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:

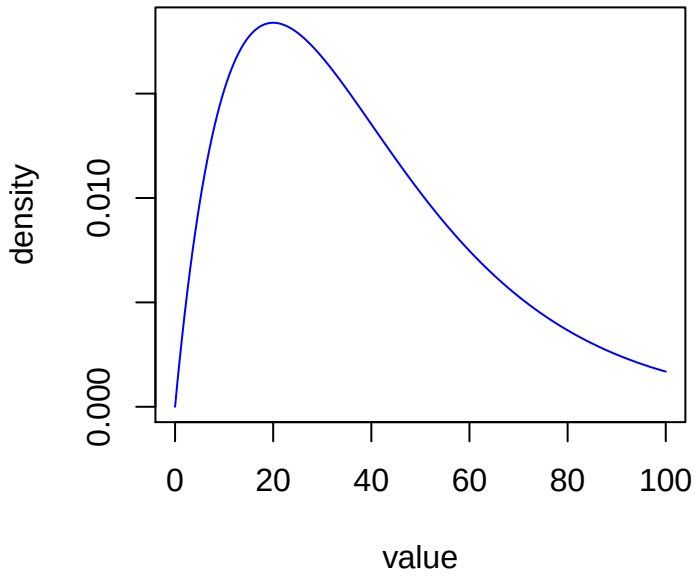
Parameter	Description
res	character or vector indicating restrictions to place on rates. If a vector is given, each element indicates an inde
resall	character indicating a rate or a non-negative number to restrict all rates to.
mrca	character or vector indicating nodes to reconstruct using the most recent 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

