

Bayesian ancestral state reconstruction

Simon Joly

BIO 6008 - Fall 2015

Contents

Bayesian ancestral state reconstructions	1
Multistate reconstruction	2
Running the analysis	2
Bayesian analysis diagnostics	4
Corelated evolution between binary traits in BayesTraits	12
Independent model	12
Dependent model	13
Running the analysis	13
Bayes Factors	19
Assignment	20
References	20

Bayesian ancestral state reconstructions

In the previous lecture, we saw how to reconstruct ancestral state reconstruction using maximum likelihood or stochastic 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. Conveniently, 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:

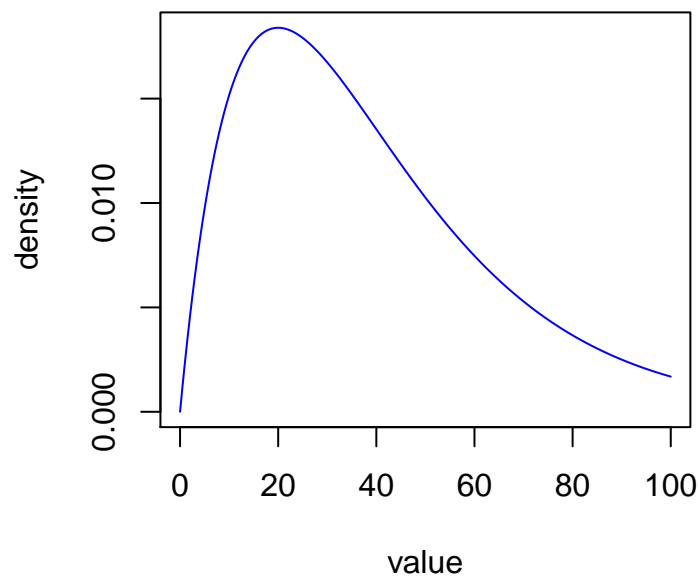
Parameter	Description
res	character or vector indicating restrictions to place on rates. If a vector is given, each element indicates an index
resall	character indicating a rate or a non-negative number to restrict all rates to.
mrca	character or vector indicating nodes to reconstruct using the most recent common ancestor approach. If a vector is given, each element indicates a node
fo	character or vector indicating nodes to fix at particular states. If a vector is given, each element is a character state
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 character string
pa	character string specifying the prior distribution for all parameters by listing first the name of the parameter, then the prior distribution
rj	toggles reversible jump model if non-empty. A character string specifying the prior distribution and it's parameters
rjhp	toggles reversible jump model with a hyper-prior if non-empty. A character string specifying the prior distribution and hyper-prior
hp	character or vector describing prior distributions and hyper-priors for model parameters. If a vector is given, each element is a character string
hpall	character string specifying the prior distribution and hyper-prior for all parameters by listing first the name of the parameter, then the prior distribution

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)

```

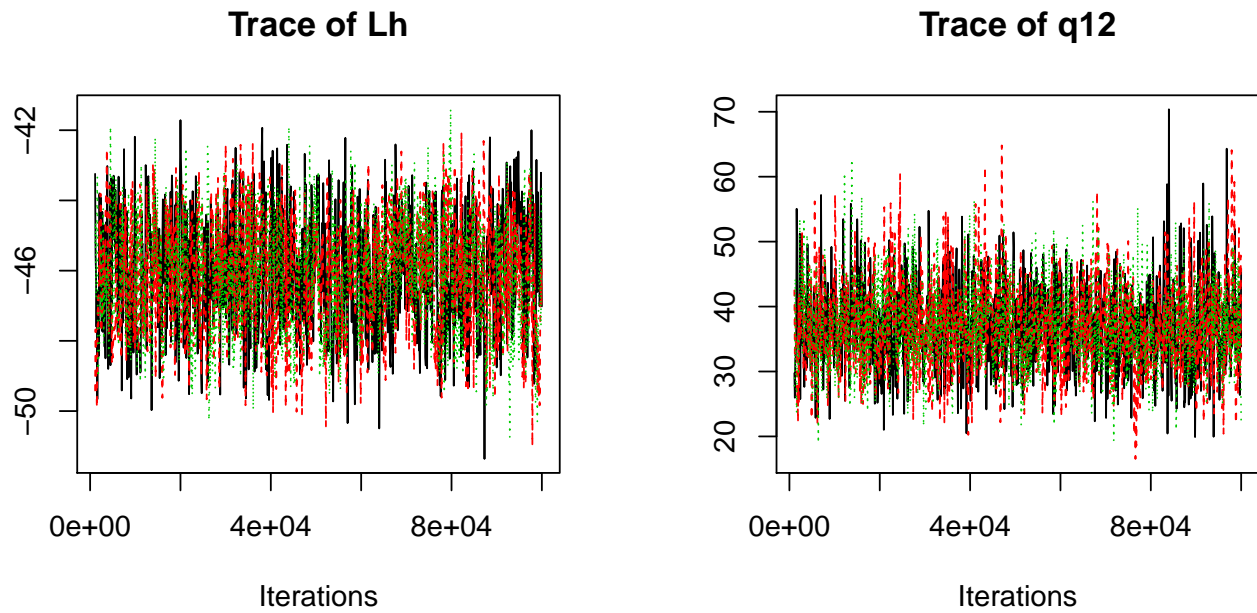
Trace plots

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)])

