Historical biogeography using data augmentation in RevBayes

Michael J. Landis
mlandis@berkeley.edu

August 18, 2014

1 Introduction

Using RevBayes, this lab describes how to perform Bayesian inference of historical biogeography using RevBayes. The analysis jointly infers the posterior range evolution parameters and ancestral ranges using the data augmentation approach described in Landis et al. (2013). To generate hypotheses, we will explore the resulting posterior using Tracer, an MCMC analysis tool created by Andrew Rambaut, and Phylowood, a Javascript web service that animates and filters phylogenetic biogeography reconstructions (Landis and Bedford 2014). Finally, we will quantify and plot certain qualities of the inferred biogeographic posterior using R.

Outline
1) Brief overview of model and method
2) Historical biogeography analysis
2.a) Input
2.b) MCMC inference
3) Review results.
3.a) Parameter output
3.b) Ancestral range output
3.c) Visualize ancestral range reconstructions.

→ These little arrows indicate lines containing key information to progress through the lab. The rest of the text gives context for why we're taking these steps or what to make of results.

1.1 Handy links for this lab

Lab zip file	[NESCent URL]
RevBayes	`https://github.com/revbayes/revbayes`
Phylowood software	`http://mlandis.github.io/phylowood`
Phylowood manual	`https://github.com/mlandis/phylowood/wiki`
Tracer	`http://tree.bio.ed.ac.uk/software/tracer`

1.2 Setting up your workspace

The practical part of the lab will analyze a small dataset of 19 taxa distributed over 4 biogeographic areas. Parts of the lab will require entering terminal commands, which will assume you are using the Unix shell `bash`. Just ask for help if the commands don't seem to work.

→ Portions of this lab require `R` is installed, along with the `stringr` and `ggplot2` packages.

→ This tutorial will assume you have successfully installed RevBayes and can be called from your current working directory.

→ Download and unzip the lab zip file, `bayarea_lab.zip`, into `~/apps/bayarea_1.0.2`.

→ Copy all files into `./examples/`

```
> ls bayarea_lab
my_data.txt                  my_run.area_states.txt
my_geo.txt                   my_run.nhx
my_lab_run.sh                my_run.parameters.txt
my_other_run.parameters.txt  my_tree.txt
my_run.area_probs.txt        pp_range.py

> cp bayarea_lab/* ./examples
```

→ Move the shell script, `my_lab_run.sh`, into the main app directory, and set the file to be executable

```
> mv examples/my_lab_run.sh .
> chmod 766 my_lab_run.sh
```

2 Model and method

This section contains a brief description of the data, model, parameters, and method used in BayArea.

First, we define the range for taxon i as the bit vector X_i, where $X_{i,j} = 1$ if the taxon is present in area j and $X_{i,j} = 0$ if the taxon is absent. Each taxon range is a bit vector of length N areas. For example, if taxon B is present only in areas 2 and 3 out of $N = 3$ areas, its range is represented as $X_B = (0, 1, 1)$, which is translated to the bit string $X_B = 011$ for short. The data matrix, $\mathbf{X}$, is analogous to a multiple sequence alignment where each element in the data matrix reports a discrete value for a homologous character shared by all taxa at column j.

Next, we need a model of range evolution. Since we have discrete characters we'll use the continuous-time Markov chain, which allows us to compute transition probability of a character changing from i to j in time t through matrix exponentiation

$$\mathbf{P}_{i,j}(t) = \left[\exp\left\{\mathbf{Q}t\right\}\right]_{i,j},$$

where $\mathbf{Q}$ is the instantaneous rate matrix defining the rates of change between all pairs of characters, and $\mathbf{P}$ is the transition probability rate matrix. This technique of matrix exponentiation is powerful because it integrates over all possible scenarios of character transitions that could occur during t so long as the chain begins in state i and ends in state j.

We can then encode range evolution events into the allowed character transitions of $\mathbf{Q}$ and parameterize the events so that we may infer their relative importance to generating our observed ranges. We'll take a simple model of range expansion (e.g. $011 \rightarrow 111$) and range contraction (e.g. $011 \rightarrow 001$). (Range expansion may also be referred to as dispersal or area gain and range contraction as extirpation or area loss.) The rates in the transition matrix for three areas might appear as

$$\mathbf{Q} = \begin{array}{c|ccccccc} & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ \hline 001 & - & 0 & \lambda & 0 & \lambda & 0 & 0 \\ 010 & 0 & - & \lambda & 0 & 0 & \lambda & 0 \\ 011 & \lambda & \lambda & - & 0 & 0 & 0 & \lambda \\ 100 & 0 & 0 & 0 & - & \lambda & \lambda & 0 \\ 101 & \lambda & 0 & 0 & \lambda & - & 0 & \lambda \\ 110 & 0 & \lambda & 0 & \lambda & 0 & - & \lambda \\ 111 & 0 & 0 & 0 & \lambda & \lambda & \lambda & - \end{array},$$

where λ is the rate of area gain or loss. This can also be represented compactly as the rate function

$$q^{(a)}_{\mathbf{y},\mathbf{z}} = \begin{cases} \lambda & \mathbf{y} \text{ and } \mathbf{z} \text{ differ in exactly one area} \\ 0 & \mathbf{y} \text{ and } \mathbf{z} \text{ differ in more than one area} \end{cases},$$

where $\mathbf{y}$ and $\mathbf{z}$ are the "from" and "to" ranges and a is the area that changes. For example, $q^{(1)}_{011,111}$ is the rate of range expansion for $011 \rightarrow 111$ to gain area 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.

There are several simple ways to extend this model. For example, the rate of area gain and area loss (λ_0 and λ_1, resp.) may differ as

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

In addition, you may want to assign transition probabilities to entering the all-zero null range, but treat it as an absorbing state with that has a zero probability of being exited. Lastly, 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.

Let's consider what happens to the size of $\mathbf{Q}$ when the number of areas, N, becomes large. For three areas, $\mathbf{Q}$ is size 8×8. For ten areas, $\mathbf{Q}$ is size 1024×1024, which approaches the largest size matrix that can be exponentiated in a practical amount of time. This is problematic, meaning we need an alternative method to integrate over historical range evolution events.

You may wonder why matrix exponentiation works fine for molecular substitution models and large multiple sequence alignments. Because recombination degrades linkage disequilibrium over geological timescales, molecular substitution models typically assume each site in the multiple sequence alignment evolves independently. Conveniently, this keeps $\mathbf{Q}$ small even for datasets with many sites.