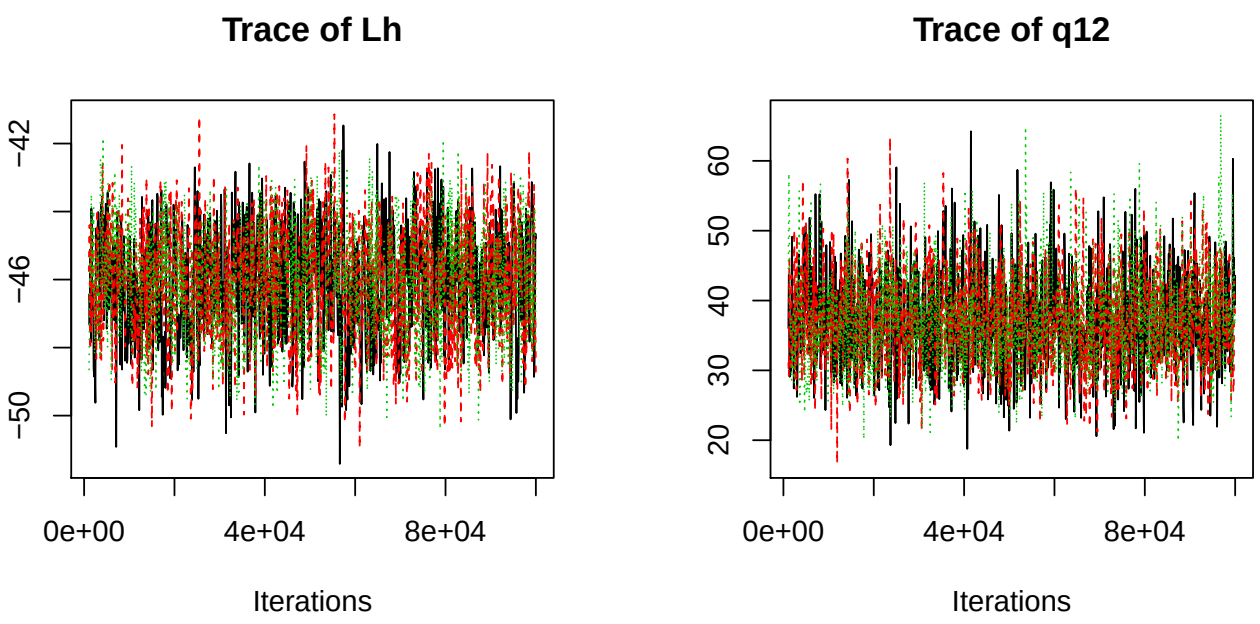
For diagnostic of Bayesian MCMC analyses, the package `coda` is very useful to look for chain convergence and calculate statistics. To be able to estimate convergence statistics, it is important to run at least 2 independent chains. This should be standard anyway to ensure convergence and thus that the 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$Iteration),
             end=max(multistate.MCMC.res3$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)])
```



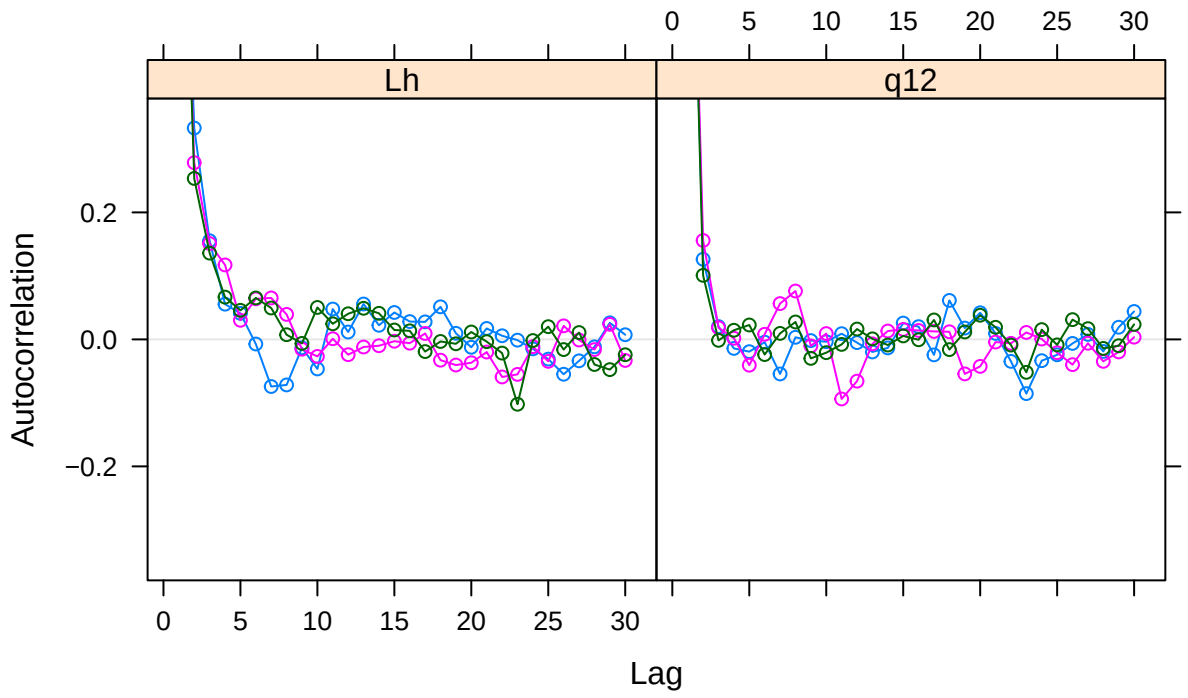

```
par(op)
```

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
##     1412.1418       75.8133     2332.0987     1370.3311     1370.3311
##           q23           q31           q32     Root.P.1.     Root.P.2.
##     1160.5912     2332.0987     1160.5912     2355.2098     2582.5137
##     Root.P.3.
##     2524.7686
```

