Markov Chain Monte Carlo Computations for Dirichlet Diffusion Trees

Radford M. Neal

Dept. of Statistics and Dept. of Computer Science
University of Toronto

http://www.cs.utoronto.ca/∼radford/
radford@stat.utoronto.ca

NTOC 2001, December 9-13, Kyoto

Bayesian Modeling of Distributions

A Bayesian model for an unknown distribution needs a prior distribution over distributions.

Some possibilities:

- Assume some simple parametric form for the distribution, and give a prior to the parameters.
- Use a general prior expressing a belief in smoothness, but with no underlying latent structure. Eg, a Gaussian process for the log of the (unnormalized) density.
- Use a latent variable model, such as a mixture of simple distributions.

Mixture-based priors are appropriate when we think that latent classes do underlie the data, especially if this latent structure is of interest, or if we just think this produces a good prior.

The Need for Hierarchical Structure

Finite mixtures can model only a limited class of distributions. Dirichlet process mixtures are more flexible, having a countably infinite number of components, but do not capture the hierarchical structure of the world:

- Organisms are grouped into families, genera, species, sub-species, etc.

- Human artifacts are often manufactured according to make, model, and customized versions of models.

Also, when non-Gaussian classes are modeled with Gaussian mixture components, each class will be modeled using many components with similar locations — a two-level hierarchy.

My objective: Generalize Dirichlet process mixtures to capture this hierarchical structure, and hence provide a richer class of priors.

Defining Priors Over Distributions Via Exchangeable Priors on Data Sets

Defining a prior distribution over distributions looks hard — how can we visualize a prior for an infinite-dimensional quantity?

Fortunately, de Finetti's Representation Theorem provides a way:

> A distribution over data sets of arbitrary size that is *exchangeable* (invariant under change in order) implicitly defines a prior distribution over distributions.

Such an exchangeable distribution for data sets can be seen as the marginal distribution for a data set that is randomly sampled from a distribution drawn randomly from this prior.

The Idea of Dirichlet Diffusion Trees

A Dirichlet diffusion tree model defines a procedure for generating data sets one point at a time — in an exchangeable way.

To generate the first point:

> Simulate Brownian motion from $t = 0$ to $t = 1$. The distribution of the end-point will be Gaussian with some variance, σ^2.

The generate the next point:

> Simulate Brownian motion again, but follow the path to the first point initially, diverging at some random time.

To generate later points:

> Follow previous paths initially, choosing randomly when they branch, but diverge at some random time.

General principle: Paths followed many times previously are more likely to be followed again.

Details of Dirichlet Diffusion Trees

The Brownian diffusion begins at $t = 0$ at the origin (more generally, at a prior guess for location), and continues to $t = 1$, with the variance over this time for variable i being σ_i^2.

A *divergence function*, $a(t)$, controls the probability of divergence from previous paths.

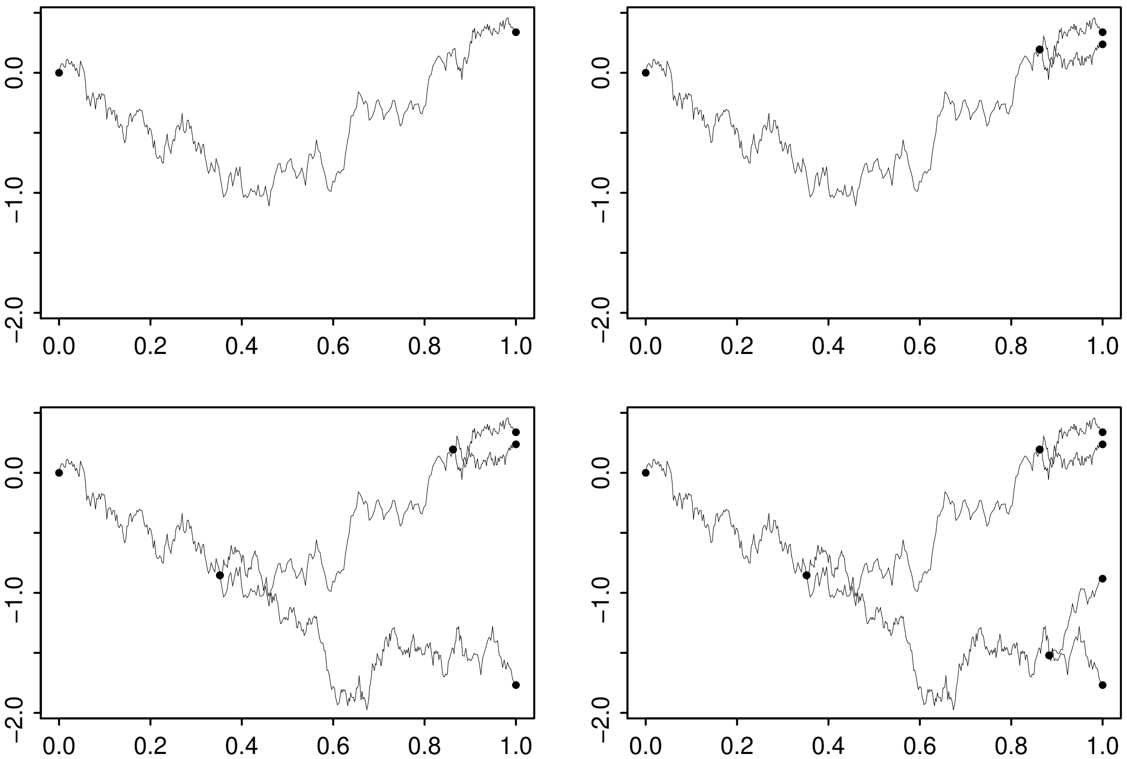
While moving over a path traversed m times previously, the probability of diverging between times t and $t + dt$ is $a(t)dt/m$.

When a branch is encountered while following previous paths, the probabilities of going each way are proportional to the numbers of paths going each way previously.

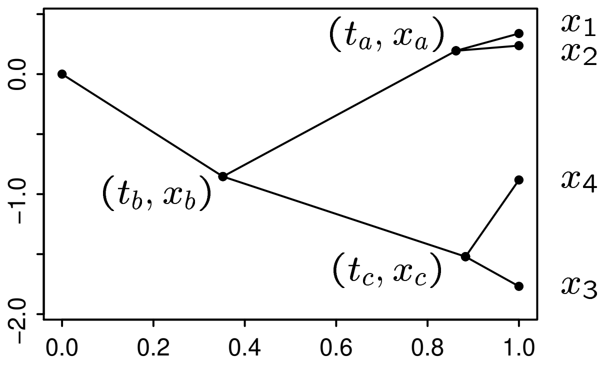
Once divergence occurs, subsequent motion is independent of previous paths.