```



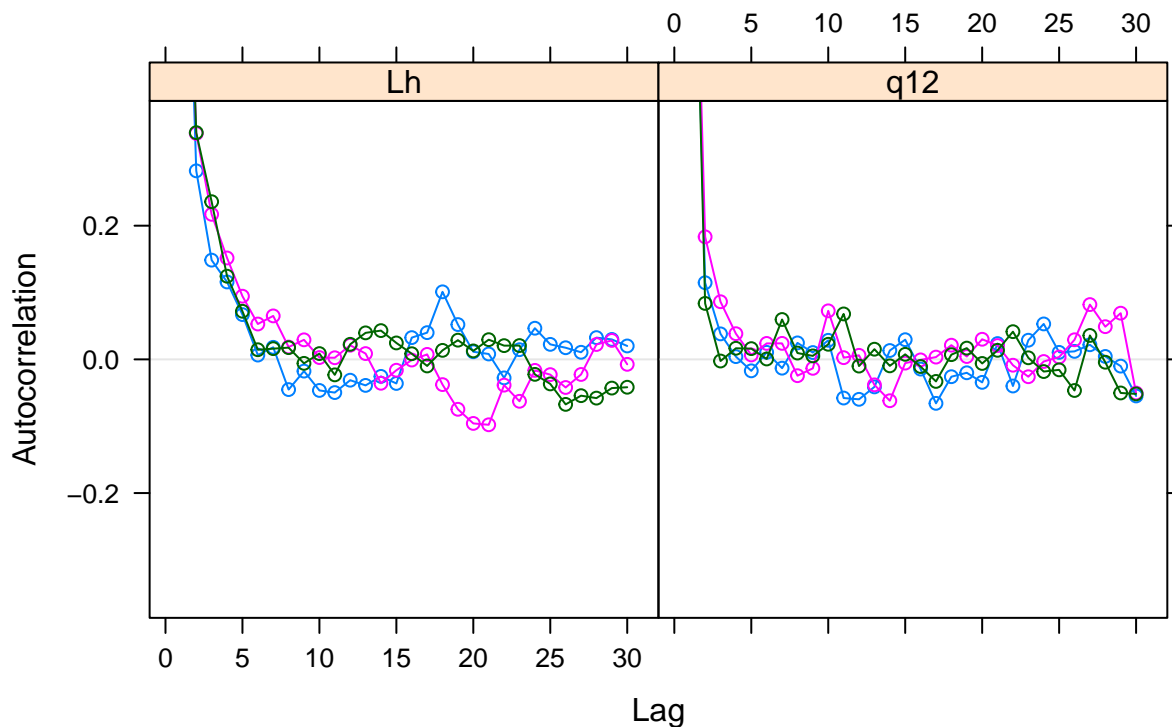
```
par(op)
```

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 diagnostics. 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 remaining 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
## 1138.0445    367.4877    2233.0336    1177.7616    1177.7616
##           q23           q31           q32    Root.P.1.    Root.P.2.
##  863.0361    2233.0336    863.0361    2308.3274    2346.3270
##    Root.P.3.
## 2205.3160
```

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.

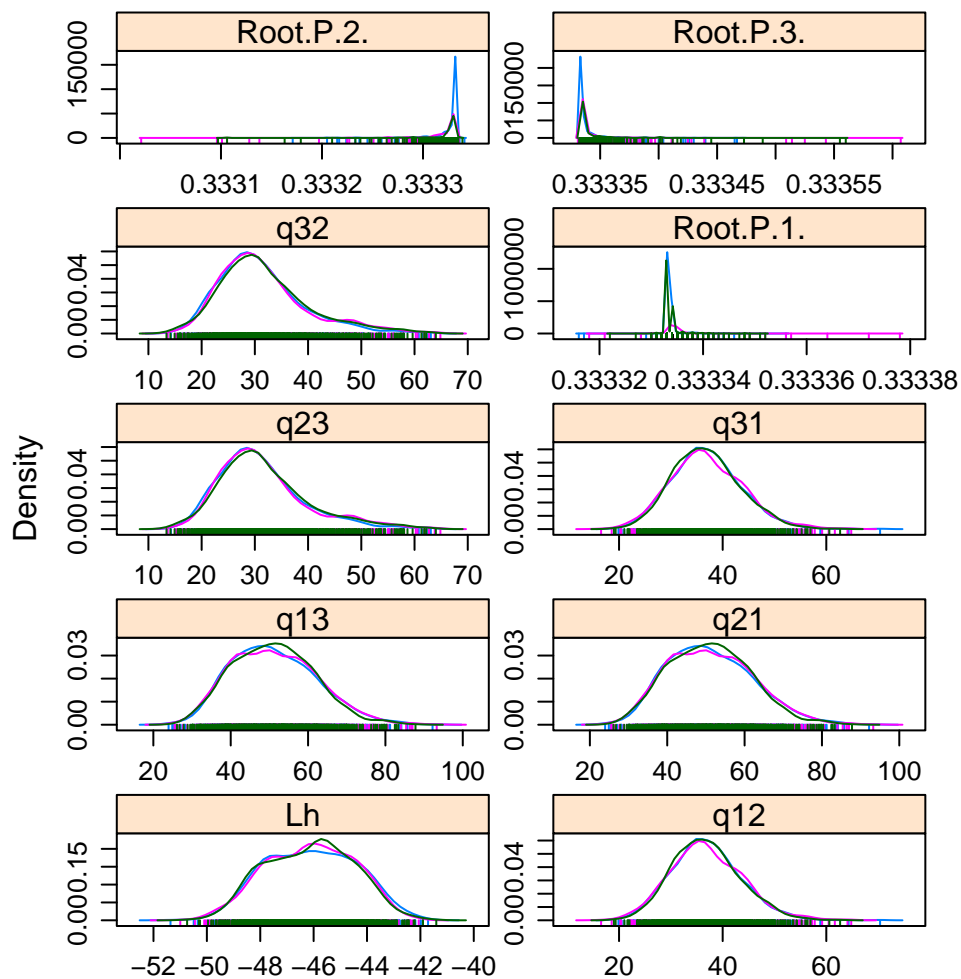
```
# Gelman and Rubin's convergence diagnostic
gelman.diag(res, autoburnin=FALSE, multivariate=FALSE)
```