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)
```

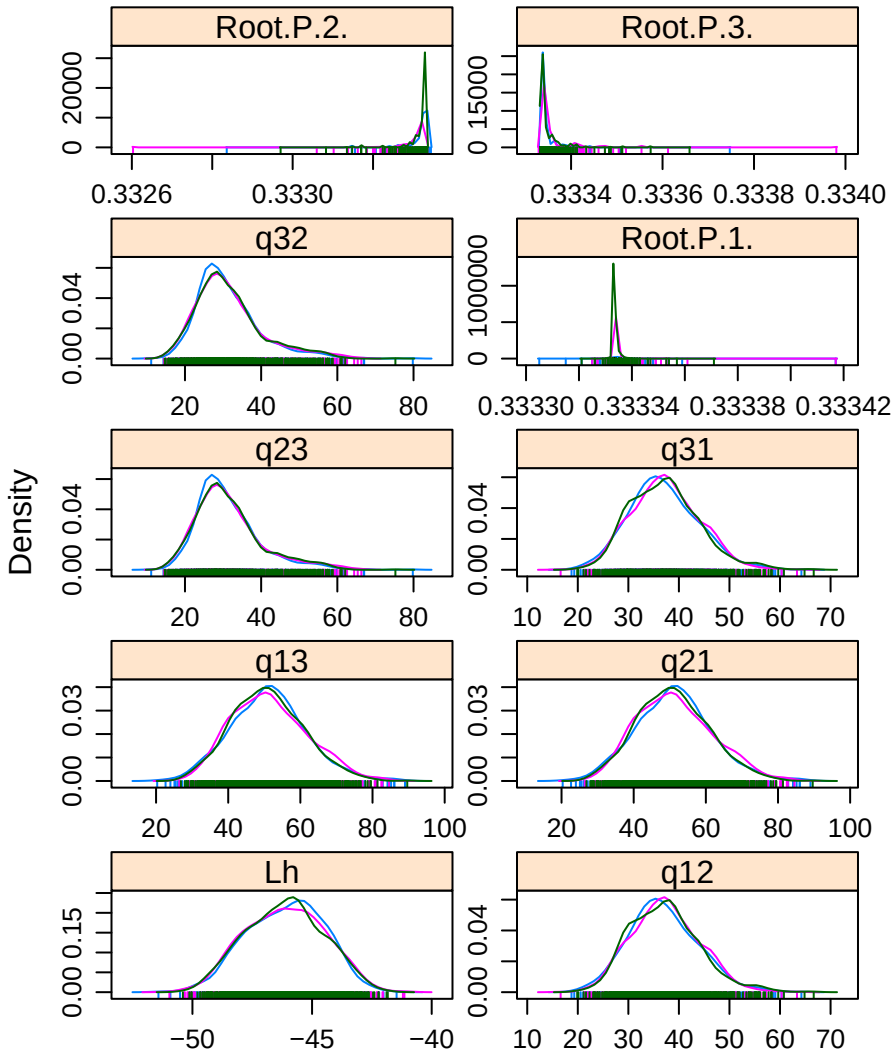
```
## Potential scale reduction factors:
```

```
##
##                 Point est. Upper C.I.
## Lh                    1.00       1.00
## Harmonic.Mean         1.41       2.23
## q12                   1.00       1.00
## q13                   1.00       1.00
## q21                   1.00       1.00
## q23                   1.00       1.00
## q31                   1.00       1.00
## q32                   1.00       1.00
## Root.P.1.             1.02       1.02
## Root.P.2.             1.03       1.03
## Root.P.3.             1.04       1.04
```

Density plots

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

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



```r
# 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             -46.1236 1.627e+00 2.986e-02      4.351e-02
## Harmonic.Mean -47.3110 1.939e-01 3.558e-03      4.518e-02
## q12            37.1792 6.822e+00 1.252e-01      1.413e-01
## q13            51.2271 1.033e+01 1.896e-01      2.803e-01
## q21            51.2271 1.033e+01 1.896e-01      2.803e-01
## q23            31.4778 8.719e+00 1.600e-01      2.594e-01
## q31            37.1792 6.822e+00 1.252e-01      1.413e-01
## q32            31.4778 8.719e+00 1.600e-01      2.594e-01
## Root.P.1.       0.3333 3.192e-06 5.858e-08      6.550e-08
## Root.P.2.       0.3333 2.517e-05 4.618e-07      4.981e-07
## Root.P.3.       0.3333 2.274e-05 4.172e-07      4.600e-07
##
## 2. Quantiles for each variable:
##
##                    2.5%      25%      50%      75%    97.5%
## Lh             -49.2245 -47.3376 -46.0729 -44.9554 -43.1323
## Harmonic.Mean -47.5415 -47.4162 -47.3409 -47.2902 -46.7293
## q12            25.3892  32.4744  36.8237  41.5310  51.3589
## q13            32.7544  43.9004  50.7917  57.8294  72.8183
## q21            32.7544  43.9004  50.7917  57.8294  72.8183
## q23            18.7282  25.6041  29.8007  35.4955  54.3993
## q31            25.3892  32.4744  36.8237  41.5310  51.3589
## q32            18.7282  25.6041  29.8007  35.4955  54.3993
## 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
```

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

