Bayesian ancestral state reconstruction

Simon Joly

BIO 6008 - Fall 2015

Contents

Bayesian ancestral state reconstuctions

In the previous lecture, we saw how to reconstruct ancestral state reconstruction using maximum likelihood or stochatic mapping, the latter of which uses monte carlo simulations. In this lecture, we will see how to reconstruct ancestral states using a full Bayesian approach (Pagel et al. 2004).

For this, we will use the program BayesTraits, written by Andrew Meade and Mark Pagel. Coveniently, Randi Griffin has written wrapper functions that allow to call BayesTrait from R. The BayesTrait wrapper function can be downloaded from her website. However, I have included modified scripts with this lecture of the BayesTraits Wrapper that include several important improvements. For instance, I modified the scripts to make the wrapper multi-platform (the original was only for OSX) and to use the second version of BayesTraits. The BayesTrait Wrapper can be found in the folder BTW.

Prepare seed plant data

Throughout this tutorial, we will use the seed plant phylogeny and trait data from Paquette et al. (2015). Let's load it and prepare it.

```
# Load ape
require(ape)
# Import datasets
seedplantstree <- read.nexus("./data/seedplants.tre")
```

```
seedplantsdata <- read.csv2("./data/seedplants.csv")
# Remove species for which we don't have complete data
seedplantsdata <- na.omit(seedplantsdata)
# Remove species in the tree that are not in the data matrix
species.to.exclude <- seedplantstree$tip.label[!(seedplantstree$tip.label %in%
                                                 seedplantsdata$Code)]
seedplantstree <- drop.tip(seedplantstree,species.to.exclude)
rm(species.to.exclude)
# Name the rows of the data.frame with the species codes used as tree labels
rownames(seedplantsdata) <- seedplantsdata$Code
seedplantsdata <- seedplantsdata[,-1]
# Order the data in the same order as the tip.label of the tree. In the present
# example, this was already the case.
seedplantsdata <- seedplantsdata[seedplantstree$tip.label,]
# Create a factor for a categorical variable
height <- factor(seedplantsdata$height)
names(height) <- rownames(seedplantsdata)
# Create a vector for a continuous character
maxH <- seedplantsdata$maxH
names(maxH) <- rownames(seedplantsdata)
```

Multistate reconstruction

We will use the MultistateMCMC R function to estimate ancestral states using a Bayesian approach in BayesTraits (Pagel et al. 2004). The first thing to do is to load all the functions from BayesTrait Wrapper. Copy the BTW folder into your working folder and then enter the following code:

```
for (n in 1:length(list.files('./BTW/R'))) {
  source(paste("./BTW/R/", list.files('./BTW/R')[n], sep=""))
}
```

You will then have to copy the "BayesTraits" programs into your working directory. You can only copy the program that corresponds to your operating system.

Running the analysis

BayesTraits has several functions that can be modified. The most important are implemented in the BTW functions, but maybe not all of them. For a complete description of the functions available in these function, you should have a look at the BTW manual in the folder `./BTW/help/BTWman.pdf`.

When running a Bayesian analysis of BayesTraits from R, the following parameters are important for Bayesian MCMC analyses:

Parameter	Description
it	integer specifying number of MCMC iterations. Default is 10000.
bi	integer specifying number of iterations to discard as burn-in. Default is 1000.
sa	integer specifying number of iterations to skip between samples. Default is 100.
rd	positive number specifying the rate deviation parameter. Default is 2.

When running the a Multistate analysis, these options are also important:

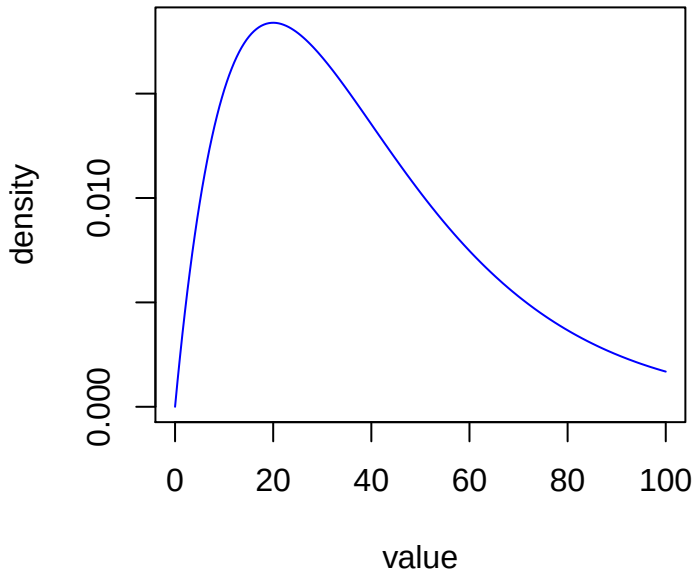
Parameter	Description
res	character or vector indicating restrictions to place on rates. If a vector is given, each element indicates an inde
resall	character indicating a rate or a non-negative number to restrict all rates to.
mrca	character or vector indicating nodes to reconstruct using the most recent common ancestor approach. If a vect
fo	character or vector indicating nodes to fix at particular states. If a vector is given, each element is a character
et	character or vector listing taxa to exclude.

And finally, the MultistateMCMC analysis have these additional parameters:

Parameter	Description
pr	character or vector describing prior distributions for model parameters. If a vector is given, each element is a c
pa	character string specifying the prior distribution for all parameters by listing first the name of the parameter, t
rj	toggles reversible jump model if non-empty. A character string specifying the prior distribution and it's param
rjhp	toggles reversible jump model with a hyper-prior if non-empty. A character string specifying the prior distribu
hp	character or vector describing prior distributions and hyper-priors for model parameters. If a vector is given, e
hpall	character string specifying the prior distribution and hyper-prior for all parameters by listing first the name of

We will now run three independent MCMC runs of BayesTraits. This is important to make sure that the analyses have converged on the same estimates.We will use the same substitution model as last week, that is with three different rates. We will run an analysis of 100000 generations (`it`), sampling the chain every 100 generations (`sa=100`), discarding the first 1000 as burnin (`bi`). Finally, a gamma prior with mean 2 and shape 20 will be given to all parameters (`pa="gamma 2 20"`). This distribution looks like the following:

```
x <- seq(0, 100, length=200)
hx <- dgamma(x,shape=2,scale=20)
plot(x,hx,type="l",ylab="density",xlab="value",col="blue")
```