```
## Potential scale reduction factors:
##
##           Point est. Upper C.I.
## Lh           1.00      1.00
## Harmonic.Mean 1.13      1.41
## q12           1.00      1.00
## q13           1.00      1.00
## q21           1.00      1.00
## q23           1.00      1.01
## q31           1.00      1.00
## q32           1.00      1.01
## Root.P.1.     1.03      1.03
## Root.P.2.     1.02      1.02
## Root.P.3.     1.02      1.02
```

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             -46.1194 1.660e+00 3.046e-02 4.931e-02
## Harmonic.Mean -47.3363 1.979e-01 3.631e-03 1.504e-02
## q12            36.8855 6.754e+00 1.239e-01 1.450e-01
## q13            51.1744 1.086e+01 1.992e-01 3.178e-01
## q21            51.1744 1.086e+01 1.992e-01 3.178e-01
## q23            31.5102 8.351e+00 1.532e-01 2.888e-01
## q31            36.8855 6.754e+00 1.239e-01 1.450e-01
## q32            31.5102 8.351e+00 1.532e-01 2.888e-01
## Root.P.1.      0.3333 2.327e-06 4.270e-08 4.936e-08
## Root.P.2.      0.3333 1.712e-05 3.142e-07 3.575e-07
## Root.P.3.      0.3333 1.545e-05 2.835e-07 3.297e-07
##
## 2. Quantiles for each variable:
##
##              2.5%      25%      50%      75%      97.5%
## Lh             -49.1823 -47.4301 -46.0681 -44.8457 -43.1174
## Harmonic.Mean -47.6211 -47.3777 -47.3318 -47.2922 -47.1022
## q12            24.8619  32.1661  36.4506  41.1692  51.2682
## q13            32.4843  42.9056  50.3869  58.5146  74.0144
## q21            32.4843  42.9056  50.3869  58.5146  74.0144
## q23            19.1070  25.8529  30.0972  35.4529  52.8376
## q31            24.8619  32.1661  36.4506  41.1692  51.2682
## q32            19.1070  25.8529  30.0972  35.4529  52.8376
## 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             -49.316206 -43.188504
## Harmonic.Mean -47.577252 -47.286384
## q12            24.160816  49.735782
## q13            31.334100  72.320082
## q21            31.334100  72.320082
## q23            17.984193  47.862917
## q31            24.160816  49.735782
## q32            17.984193  47.862917
```



```
## Root.P.1.      0.333333  0.333337
## Root.P.2.      0.333307  0.333333
## Root.P.3.      0.333332  0.333353
## attr("Probability")
## [1] 0.9494949
##
## [[2]]
##              lower      upper
## Lh            -49.029249 -43.000822
## Harmonic.Mean -47.647020 -47.242386
## q12            22.035225  49.395146
## q13            31.162741  71.726894
## q21            31.162741  71.726894
## q23            17.330880  49.912151
## q31            22.035225  49.395146
## q32            17.330880  49.912151
## Root.P.1.      0.333332  0.333337
## Root.P.2.      0.333311  0.333335
## Root.P.3.      0.333331  0.333351
## attr("Probability")
## [1] 0.9494949
##
## [[3]]
##              lower      upper
## Lh            -49.060614 -43.252369
## Harmonic.Mean -47.358254 -47.076826
## q12            24.809747  49.675654
## q13            31.710658  69.977005
## q21            31.710658  69.977005
## q23            15.505196  49.121885
## q31            24.809747  49.675654
## q32            15.505196  49.121885
## Root.P.1.      0.333330  0.333336
## Root.P.2.      0.333305  0.333334
## 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
## 34.480617 11.771899 26.289773 6.705132 6.705132
##          q23          q31          q32  Root.P.1.  Root.P.2.
## 79.441516 26.289773 79.441516 1501.634948 1752.841249
##  Root.P.3.
## 1757.998364