Remember matrix exponentiation integrates over all *unobserved* transition events during time t. The likelihood of beginning in character i and ending in character 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 is the strategy used in BayArea to infer the posterior distribution approximated is $\text{Prob}(\mathbf{X}_{aug}, \theta \mid \mathbf{X}_{obs}, T, M)$, 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. Ancestral range reconstructions are often of primary interest in phylogenetic biogeographic analyses, which are generated with support values as a by-product of the MCMC analysis.

The rest of the tutorial will describe how to assemble the input, run the analysis, assess the output, and visualize the results.

3 Input

For this tutorial, we'll use a dataset for 19 species of *Psychotria* whose range spans the Hawaiian archipelago. The dataset was originally reported in Nepokroeff et al. (2003) and analyzed using the likelihood-based historical biogeography method, LAGRANGE, by Ree and Smith (2008). We'll use this dataset for three reasons. First, it is relatively small, meaning we can produce results quickly. Second, the Hawaiian archipelago can be broken into naturally discrete areas and has a well-characterized geographical history that is uncomplicated to model. Third, it has previously been analyzed, which provides some basis for comparison to other methods.

The model will assume presence-absence characters are recorded without error.

3.1 Nexus file

The data file contains a matrix of binary characters corresponding to the observed ranges of the study taxa.

→ Open the file `examples/psychotria_range.txt`.

```
#NEXUS

begin data;
  dimensions ntax=19 nchar=4;
  format datatype=standard symbols = "01";
  matrix
    P_fauriei2          0100
    P_grandiflora_Kal2  1000
    ...
    P_wawraeDL7428      1000
  ;
end;

Begin trees;
        TREE tree1 = (((((((P_hawaiiensis_WaikamoiL1:1.0853
        ,P_mauiensis_Eke:1.0853)N2:0.7964,(P_fauriei2:1.3826,
        P_hathewayi_1:1.3826)N5:0.4991)N6:0.1986,(
        ...
        2.6568)N33:0.5204,P_hexandra_Oahu:3.1772)N35:2.6659)N36;
End;
```

Range data is stored in standard Nexus format. In the `data` block, the first line gives the dimensions of the data matrix and the second line indicates we will be using binary characters. The four characters correspond to areas defined by the geography file (next subsection). Rows in the `matrix` block correspond to taxa and their range data, while columns give in which areas each taxon is present (1) or absent (0). For example, taxon `P_fauriei2` is present only in area 2.

The `trees` block gives the tree describing the shared ancestry of the study species. Because range evolution occurs in units of geological time, the analysis in this tutorial requires a high-quality time-calibrated phylogeny. This typically requires a multiple sequence alignment over several loci plus fossils for calibration. Since this data availability is often the limiting factor for which taxa to include for your analysis, it is best to produce the phylogeny first. Only afterwards should you begin assembling data for your data matrix. If your phylogeny cannot be calibrated (e.g. it has no fossils) your best alternative is to proceed with a time tree resulting from a divergence time estimation analysis. For this tutorial, the phylogeny is assumed to contain no uncertainty.

3.2 Atlas file

The geography used in this tutorial represents the Hawaiian archipelago. Beneath Hawaii, currently the largest and youngest island, is a volcanic hotspot that periodically creates new islands. The ages of these islands are fairly well known, meaning we can model range availability as a function of time. Following Ree and Smith (2008), we will lump groups of smaller islands into single areas to simplify the analysis, leaving us with four areas: Hawaii (H), Oahu (O), Maui (M; this includes Molokai and Lanai), and Kauai (K; this includes Niihau). These areas are modeled have arisen 0.5, 1.9, 3.7, and 5.5 million years ago, respectively.

Although the model will use discrete-state biogeographic ranges, geographical area is naturally continuous. This means we must impose some discretization upon the geography to designate a set of biogeographically meaningful characters called areas. Different methods use different criteria for this discretization, so it is best to perform the discretization yourself rather than blindly using the discretization given from a previous study or method (but do blindly use the dataset included in this tutorial). Some geographies have natural discretizations: for instance, the Hawaiian archipelago forms naturally discrete areas on the basis of islands. For many geographies, however, it may unclear how to perform this discretization. Much like morphological analyses, you might choose to choose areas based on expert opinion, based on some model, or using some "naive" uniform discretization. This procedure is not part of the tutorial, but you should be aware that area definitions are not always obvious or objective.

→ Open the file `examples/hawaii_dynamic.atlas.txt`.

```
{
  "name":"HawaiianArchipelago10my",
  "epochs":
  [{
    "name":"epoch1",
    "start_age":5.5,
    "end_age":3.7,
    "areas":
      [{ "name": "Kauai",  "latitude":22.08, "longitude": −159.50,
         "dispersalValues": [ 1,0,0,0 ] },
       { "name": "Oahu",   "latitude":21.47, "longitude": −157.98, "
          dispersalValues": [ 0,0,0,0 ] },
       { "name": "Maui",   "latitude":20.80, "longitude": −156.33, "
          dispersalValues": [ 0,0,0,0 ] },
       { "name": "Hawaii", "latitude":19.57, "longitude": −155.50, "
          dispersalValues": [ 0,0,0,0 ] }]
  },
  ... epoch2 and epoch3 omitted ...
  {
    "name":"epoch4",
    "start_age":0.5,
    "end_age":0.0,
    "areas":
      [{ "name": "Kauai",  "latitude":22.08, "longitude": −159.50,
         "dispersalValues": [ 1,1,1,1 ] },
       { "name": "Oahu",   "latitude":21.47, "longitude": −157.98, "
          dispersalValues": [ 1,1,1,1 ] },
       { "name": "Maui",   "latitude":20.80, "longitude": −156.33, "
          dispersalValues": [ 1,1,1,1 ] },
       { "name": "Hawaii", "latitude":19.57, "longitude": −155.50, "
          dispersalValues": [ 1,1,1,1 ] }]  }
  ]
}
```

This is called the Atlas file, which uses a file format called JSON. JSON is a lightweight format used to assign values to variables in a hierarchical manner. There are three main tiers to the hierarchy in the Atlas file: the atlas, the epoch, and the area. In the lowest tier, each area corresponds to a character in the model and is assigned it's own properties. In the middle tier, each epoch contains the set of homologous areas (characters) that may be part of a species' range, but importantly the properties of these areas may take on different values during different intervals of time, as given by the `start_age` and `end_age` variables. Because the tree and range evolution model also operate on units of geological time, the rates of area gain and loss can condition on

areas' properties as a function of time. Sometimes these models are called stratified models or epochal models. Finally, the atlas contains the array of epochs in the highest tier.