Now, let's run BayesTraits.

```
height.dat<-data.frame(code=rownames(seedplantsdata),height=as.numeric(seedplantsdata$height))
# Model: rate constraints
constraints <- c("q21 q13", "q23 q32", "q12 q31")
# Run three independent analyses (MCMC chains)
multistate.MCMC.res1 <- MultistateMCMC(seedplantstree, height.dat, res=constraints,
        it = 100000, bi = 1000, sa = 100, rd = 2, pa = "gamma 2 20", silent = TRUE)
multistate.MCMC.res2 <- MultistateMCMC(seedplantstree, height.dat, res=constraints,
        it = 100000, bi = 1000, sa = 100, rd = 2, pa = "gamma 2 20", silent = TRUE)
multistate.MCMC.res3 <- MultistateMCMC(seedplantstree, height.dat, res=constraints,
        it = 100000, bi = 1000, sa = 100, rd = 2, pa = "gamma 2 20", silent = TRUE)
```

Bayesian analysis diagnostics

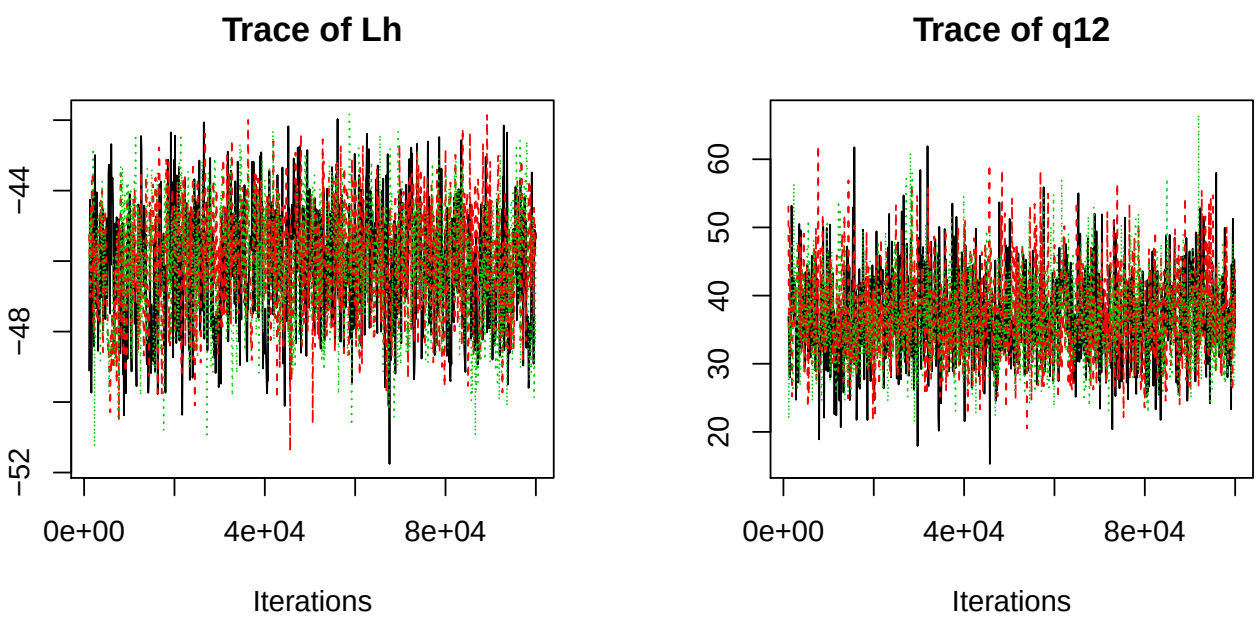
For diagnostic of Bayesian MCMC analyses, the package `coda` is very useful to look for chain convergence and calculate statistics. To be able to estimate convergence statistics, it is important to run at least 2 independent chains. This should be standard anyway to ensure convergence and thus that the results are reliable. First, let's convert the output of BayesTraits into `coda` format.

```
require(lattice)
require(coda)
# Read the BayesTrait results in coda format
res1 <- mcmc(multistate.MCMC.res1$Results[,c(-1,-4)],
             start=min(multistate.MCMC.res1$Results$Iteration),
             end=max(multistate.MCMC.res1$Results$Iteration),thin=100)
res2 <- mcmc(multistate.MCMC.res2$Results[,c(-1,-4)],
             start=min(multistate.MCMC.res2$Results$Iteration),
             end=max(multistate.MCMC.res2$Results$Iteration),thin=100)
res3 <- mcmc(multistate.MCMC.res3$Results[,c(-1,-4)],
             start=min(multistate.MCMC.res3$Results$Iteration),
             end=max(multistate.MCMC.res2$Results$Iteration),thin=100)
# Combine the three chains
res <- mcmc.list(res1,res2,res3)
```

Trace plots

Now, we can have a look at the results. Let's start by looking at the values of two parameters along the MCMC chain.

```
# Look at the trace plots for some characters
op <- par(mfrow=c(1,2))
traceplot(res[,c(1,3)])
```



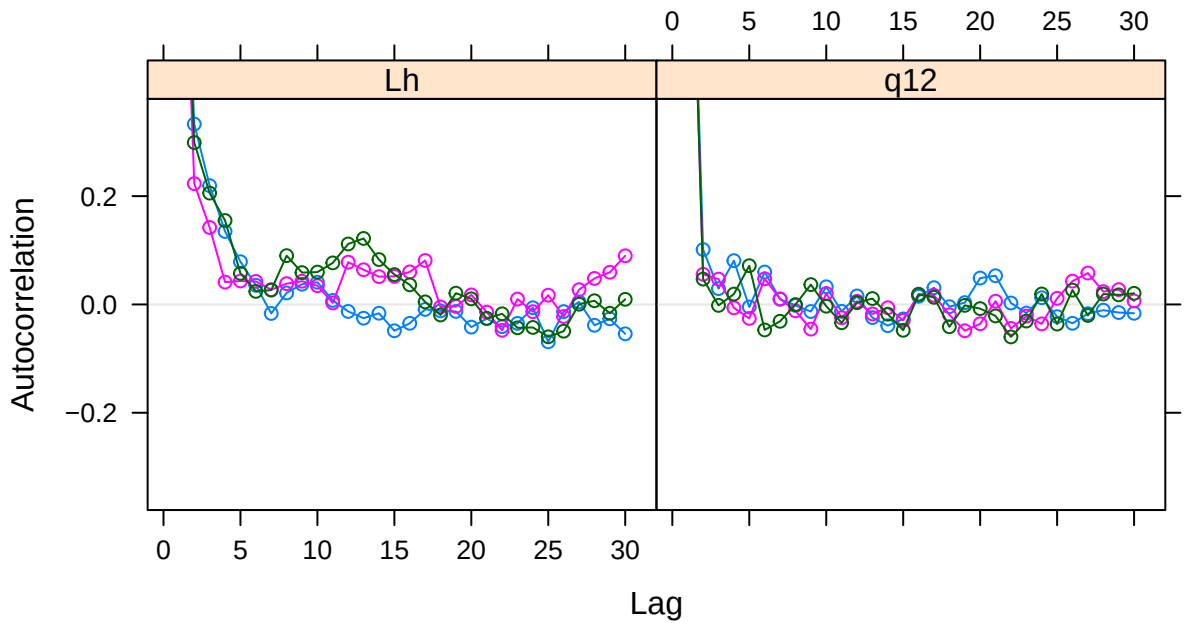

```
par(op)
```

The different colors on the plot represent the different chains. You can see that the values go up-and-down a lot, which is a sign that the chain is mixing well. The opposite would give a lot of correlations between successive samples and would give poor estimates of the parameters.