```

```

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

```

```

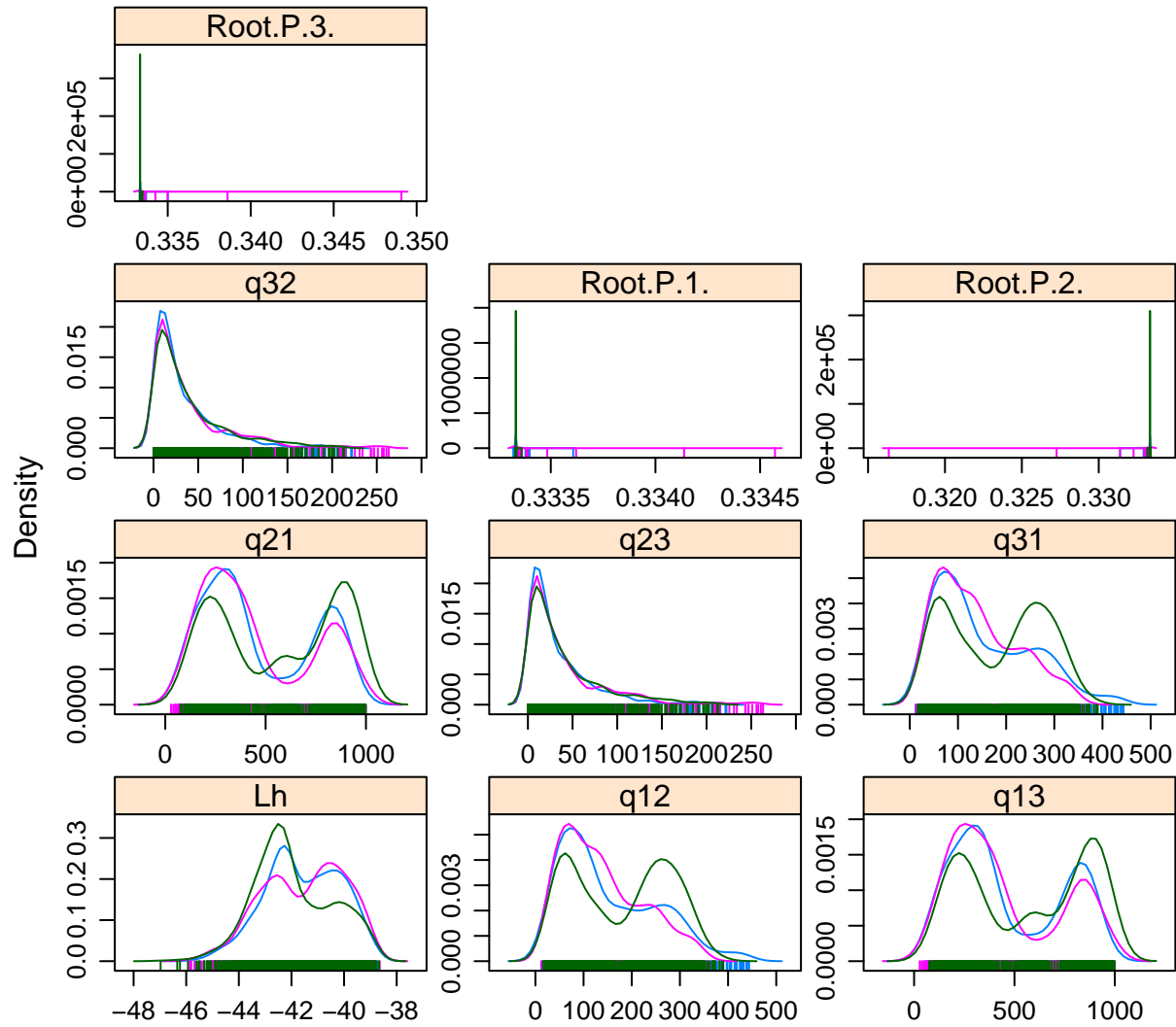
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## Lh          1.02      1.06
## Harmonic.Mean 1.22      1.66
## q12          1.04      1.13
## q13          1.04      1.12
## q21          1.04      1.12
## q23          1.01      1.01
## q31          1.04      1.13
## q32          1.01      1.01
## Root.P.1.    1.27      1.33
## Root.P.2.    1.28      1.34
## Root.P.3.    1.29      1.34

```

```

# 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          -41.6821 1.497e+00 2.747e-02    2.754e-01
## Harmonic.Mean -41.9297 6.664e-01 1.223e-02    2.358e-01
## q12          156.6974 9.693e+01 1.779e+00    1.942e+01
## q13          494.7281 2.888e+02 5.299e+00    1.128e+02
## q21          494.7281 2.888e+02 5.299e+00    1.128e+02
## q23           37.5943 4.167e+01 7.645e-01    4.727e+00
## q31          156.6974 9.693e+01 1.779e+00    1.942e+01
```

```
## q32          37.5943 4.167e+01 7.645e-01      4.727e+00
## Root.P.1.    0.3333 2.829e-05 5.192e-07      8.549e-07
## Root.P.2.    0.3333 3.357e-04 6.160e-06      8.336e-06
## Root.P.3.    0.3333 3.085e-04 5.660e-06      7.591e-06
##
## 2. Quantiles for each variable:
##
##           2.5%      25%      50%      75%      97.5%
## Lh          -44.5583 -42.7616 -41.8530 -40.4264 -39.1095
## Harmonic.Mean -42.9939 -42.5133 -41.8542 -41.4799 -40.4766
## q12          27.6637  71.3503 131.6632 243.0298 339.5339
## q13          101.3750 243.9400 389.3513 807.2932 963.4725
## q21          101.3750 243.9400 389.3513 807.2932 963.4725
## q23           1.2341   9.3068  22.7443  49.6963 155.1684
## q31          27.6637  71.3503 131.6632 243.0298 339.5339
## q32           1.2341   9.3068  22.7443  49.6963 155.1684
## 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 used 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 dependent 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
##      164.14344      16.71548      342.90645      412.42237      357.72912
##           q24           q31           q34           q42           q43
##      412.42237      374.25951      342.90645      374.25951      357.72912
## Root...P.0.0. Root...P.0.1. Root...P.1.0. Root...P.1.1.
##      1516.81754      1523.77962      1523.78272      1516.81754
```

```
effectiveSize(dep)
```

```
##           Lh Harmonic.Mean           q12           q13           q21
##       87.74305      18.70751      234.76651      245.33978      260.18121
##           q24           q31           q34           q42           q43
##      252.74302      192.05650      197.24660      215.08936      224.91006
## Root...P.0.0. Root...P.0.1. Root...P.1.0. Root...P.1.1.
##      1182.88776      1200.48994      1155.14842      1351.19982
```