```
## [[1]]
##                     lower      upper
## Lh             -48.996878 -43.165809
## Harmonic.Mean -47.608705 -47.338061
## q12            23.021329  49.639770
## q13            31.690352  72.956165
## q21            31.690352  72.956165
## q23            15.911592  48.051960
## q31            23.021329  49.639770
## q32            15.911592  48.051960
```

```
## Root.P.1.       0.333332   0.333337
## Root.P.2.       0.333311   0.333334
## Root.P.3.       0.333332   0.333353
## attr(,"Probability")
## [1] 0.9494949
##
## [[2]]
##                     lower      upper
## Lh             -49.254204 -43.200092
## Harmonic.Mean -47.430681 -46.921486
## q12             25.048473  49.594601
## q13             33.471411  72.818220
## q21             33.471411  72.818220
## q23             15.597003  50.930377
## q31             25.048473  49.594601
## q32             15.597003  50.930377
## Root.P.1.       0.333333   0.333337
## Root.P.2.       0.333306   0.333334
## Root.P.3.       0.333333   0.333359
## attr(,"Probability")
## [1] 0.9494949
##
## [[3]]
##                     lower      upper
## Lh             -49.344520 -43.283237
## Harmonic.Mean -47.368667 -46.691168
## q12             25.232160  50.992961
## q13             32.689210  70.929942
## q21             32.689210  70.929942
## q23             16.265043  50.467956
## q31             25.232160  50.992961
## q32             16.265043  50.467956
## Root.P.1.       0.333331   0.333337
## Root.P.2.       0.333305   0.333334
## Root.P.3.       0.333333   0.333359
## 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),
```

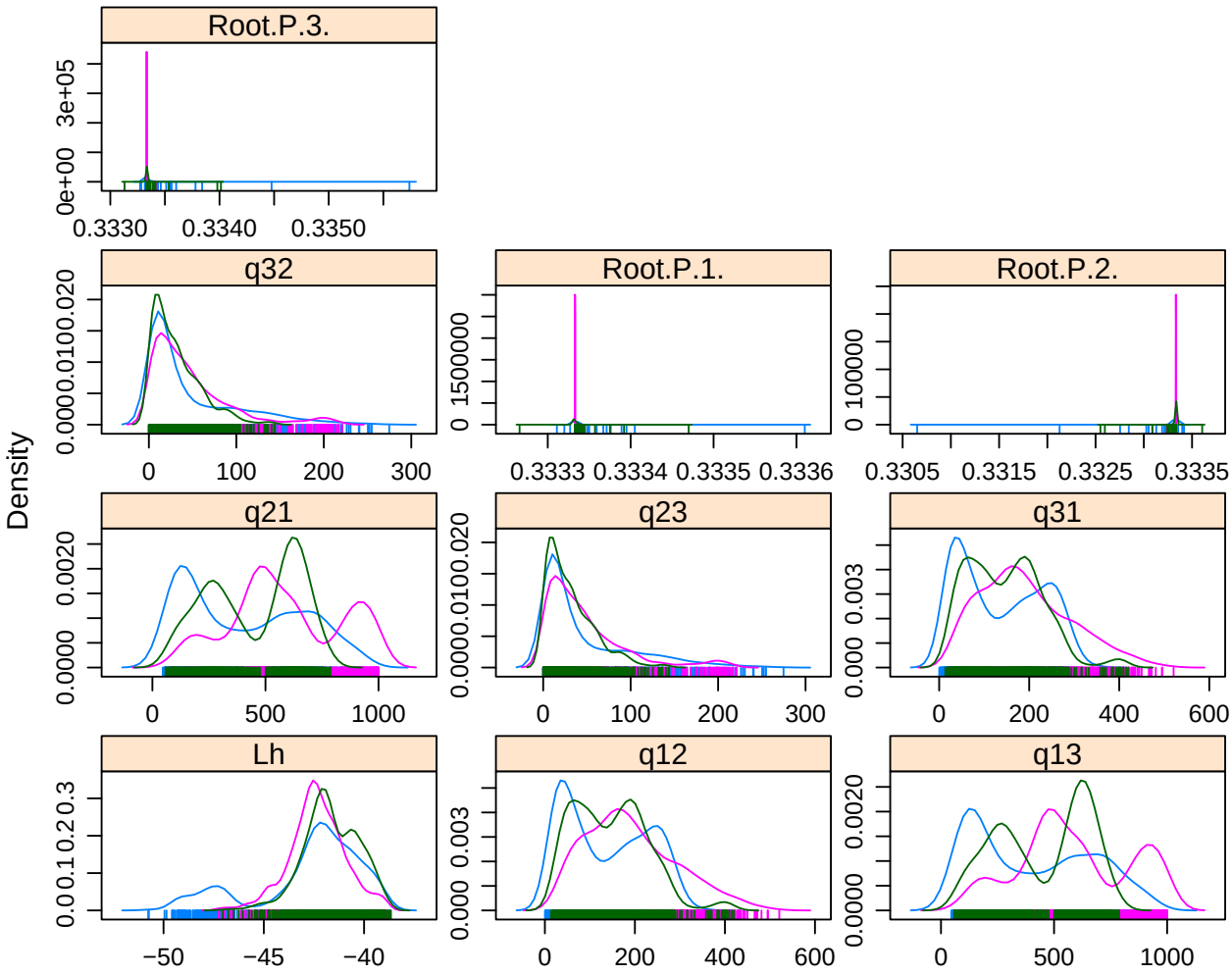
```
            end=max(multistate.MCMC.res1$Results$Iteration),thin=100)
res2 <- mcmc(multistate.MCMC.res2$Results[,c(-1,-4)],
            start=min(multistate.MCMC.res2$Results$Iteration),
            end=max(multistate.MCMC.res2$Results$Iteration),thin=100)
res3 <- mcmc(multistate.MCMC.res3$Results[,c(-1,-4)],
            start=min(multistate.MCMC.res3$Results$Iteration),
            end=max(multistate.MCMC.res3$Results$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
##     37.229224     10.509809     25.170421      8.534771      8.534771
##           q23           q31           q32     Root.P.1.     Root.P.2.
##    119.025292     25.170421    119.025292   1739.001367   1761.358625
##     Root.P.3.
##   1727.878694
```

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