Autocorrelation plots

You can see how the correlation drops between successive samples by using the function `acfplot`.

```
acfplot(res[,c(1,3)])
```



You can see that when samples are approximately 5 samples apart, they are not much correlated.

Convergence diagnostics

Let's now look at some convergence disgnostics. The effective size of the parameter represents the estimated number of independent samples that are used to estimate the parameter's mean. Because parameter values are sampled from a chain, values sampled consecutively along the chain are generally correlated. The effective size is the estimated number of independent samples remainning once that autocorrelation is removed (this is inferred, of course). You generally want to have at least 200 of effective size to believe in your results (the more the better).

```
# Get effective sizes (should be > 200)
effectiveSize(res)
```

```
##            Lh Harmonic.Mean           q12           q13           q21
##    1111.59179      65.92722    2339.81105    1308.86554    1308.86554
##           q23           q31           q32     Root.P.1.     Root.P.2.
##    1088.55891    2339.81105    1088.55891    2446.53660    2158.64016
##     Root.P.3.
##    2161.96459
```

The Gelman and Rubin's Potential Scale Reduction Factor (PSRF) is based on a comparison of within-chain vs. between-chain variance. If the chains have converged, then the potential scale reduction factor should be 1. If the values are above 1.05, this means you should run the chains longer.

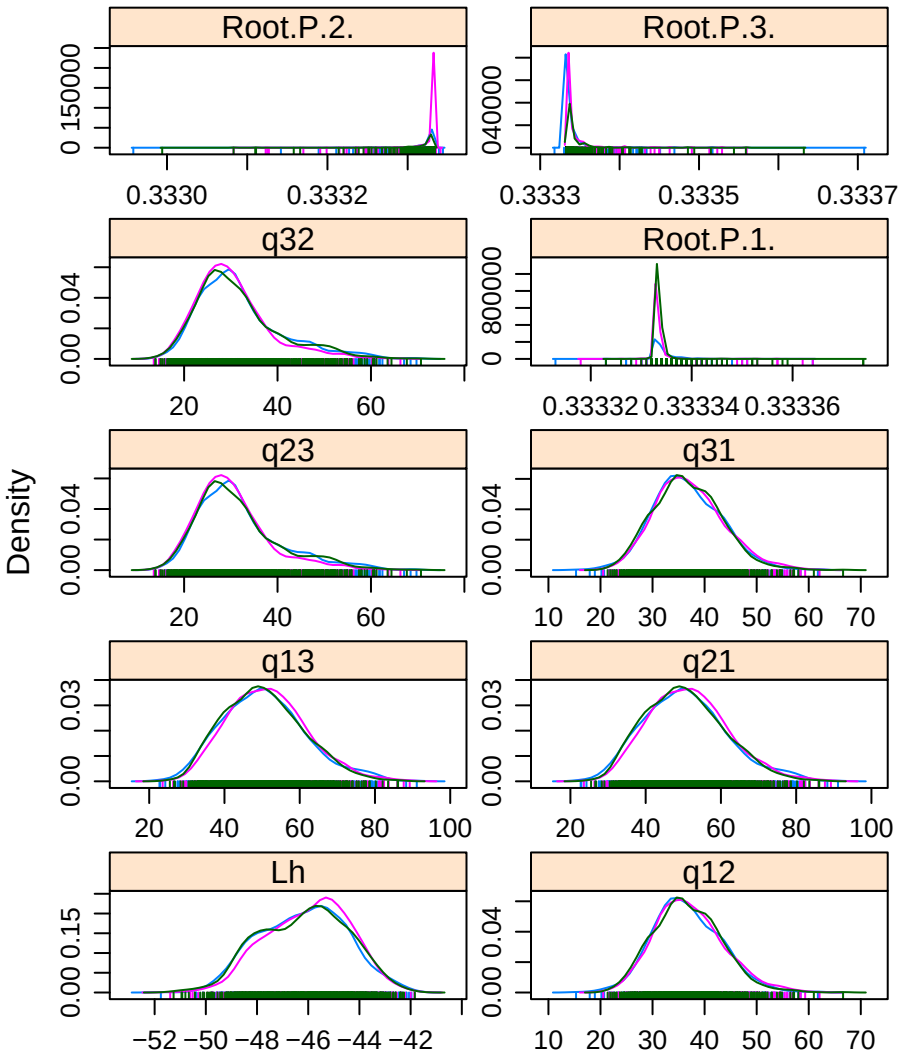
```
# Gelman and Rubin's convergence disgnostic
gelman.diag(res,autoburnin=FALSE,multivariate=FALSE)
```

```
## Potential scale reduction factors:
##
##               Point est. Upper C.I.
## Lh                  1.00       1.01
## Harmonic.Mean       1.15       1.45
## q12                 1.00       1.00
## q13                 1.00       1.01
## q21                 1.00       1.01
## q23                 1.01       1.03
## q31                 1.00       1.00
## q32                 1.01       1.03
## Root.P.1.           1.01       1.02
## Root.P.2.           1.00       1.00
## Root.P.3.           1.00       1.00
```