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

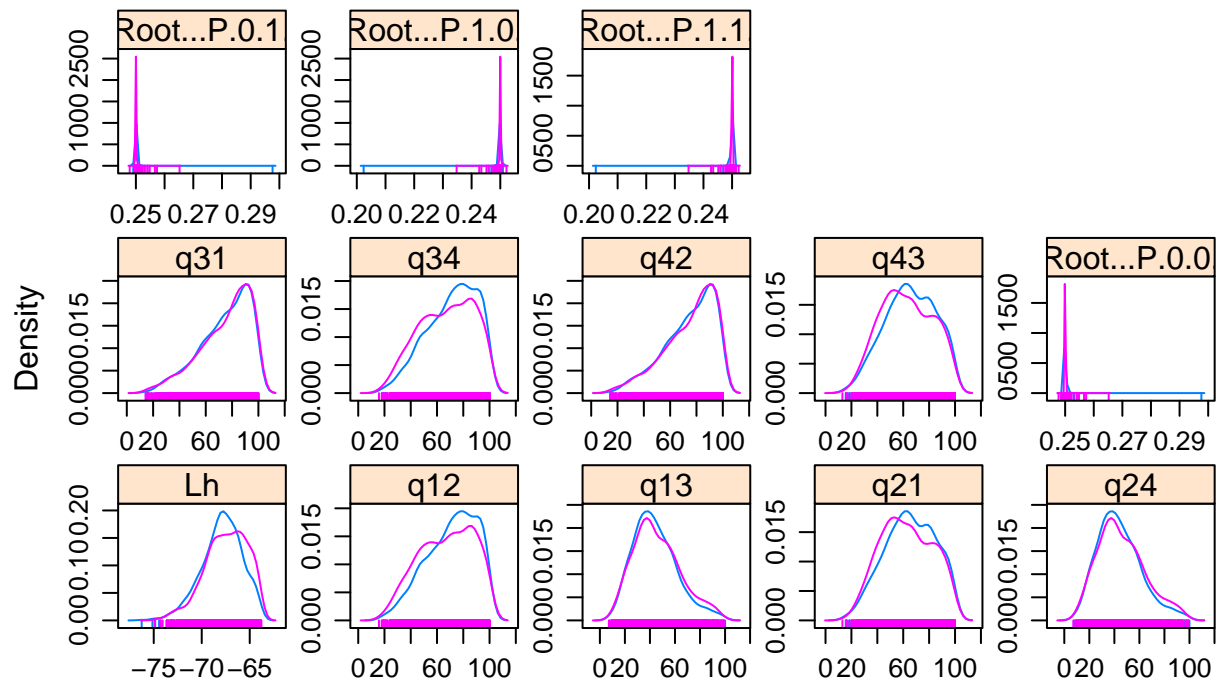
```
## Potential scale reduction factors:
##
##           Point est. Upper C.I.
## Lh           1.02      1.08
## Harmonic.Mean 1.12      1.43
## q12           1.02      1.09
## q13           1.01      1.02
## q21           1.01      1.04
## q24           1.01      1.02
## q31           1.00      1.00
## q34           1.02      1.09
## q42           1.00      1.00
## q43           1.01      1.04
## Root...P.0.0. 1.18      1.18
## Root...P.0.1. 1.18      1.18
## Root...P.1.0. 1.18      1.18
## Root...P.1.1. 1.18      1.18
```