Each area is assigned a `latitude` and `longitude` to represent its geographical coordinates, ideally the area's centroid. If a centroid does not represent the distance between areas, splitting the area into multiple smaller areas is reasonable. The data augmentation approach used in this analysis allows you to use more areas as desired, whereas matrix exponentiation methods are limited to approximately ten areas. The distance between any two areas' coordinates inform to distance-dependent dispersal parameter (β from the $\eta(\cdot)$ function) for range expansion events, so coordinates roughly close to the center of the area suffice.

In addition, each area is marked as habitable or not using the `dispersalValues` array. The elements in the array correspond to the other areas defined in the analysis. For example, in `epoch1`, Kauai's `dispersalValues` is equal to `[ 1,0,0,0 ]`, which indicates Kauai exists at that point in time but it is not in contact with any other areas, i.e. the range in that area cannot expand into other areas. The `dispersalValues` for Oahu, Maui, and Hawaii are all equal to `[ 0,0,0,0 ]`, meaning no species may be present in that area during the time interval of e poch1 during ages from 10.0 to 3.7. In contrast, `epoch4`, from ages 0.5 to the present, range expansions may occur between any pair of areas and any area may be included in a species' range.

4 RevBayes Analysis

There are five major parts to the analysis.

First, we need to read in the input files and assign analysis settings. Second, we need to construct our model. Third, we need to assign moves and monitors to our model parameters for use with Markov chain Monte Carlo (MCMC). Fourth, we will run an MCMC analysis assuming a complex model. Fifth, we will compare the complex model with a simple model using Bayes factors. Finally, we'll analyze our MCMC output.

4.1 Analysis settings

Open the RevBayes console

```
$ rb-extended
```

First, we'll assign all our input files to `String` variables.

```
RevBayes > in_fp   <- "/filepath/to/your/input/files/"
RevBayes > data_fn <- "psychotria_range.nex"
RevBayes > area_fn <- "hawaii_dynamic.atlas.txt"
```

Then we'll create our range data, tree, and atlas objects

```
RevBayes > data  <- readCharacterData(in_fp + data_fn)
RevBayes > tree  <- readTrees(in_fp + data_fn)[1]
RevBayes > atlas <- readAtlas(in_fp + area_fn)
```

Verify the data and tree share the same number of taxa and the data and atlas share the same number of characters

```
RevBayes > data.ntaxa() == tree.ntips()
   true
RevBayes > data.nchar() == atlas.nareas()
   true
```

4.2 Creating the model

Here, we will compose our rate matrix, $\mathbf{Q}$, parameterized by the transition rates, λ, and the distance dependent dispersal power parameter, β.

First, for λ, we will create a vector of two rates, where `glr[1]` corresponds to the rate of area loss (local extinction) and `glr[2]` corresponds to the rate of area gain (dispersal). Importantly, we must assign prior distributions to these parameters. Here, we'll use an exponential distribution with rate 10.0, which has a mean of 0.1. Because our tree is in units of millions of years, this means our prior expectation is that any given species undergoes one dispersal event and one extinction event per area per ten million years. To introduce this to the model, type

```
RevBayes > for (i in 1:2) glr[i] ~ dnExponential(10.0)
```

Next, we will create `dp`, 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 rate 10.0 (mean 0.1) as a prior for `dp`.

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.

```
RevBayes > dp ~ dnExponential(10.0)
RevBayes > grm := fnBiogeoGRM(atlas=atlas, distancePower=dp,
   useAvailable=true, 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 `fnBiogeoDEC` to assign our model parameters to transition rates as described in the introduction. As input, we'll pass our gain and loss rates, `glr`, and our geographical rate modifier, `grm`. 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).

```
RevBayes > Q_like := fnBiogeoDEC(gainLossRates=glr, rootFrequencies=pi,
    geoRateMod=grm, numAreas=4, forbidExtinction=true)
```

For the model's final node, we create the stochastic node for the continuous-time Markov chain (CTMC). This node's distribution is `dnPhyloDACTMC` where `DA` indicates the CTMC uses data-augmentation to compute the likelihood rather than Felsenstein's pruning algorithm. To create the distribution, we must pass it our `tree` and `Q_like` 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 Ree and Smith (2008) (`useCladogenesis=true`), and that it will assign zero probability to a transition away from the null range state.

```
RevBayes > M ~ dnPhyloDACTMC(tree=tree, Q=Q_like, 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 will prime the model with data-augmented character histories. Now `M` has a defined likelihood value.

```
RevBayes > M.clamp(data)
RevBayes > M.lnProb
   −76.0193
```

Finally, we encapsulate our graphical model into a `Model` object, which can learn the model's structure and dependencies from any model parameter.

```
RevBayes > my_model <− model(glr)
```

4.3 Running an MCMC analysis

Now that we have our `Model` object, we can soon run an MCMC analysis. Remember that MCMC approximates the posterior distribution by repeatedly proposing new model parameter values, accepting or rejecting those new parameter values based on the model likelihood (and on biases in the proposal distribution), then reporting the sampled parameter values.

First, let's assign moves to our model parameters. These parameters are all supported for real positive values, which is appropriate for use with scale-multipler proposal, `mvScale()`. To inspect our model parameter types and the proposal argument types, enter

```
RevBayes > type(dp)
   RealPos
RevBayes > type(glr)
   RealPos[]
RevBayes > mvScale
   Move_Scale function (RealPos x, RealPos lambda, Bool tune, RealPos
      weight)
```

The arguments for `mvScale` are fairly typical as far as RevBayes `Move` objects go: `x` is the stochastic node the `Move` will update, `lambda` is proportional to how radically different proposed parameter values will tend to be, `tune` allows `lambda` to be adjusted automatically as the MCMC runs, and `weight` tells the MCMC how many times to perform the `Move` during a single MCMC generation (e.g., `weight=2.0)` means each generation will call that `Move` for the parameter `x` twice).

```
RevBayes > moves[1] <- mvScale(x=glr[1], lambda=0.5, tune=false, weight
   =2.0)
RevBayes > moves[2] <- mvScale(x=glr[2], lambda=0.5, tune=false, weight
   =2.0)
RevBayes > moves[3] <- mvScale(x=dp,      lambda=0.5, tune=false,
   weight=2.0)
```

In addition to proposing new model parameter values, we must also propose new data-augmented states and events to properly integrate over the space of possible range histories. The major challenge to sampling character histories is ensuring the character histories are consistent with the observations at the tip 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 matrices.