Density plots

Now, let's look at the density plots for the parameters.

```
# Density Plots
densityplot(res[,-2])
```




```
# Parameter summary
summary(res)
```

```
##
## Iterations = 1100:1e+05
## Thinning interval = 100
## Number of chains = 3
## Sample size per chain = 990
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                    Mean        SD  Naive SE Time-series SE
## Lh             -46.1118 1.665e+00 3.054e-02      5.399e-02
## Harmonic.Mean -47.5270 2.611e-01 4.790e-03      3.194e-02
## q12            36.8631 6.521e+00 1.196e-01      1.366e-01
## q13            50.8819 1.069e+01 1.962e-01      2.962e-01
## q21            50.8819 1.069e+01 1.962e-01      2.962e-01
## q23            31.1989 8.663e+00 1.590e-01      2.713e-01
## q31            36.8631 6.521e+00 1.196e-01      1.366e-01
```

```
## q32            31.1989 8.663e+00 1.590e-01      2.713e-01
## Root.P.1.       0.3333 2.610e-06 4.790e-08      5.398e-08
## Root.P.2.       0.3333 2.049e-05 3.760e-07      4.408e-07
## Root.P.3.       0.3333 1.873e-05 3.436e-07      4.037e-07
##
## 2. Quantiles for each variable:
##
##                    2.5%      25%      50%      75%    97.5%
## Lh             -49.1954 -47.3826 -45.9838 -44.8659 -43.0912
## Harmonic.Mean -48.0323 -47.6418 -47.5069 -47.4339 -47.2737
## q12            25.5061  32.3509  36.3037  41.0941  50.5212
## q13            32.9264  43.1640  50.2819  57.6172  74.0675
## q21            32.9264  43.1640  50.2819  57.6172  74.0675
## q23            18.7377  25.2054  29.5799  34.8674  53.3995
## q31            25.5061  32.3509  36.3037  41.0941  50.5212
## q32            18.7377  25.2054  29.5799  34.8674  53.3995
## Root.P.1.       0.3333   0.3333   0.3333   0.3333   0.3333
## Root.P.2.       0.3333   0.3333   0.3333   0.3333   0.3333
## Root.P.3.       0.3333   0.3333   0.3333   0.3333   0.3334
```

```
# Highest Posterior Density intervals
HPDinterval(res)
```

```
## [[1]]
##                     lower      upper
## Lh             -49.409696 -43.118669
## Harmonic.Mean -47.843079 -47.398942
## q12             24.544150  49.682653
## q13             29.902705  73.457171
## q21             29.902705  73.457171
## q23             17.606525  52.815803
## q31             24.544150  49.682653
## q32             17.606525  52.815803
## Root.P.1.        0.333332   0.333337
## Root.P.2.        0.333309   0.333334
## Root.P.3.        0.333332   0.333356
## attr(,"Probability")
## [1] 0.9494949
##
## [[2]]
##                     lower      upper
## Lh             -48.880845 -43.082262
## Harmonic.Mean -48.056482 -47.272596
## q12             25.153021  50.223431
## q13             32.693461  72.043843
## q21             32.693461  72.043843
## q23             17.093492  47.089023
## q31             25.153021  50.223431
## q32             17.093492  47.089023
## Root.P.1.        0.333333   0.333338
## Root.P.2.        0.333302   0.333335
## Root.P.3.        0.333333   0.333361
## attr(,"Probability")
## [1] 0.9494949
```

```
##
## [[3]]
##                     lower      upper
## Lh             -49.114864 -43.067048
## Harmonic.Mean -48.062377 -47.422544
## q12            24.445523  48.181663
## q13            31.286293  69.231366
## q21            31.286293  69.231366
## q23            18.322521  52.139475
## q31            24.445523  48.181663
## q32            18.322521  52.139475
## Root.P.1.       0.333332   0.333337
## Root.P.2.       0.333305   0.333333
## Root.P.3.       0.333333   0.333357
## attr(,"Probability")
## [1] 0.9494949
```

The density plots show that the runs have converged on very similar posterior distributions, which confirms the convergence diagnostic stats. The summary gives the quantiles and the median value.

Interestingly, whereas the transition rate parameters are in the same order as with likelihood inference, the variation is much smaller... Actually, this is a consequence of the prior used for the rate variation. If we take a flat prior instead between 0 and 1000, here is what we would get:

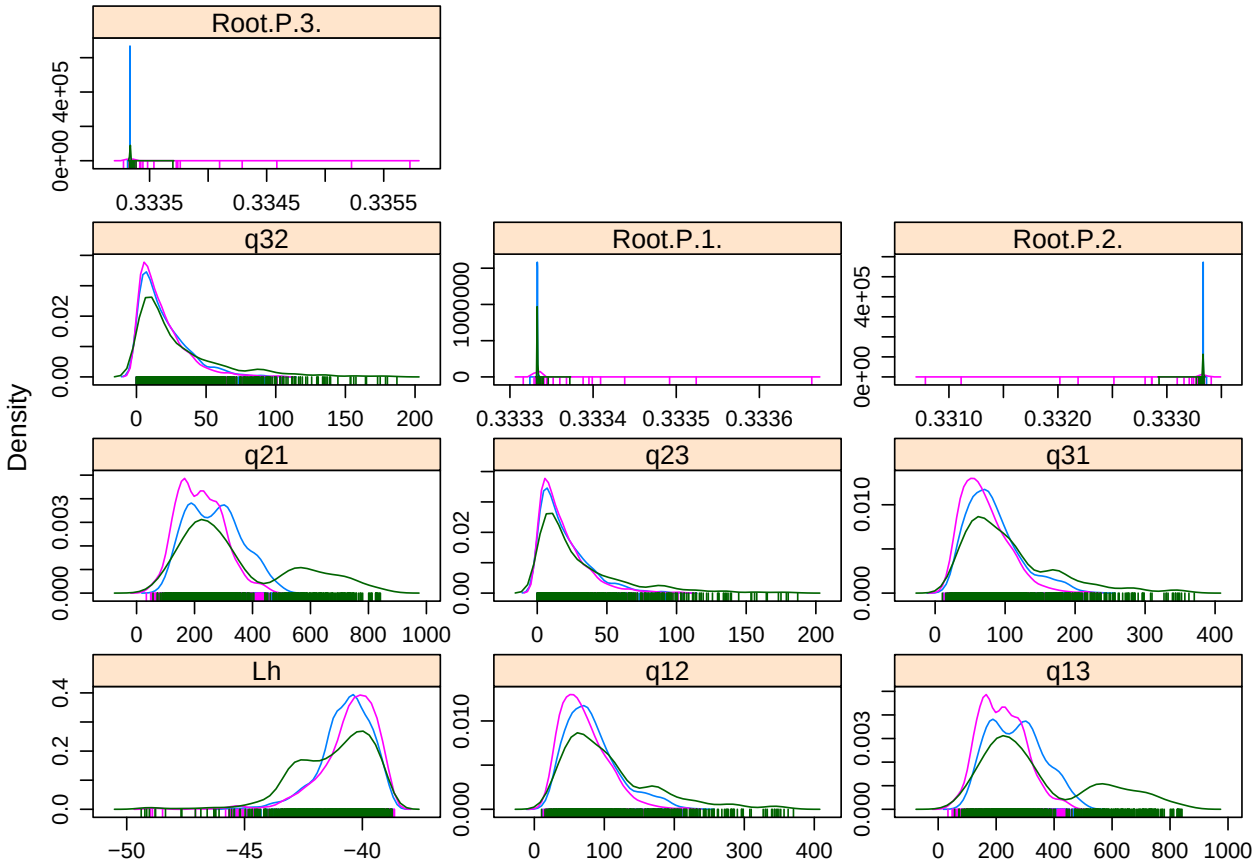
```
multistate.MCMC.res1 <- MultistateMCMC(seedplantstree, height.dat, res=constraints,
    it = 100000, bi = 1000, sa = 100, rd = 2, pa = "uniform 0 1000", silent = TRUE)
multistate.MCMC.res2 <- MultistateMCMC(seedplantstree, height.dat, res=constraints,
    it = 100000, bi = 1000, sa = 100, rd = 2, pa = "uniform 0 1000", silent = TRUE)
multistate.MCMC.res3 <- MultistateMCMC(seedplantstree, height.dat, res=constraints,
    it = 100000, bi = 1000, sa = 100, rd = 2, pa = "uniform 0 1000", silent = TRUE)
# Read the BayesTrait results in coda format
res1 <- mcmc(multistate.MCMC.res1$Results[,c(-1,-4)],
             start=min(multistate.MCMC.res1$Results$Iteration),
             end=max(multistate.MCMC.res1$Results$Iteration),thin=100)
res2 <- mcmc(multistate.MCMC.res2$Results[,c(-1,-4)],
             start=min(multistate.MCMC.res2$Results$Iteration),
             end=max(multistate.MCMC.res2$Results$Iteration),thin=100)
res3 <- mcmc(multistate.MCMC.res3$Results[,c(-1,-4)],
             start=min(multistate.MCMC.res3$Results$Iteration),
             end=max(multistate.MCMC.res2$Results$Iteration),thin=100)
# Combine the three chains
res <- mcmc.list(res1,res2,res3)
# Get effective sizes (should be > 200)
effectiveSize(res)
```

```
##           Lh Harmonic.Mean          q12          q13          q21
##     295.25416      29.76475      98.71270      34.06691      34.06691
##          q23           q31          q32     Root.P.1.     Root.P.2.
##     344.78014      98.71270     344.78014    8730.01222    2735.26034
##     Root.P.3.
##    1694.04306
```

```
# Gelman and Rubin's convergence disgnostic
gelman.diag(res,autoburnin=FALSE,multivariate=FALSE)
```

```
## Potential scale reduction factors:
##
##                Point est. Upper C.I.
## Lh                   1.08       1.21
## Harmonic.Mean        1.33       2.46
## q12                  1.18       1.54
## q13                  1.29       2.05
## q21                  1.29       2.05
## q23                  1.15       1.40
## q31                  1.18       1.54
## q32                  1.15       1.40
## Root.P.1.            1.29       1.48
## Root.P.2.            1.29       1.47
## Root.P.3.            1.29       1.47
```

```
# Density Plots
densityplot(res[,-2])
```




```
# Parameter summary
summary(res)
```

```
##
```

```
## Iterations = 1100:1e+05
## Thinning interval = 100
## Number of chains = 3
## Sample size per chain = 990
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                    Mean        SD  Naive SE Time-series SE
## Lh             -40.8336 1.404e+00 2.576e-02      1.100e-01
## Harmonic.Mean -42.0985 1.051e+00 1.929e-02      3.300e-01
## q12            86.6159 5.054e+01 9.273e-01      6.079e+00
## q13           279.8109 1.435e+02 2.633e+00      3.551e+01
## q21           279.8109 1.435e+02 2.633e+00      3.551e+01
## q23            22.4714 2.362e+01 4.335e-01      1.696e+00
## q31            86.6159 5.054e+01 9.273e-01      6.079e+00
## q32            22.4714 2.362e+01 4.335e-01      1.696e+00
## Root.P.1.       0.3333 8.281e-06 1.519e-07      2.440e-07
## Root.P.2.       0.3333 7.377e-05 1.354e-06      2.409e-06
## Root.P.3.       0.3333 6.641e-05 1.219e-06      2.158e-06
##
## 2. Quantiles for each variable:
##
##                    2.5%      25%      50%      75%    97.5%
## Lh             -44.0686 -41.4847 -40.5514 -39.8349 -38.9699
## Harmonic.Mean -44.4427 -42.9882 -41.6112 -41.4545 -40.3964
## q12            25.6362  52.7689  74.6475 106.4892 217.7181
## q13           105.5286 181.3218 248.9100 325.9098 705.9384
## q21           105.5286 181.3218 248.9100 325.9098 705.9384
## q23             0.9374   6.8686  15.0954  29.6689  89.4593
## q31            25.6362  52.7689  74.6475 106.4892 217.7181
## q32             0.9374   6.8686  15.0954  29.6689  89.4593
## Root.P.1.       0.3333   0.3333   0.3333   0.3333   0.3333
## Root.P.2.       0.3333   0.3333   0.3333   0.3333   0.3333
## Root.P.3.       0.3333   0.3333   0.3333   0.3333   0.3333
```

We can conclude two things from this analysis. First, the posterior distribution is strongly affected by the prior used. This is problematic and it suggest that there may not be enough information in the data to properly estimate the transition rate parameters. This is one of the advantage of the Bayesian approach as you can more easily see when it is the case. Second, the chains have not converged as well with the flat prior. This also likely reflect the little information present in the data. Consequently, you should interpret these results with much caution (if at all!).

Corelated evolution between binary traits in BayesTraits

A common application of phylogenetic methods is to study the correlation of characters (and their evolution). A very popular model for binary traits is that of Pagel (1994). The idea is to test if two traits evolved in a correlated fashion or independently.

The test is run using the Discrete function of BayesTraits. The example below will focus on a Bayesian approach, but this can also be done with ML. The idea is to evaluate two models: one in which the traits evolve independently and another one where the traits evolved in a correlative way.

Independent model

The simpler model is the independent one. In this model, there are four paramters:

Parameter	Trait	Transitions
α_1	1	$0 \rightarrow 1$
β_1	1	$1 \rightarrow 0$
α_2	2	$0 \rightarrow 1$
β_2	2	$1 \rightarrow 0$

This can be represented in a double transition matrix:

	0,0	0,1	1,0	1,1
0,0	-	α_2	α_1	0
0,1	β_2	-	0	α_1
1,0	β_1	0	-	α_2
1,1	0	β_1	β_2	-

Note that the transitions where both characters would have to evolve at the same time are set to zero as this is impossible in an infinitesimal amount of time.

Dependent model

The dependent model is more complex. It assumes that the rate of change in one character depends on the state of the other character.