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

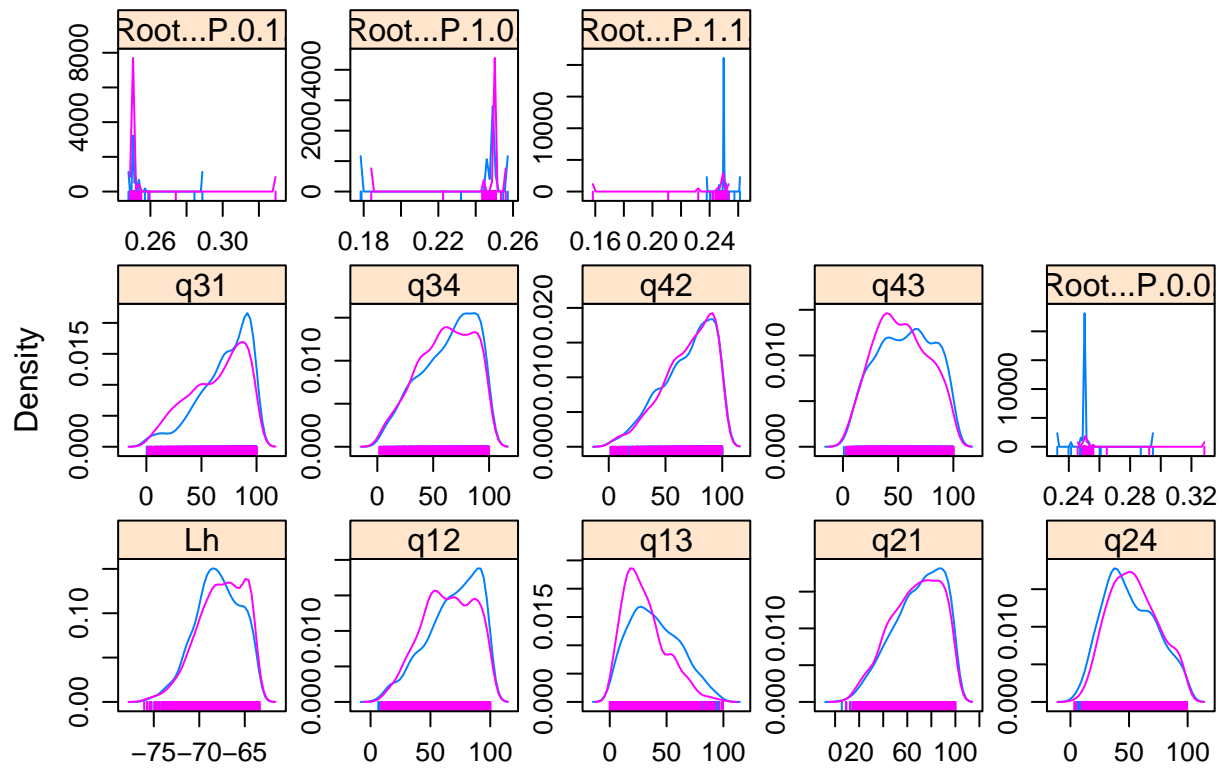
```
## Potential scale reduction factors:
##
##           Point est. Upper C.I.
## Lh           1.01      1.05
## Harmonic.Mean 1.40      3.14
## q12           1.02      1.08
## q13           1.09      1.35
## q21           1.00      1.03
## q24           1.01      1.04
```