Generating a Data Set of Four Points From a Dirichlet Diffusion Tree

The prior here uses $\sigma = 1$ and $a(t) = 1/(1-t)$.



A picture with the details suppressed:



Dirichlet Diffusion Tree Priors With $a(t) = c/(1-t)$

Consider a divergence function of the form $a(t) = c/(1-t)$, with $c > 0$. The corresponding *cumulative divergence function* is

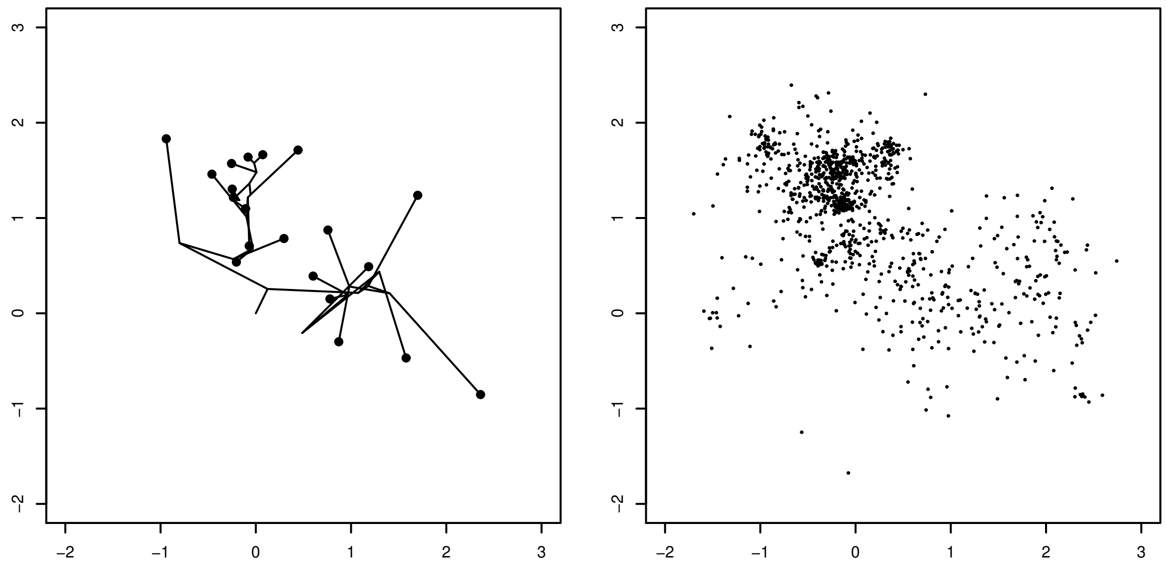
$$A(t) \;=\; \int_0^t a(u)du \;=\; -c\log(1-t)$$

The probability that the second path will not diverge from the first path before time t is $\exp(-A(t))$. As $t \to 1$, $a(t)$ and $A(t)$ go to infinity, and the probability of divergence goes to one.

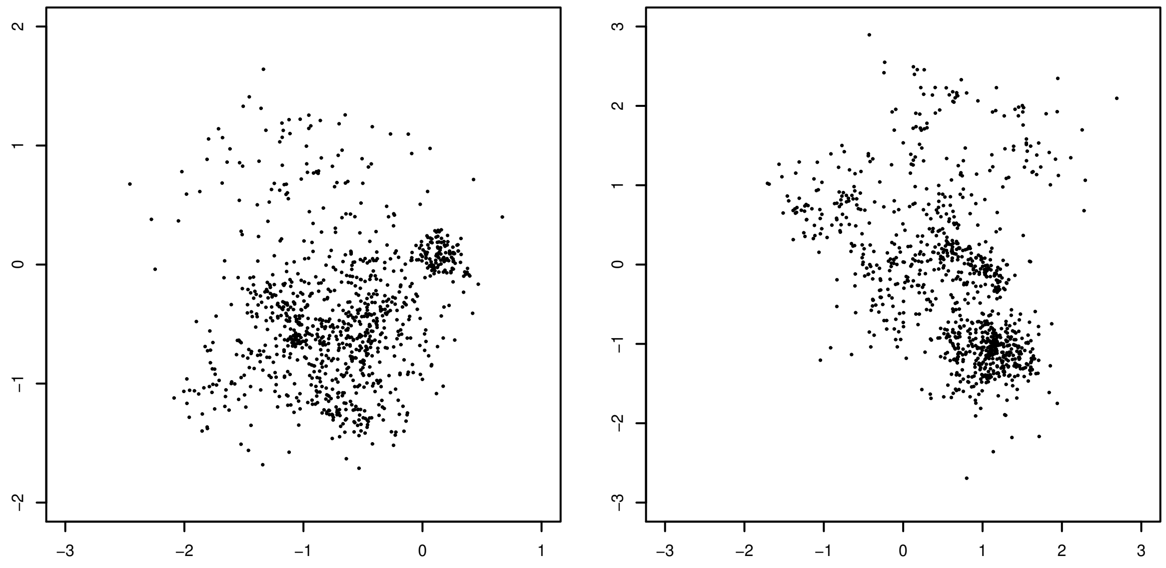
When c is small, the fairly slow divergence of $A(t)$ results in tight clusters of points, since the paths sometimes don't diverge until close to $t = 1$.

Larger values of c give more smooth-looking distributions.