```
## Potential scale reduction factors:
##
##               Point est. Upper C.I.
## Lh                  1.15       1.39
## Harmonic.Mean       6.36      12.82
## q12                 1.09       1.26
## q13                 1.13       1.38
## q21                 1.13       1.38
## q23                 1.07       1.17
## q31                 1.09       1.26
## q32                 1.07       1.17
## Root.P.1.           1.17       1.19
## Root.P.2.           1.23       1.26
## Root.P.3.           1.23       1.27
```

```
# 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             -42.2394 1.976e+00 3.626e-02      4.033e-01
## Harmonic.Mean -44.1566 2.453e+00 4.500e-02      2.404e-01
## q12           157.7984 9.536e+01 1.750e+00      1.889e+01
## q13           480.1138 2.506e+02 4.599e+00      8.399e+01
## q21           480.1138 2.506e+02 4.599e+00      8.399e+01
## q23            41.6890 4.431e+01 8.130e-01      5.251e+00
## q31           157.7984 9.536e+01 1.750e+00      1.889e+01
## q32            41.6890 4.431e+01 8.130e-01      5.251e+00
## Root.P.1.       0.3333 6.490e-06 1.191e-07      1.652e-07
```

```
## Root.P.2.       0.3333 6.054e-05 1.111e-06      1.571e-06
## Root.P.3.       0.3333 5.441e-05 9.984e-07      1.415e-06
##
## 2. Quantiles for each variable:
##
##                    2.5%      25%      50%      75%    97.5%
## Lh             -47.8739 -42.9565 -42.0559 -40.9590 -39.2463
## Harmonic.Mean -48.5052 -46.9362 -43.0291 -42.2816 -40.7952
## q12             20.0634  75.6197 154.8932 223.7319 374.0282
## q13             82.6910 254.3228 497.0869 652.5884 959.1860
## q21             82.6910 254.3228 497.0869 652.5884 959.1860
## q23              1.4314  10.7617  26.6740  56.2958 176.0072
## q31             20.0634  75.6197 154.8932 223.7319 374.0282
## q32              1.4314  10.7617  26.6740  56.2958 176.0072
## 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