## q31	1.03	1.14
## q34	1.00	1.02
## q42	1.00	1.00
## q43	1.01	1.05
## Root...P.0.0.	1.05	1.06
## Root...P.0.1.	1.08	1.08
## Root...P.1.0.	1.05	1.05
## Root...P.1.1.	1.26	1.33

```
# 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          -67.6786   2.148384 4.828e-02    1.694e-01
## Harmonic.Mean -70.6276   0.526004 1.182e-02    1.288e-01
## q12          69.1093  19.811634 4.452e-01    1.068e+00
## q13          45.0115  18.237834 4.099e-01    8.988e-01
## q21          63.8776  19.510321 4.385e-01    1.033e+00
## q24          45.0115  18.237834 4.099e-01    8.988e-01
## q31          74.7555  19.048815 4.281e-01    9.908e-01
## q34          69.1093  19.811634 4.452e-01    1.068e+00
## q42          74.7555  19.048815 4.281e-01    9.908e-01
## q43          63.8776  19.510321 4.385e-01    1.033e+00
## Root...P.0.0.   0.2501   0.001175 2.640e-05    2.896e-05
## Root...P.0.1.   0.2501   0.001174 2.639e-05    2.893e-05
## Root...P.1.0.   0.2499   0.001174 2.639e-05    2.893e-05
## Root...P.1.1.   0.2499   0.001175 2.640e-05    2.896e-05
##
## 2. Quantiles for each variable:
```

```
##
##          2.5%    25%    50%    75%    97.5%
## Lh        -72.4405 -69.00 -67.57 -66.08 -64.1686
## Harmonic.Mean -71.3514 -70.95 -70.70 -70.32 -69.2547
## q12        29.1818  54.37  71.34  85.59  98.7065
## q13        15.9593  31.90  42.39  56.26  87.6402
## q21        27.0116  49.32  63.71  79.69  97.5453
## q24        15.9593  31.90  42.39  56.26  87.6402
## q31        30.5138  62.73  78.84  90.33  99.0611
## q34        29.1818  54.37  71.34  85.59  98.7065
## q42        30.5138  62.73  78.84  90.33  99.0611
## q43        27.0116  49.32  63.71  79.69  97.5453
## Root...P.0.0.  0.2500  0.25  0.25  0.25  0.2503
## Root...P.0.1.  0.2500  0.25  0.25  0.25  0.2505
## Root...P.1.0.  0.2495  0.25  0.25  0.25  0.2500
## Root...P.1.1.  0.2497  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        -67.7906   2.517240 5.657e-02    2.827e-01
## Harmonic.Mean -70.9905   0.436637 9.813e-03    1.294e-01
## q12        66.7586  21.676776 4.871e-01    1.407e+00
## q13        35.4001  21.079065 4.737e-01    1.321e+00
## q21        68.0117  20.566248 4.622e-01    1.280e+00
## q24        52.5255  21.584768 4.851e-01    1.371e+00
## q31        67.3054  24.653718 5.541e-01    1.782e+00
## q34        62.0005  24.561308 5.520e-01    1.752e+00
## q42        69.3713  22.710446 5.104e-01    1.554e+00
## q43        54.5774  24.602456 5.529e-01    1.637e+00
## Root...P.0.0.  0.2501  0.002519 5.662e-05    8.472e-05
## Root...P.0.1.  0.2502  0.002241 5.037e-05    7.160e-05
## Root...P.1.0.  0.2498  0.002854 6.414e-05    8.384e-05
## Root...P.1.1.  0.2499  0.002359 5.301e-05    8.646e-05
##
## 2. Quantiles for each variable:
##
##          2.5%    25%    50%    75%    97.5%
## Lh        -73.1624 -69.47 -67.69 -65.76 -63.9364
## Harmonic.Mean -71.5385 -71.23 -71.13 -70.91 -69.6970
## q12        19.9765  51.52  69.03  85.10  98.3807
## q13         4.3070  18.90  31.90  50.02  82.2179
## q21        24.5027  52.99  70.48  85.39  98.3396
## q24        16.1279  35.97  50.62  68.29  95.2071
## q31        10.9480  50.40  72.49  88.21  98.6546
```

```
## q34          11.0096  43.35  64.83  82.80  97.5915
## q42          18.5020  54.64  73.71  88.49  99.0081
## q43          10.9246  35.33  54.58  74.25  97.2360
## Root...P.0.0.  0.2499   0.25   0.25   0.25   0.2505
## Root...P.0.1.  0.2500   0.25   0.25   0.25   0.2506
## Root...P.1.0.  0.2491   0.25   0.25   0.25   0.2500
## Root...P.1.1.  0.2496   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:

$$2\ln BF = 2(\ln L_{\text{complex model}} - \ln L_{\text{simpler model}})$$

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] -70.76578
```

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

```
## [1] -71.14205
```

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

```
## [1] -0.75253
```

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

Please reconstruct the ancestral states of one character on your tree using the method of your choice. Present the analyses and the results in a R Markdown document.

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.