the distribution requires data augmentation to compute the likelihood rather than Felsenstein's pruning algorithm. To create the distribution, we must pass it our `tree` and `q_range` objects, but additionally inform the distribution that it will be using a biogeographic model, that it will introduce the simple cladogenic range evolution events described in `?` (`useCladogenesis=true`), and that it will assign zero probability to a transition away from the null range state.

```
m ~ dnPhyloDACTMC(tree=psi, Q=q_range, cladoProbs=clado_prob, type="Biogeo",
   forbidExtinction=true, useCladogenesis=true)
```

So we may evaluate the graphical model's likelihood, we tell the CTMC to observe the `data` object, which then primes the model with data-augmented character histories.

Now `m` has a defined likelihood value.

```
m.clamp(data)
m.lnProbability()
   -156.0288
```

To integrate over the space of possible range histories, we still need to add moves to propose new data augmented histories. The major challenge to sampling character histories is ensuring the character histories are consistent with the observations at the tips of the tree. The proposals in this tutorial use `?`'s rejection sampling algorithm, with some modifications to account for cladogenic events and epoch-based rate maps.

The basic idea is simple. Each time a character history proposal is called, it selects a node at random from the tree. Branch history proposals propose a new character history to a single randomly-chosen branch.

Node history proposals propose a new character history for the three branches connecting to a randomly chosen node, in addition to the cladogenic state of the node itself.

For each proposal, each character's history is resampled with probability `lambda` – i.e. `lambda=0.01` resamples 1% of characters, while `lambda=1.` resamples all characters. Once the new character history is proposed, the likelihood of the model is evaluated and the MCMC accepts or rejects the new state according to e.g. the Metropolis-Hastings algorithm.

Cladogenic events require special considerations during sampling, so we indicate `type="Biogeo"`. Currently, only rejection-sampling is available for cladogenic histories, hence `proposal="rejection"`. Finally, we apply high `weight` values to the proposals since the larger the state space is, the more character history proposals needed to effectively integrate over the space of sample paths.

Let's create the character history moves as follows: conservative character history updates for paths and nodes, with `lambda=0.05`

```
moves[++mvi] = mvCharacterHistory(ctmc=m, qmap=q_range, tree=psi, lambda=0.05,
        type="Biogeo", graph="node", proposal="rejection", weight=n_nodes*5)
moves[++mvi] = mvCharacterHistory(ctmc=m, qmap=q_range, tree=psi, lambda=0.05,
        type="Biogeo", graph="branch", proposal="rejection", weight=n_nodes)
```

and the same proposals for moderately-sized character history updates, with `lambda=0.2`

```
moves[++mvi] = mvCharacterHistory(ctmc=m, qmap=q_range, tree=psi, lambda=0.2,
        type="Biogeo", graph="node", proposal="rejection", weight=n_nodes*5)
moves[++mvi] = mvCharacterHistory(ctmc=m, qmap=q_range, tree=psi, lambda=0.2,
        type="Biogeo", graph="branch", proposal="rejection", weight=n_nodes)
```

and radical updates

```
moves[++mvi] = mvCharacterHistory(ctmc=m, qmap=q_range, tree=psi, lambda=1.0,
        type="Biogeo", graph="node", proposal="rejection", weight=n_nodes*5)
moves[++mvi] = mvCharacterHistory(ctmc=m, qmap=q_range, tree=psi, lambda=1.0,
        type="Biogeo", graph="branch", proposal="rejection", weight=n_nodes)
```

Next, create the model object,

```
my_model = model(m)
```

and monitors for our simple parameters.

```
monitors[1] = mnScreen(clock_bg, r_gain, r_loss, dp, subset_sympatry, allopatry,
    widespread_sympatry, printgen=100)
monitors[2] = mnFile(clock_bg, r_gain, r_loss, dp, subset_sympatry, allopatry,
    widespread_sympatry, filename=out_fn+".params.txt", printgen=10)
```

Like any parameter, we can sample the augmented range histories from the MCMC to approximate the posterior distribution of range histories. This is statistically equivalent to generating ancestral state reconstructions from a posterior distribution via stochastic mapping. We will extract these reconstructions using special monitors designed for the `dnPhyloDACTMC` distribution.

Next, we will create `mnCharHistoryNewick` monitors to record the sampled character history states for each node in the tree. This monitor has two `style` options: `counts` reports the number of gains and losses per branch in a tab-delimited Tracer-readable format; `events` reports richer information of what happens along a branch, anagenically and cladogenically, using an extended Newick format. How to read these file formats will be discussed in more detail in Section 1.4.

```
monitors[3] = mnCharHistoryNewick(filename=out_fn+".events.txt", ctmc=m, tree=psi,
    printgen=100, style="events")
monitors[4] = mnCharHistoryNewick(filename=out_fn+".counts.txt", ctmc=m, tree=psi,
    printgen=100, style="counts")
```

As our last monitor,`mnCharHistoryNhx` records character history values throughout the MCMC analysis, then stores some simple posterior summary statistics as a Nexus file. These summary statistics could be computed from the previously mentioned monitor output files, but `mnCharHistoryNhx` provides a simple way to produce Phylowood-compatible files. We will also discuss this file's format in more detail later in the tutorial.

```
monitors[5] = mnCharHistoryNhx(filename=out_fn+".phw.txt", ctmc=m, tree=psi, atlas=atlas
    , samplegen=100, maxgen=ngen, burnin=0.5)
```

Finally, create the MCMC object

```
my_mcmc = mcmc(my_model, monitors, moves)
```

and run

```
my_mcmc.run(generations=ngen)
```

The posterior surface over character histories is highly multimodal, so this analysis takes a long time to mix. If you're following this tutorial in a lab setting, you should proceed using the pre-analysed output files provided in `./example_output`.