The basic idea is simple. Each time a character history proposal is called, it selects a node at random from the tree. Path history proposals (`mvPathCHRS()`) propose a new character history for the lineage leading to that node. Node history proposals (`mvNodeCHRS()`) propose a new character history for the node and for the three lineages incident to that node. The character history proposal also samples some number of areas to update, ranging from one to all of the areas. 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.

Because these `Move` objects update the character histories stored in the data-augmented CTMC node, e.g. `M`, they require access to a `TimeTree` object to know which lineages are sisters and whether the lineages span various epochs, and a `RateMap_Biogeography` object to propose new character histories. The `lambda` argument gives what proportion of areas' character histories to update. Here, if `lambda=0.2`, then the proposal will redraw character histories for each area with probability 0.2 (in addition to one random area with probability 1). Below, we use two moves of each type with `lambda=0.2` and `lambda=1.0` for partial and full character history updates, respectively. Indicating `type="biogeo"` informs the `Move` object to be aware of special character history constraints, such as cladogenic events and forbidden null ranges. The `weight` parameter should be assigned a value proportional to the number of nodes in the analysis to ensure proper mixing.

Let's create the character history moves as follows.

```
RevBayes > n_nodes  <- tree.nnodes()
RevBayes > moves[4] <- mvNodeCHRS(ctmc=M, qmap=Q_like, tree=tree,
   lambda=0.2, type="biogeo", weight=2.0*n_nodes)
RevBayes > moves[5] <- mvPathCHRS(ctmc=M, qmap=Q_like, tree=tree,
   lambda=0.2, type="biogeo", weight=2.0*n_nodes)
RevBayes > moves[6] <- mvNodeCHRS(ctmc=M, qmap=Q_like, tree=tree,
   lambda=1.0, type="biogeo", weight=n_nodes)
RevBayes > moves[7] <- mvPathCHRS(ctmc=M, qmap=Q_like, tree=tree,
   lambda=1.0, type="biogeo", weight=n_nodes)
```

Now that we have moves for all our parameters and the character histories, we'll proceed with assigning `Monitor` objects to record their values. The first two `Monitor` objects are fairly standard and found in most RevBayes MCMC analyses. `mnScreen` reports the values for any nodes assigned to `RevObject ...` every `printgen` generations to the terminal screen. `mnModel` reports the values for all nodes in the `Model` object every `printgen` generations to the file assigned to `filename`, which is delimited by the `separator` character.

```
RevBayes > mnScreen
   Mntr_Screen function (RevObject ..., Natural printgen, Bool
      posterior, Bool
   likelihood, Bool prior)
RevBayes > monitors[1] <- mnScreen(printgen=10, glr, dp, pi)
RevBayes > mnModel
   Mntr_Model function (String filename, Natural printgen, String
      separator,
   Bool posterior, Bool likelihood, Bool prior, Bool append, Bool
   stochasticOnly)
RevBayes > monitors[2] <- mnModel(filename=out_fp+params_fn, 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 `Mntr_CharacterHistoryNewickFile` objects 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 **??**.

```
RevBayes > mnCharHistoryNewick
   Mntr_CharacterHistoryNewickFile function (String filename,
```

```
   AbstractCharacterData ctmc, TimeTree tree, Natural printgen, String
   separator, Bool posterior, Bool likelihood, Bool prior, Bool append,
       String
   style = events|counts
   , String type = biogeo
   )
RevBayes > monitors[3] <- mnCharHistoryNewick(filename="psychotria.
   counts.txt", ctmc=M, tree=tree, printgen=100, style="events")
RevBayes > monitors[4] <- mnCharHistoryNewick(filename="psychotria.
   counts.txt", ctmc=M, tree=tree, printgen=100, style="counts")
```

As our last monitor, the `Mntr_CharacterHistoryNhxFile` 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 in Section **??**.

```
RevBayes > mnCharHistoryNhx
   Mntr_CharacterHistoryNhxFile function (String filename,
      AbstractCharacterData
   ctmc, TimeTree tree, RlAtlas atlas, Natural samplegen, Natural
      maxgen,
   Probability burnin, String separator, Bool posterior, Bool
      likelihood, Bool
   prior, String type = biogeo
   )
RevBayes > monitors[5] <- mnCharHistoryNhx(filename="psychotria.nhx.txt
   ", ctmc=M, tree=tree, atlas=atlas, samplegen=100, maxgen=nGens,
   burnin=0.25)
```

4.4 Running an MCMC analysis

Now all that's left is to configure and run our MCMC analysis. For this, we create an `Mcmc` object, which we give our `Move` vector, our `Monitor` vector, and our `Model object`

```
RevBayes > mcmc
   MCMC function (Model model, Monitor[] monitors, Move[] moves, String
   moveschedule = sequential|random|single
   )
RevBayes > my_mcmc <- mcmc(my_model, monitors, moves)
```

MCMC typically requires some period of burn-in before it reaches stationarity, i.e. from a random starting point, it takes some time for the chain to produce valid samples from the posterior distribution. By running `burnin()`, we tell the `Mcmc` object to propose and reject new states but *not* to

record anything to file. After burn-in is complete, we call `run()`, where we begin recording valid posterior samples under our model.