Parameter	Dependent on	Trait	Transitions
$q_{1,2}$	Trait 1 = 0	2	$0 \rightarrow 1$
$q_{1,3}$	Trait 2 = 0	1	$0 \rightarrow 1$
$q_{2,1}$	Trait 1 = 0	2	$1 \rightarrow 0$
$q_{2,4}$	Trait 2 = 1	1	$0 \rightarrow 1$
$q_{3,1}$	Trait 2 = 0	1	$1 \rightarrow 0$
$q_{3,4}$	Trait 1 = 1	2	$0 \rightarrow 1$
$q_{4,2}$	Trait 2 = 1	1	$1 \rightarrow 0$
$q_{4,3}$	Trait 1 = 1	2	$1 \rightarrow 0$

As you can see, it has 8 parameters instead of 4. This model results in the following double transition matrix:

	0,0	0,1	1,0	1,1
0,0	-	$q_{1,2}$	$q_{1,3}$	0
0,1	$q_{2,1}$	-	0	$q_{2,4}$
1,0	$q_{3,1}$	0	-	$q_{3,4}$
1,1	0	$q_{4,2}$	$q_{4,3}$	-

Now, BayesTraits can be use to calculate the fit of the two models and then compare them to select the best one.

Running the analysis

The idea is to run both models separately and compare their fit. In the Bayesian framework, one uses Bayes Factors to compare the models. Let's first fit the model. For this, we will use 500 trees sampled from the posterior distribution of trees. This has the advantage that the analysis will also integrate phylogenetic uncertainty in the model. This is especially important if the support for the groups in the tree are not all very strong. By doing so, the results obtained integrate over all possible tree topology and accounts for phylogenetic uncertainty.

We'll have to import these trees. Such posterior samples of trees can be obtained using Bayesian phylogenetic methods (BEAST, MrBayes).

```
pdtrees <- read.nexus("./data/pd_500.trees")
species.to.exclude <- pdtrees[[1]]$tip.label[!(pdtrees[[1]]$tip.label %in%
                                                rownames(seedplantsdata))]
pdtrees<-lapply(pdtrees,drop.tip,tip=species.to.exclude)
class(pdtrees)<-"multiPhylo"
attr(pdtrees,"TipLabel") <- pdtrees[[1]]$tip.label
rm(species.to.exclude)
# Need two binary variables
# Start by converting the height variable in a binary variable (0/1)
height2 <- as.numeric(seedplantsdata$height)
height2[height2==2] <- 3
height2[height2==3] <- 0
thedata<-data.frame(code=rownames(seedplantsdata),
           height=height2,ShadeTolerance=as.numeric(seedplantsdata$ShadeTol)-1)
```

Now, we can perform the BayesTraits analyses using this distribution of trees. For this, we will use the DiscreteMCMC function. The settings are as for above, with a flat prior. The independent and dependent models can be set using the parameter `dependent=FALSE` or `dependent=TRUE`.

```
# Fit independent model
ind.res1 <- DiscreteMCMC(pdtrees, thedata, dependent = FALSE, it = 100000, bi = 1000,
                         sa = 100, rd = 2, pa = "uniform 0 100", silent = TRUE)
ind.res2 <- DiscreteMCMC(pdtrees, thedata, dependent = FALSE, it = 100000, bi = 1000,
                         sa = 100, rd = 2, pa = "uniform 0 100", silent = TRUE)
# Fit dependent model
dep.res1 <- DiscreteMCMC(pdtrees, thedata, dependent = TRUE, it = 100000, bi = 1000,
                         sa = 100, rd = 2, pa = "uniform 0 100", silent = TRUE)
dep.res2 <- DiscreteMCMC(pdtrees, thedata, dependent = TRUE, it = 100000, bi = 1000,
                         sa = 100, rd = 2, pa = "uniform 0 100", silent = TRUE)
```

Now, let's read the results with the `coda` package.

```
require(coda)
require(lattice)
# Read the BayesTrait results of the independent model in coda format
ind1 <- mcmc(ind.res1$Results[,c(-1,-4)],start=min(ind.res1$Results$Iteration),
             end=max(ind.res1$Results$Iteration),thin=100)
ind2 <- mcmc(ind.res2$Results[,c(-1,-4)],start=min(ind.res2$Results$Iteration),
             end=max(ind.res2$Results$Iteration),thin=100)
# Combine the three chains
ind <- mcmc.list(ind1,ind2)
```

```
# Read the BayesTrait results of the independent model in coda format
dep1 <- mcmc(dep.res1$Results[,c(-1,-4)],start=min(dep.res1$Results$Iteration),
             end=max(dep.res1$Results$Iteration),thin=100)
dep2 <- mcmc(dep.res2$Results[,c(-1,-4)],start=min(dep.res2$Results$Iteration),
             end=max(dep.res2$Results$Iteration),thin=100)
# Combine the three chains
dep <- mcmc.list(dep1,dep2)
```

Now, we can look for convergence of the two sets of analyses

```
# Get effective sizes (should be > 200)
effectiveSize(ind)
```

```
##            Lh Harmonic.Mean           q12           q13           q21
##     144.47319      68.25323     308.70864     445.42356     389.48433
##           q24           q31           q34           q42           q43
##     445.42356     371.50398     308.70864     371.50398     389.48433
## Root...P.0.0. Root...P.0.1. Root...P.1.0. Root...P.1.1.
##     768.09519     773.10562     773.10347     768.09049
```

```
effectiveSize(dep)
```

```
##            Lh Harmonic.Mean           q12           q13           q21
##     136.56310      22.60326     266.34646     233.67964     305.19543
##           q24           q31           q34           q42           q43
##     238.72866     170.19597     233.35761     216.38000     208.86778
## Root...P.0.0. Root...P.0.1. Root...P.1.0. Root...P.1.1.
##    1004.13655    1000.21201    1019.04752     882.53535
```