A Two-Dimensional Example

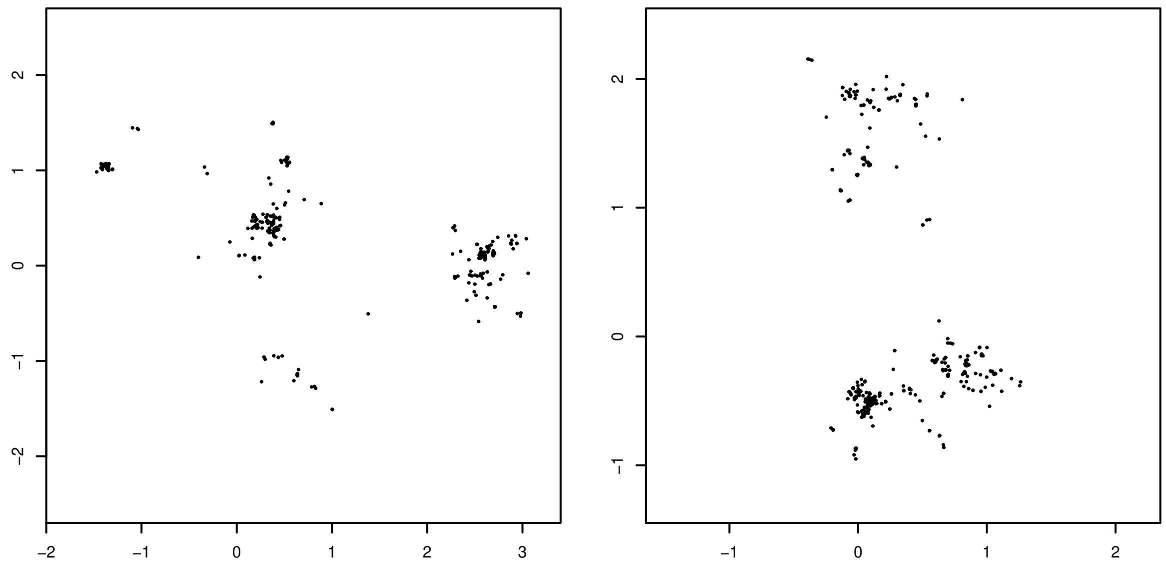


A data set generated with $c = 1$

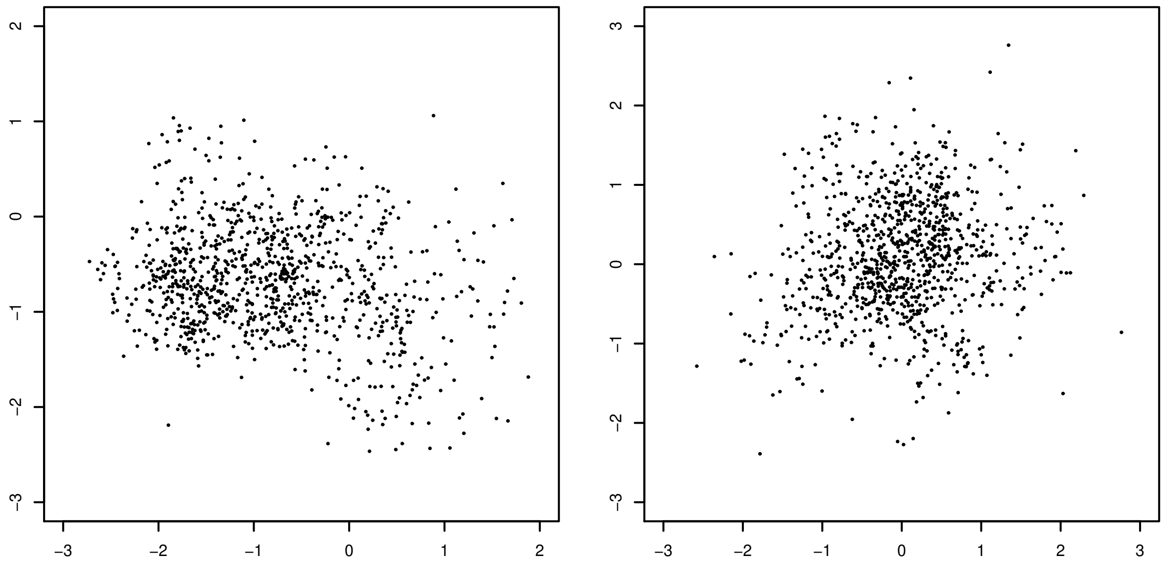


Two more data sets generated with $c = 1$

More Two-Dimensional Examples



Two data sets generated with $c = 1/4$

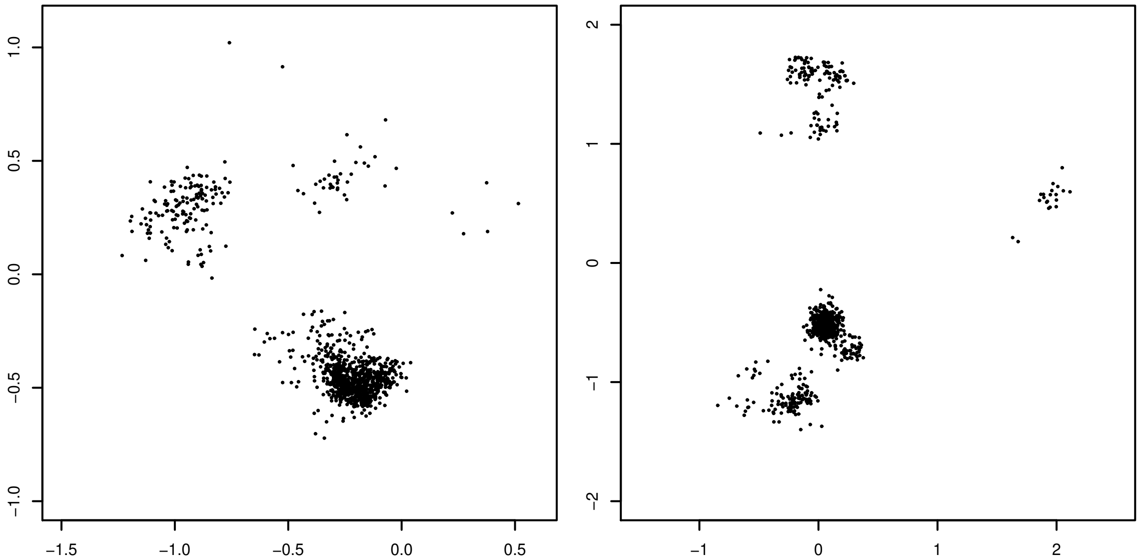


Two data sets generated with $c = 3/2$

Dirichlet Diffusion Tree Priors With $a(t) = b + d/(1-t)^2$

The divergence function $a(t) = b + d/(1-t)^2$ can produce data sets with well-separated clusters that have internal structure, but which appear smooth at a small scale.

For example:



Two data sets generated with $b = 1/2$ and $d = 1/200$

Continuity and Absolute Continuity

A distribution drawn from a Dirichlet diffusion tree with $\int_0^1 a(t)dt = \infty$ will be *continuous* — ie, two points drawn from the distribution will have zero probability of being identical.

But when will these distributions be *absolutely continuous* — ie, have density functions?

Here are my conjectures (empirically supported) about p-dimensional distributions:

If the divergence function is $a(t) = c/(1-t)$:

1) If $c > p/2$, a distribution drawn from the prior will almost surely be absolutely continuous.

2) If $c < p/2$, a distribution drawn from the prior will almost surely not be absolutely continuous.

Gareth Roberts has recently proved (2) above.

Models Using Dirichlet Diffusion Trees

Dirichlet Diffusion tree priors can directly model the distribution of real-valued data, or they can instead model the distribution of latent values that underlie the data:

- If the observed data is noisy, the latent values would be the noise-free values.

- If the observed data is categorical, the latent values would define the category probabilities — eg, using a logit model.