	0,0	0,1	1,0	1,1
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
##     216.77551      38.89503     330.40364     394.00370     335.11918
##           q24           q31           q34           q42           q43
##     394.00370     375.17067     330.40364     375.17067     335.11918
## Root...P.0.0. Root...P.0.1. Root...P.1.0. Root...P.1.1.
##    1450.89378    1033.66372    1035.61964    1446.85109
```

```
effectiveSize(dep)
```

```
##            Lh Harmonic.Mean           q12           q13           q21
##     234.76171      14.53585     237.15719     228.40602     298.53284
##           q24           q31           q34           q42           q43
##     225.58948     217.46602     166.88288     239.16590     220.55682
## Root...P.0.0. Root...P.0.1. Root...P.1.0. Root...P.1.1.
##     817.93370     827.04936     851.72739     663.32531
```

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

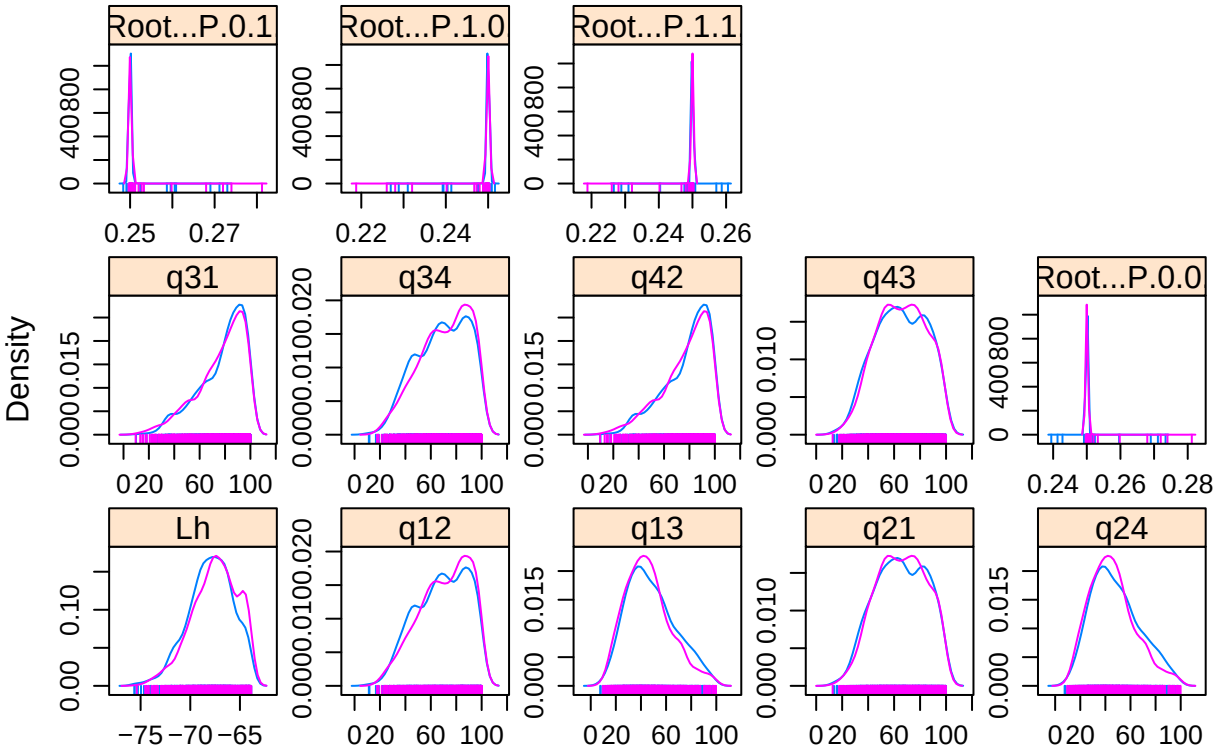
```
## Potential scale reduction factors:
##
##               Point est. Upper C.I.