```
RevBayes > my_mcmc.burnin(generations=1000, tuningInterval=100)
RevBayes > my_mcmc.run(generations=10000)
```

Everything we've done is contained in the file `biogeography_M1.Rev`. You can modify this file as you like then re-run the analysis by typing

```
RevBayes > source("biogeography_M1.Rev")
```

4.5 Model selection using Bayes factors

Bayes factors (BFs) are used to select which of two models better describes the observed data, $\mathbf{X}_{obs}$, and are computed as the ratio of marginal likelihoods for those two models. One might prefer to analytically compute the marginal likelihood as

$$\text{Prob}(\mathbf{X}_{obs} \mid M) = \int_{\mathbf{X}_{aug}} \int_{\theta} \text{Prob}(\mathbf{X}_{obs}, \mathbf{X}_{aug}, \theta \mid M)\, d\theta\, d\mathbf{X}_{aug}$$

but the marginal likelihood is the same intractable quantity we intentionally avoid computing when using MCMC in a Bayesian context. Instead, we must estimate the marginal likelihood from our posterior distribution samples. Here, we will use thermodynamic integration (**?**) and stepping-stone approximation (**?**). The exact details of these techniques will not be covered here, but there is an important practical point to mention: both methods rely on computing a number of "power posterior" distributions. Computing more power posteriors increases the marginal likelihood estimator's accuracy at the cost of computational time. (For more discussion on model selection and marginal likelihood estimation in RevBayes, see XXXX).

Moving on, we'll compute the Bayes factor to compare a simple one-rate model, which asserts the rate of area gain and loss are always equal, to a two-rate model which allows these rates to vary independently. Rather than specifying the model manually, we will load (source) the model definition from a file then enter the commands to compute its marginal likelihood. For faster results, we will use two separate RevBayes sessions, one for each model. For each session, the power posterior analysis run for 1000 generations during burn-in then 1000 generations per each of 30 power posterior categories.

```
RevBayes > source("biogeography_1rate.Rev")
RevBayes > pow_p <- powerPosterior(my_model, moves, out_fp+out_pp_fn,
   cats=30)
RevBayes > pow_p.burnin(generations=1000,tuningInterval=100)
RevBayes > pow_p.run(generations=1000)
```

```
RevBayes > source("biogeography_2rate.Rev")
RevBayes > pow_p <- powerPosterior(my_model, moves, out_fp+out_pp_fn,
   cats=30)
RevBayes > pow_p.burnin(generations=1000,tuningInterval=100)
RevBayes > pow_p.run(generations=1000)
```

Each power posterior analysis will write their contents to the file given in `out_pp_fn`. These files are `pp_1rate_out.txt` and `pp_2rate_out.txt` for the simple and complex models, respectively. This may take a few minutes. When complete, the power posterior files may then be used to compute marginal likelihoods. For example, from the RevBayes session analyzing the simple one-rate model

```
RevBayes > ss <- steppingStoneSampler(file=out_fp+out_pp_fn)
RevBayes > ss.marginal()
RevBayes > ps <- pathSampler(file=out_fp+ouf_pp_fn)
RevBayes > ps.marginal()
```

For a given model, the path sampling and stepping stone sampling methods should produce similar marginal likelihood estimates. Values should be within one log likelihood unit of one another. If the values are extremely different, this may indicate `powerPosterior` should be re-run with a larger number of `cats`. We chose `cats=30` which should suffice, and we see no problem. Then from the complex two-rate model RevBayes session

```
RevBayes > ss <- steppingStoneSampler(file=out_fp+out_pp_fn)
RevBayes > ss.marginal()
RevBayes > ps <- pathSampler(file=out_fp+out_pp_fn)
RevBayes > ps.marginal()
```

Finally, we can compute the Bayes factor, which is simply the ratio of marginal likelihoods.