Gaussian noise is easily handled by absorbing it into the Gaussian diffusion process.

If the noise is non-Gaussian (eg, t distributed), the noise-free latent values need to be explicitly represented. The same is true for latent values in a logit model for binary data.

Learning Structure by Inferring Values of Hyperparameters

Most models will have various unknown hyperparameters, such as:

- Parameters of the divergence function — for example, c in $a(t) = c/(1-t)$.
- Diffusion standard deviations for each variable.
- Noise standard deviations for each variable.

We can give priors to these hyperparameters, and allow their values to be learned from the data.

One benefit: We may learn that some of the variables are mostly "noise" (ie, variation unrelated to other variables). These will then not disturb the clustering based on the other variables.

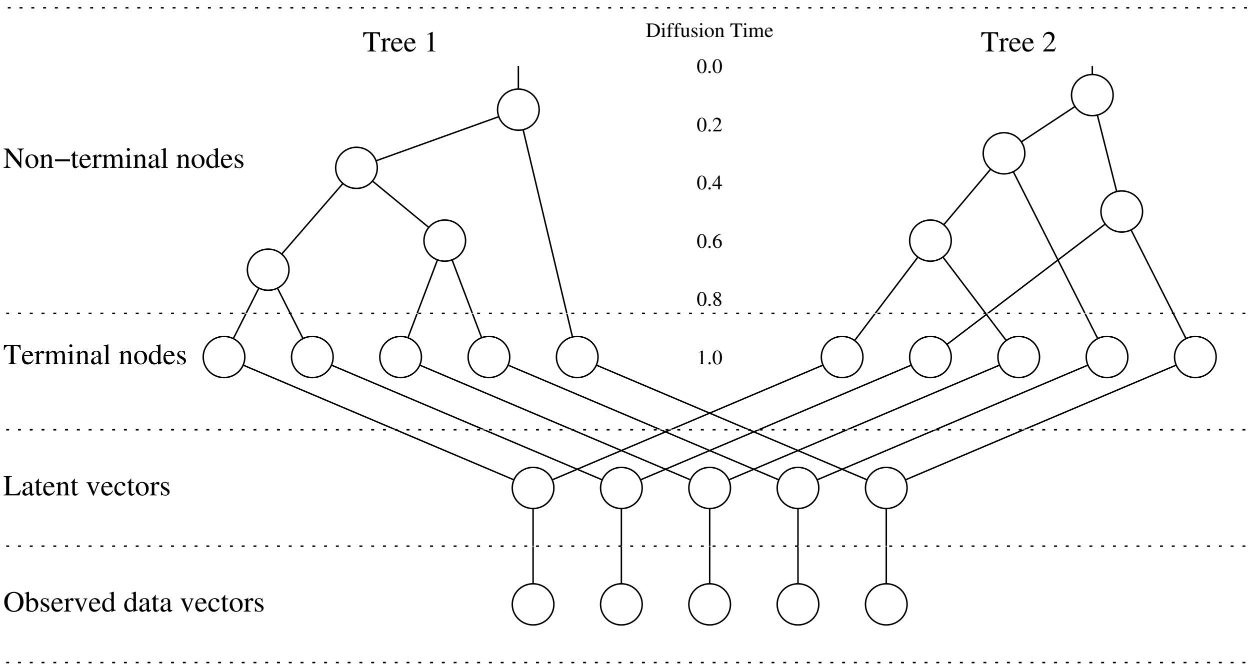
Models With More Than One Tree

We can build models with more than one tree:

- to model clusters that have similar shapes
- to handle data that clusters in more than one way (eg, by disease and by ethnicity).

The terminal nodes are added together to produce the final observed or latent values.

An example for a data set with five cases,



Some computations are more difficult when there is more than one tree.

Using MCMC for Computations

Computation for these models seems to require Monte Carlo methods, and only Markov chain Monte Carlo seems feasible.

We simulate a Markov chain that converges to the posterior distribution over underlying trees, hyperparameters, etc. After discarding a burn-in period, we use later states to:

- Interpret the posterior distribution — eg, look at the likely ways that items may be clustered.
- Make predictions for new data.

Given a sample of trees underlying the data, predictions can be made by simulating the data generation process described earlier.

The State of the Markov Chain

The state of the Markov chain we simulate will need to contain at least the following:

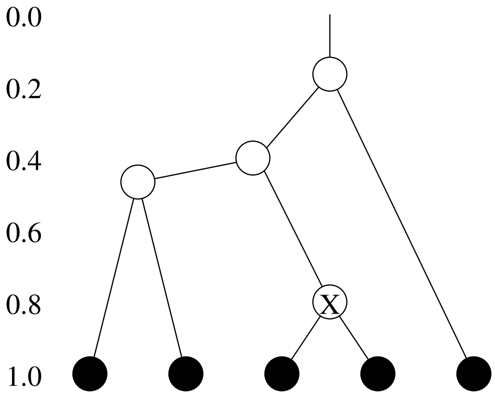
- The *structure* of the tree — ie, the hierarchical organization of points.
- The *divergence times* for nonterminal nodes.

Depending on the model, and on the Markov chain updates used, the state may also need to contain

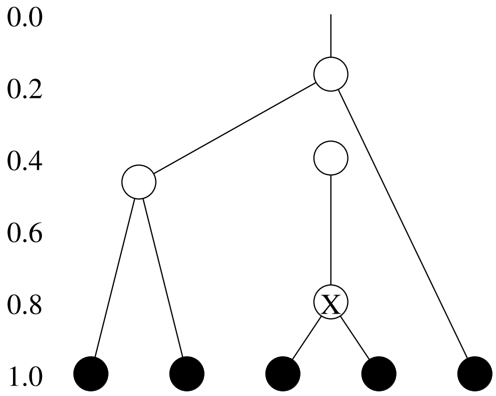
- The *locations* associated with nonterminal nodes.
- The *latent vectors* associated with data items.
- The values of various *hyperparameters*.