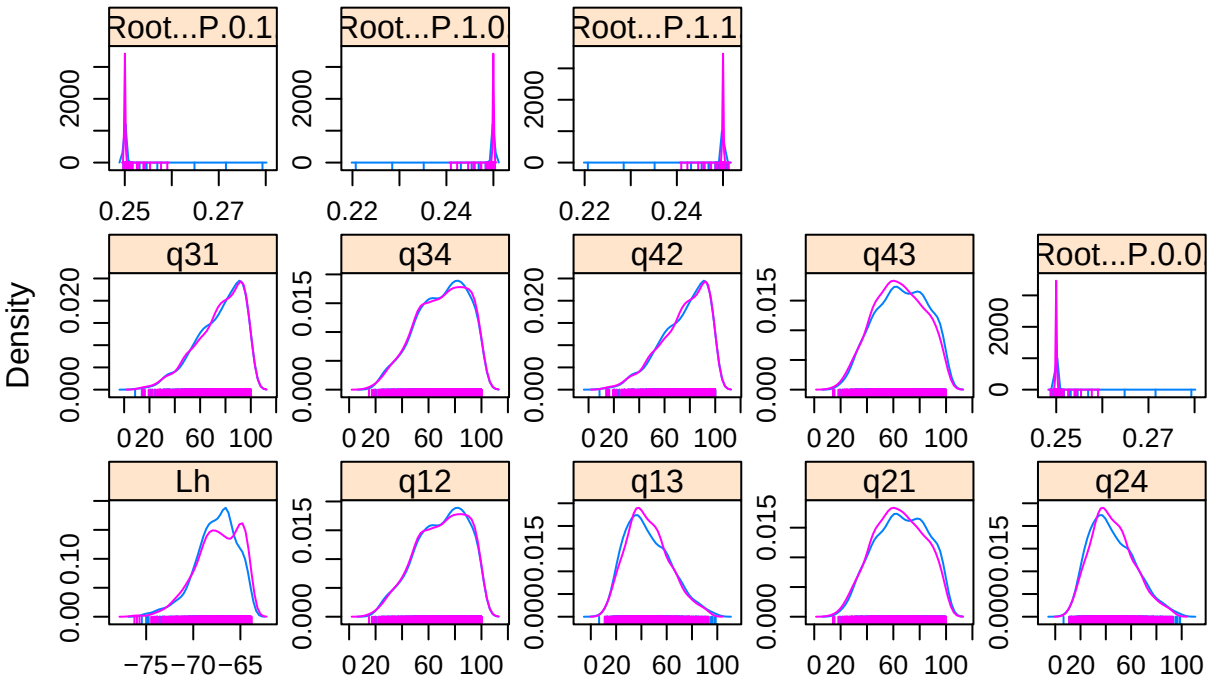
```
# Gelman and Rubin's convergence disgnostic
gelman.diag(ind,autoburnin=FALSE,multivariate=FALSE)
```

```
## Potential scale reduction factors:
##
##                Point est. Upper C.I.
## Lh                   1.01       1.02
## Harmonic.Mean        1.01       1.01
## q12                  1.00       1.00
## q13                  1.00       1.00
## q21                  1.00       1.02
## q24                  1.00       1.00
## q31                  1.00       1.00
## q34                  1.00       1.00
## q42                  1.00       1.00
## q43                  1.00       1.02
## Root...P.0.0.        1.20       1.21
## Root...P.0.1.        1.20       1.21
## Root...P.1.0.        1.20       1.21
## Root...P.1.1.        1.20       1.21
```