## Lh                  1.01       1.06
## Harmonic.Mean       1.20       1.63
## q12                 1.00       1.02
## q13                 1.01       1.04
## q21                 1.00       1.00
## q24                 1.01       1.04
## q31                 1.00       1.01
## q34                 1.00       1.02
## q42                 1.00       1.01
## q43                 1.00       1.00
## Root...P.0.0.       1.02       1.02
## Root...P.0.1.       1.01       1.01
## Root...P.1.0.       1.01       1.01
## Root...P.1.1.       1.02       1.02
```

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

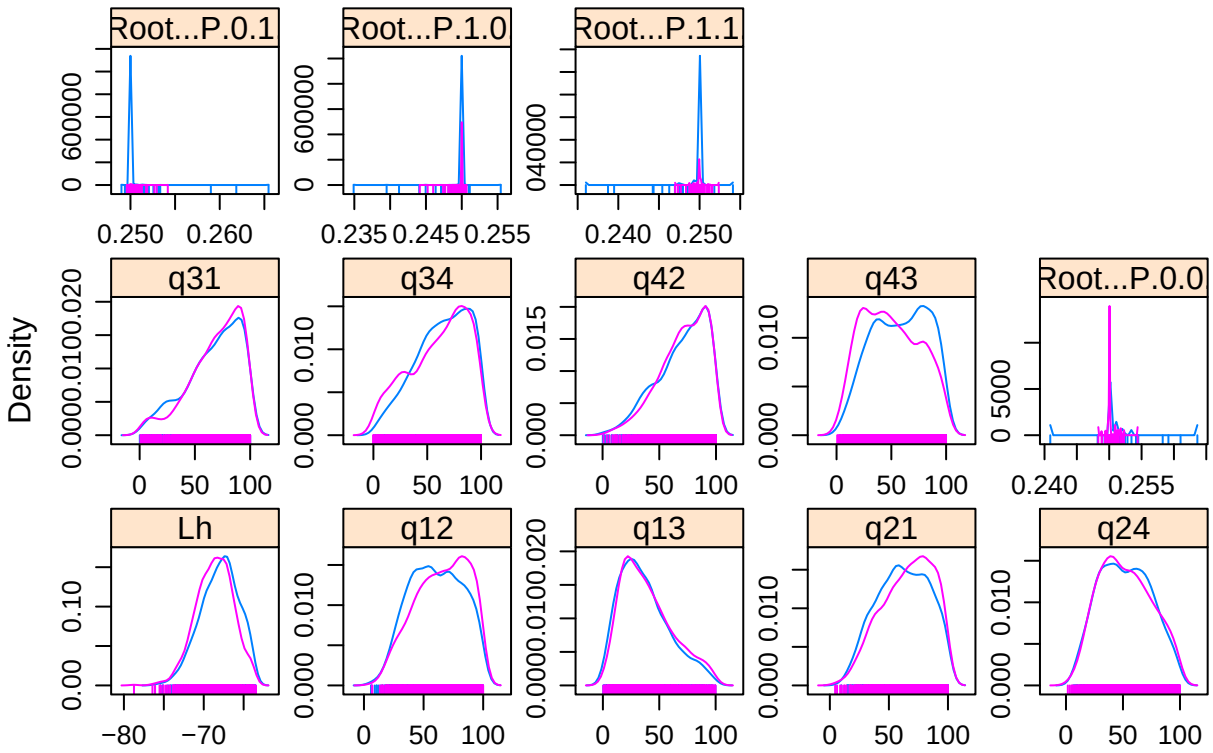
```
## Potential scale reduction factors:
##
##               Point est. Upper C.I.
## Lh                  1.03       1.13
## Harmonic.Mean       2.50       5.38
## q12                 1.02       1.09
## q13                 1.01       1.03
## q21                 1.01       1.06
## q24                 1.00       1.00
## q31                 1.00       1.01
## q34                 1.01       1.04
## q42                 1.00       1.00
## q43                 1.05       1.20
```

```
## Root...P.0.0.        1.23        1.31
## Root...P.0.1.        1.22        1.24
## Root...P.1.0.        1.16        1.16
## Root...P.1.1.        1.23        1.34
```