Updating the Tree Using Parent Moves

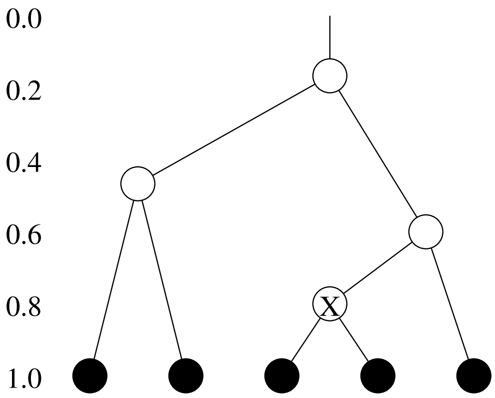
The tree structure, divergence times, and node locations can all be updated by means of *parent moves*, illustrated below:



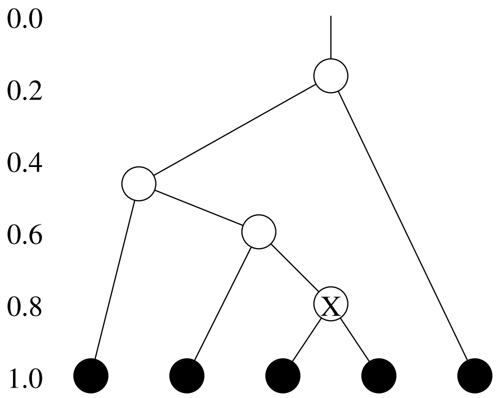
The original tree



After removing X's parent



A tree with X's parent added back in



Another tree with X's parent added back in

Types of Parent Move Updates

Several ways of deciding where to move the parent are possible.

In a *Metropolis-Hastings* move we:

- pick a new segment of the tree and a new divergence time within this segment, using some proposal distribution,
- decide whether or not to accept this move, based on the prior and likelihood for the new and old position, and on the proposal probabilities.

We can pick a segment and time uniformly, or by simulating the data generation process, which makes the prior cancel out.

Slice sampling can also be used. We pick a path from the root to a terminal node and some node along this path, then update this node's position on the path by slice sampling.

Node Locations in Parent Moves

Because the diffusion process is Gaussian, node locations can be analytically integrated away if desired. But this might be slower per iteration than keeping node locations explicitly.

Three possible ways of doing parent moves:

1) Keep all node locations around. Accept/reject based on the likelihood with these locations fixed. (Sample new locations in other updates.)

2) Keep all node locations around, *except* the location of the parent node. Accept/reject based on the likelihood integrating over this location. Sample a new location for the parent at the end of each update.

3) Don't keep any node locations around. Accept/reject based on the likelihood integrating over all node locations.

Likelihood Without Node Locations

The tree structure allows the likelihood with node locations integrated away to be computed in linear time. If diffusions are independent for different variables, we can handle each variable separately.

The likelihood function for a node's location is Gaussian in shape, characterized by mean, variance, and maximum height.

We recursively compute the likelihood function each node from the likelihood functions for its children.

Once we have likelihoods functions for all nodes, we can *update* them in a parent move in less than linear time — we just update the likelihood functions for the ancestors of the parent being moved (in both its old and its new position).

Sampling for Locations & Updating Hyperparameters

By exploiting the tree structure, we can also sample values for all node locations in linear time (independently of previous values, if any).

We can sample new values for the *diffusion variances* once we have node locations, if we use a conjugate inverse gamma prior.
Each node provides an independent data point regarding the diffusion variance.

The *noise variances* (if the model includes noise) can be sampled similarly.

We can update parameters of the *divergence function* by Metropolis or slice sampling methods, given only the tree structure and divergence times.

Testing the Methods on the Iris Data

I looked at autocorrelation times for different methods on the famous Iris data (the 100 observations on the two easily-confused species).

		mean divt	mean depth	divt 2,3	I1,2 :100
WITH NO NODE LOCATIONS	6 x slice	5.2	3.1	6.5	4.8
	6 x met-t	13.8	6.5	4.6	3.4
	6 x met-nt	12.0	5.3	8.1	6.6
	3 x (met-t+met-nt)	8.8	3.3	3.0	3.3
	3 x (slice+met-nt)	6.3	2.9	4.6	3.9
	3 x (slice+met-t)	5.0	2.9	3.2	3.0
	2 x (slice+met-t+met-nt)	6.2	3.3	3.1	3.2
WITH NODE LOCATIONS	6 x slice	3.5	2.4	3.7	2.7
	6 x met-t	9.3	4.6	2.1	1.8
slice 4 times	6 x met-nt	11.5	5.6	9.7	8.5
met-t 8 times	3 x (met-t+met-nt)	3.9	2.4	1.5	1.6
met-nt 4 times	3 x (slice+met-nt)	4.5	2.7	4.1	3.2
	3 x (slice+met-t)	2.3	2.2	1.5	1.6
	2 x (slice+met-t+met-nt)	2.9	2.3	1.5	1.6
WITH NODE LOCATIONS,	6 x slice	3.2	2.2	3.2	2.8
SAMPLED EACH ITERATION	6 x met-t	8.4	4.2	2.1	1.4
	6 x met-nt	12.7	5.0	9.8	6.9
slice 4 times	3 x (met-t+met-nt)	4.1	2.4	1.7	1.4
met-t 8 times	3 x (slice+met-nt)	4.1	2.5	4.2	3.2
met-nt 4 times	3 x (slice+met-t)	2.3	2.1	1.5	1.5
	2 x (slice+met-t+met-nt)	2.9	2.2	1.6	1.5

Clustering Gene Expression Data

I have applied Dirichlet diffusion tree models to data on gene expression in leukemia cells from Golub, *et al* (*Science*, 15 October 1999).

The data has 72 cases (leukemia cells from different patients). After some selection, there is data on the level of expression in these cells of 3571 genes. Each case was classified as one of three types — AML, ALL-B, or ALL-T — based on data other than the gene expression.

By clustering the gene expression data, can we rediscover these three known types?

Tests Using Only 200 Genes

I first did some tests using a random subset of only 200 genes.

The expression levels were log transformed, standardized to unit variance within each case (row), then standardized within each column.

The model included Gaussian noise. The noise levels, the diffusion standard deviations, and the parameters of the divergence function were given fairly broad priors.

I used parent moves with Metropolis-Hastings, for both terminal and nonterminal nodes, and also slice sampling. I tried runs with and without nonterminal node locations present.

These MCMC runs took about an hour on an 866 MHz Pentium III, which superficially seemed to be long enough for good sampling. To check this, I did two runs with different random number seeds.