```
RevBayes > exp(−45.5) / exp(−46.7)
```

A value of one would mean both models had equal marginal likelihoods. A value less than one would indicate the first model, the simple model, had a larger marginal likelihood, and was therefore favored by model testing. But that's not the case, the value is greater than one, and the complex two-rate model is favored. Similar to frequentist interpretations of significance for p-values, there is no universal and objective criterion of significance with Bayes Factors, but most would agree a factor of 10 (or 0.1) indicates strong support for one model over the other.

5 Output

5.1 Screen

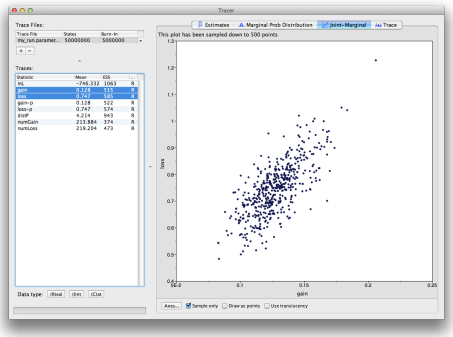
There are three types of MCMC proposals, the first being parameter proposals that act upon the model parameters, λ_0, λ_1, and β. The second type of proposals act upon our sampling parameters, λ_0^* and λ_1^*, which we use to sample range histories but do not directly affect our model likelihood. The last type, ancestral range history updates, occur in very high proportion to all other updates.

Recall from the Introduction that this is necessary to effectively integrate over all ancestral ranges of moderate to high posterior probability. The screen output reports whether the proposal for that iteration was accepted or rejected. If you notice the MCMC is rejecting most proposals, it will probably not produce an accurate approximation of the posterior. Pay particular attention to range history proposals. Range history proposals tend to introduce very small differences to the likelihood so they are generally accepted easily. If you notice very few range history proposals are accepted, you will improve the acceptance probability by reducing the `areaProposalTuner` value.

5.2 Sampled parameters (`my_run.parameters.txt`)

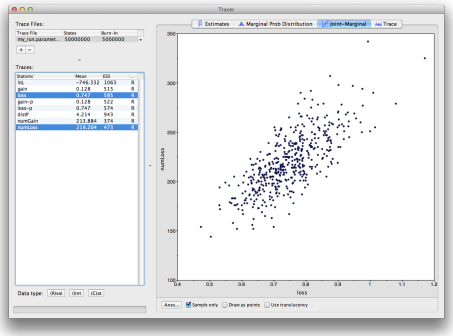
This tab-delimited file contains the parameters sampled at regular intervals from the MCMC (set by `parameterSampleFrequency`). Open the `my_run.parameters.txt` in Tracer. First, we see the effective sample size is only moderate for 50×10^6 cycles when sampling every 1000 cycles. This is because the vast majority of MCMC updates are range history updates.

→ Highlight the "gain" and "loss" parameters then click Joint-Marginal.



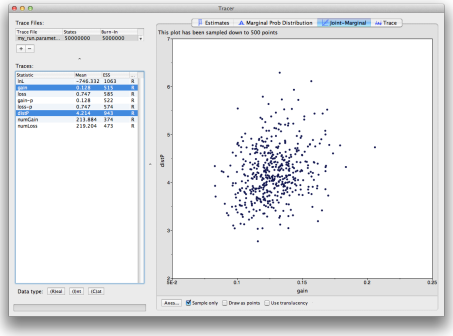
The Joint-Marginal of rate of area gain and loss parameters shows strong positive correlation. This correlation arises partly because the proportion of occupied areas is greatly influenced by the ratio of rate of area gain to rate of area loss. For example, when the rates of area gain and loss are equal, range sizes tend to equal half the total number of areas.

→ Highlight the "loss" and "numLoss" parameters then click Joint-Marginal. Do the same for "gain" and "numGain".



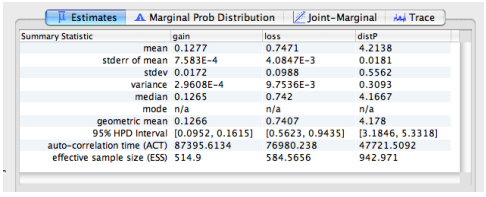
Again, we see strong positive correlation. This is reassuring, since we expect to see many sampled range gain and loss events when the corresponding rates are high.

→ Highlight the "gain" and "distP" parameters then click Joint-Marginal. Do the same for "loss" and "distP".



For the simulated data we don't see any strong correlation between either rate of loss or gain and the distance power parameter, but this is not always the case. Large distance power parameters sometimes results in rates of area gain and loss being underestimated in value, resulting in a mild negative correlation.

→ Highlight all parameters and click Estimates.



Estimates | Marginal Prob Distribution | Joint-Marginal | Trace

Summary Statistic	gain	loss	distP
mean	0.1277	0.7471	4.2138
stderr of mean	7.583E-4	4.0847E-3	0.0181
stdev	0.0172	0.0988	0.5562
variance	2.9608E-4	9.7536E-3	0.3093
median	0.1265	0.742	4.1667
mode	n/a	n/a	n/a
geometric mean	0.1266	0.7407	4.178
95% HPD Interval	[0.0952, 0.1615]	[0.5623, 0.9435]	[3.1846, 5.3318]
auto-correlation time (ACT)	87395.6134	76980.238	47721.5092
effective sample size (ESS)	514.9	584.5656	942.971

This gives you a handy selection of summary statistics. Of course we typically don't know the true parameter values, but we should be reassured to learn the true parameter values for gain, loss, and distP (0.1, 0.7, and 4.0, respectively) reside within their corresponding 95% HPD Intervals. It's

also interesting to note that the distance power parameter has a much wider interval that the rate parameters, suggesting there is less information in the data for this parameter.

5.3 Sampled range histories (`my_run.area_states.txt`)

This file reports the stochastically mapped range histories sampled every `historySampleFrequency` iterations. From this we can compute the *marginal* posterior probability of any given area being occupied simply by finding the frequency the area was marked present (1), which we'll call the *marginal area posterior probability*. You may be interested in not only the marginal posterior of area occupancy, but the marginal posterior of the joint distribution of areas, which we'll call the *marginal range posterior probability*. For example, say your range contains 4 areas, and taxon A is present only in areas 1 and 2 for half of all MCMC samples and only present in areas 3 and 4 for the remaining half. The marginal area posterior range would report a 0.5 probability of any given area being occupied. The marginal range posterior would give you richer information, that the range was either composed of areas 1 and 2 or of 3 and 4, but never 1 and 3, 1 and 4, 2 and 3, or 2 and 4.

Here we will use Python to report the marginal range probability for the range belonging to the most recent common ancestor (MRCA) shared by the species named `t1` and `t2` ordered by their posterior probabilities. We'll call this node MRCA(t1,t5).

→ Change directories to `examples`.

```
> cd examples
```

→ Open the Python console and enter the following commands.

```
>>> from pp_range import *
>>> range_dict = make_range_dict('my_run.area_states.txt', fburn
   =0.25)
>>> area_coords = get_area_coords('my_geo.txt')
>>> idx_list = get_node_idx_list('my_run.area_states.txt')
>>> node_key = get_node_key('my_tree.txt', 't1', 't5', idx_list)
>>> best_ranges = get_best_ranges(range_dict, node_key)
>>> prob_area_pairs = make_posterior_area_pairs(range_dict, node_key
   )
>>> best_area_pairs = get_best_area_pairs(prob_area_pairs)
>>> best_areas = get_best_area_pairs(prob_area_pairs, show_marginal=
   True)
```

→ To list the five most probable ranges for MRCA(t1,t5), type

```
>>> best_ranges[0:5]
[('00000000000000000000000000001111000', 0.022660623833644103),
 ('00000000000000000000000000001011000', 0.01759530791788836),
 ('00000000000000000000000000011111000', 0.01652892561983452),
 ('00000000000000000000000000011011000', 0.0146627565982403),
 ('00000000000000000000100000001111000', 0.01226339642761916)]

>>> sum([p[1] for p in best_ranges[0:5]])
0.08371101039722645
```

The second column gives the probability of the entire MRCA(t1,t5) range shown in the first column. There's a fair deal of uncertainty the MRCA(t1,t5) range reconstruction, since these best five ranges only account for about 0.08 of the probability.

→ To list the five most probable area co-occurrences in the MRCA(t1,t5) range, type

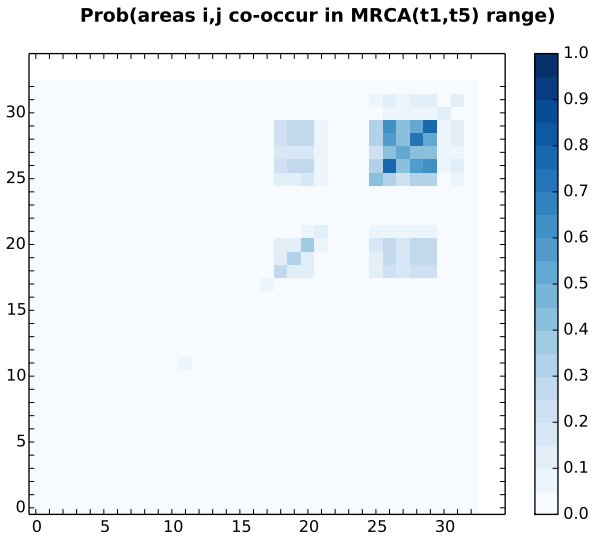
```
>>> best_area_pairs[0:5]
[((26, 29), 0.60650493201812505),
 ((26, 28), 0.57717941882164425),
 ((28, 29), 0.54385497200746136),
 ((26, 27), 0.42282058117834981),
 ((27, 28), 0.41242335377232475)]