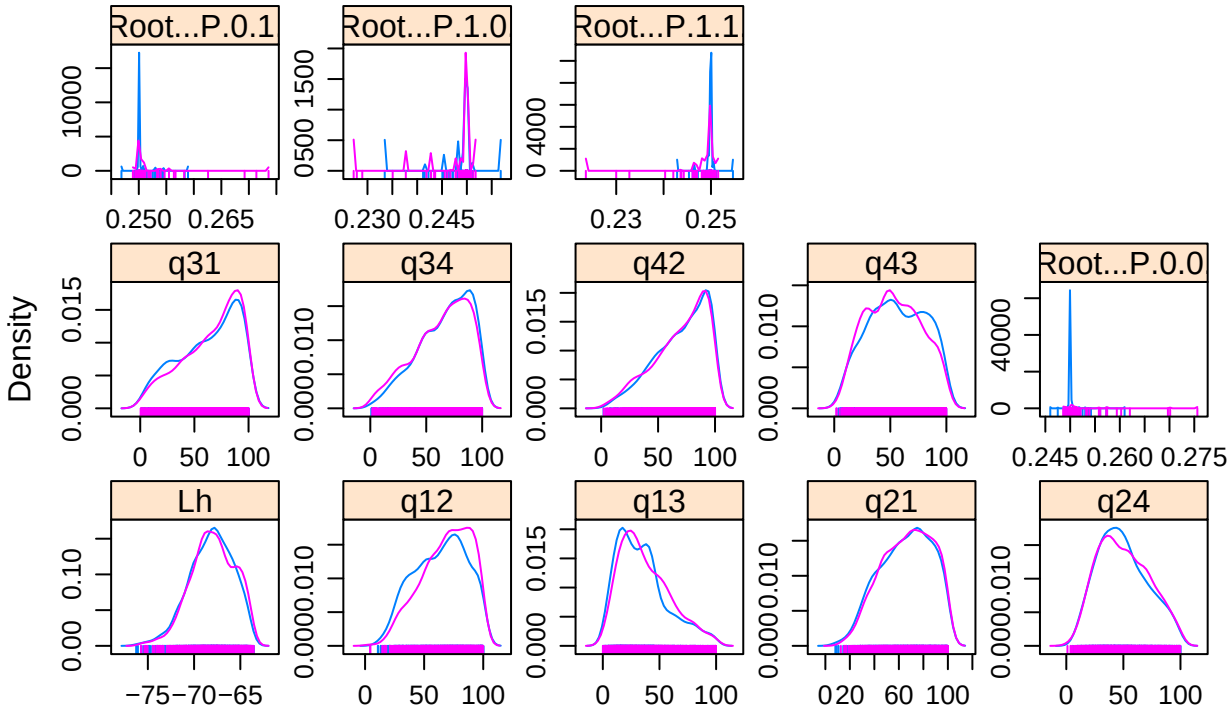
```
gelman.diag(dep,autoburnin=FALSE,multivariate=FALSE)
```

```
## Potential scale reduction factors:
##
##                 Point est. Upper C.I.
## Lh                    1.00       1.01
## Harmonic.Mean         1.01       1.03
## q12                   1.02       1.10
## q13                   1.01       1.03
## q21                   1.00       1.01
## q24                   1.00       1.00
## q31                   1.00       1.02
## q34                   1.01       1.03
## q42                   1.00       1.00
## q43                   1.01       1.02
## Root...P.0.0.         1.19       1.22
## Root...P.0.1.         1.20       1.24
## Root...P.1.0.         1.12       1.13
## Root...P.1.1.         1.23       1.31
```

```
# Density Plots
densityplot(ind[,-2])
```




```
densityplot(dep[,-2])
```




```
# Parameter summary
summary(ind)
```

```
##
## Iterations = 1100:1e+05
## Thinning interval = 100
## Number of chains = 2
## Sample size per chain = 990
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                     Mean        SD  Naive SE Time-series SE
## Lh             -67.4811 2.267e+00 5.095e-02      1.898e-01
## Harmonic.Mean -71.0564 5.420e-01 1.218e-02      7.153e-02
## q12            69.8105 1.915e+01 4.304e-01      1.090e+00
## q13            45.2592 1.724e+01 3.873e-01      8.177e-01
## q21            64.9837 1.907e+01 4.286e-01      9.661e-01
## q24            45.2592 1.724e+01 3.873e-01      8.177e-01
## q31            75.2184 1.825e+01 4.100e-01      9.478e-01
## q34            69.8105 1.915e+01 4.304e-01      1.090e+00
## q42            75.2184 1.825e+01 4.100e-01      9.478e-01
## q43            64.9837 1.907e+01 4.286e-01      9.661e-01
## Root...P.0.0.   0.2501 9.682e-04 2.176e-05      3.461e-05
## Root...P.0.1.   0.2501 9.678e-04 2.175e-05      3.445e-05
## Root...P.1.0.   0.2499 9.678e-04 2.175e-05      3.445e-05
## Root...P.1.1.   0.2499 9.682e-04 2.176e-05      3.461e-05
##
## 2. Quantiles for each variable:
##
##                    2.5%     25%     50%     75%     97.5%
```

```
## Lh            -72.5875 -68.89 -67.29 -65.69 -64.1258
## Harmonic.Mean -71.7799 -71.28 -71.05 -70.89 -69.8508
## q12            29.7903  55.50  71.91  85.52  98.8694
## q13            17.8751  32.43  42.89  56.57  83.3984
## q21            28.7689  50.48  64.96  80.07  97.3984
## q24            17.8751  32.43  42.89  56.57  83.3984
## q31            33.2227  63.14  78.66  90.52  98.9199
## q34            29.7903  55.50  71.91  85.52  98.8694
## q42            33.2227  63.14  78.66  90.52  98.9199
## q43            28.7689  50.48  64.96  80.07  97.3984
## Root...P.0.0.   0.2500   0.25   0.25   0.25   0.2502
## Root...P.0.1.   0.2500   0.25   0.25   0.25   0.2502
## Root...P.1.0.   0.2498   0.25   0.25   0.25   0.2500
## Root...P.1.1.   0.2498   0.25   0.25   0.25   0.2500
```

```
summary(dep)
```

```
##
## Iterations = 1100:1e+05
## Thinning interval = 100
## Number of chains = 2
## Sample size per chain = 990
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                   Mean        SD  Naive SE Time-series SE
## Lh            -68.0716 2.351e+00 5.283e-02      2.038e-01
## Harmonic.Mean -71.0905 3.618e-01 8.131e-03      9.942e-02
## q12            66.1112 2.131e+01 4.789e-01      1.297e+00
## q13            36.1986 2.222e+01 4.994e-01      1.453e+00
## q21            66.2680 2.054e+01 4.615e-01      1.176e+00
## q24            49.9148 2.186e+01 4.913e-01      1.420e+00
## q31            64.2013 2.670e+01 6.001e-01      2.069e+00
## q34            64.8739 2.399e+01 5.392e-01      1.568e+00
## q42            69.8993 2.291e+01 5.148e-01      1.559e+00
## q43            54.5991 2.457e+01 5.521e-01      1.718e+00
## Root...P.0.0.   0.2501 1.096e-03 2.463e-05      3.557e-05
## Root...P.0.1.   0.2501 1.026e-03 2.306e-05      3.292e-05
## Root...P.1.0.   0.2499 1.189e-03 2.671e-05      3.852e-05
## Root...P.1.1.   0.2499 9.848e-04 2.213e-05      3.141e-05
##
## 2. Quantiles for each variable:
##
##                   2.5%    25%    50%    75%    97.5%
## Lh            -73.1018 -69.64 -67.97 -66.32 -64.1781
## Harmonic.Mean -71.7112 -71.29 -71.10 -70.90 -70.4739
## q12            24.7641  50.45  68.59  83.32  98.3884
## q13             5.1263  18.68  32.18  48.97  88.8117
## q21            26.3449  50.51  67.76  83.27  98.4334
## q24            13.6725  32.96  47.51  65.95  94.0125
## q31             8.3441  44.27  69.47  87.17  98.7546
## q34            11.9933  48.80  69.15  85.19  98.2954
## q42            18.9107  54.11  74.72  89.73  99.0225
```

```
## q43            11.6623  34.53  53.80  74.84  97.2785
## Root...P.0.0.   0.2499   0.25   0.25   0.25   0.2509
## Root...P.0.1.   0.2500   0.25   0.25   0.25   0.2507
## Root...P.1.0.   0.2491   0.25   0.25   0.25   0.2500
## Root...P.1.1.   0.2493   0.25   0.25   0.25   0.2501
```

You can see that the chains have converged well for both models.

Bayes Factors

To calculate whether there is support for the more complex correlated model, we will use Bayes Factors, which is common for Bayesian analyses. The Bayes Factor (BF) can be calculated the following way:

$$2lnBF = 2(lnL_{complex\ model} - lnL_{simpler\ model})$$

To calculate the BF, it is common to use the harmonic mean of the likelihood of each run (but see below). For this, you only use the last value from the complete run.

```
# Harmonic mean of the independent model
(ind_harm<-ind.res1$Results$Harmonic.Mean[length(ind.res1$Results$Harmonic.Mean)])
```

```
## [1] -70.77094
```

```
# Harmonic mean of the dependent model
(dep_harm<-dep.res1$Results$Harmonic.Mean[length(ind.res1$Results$Harmonic.Mean)])
```

```
## [1] -71.37005
```

```
# Bayes Factor
BF = 2*(dep_harm-ind_harm)
BF
```

```
## [1] -1.198204
```

Following Kass and Raftery (1995), BayesFactors can be interpreted the following way:

2 ln BF	Interpretation
0 to 2	Not worth more than a mention
2 to 6	Positive evidence
6 to 10	Strong evidence
> 10	Very strong evidence

Consequently, you can see that with the present case, there is not support for the more complex model.

> The harmonic mean is not a very good estimator of the likelihood of the model and many suggest it should not be used. BayesTraits has a stepping stone function to better estimate the likelihood and if you plan publishing using Bayes Factors, this is what you should use. Unfortunately, there is no wrapper yet to run this function from R.

Assignment

No assignment today!

References

Kass R.E., A.E. Raftery. 1995. Bayes factor. *Journal of the American Statistical Association* 90:773–795.

Pagel M. 1994. Detecting Correlated Evolution on Phylogenies: A General Method for the Comparative Analysis of Discrete Characters. *Proceedings of the Royal Society B* 255:37–45.

Pagel M., A. Meade, D. Barker. 2004. Bayesian estimation of ancestral character states on phylogenies. *Systematic Biology*. 53:673–684.

Paquette A., S. Joly, C. Messier. 2015. Explaining forest productivity using tree functional traits and phylogenetic information: two sides of the same coin over evolutionary scale? *Ecology and Evolution* 5:1774–1783.