Data set with 200 genes

Last tree from run #1
(with no node locations)

```
72| 49| 27| 21| 20| 16| 15|  9|  8|  7|  5|  3|  2| 13:ALL-B
  |   |   |   |   |   |   |   |   |   |   |   |   | 24:ALL-B
  |   |   |   |   |   |   |   |   |   |   |   | 69:ALL-B
  |   |   |   |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |   |   |   |  2| 48:ALL-B
  |   |   |   |   |   |   |   |   |   |   |   | 68:ALL-B
  |   |   |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |   |   |  2| 5:ALL-B
  |   |   |   |   |   |   |   |   |   |   | 15:ALL-B
  |   |   |   |   |   |   |   |   | 41:ALL-B
  |   |   |   |   |   |   |   | 20:ALL-B
  |   |   |   |   |   |   |
  |   |   |   |   |   |   |  6|  4|  3|  2| 44:ALL-B
  |   |   |   |   |   |   |   |   |   |   | 70:ALL-B
  |   |   |   |   |   |   |   |   |   | 46:ALL-B
  |   |   |   |   |   |   |   |   | 45:ALL-B
  |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |  2| 16:ALL-B
  |   |   |   |   |   |   |   |   | 19:ALL-B
  |   |   |   |   |   | 47:ALL-B
  |   |   |   |   |
  |   |   |   |   |  4|  2| 42:ALL-B
  |   |   |   |   |   |   | 43:ALL-B
  |   |   |   |   |   |
  |   |   |   |   |   |  2| 29:AML
  |   |   |   |   |   |   | 66:AML
  |   |   |   | 21:ALL-B
  |   |   |
  |   |   |  6|  4|  3|  2| 71:ALL-B
  |   |   |   |   |   |   | 72:ALL-B
  |   |   |   |   |   | 67:ALL-T
  |   |   |   |   | 18:ALL-B
  |   |   |   |
  |   |   |   |  2| 2:ALL-T
  |   |   |   |   | 14:ALL-T
  |   |
  |   | 22| 15| 14| 13| 12|  7|  5|  4|  3|  2| 1:ALL-B
  |   |   |   |   |   |   |   |   |   |   |   | 4:ALL-B
  |   |   |   |   |   |   |   |   |   |   | 8:ALL-B
  |   |   |   |   |   |   |   |   |   | 7:ALL-B
  |   |   |   |   |   |   |   |   | 40:ALL-B
  |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |  2| 27:ALL-B
  |   |   |   |   |   |   |   |   | 49:ALL-B
  |   |   |   |   |   |   |
  |   |   |   |   |   |   |  5|  4|  3|  2| 26:ALL-B
  |   |   |   |   |   |   |   |   |   |   | 56:ALL-B
  |   |   |   |   |   |   |   |   |   | 25:ALL-B
  |   |   |   |   |   |   |   |   | 39:ALL-B
  |   |   |   |   |   |   |   | 55:ALL-B
  |   |   |   |   |   | 22:ALL-B
  |   |   |   |   | 12:ALL-B
  |   |   |   | 59:ALL-B
  |   |   |
  |   |   |  7|  6|  4|  2| 10:ALL-T
  |   |   |   |   |   |   | 23:ALL-T
  |   |   |   |   |   |
  |   |   |   |   |   |  2| 6:ALL-T
  |   |   |   |   |   |   | 11:ALL-T
  |   |   |   |   |
  |   |   |   |   |  2| 3:ALL-T
  |   |   |   |   |   | 9:ALL-T
  |   |   |   | 17:ALL-B
  |
  | 23| 20| 17| 10|  9|  8|  6|  4|  2| 33:AML
  |   |   |   |   |   |   |   |   |   | 53:AML
  |   |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |   |  2| 37:AML
  |   |   |   |   |   |   |   |   |   | 38:AML
  |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |  2| 50:AML
  |   |   |   |   |   |   |   |   | 51:AML
  |   |   |   |   |   |   |
  |   |   |   |   |   |   |  2| 28:AML
  |   |   |   |   |   |   |   | 32:AML
  |   |   |   |   |   | 36:AML
  |   |   |   |   | 30:AML
  |   |   |   |
  |   |   |   |  7|  4|  3|  2| 34:AML
  |   |   |   |   |   |   |   | 35:AML
  |   |   |   |   |   |   | 64:AML
  |   |   |   |   |   | 31:AML
  |   |   |   |   |
  |   |   |   |   |  3|  2| 58:AML
  |   |   |   |   |   |   | 61:AML
  |   |   |   |   |   | 57:AML
  |   |   |
  |   |   |  3|  2| 62:AML
  |   |   |   |   | 63:AML
  |   |   |   | 52:AML
  |   |
  |   |  3|  2| 54:AML
  |   |   |   | 60:AML
  |   |   | 65:AML
```

```
72| 38| 23| 21| 16| 15| 12|  8|  6|  4|  2| 37:AML
  |   |   |   |   |   |   |   |   |   |   | 38:AML
  |   |   |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |   |   |  2| 33:AML
  |   |   |   |   |   |   |   |   |   |   | 53:AML
  |   |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |   |  2| 50:AML
  |   |   |   |   |   |   |   |   |   | 51:AML
  |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |  2| 28:AML
  |   |   |   |   |   |   |   |   | 32:AML
  |   |   |   |   |   |   |
  |   |   |   |   |   |   |  4|  3|  2| 34:AML
  |   |   |   |   |   |   |   |   |   | 35:AML
  |   |   |   |   |   |   |   |   | 64:AML
  |   |   |   |   |   |   |   | 31:AML
  |   |   |   |   |   |
  |   |   |   |   |   |  3|  2| 57:AML
  |   |   |   |   |   |   |   | 61:AML
  |   |   |   |   |   |   | 58:AML
  |   |   |   |   | 65:AML
  |   |   |   |
  |   |   |   |  5|  3|  2| 62:AML
  |   |   |   |   |   |   | 63:AML
  |   |   |   |   |   | 52:AML
  |   |   |   |   |
  |   |   |   |   |  2| 30:AML
  |   |   |   |   |   | 36:AML
  |   |   |
  |   |   |  2| 54:AML
  |   |   |   | 60:AML
  |   |
  |   | 15| 13| 12| 11|  6|  4|  3|  2| 4:ALL-B
  |   |   |   |   |   |   |   |   |   | 7:ALL-B
  |   |   |   |   |   |   |   |   | 8:ALL-B
  |   |   |   |   |   |   |   | 1:ALL-B
  |   |   |   |   |   |   |
  |   |   |   |   |   |   |  2| 27:ALL-B
  |   |   |   |   |   |   |   | 49:ALL-B
  |   |   |   |   |   |
  |   |   |   |   |   |  5|  4|  2| 26:ALL-B
  |   |   |   |   |   |   |   |   | 39:ALL-B
  |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |  2| 55:ALL-B
  |   |   |   |   |   |   |   |   | 56:ALL-B
  |   |   |   |   |   |   | 25:ALL-B
  |   |   |   |   | 40:ALL-B
  |   |   |   | 22:ALL-B
  |   |   |
  |   |   |  2| 12:ALL-B
  |   |   |   | 59:ALL-B
  |
  | 34| 29| 21| 20| 17| 15|  9|  8|  6|  5|  4|  3|  2| 15:ALL-B
  |   |   |   |   |   |   |   |   |   |   |   |   |   | 48:ALL-B
  |   |   |   |   |   |   |   |   |   |   |   |   | 24:ALL-B
  |   |   |   |   |   |   |   |   |   |   |   | 68:ALL-B
  |   |   |   |   |   |   |   |   |   |   | 13:ALL-B
  |   |   |   |   |   |   |   |   |   | 69:ALL-B
  |   |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |   |  2| 5:ALL-B
  |   |   |   |   |   |   |   |   |   | 41:ALL-B
  |   |   |   |   |   |   |   | 20:ALL-B
  |   |   |   |   |   |   |
  |   |   |   |   |   |   |  6|  4|  2| 45:ALL-B
  |   |   |   |   |   |   |   |   |   | 46:ALL-B
  |   |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |   |  2| 44:ALL-B
  |   |   |   |   |   |   |   |   |   | 70:ALL-B
  |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |  2| 16:ALL-B
  |   |   |   |   |   |   |   |   | 19:ALL-B
  |   |   |   |   |   |
  |   |   |   |   |   |  2| 42:ALL-B
  |   |   |   |   |   |   | 43:ALL-B
  |   |   |   |   |
  |   |   |   |   |  3|  2| 29:AML
  |   |   |   |   |   |   | 66:AML
  |   |   |   |   |   | 21:ALL-B
  |   |   |   | 47:ALL-B
  |   |   |
  |   |   |  8|  6|  4|  2| 11:ALL-T
  |   |   |   |   |   |   | 23:ALL-T
  |   |   |   |   |   |
  |   |   |   |   |   |  2| 6:ALL-T
  |   |   |   |   |   |   | 10:ALL-T
  |   |   |   |   |
  |   |   |   |   |  2| 3:ALL-T
  |   |   |   |   |   | 9:ALL-T
  |   |   |   |
  |   |   |   |  2| 2:ALL-T
  |   |   |   |   | 14:ALL-T
  |   |
  |   |  5|  4|  3|  2| 71:ALL-B
  |   |   |   |   |   | 72:ALL-B
  |   |   |   |   | 67:ALL-T
  |   |   |   | 18:ALL-B
  |   |   | 17:ALL-B
```