>>> best_areas[0:5]
[((26, 26), 0.78459077579311998),
 ((29, 29), 0.77179418821647394),
 ((28, 28), 0.74193548387096575),
 ((26, 29), 0.60650493201812505),
 ((26, 28), 0.57717941882164425)]
```

Note the values in `best_areas` are the marginal area probabilities. These correspond to the diagonal entries in the heatmap below and to the values in `my_run.area_probs.txt` and `my_run.nhx`.

→ To visualize these probabilities as a heatmap, type

```
>>> title_str = 'Prob(areas i,j co—occur in MRCA(t1,t5) range)'
>>> plot_posterior_area_pair(prob_area_pairs, title_str)
```



As indicated from the list produced by `get_best_area_pairs()`, we see the inference gives moderate support for areas 26, 28, and 29 being co-occupied by MRCA(t1,t5). Referring to `my_geo.txt`, we see these areas are geographically close and located in south-eastern Australia. You can also retrieve the coordinates through the `area_coords` dictionary.

→ To query the geographical coordinates for an area, type

```
>>> area_coords[26]
{'lat': -32.5, 'lon': 137.5}

>>> area_coords[28]
{'lat': -32.5, 'lon': 147.5}

>>> area_coords[29]
{'lat': -32.5, 'lon': 152.5}
```

5.4 Posterior probabilities of range histories (`my_run.area_probs.txt`)

Each row of this tab-delimited file contains information about that node's range. The rows are ordered in postorder traversal (aka pruningwise). The first column gives the taxon name. The remaining columns correspond to areas, maintaining the same order as all other files (e.g. the second column is the first area, the nth column is the $(n-1)$th area). These entries report the marginal area probability of the taxon (row) occupying the area (column). These probabilities are computed the frequency that the area was marked present from the data contained in `my_run.area_probs.txt`, a quantity that cannot be computed until the MCMC analysis completes. Because MCMC does not accurately sample from posterior during burn-in, some number of initial cycles are discarded as specified by the `probBurnIn` flag.

5.5 New Hampshire extended format file (`my_run.nhx`)

This file summarizes the input and output from a BayArea analysis using NEXUS format containing a New Hampshire eXtended (NHX) tree string. NHX allows you to annotate nodes in a Newick string with meta-information, which BayArea uses to report the probabilities in the `my_run.area_probs.txt` file. The `geo` block gives the geographical latitudes and longitudes for the areas in the order they are reported as probabilities. Like the `my_run.area_probs.txt` file, this file is not written until the analysis is complete. This annotation is used for the two visualization programs covered in the next section, Phylowood and BayArea-Fig. The anatomy of the Phylowood and BayArea-Fig settings blocks will also be explained there.

6 Visualization

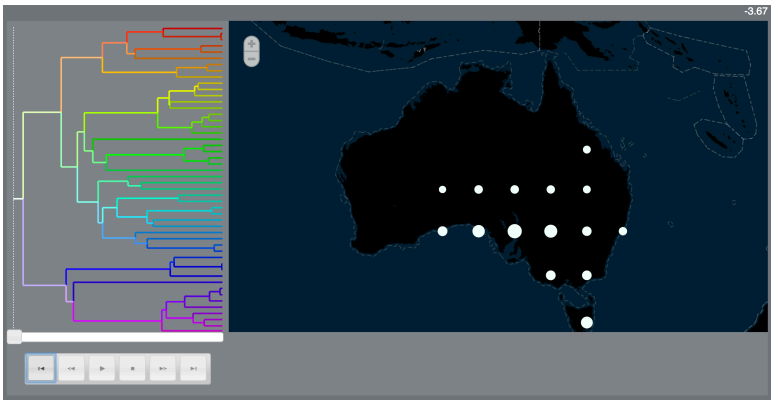
Here we'll explore two options for visualizing ancestral range reconstructions. I'll walk you through some of the basic functionality, but feel free to play around as you like.

6.1 Phylowood

Phylowood generates interactive animations to explore biogeographic reconstructions.

→ Open `http://mlandis.github.io/phylowood`.

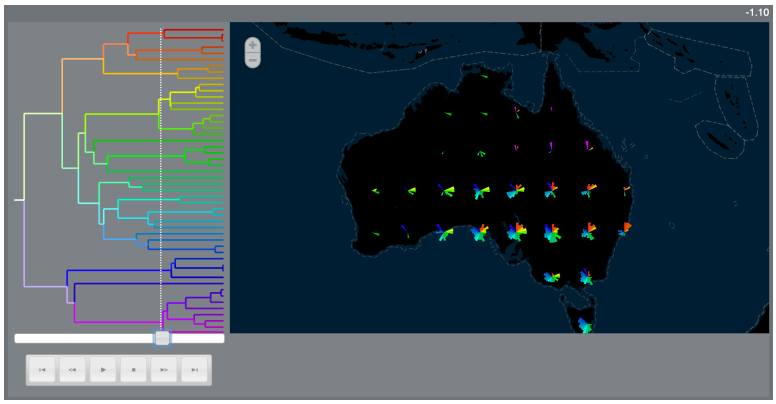
→ Drag and drop `my_run.nhx` into the text field.



→ Click the Play button to view the animation.

There are three control panels to help you filter data: the media panel, the map panel, and the phylogeny panel. The media buttons correspond to Beginning, Slow/Rewind, Play, Stop, Fast Forward, Ending (from left to right). The animation will play the timeframe corresponding to the slider.

→ Drag the slider to the time marked “-1.1” in the upper right corner.



→ Pan and zoom around the map.

Marker colors correspond to the phylogenetic lineages in the phylogeny panel. Markers are split into slices and (loosely) sorted phylogenetically, so nearby slices are generally closely related. At divergence events, a marker’s radius is proportional to the marginal posterior probability the node was present in the area at that time. Between divergence events, marker’s radius is simply an interpolation of the values at the two endpoints.

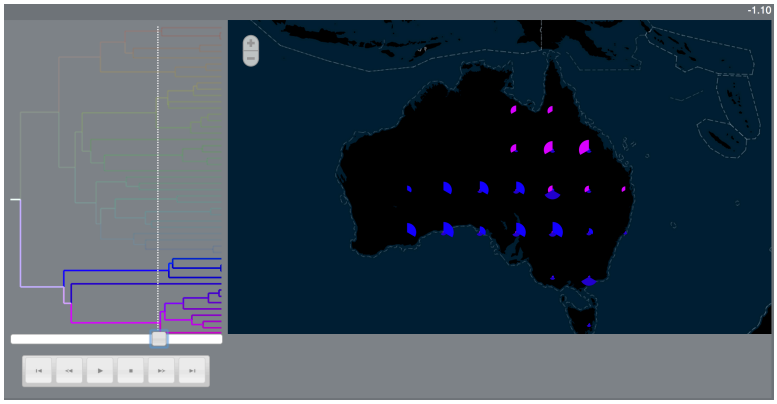
→ Mouseover an area to learn which lineage it belongs to and its presence probability.

Since it’s difficult to see how specific clades evolve with so many taxa, Phylowood offers two ways to filter taxa from the animation. We call the set of a lineage, all its ancestral lineages towards the root, and all descendant lineages a phylogenetic heritage. The root’s heritage is the entire clade. A leaf node’s heritage is a path from the tip to the root.

→ Mouseover a lineage to temporarily highlight the lineage’s heritage. Remove the mouseover to remove the highlight effect.

The highlight effect is temporary and quickly allows you to single out lineages of interest during animation. Phylowood also offers a masking effect that persists until an unmask command is issued.

→ Double-click the white root branch to mask the root node’s heritage (all lineages). Single click a lineage to unmask that lineage’s heritage.



Now that the masking effects are in place, you're free to interact with other map components. In addition, the area of marker sizes is only distributed among unmasked lineages.

→ Visit `https://github.com/mlandis/phylowood/wiki` to learn more about Phylowood.

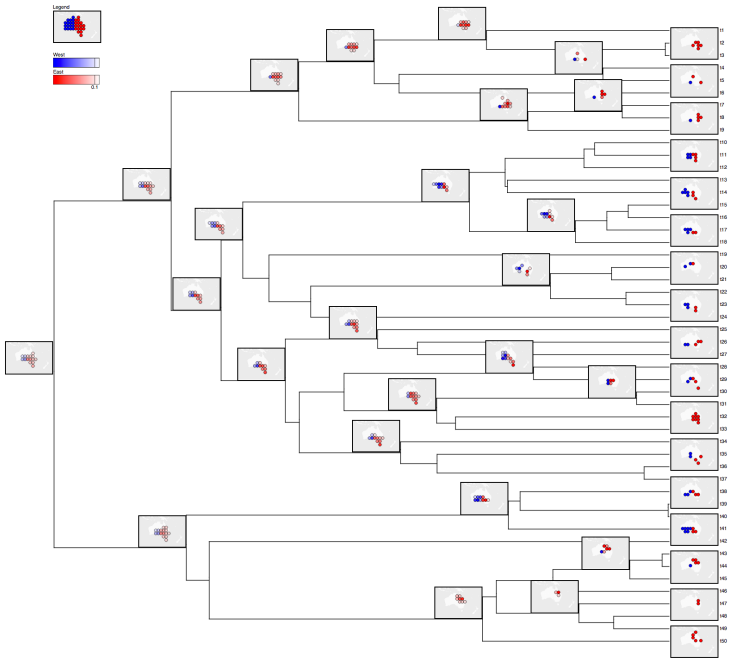
6.2 BayArea-Fig

BayArea-Fig is a simple Javascript utility to help generate ancestral range reconstruction figures for publications. BayArea 1.0.2 does not automatically generate the NEXUS settings block for BayArea-Fig, so I added the following block to `my_run.nhx`.