1.4 Analysis output

We'll focus some attention on node 42, the most recent common ancestor of chimps and macaques, designated as the Old World most recent common ancestor (MRCA).

1.4.1 Biogeographic event counts from mnCharHistoryNewick

Recording stochastic mappings in a Tracer-compatible format requires some summarization. This monitor generates a tab-delimited file where the number of events of each type for each branch is recorded.

Open `./output/bg_3.counts.txt` in a text editor.

```
Iter   Posterior  Likelihood Prior  t_s0   t_s1   t_c0   t_c1   t_c2   t_c3   b0_s0
     b0_s1  b0_c   ...
0   -6518.54   -6519.25   0.706823   977 839 22 0   0   0   38  28  0   ...
10  -583.147   -585.333   2.186240   66  37  3   7   9   3   2   1   2   ...
20  -296.540   -297.340   0.799755   20  25  15  1   3   3   0   1   0   ...
30  -276.284   -277.257   0.972918   17  24  14  0   4   4   0   1   0   ...
40  -266.569   -266.948   0.379624   18  23  15  1   3   3   0   1   3   ...
...
```

For example, `b0_s1` gives the number of areas that are gained for the branch leading to the node indexed 0. Referencing FigTree, we see this corresponds to chimpanzees. `b0_c` gives the cladogenic event type that gives rise to the chimp lineage, where narrow sympatry, widespread sympatry, subset sympatry, and allopatry are recorded as `0`, `1`, `2`, and `3`, respectively. The columns `t_s0` and `t_s1` give the sum of events over all branches. `t_c0`, `t_c1`, `t_c2`, and `t_c3` give the total number of narrow sympatric, widespread sympatric, subset sympatric, and allopatric cladogenic events over the entire tree.

1.5 Biogeographic event histories from mnCharHistoryNewick

For more detailed data exploration, this analysis also provides annotated Newick strings with the complete character mappings for the tree. (This file format is a bit unwieldy and under revision.)

→ Open `./output/bg_3.events.txt` in a text editor.

Each iteration records the data-augmented character history (stochastic mapping) using metadata labels, which, for an internal node, looks like

```
[&index=42;nd=000000000000101000000000;pa=000000000000100000000000;ch0
   =000000000000101000000000;ch1=000000000000001000000000;cs=s;bn=41;ev={{t:0.138033,
   a:39.4458,s:1,i:15}}]
```

Having consulted FigTree earlier, we know node 42 is the Old World MRCA. The branch began with the range `pa=000000000000100000000000` and terminated in the state `nd=000000000000101000000000`.

Since this node is not a tip node, it represents a speciation event, so the daughter ranges are also given, `ch0=000000000000000101000000000` and `ch1=0000000000000000001000000000`. The cladogenic state for this speciation event was subset sympatric, `cs=s`, rather than sympatric (wide or narrow; `w` or `n`) or allopatric (`a`).

Anagenic dispersal and extinction events occurring along the lineage leading to node 42 are recorded in `ev`, where each event has a time (relative to the absolute branch length), absolute age, state (into), and character index (`t`, `a`, `s`, `i`, resp.). Potentially, `ev` contains multiple events. For this posterior sample, the MRCA of chimps and macaques dispersed into East Africa 39.4458 million years ago, which is consistent with the ranges recorded by `pa` and `nd`.

Although we have stochastic mappings under the posterior distribution, the remaining challenge is to summarize them into something useful. The Python script `bg_parse.py` is provided to manipulate this data format. Below are a few examples of interesting features of the posterior.

→ Open a Python console and read in the events.

```
> cd RevBayes_scripts
> python

...

>>> from bg_parse import *
>>> dd=get_events(fn="../output/bg_3.events.txt")
```

By default, `get_events()` extracts a dictionary-of-dictionaries from the posterior event samples. A dictionary is very much like a vector, except values are selected by keys, so this can be thought of as similar to a two-dimensional matrix. The first key corresponds to the node (branch) whose MCMC samples you'd like to retrieve, while the second dictionary's keys are the columns correspond to MCMC states and values specific to that node. For example, say we are interested in the last five cladogenic states sampled for node 42, Old World MRCA,

```
>>> dd[42].keys()
  ['ch1', 'iteration', 'bn', 'nd', 'ch0', 'prior', 'posterior', 'pa', 'cs', 'ev', '
    likelihood']
>>> dd[42]['cs'][-5:]
  ['widespread_sympatry', 'widespread_sympatry', 'widespread_sympatry', '
    widespread_sympatry', 'widespread_sympatry']
```

To get the `n=1` highest-valued sample for a branch by its posterior value

```
>>> get_best(dd[42],n=1,p='posterior')
```

{'prior': [-0.524716], 'iteration': [46700], 'bn': [41], 'nd': [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]], 'ch0': [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]], 'ch1': [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]], 'posterior': [-95.7676], 'pa': [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]], 'cs': ['widespread_sympatry'], 'ev': [[]], 'likelihood': [-95.2429]}

More data-exploration functions are found in `bg_parse.py`.

1.6 Exercises

- (Under construction.)

References

References

Version dated: July 10, 2016