Data set with 200 genes

Last tree from run #2
(with no node locations)

Conclusions from the Tests Using Only 200 Genes

- The results obtained were plausible in terms of pre-existing knowledge.

- The posterior distributions of the hyperparameters for noise and diffusion standard deviations varied between genes, indicating that the model has inferred which are more relevant to the clustering.

- The two runs were consistent, showing no signs that the Markov chains had not reached equilibrium — though it's hard to be sure!

- Contrary to the results on the Iris data, there was no clear benefit from having node locations present.

Results Using All 3571 Genes

I also applied the same model to the full data set with all 3571 genes.

I let this MCMC run go for about ten hours, which by some indications seemed plenty of time.

However, the results were no better, and perhaps a bit worse, than with only 200 genes.

Could this be because the MCMC runs were trapped in local optima? Or with more genes, is the AML, ALL-B, ALL-T distinction somewhat obscured by smaller effects?

From looking at runs with different random seeds, it seems that Markov chain convergence was indeed a problem, since some details differed between runs.

```
 72| 37| 29| 15| 11|  9|  7|  4|  3|  2| 4:ALL-B
   |   |   |   |   |   |   |   |   |   | 7:ALL-B
   |   |   |   |   |   |   |   |   | 1:ALL-B
   |   |   |   |   |   |   |   | 8:ALL-B
   |   |   |   |   |   |   |
   |   |   |   |   |   |   |  3|  2| 27:ALL-B
   |   |   |   |   |   |   |   |   | 49:ALL-B
   |   |   |   |   |   |   |   | 56:ALL-B
   |   |   |   |   |   |
   |   |   |   |   |   |  2| 22:ALL-B
   |   |   |   |   |   |   | 39:ALL-B
   |   |   |   |   |
   |   |   |   |   |  2| 58:AML
   |   |   |   |   |   | 61:AML
   |   |   |   |
   |   |   |   |  4|  3|  2| 34:AML
   |   |   |   |   |   |   | 35:AML
   |   |   |   |   |   | 64:AML
   |   |   |   |   | 40:ALL-B
   |   |   |
   |   |   | 14| 12| 10|  9|  7|  6|  3|  2| 37:AML
   |   |   |   |   |   |   |   |   |   |   | 38:AML
   |   |   |   |   |   |   |   |   |   | 33:AML
   |   |   |   |   |   |   |   |   |
   |   |   |   |   |   |   |   |   |  3|  2| 50:AML
   |   |   |   |   |   |   |   |   |   |   | 51:AML
   |   |   |   |   |   |   |   |   |   | 53:AML
   |   |   |   |   |   |   |   | 28:AML
   |   |   |   |   |   |   |
   |   |   |   |   |   |   |  2| 30:AML
   |   |   |   |   |   |   |   | 36:AML
   |   |   |   |   |   | 32:AML
   |   |   |   |   |
   |   |   |   |   |  2| 31:AML
   |   |   |   |   |   | 65:AML
   |   |   |   |
   |   |   |   |  2| 12:ALL-B
   |   |   |   |   | 25:ALL-B
   |   |
   |   |  8|  7|  4|  3|  2| 54:AML
   |   |   |   |   |   |   | 60:AML
   |   |   |   |   |   | 57:AML
   |   |   |   |   | 55:ALL-B
   |   |   |   |
   |   |   |   |  3|  2| 62:AML
   |   |   |   |   |   | 63:AML
   |   |   |   |   | 52:AML
   |   |   | 18:ALL-B
   |
   | 35| 26| 23| 22| 18| 13|  9|  5|  4|  3|  2| 5:ALL-B
   |   |   |   |   |   |   |   |   |   |   |   | 15:ALL-B
   |   |   |   |   |   |   |   |   |   |   | 41:ALL-B
   |   |   |   |   |   |   |   |   |   | 24:ALL-B
   |   |   |   |   |   |   |   |   | 68:ALL-B
   |   |   |   |   |   |   |   |
   |   |   |   |   |   |   |   |  4|  3|  2| 13:ALL-B
   |   |   |   |   |   |   |   |   |   |   | 48:ALL-B
   |   |   |   |   |   |   |   |   |   | 69:ALL-B
   |   |   |   |   |   |   |   |   | 20:ALL-B
   |   |   |   |   |   |   |
   |   |   |   |   |   |   |  4|  3|  2| 16:ALL-B
   |   |   |   |   |   |   |   |   |   | 19:ALL-B
   |   |   |   |   |   |   |   |   | 26:ALL-B
   |   |   |   |   |   |   |   | 47:ALL-B
   |   |   |   |   |   |
   |   |   |   |   |   |  5|  4|  2| 44:ALL-B
   |   |   |   |   |   |   |   |   | 45:ALL-B
   |   |   |   |   |   |   |   |
   |   |   |   |   |   |   |   |  2| 46:ALL-B
   |   |   |   |   |   |   |   |   | 70:ALL-B
   |   |   |   |   |   |   | 59:ALL-B
   |   |   |   |   |
   |   |   |   |   |  4|  2| 29:AML
   |   |   |   |   |   |   | 66:AML
   |   |   |   |   |   |
   |   |   |   |   |   |  2| 42:ALL-B
   |   |   |   |   |   |   | 43:ALL-B
   |   |   |   | 21:ALL-B
   |   |   |
   |   |   |  3|  2| 71:ALL-B
   |   |   |   |   | 72:ALL-B
   |   |   |   | 67:ALL-T
   |   |
   |   |  9|  8|  7|  6|  4|  3|  2| 6:ALL-T
   |   |   |   |   |   |   |   |   | 23:ALL-T
   |   |   |   |   |   |   |   | 3:ALL-T
   |   |   |   |   |   |   | 10:ALL-T
   |   |   |   |   |   |
   |   |   |   |   |   |  2| 9:ALL-T
   |   |   |   |   |   |   | 11:ALL-T
   |   |   |   |   | 14:ALL-T
   |   |   |   | 2:ALL-T
   |   |   | 17:ALL-B
```