```
Begin bayarea-fig;
        mapheight 100
        mapwidth 150
        canvasheight 2000
        canvaswidth 2000
        minareaval 0.1
        areacolors blue red
        areatypes 0 1 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 0 0 0
           0 1 1 1 1 1 1 1
        areanames West East
End;
```

→ Open `http://mlandis.github.io/bayarea-fig` in a web browser.

→ Drag and drop `my_run.nhx` into the text field.



The generated figure reports the marginal area probabilities for each node in the phylogeny with a miniature map. Because of the limited real estate in the figure, you may only wish to show a subset of ancestral ranges.

→ Click unwanted ancestral ranges to delete them.

Depending on the purpose for the figure, you may wish to alter its size. The `mapheight`, `mapwidth`, `canvasheight`, and `canvaswidth` settings give the height and widths for the node-maps and the entire figure, respectively.

If you would like to differentiate areas (e.g. East from West, as above), give an ordered list of colors using the `areacolors` setting. Next, in the order areas appear in the `GEO` block, provide `areatypes` a list of assigments to area types. In the above example "0" corresponds to the 0th `areacolors` group, "blue", and "1" corresponds to the 1st group, "red".

→ Add a new color as the third `areacolors` entry. Add "Tasmania" as a third entry to `areanames`. Change the last area in `areatypes` to equal 2.

We've seen that ancestral range reconstructions contain a great deal of uncertainty. Filtering out low probability ranges may make your figure easier to interpret. If the marginal probability of presence for a node-area pair is less than `minareaval`, it is not shown. The probabilities and the threshold value are shown in the upper left corner.

After arranging your figure, you'll want to save it to file. How to accomplish this varies across operating systems and web browsers. Generally, the figure is most easily saved from the browser by printing the file to pdf. In this case the image size is equal to the paper size. If the standard 8.5×11 inch image is too small, you will need to create and apply a sufficiently large custom paper

size.

References

Landis, M. J. and T. Bedford. 2014. Phylowood: interactive web-based animations of biogeographic and phylogeographic histories. Bioinformatics 30:123–124.

Landis, M. J., N. J. Matzke, B. R. Moore, and J. P. Huelsenbeck. 2013. Bayesian analysis of biogeography when the number of areas is large. Systematic biology 62:789–804.

Nepokroeff, M., K. J. Sytsma, W. L. Wagner, and E. A. Zimmer. 2003. Reconstructing ancestral patterns of colonization and dispersal in the hawaiian understory tree genus psychotria (rubiaceae): a comparison of parsimony and likelihood approaches. Systematic Biology 52:820–838.

Ree, R. H. and S. A. Smith. 2008. Maximum likelihood inference of geographic range evolution by dispersal, local extinction, and cladogenesis. Systematic Biology 57:4–14.