Data set using all genes

Last tree from run #1
(with no node locations)

```
72| 37| 29| 27| 14| 12| 11|  9|  7|  6|  3|  2| 50:AML
  |   |   |   |   |   |   |   |   |   |   |   | 51:AML
  |   |   |   |   |   |   |   |   |   |   | 53:AML
  |   |   |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |   |   |  3|  2| 37:AML
  |   |   |   |   |   |   |   |   |   |   |   | 38:AML
  |   |   |   |   |   |   |   |   |   |   | 33:AML
  |   |   |   |   |   |   |   |   | 28:AML
  |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |  2| 30:AML
  |   |   |   |   |   |   |   |   | 36:AML
  |   |   |   |   |   |   |
  |   |   |   |   |   |   |  2| 31:AML
  |   |   |   |   |   |   |   | 32:AML
  |   |   |   |   |   | 65:AML
  |   |   |   |   |
  |   |   |   |   |  2| 58:AML
  |   |   |   |   |   | 61:AML
  |   |   |   |
  |   |   |   | 13|  9|  7|  4|  3|  2| 4:ALL-B
  |   |   |   |   |   |   |   |   |   | 7:ALL-B
  |   |   |   |   |   |   |   |   | 1:ALL-B
  |   |   |   |   |   |   |   | 8:ALL-B
  |   |   |   |   |   |   |
  |   |   |   |   |   |   |  3|  2| 27:ALL-B
  |   |   |   |   |   |   |   |   | 49:ALL-B
  |   |   |   |   |   |   |   | 56:ALL-B
  |   |   |   |   |   |
  |   |   |   |   |   |  2| 22:ALL-B
  |   |   |   |   |   |   | 39:ALL-B
  |   |   |   |   |
  |   |   |   |   |  4|  3|  2| 34:AML
  |   |   |   |   |   |   |   | 35:AML
  |   |   |   |   |   |   | 64:AML
  |   |   |   |   |   | 40:ALL-B
  |   |   |
  |   |   |  2| 12:ALL-B
  |   |   |   | 25:ALL-B
  |   |
  |   |  8|  7|  4|  3|  2| 54:AML
  |   |   |   |   |   |   | 60:AML
  |   |   |   |   |   | 57:AML
  |   |   |   |   | 55:ALL-B
  |   |   |   |
  |   |   |   |  3|  2| 62:AML
  |   |   |   |   |   | 63:AML
  |   |   |   |   | 52:AML
  |   |   | 18:ALL-B
  |
  | 35| 26| 23| 22| 18| 13|  9|  8|  7|  4|  2| 68:ALL-B
  |   |   |   |   |   |   |   |   |   |   |   | 69:ALL-B
  |   |   |   |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |   |   |   |  2| 5:ALL-B
  |   |   |   |   |   |   |   |   |   |   |   | 15:ALL-B
  |   |   |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |   |   |  3|  2| 13:ALL-B
  |   |   |   |   |   |   |   |   |   |   |   | 48:ALL-B
  |   |   |   |   |   |   |   |   |   |   | 20:ALL-B
  |   |   |   |   |   |   |   |   | 24:ALL-B
  |   |   |   |   |   |   |   | 41:ALL-B
  |   |   |   |   |   |   |
  |   |   |   |   |   |   |  4|  3|  2| 16:ALL-B
  |   |   |   |   |   |   |   |   |   | 19:ALL-B
  |   |   |   |   |   |   |   |   | 26:ALL-B
  |   |   |   |   |   |   |   | 47:ALL-B
  |   |   |   |   |   |
  |   |   |   |   |   |  5|  4|  2| 46:ALL-B
  |   |   |   |   |   |   |   |   | 70:ALL-B
  |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   |  2| 44:ALL-B
  |   |   |   |   |   |   |   |   | 45:ALL-B
  |   |   |   |   |   |   | 59:ALL-B
  |   |   |   |   |
  |   |   |   |   |  4|  2| 42:ALL-B
  |   |   |   |   |   |   | 43:ALL-B
  |   |   |   |   |   |
  |   |   |   |   |   |  2| 29:AML
  |   |   |   |   |   |   | 66:AML
  |   |   |   | 21:ALL-B
  |   |   |
  |   |   |  3|  2| 71:ALL-B
  |   |   |   |   | 72:ALL-B
  |   |   |   | 67:ALL-T
  |   |
  |   |  9|  8|  7|  6|  4|  3|  2| 6:ALL-T
  |   |   |   |   |   |   |   |   | 23:ALL-T
  |   |   |   |   |   |   |   | 3:ALL-T
  |   |   |   |   |   |   | 10:ALL-T
  |   |   |   |   |   |
  |   |   |   |   |   |  2| 9:ALL-T
  |   |   |   |   |   |   | 11:ALL-T
  |   |   |   |   | 14:ALL-T
  |   |   |   | 2:ALL-T
  |   |   | 17:ALL-B
```

Data set using all genes

Last tree from run #2
(with no node locations)

Future Work

Theory:

- Confirm or refute the conjectures about absolute continuity.

Computation:

- Try out MCMC methods that have been used for phylogenetic trees.
- See if annealing/tempering can remedy the converge problems with large data sets.
- See if circular coupling or coupling from the past can be made to work.

Modeling:

- Figure out how to visualize a posterior distribution over hierarchical structures.
- Consider models in which the Brownian motion is replaced by a continuous-time Markov process with a categorical state.