```
# 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             -67.7520  2.261793 5.083e-02      1.594e-01
## Harmonic.Mean -70.6596  0.624675 1.404e-02      9.474e-02
## q12            69.8698 19.737675 4.436e-01      1.098e+00
## q13            47.0042 18.695147 4.201e-01      9.479e-01
## q21            65.2890 19.105556 4.294e-01      1.054e+00
## q24            47.0042 18.695147 4.201e-01      9.479e-01
## q31            77.1914 18.424610 4.141e-01      9.516e-01
## q34            69.8698 19.737675 4.436e-01      1.098e+00
## q42            77.1914 18.424610 4.141e-01      9.516e-01
## q43            65.2890 19.105556 4.294e-01      1.054e+00
## Root...P.0.0.   0.2501  0.001450 3.258e-05      4.614e-05
## Root...P.0.1.   0.2501  0.001457 3.274e-05      4.891e-05
## Root...P.1.0.   0.2499  0.001457 3.274e-05      4.888e-05
## Root...P.1.1.   0.2499  0.001449 3.256e-05      4.615e-05
##
## 2. Quantiles for each variable:
```

```
##
##                    2.5%    25%    50%    75%    97.5%
## Lh             -72.5325 -69.20 -67.60 -66.11 -64.1023
## Harmonic.Mean -71.1332 -70.93 -70.78 -70.55 -69.8445
## q12             30.3780  55.44  71.65  86.33  98.7847
## q13             16.8409  33.16  44.67  58.15  89.5381
## q21             29.3801  50.54  65.71  80.84  97.1103
## q24             16.8409  33.16  44.67  58.15  89.5381
## q31             34.1042  66.09  82.17  91.98  99.3152
## q34             30.3780  55.44  71.65  86.33  98.7847
## q42             34.1042  66.09  82.17  91.98  99.3152
## q43             29.3801  50.54  65.71  80.84  97.1103
## Root...P.0.0.    0.2500   0.25   0.25   0.25   0.2501
## 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.    0.2499   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             -68.3337 2.355e+00 5.293e-02      1.526e-01
## Harmonic.Mean -71.3288 7.317e-01 1.644e-02      1.570e-01
## q12             62.8700 2.179e+01 4.897e-01      1.409e+00
## q13             38.3978 2.241e+01 5.036e-01      1.483e+00
## q21             63.8917 2.147e+01 4.825e-01      1.239e+00
## q24             52.1934 2.245e+01 5.044e-01      1.495e+00
## q31             67.6367 2.395e+01 5.382e-01      1.628e+00
## q34             62.2672 2.575e+01 5.786e-01      2.020e+00
## q42             70.3658 2.142e+01 4.813e-01      1.418e+00
## q43             54.0018 2.570e+01 5.777e-01      1.709e+00
## Root...P.0.0.    0.2501 6.003e-04 1.349e-05      2.210e-05
## Root...P.0.1.    0.2501 5.406e-04 1.215e-05      1.887e-05
## Root...P.1.0.    0.2499 5.850e-04 1.315e-05      2.082e-05
## Root...P.1.1.    0.2500 5.761e-04 1.295e-05      2.181e-05
##
## 2. Quantiles for each variable:
##
##                    2.5%    25%    50%    75%    97.5%
## Lh             -73.2256 -69.95 -68.23 -66.67 -64.1233
## Harmonic.Mean -72.4960 -71.67 -71.25 -71.00 -68.8548
## q12             22.0143  45.59  63.74  81.33  97.5985
## q13              5.6621  20.77  34.23  52.04  89.7767
## q21             22.3467  47.63  65.34  81.28  98.1083
## q24             13.8632  34.20  51.33  69.45  94.9974
## q31             10.3138  52.90  72.13  87.54  98.4383
```

```
## q34             7.2938  44.61  65.98  83.60  98.2063
## q42            22.0073  56.48  74.34  88.36  98.7528
## q43             9.6942  32.81  53.19  76.63  97.4442
## Root...P.0.0.   0.2499   0.25   0.25   0.25   0.2504
## Root...P.0.1.   0.2500   0.25   0.25   0.25   0.2504
## Root...P.1.0.   0.2494   0.25   0.25   0.25   0.2500
## Root...P.1.1.   0.2497   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] -70.87435
```

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

```
## [1] -71.02755
```

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

```
## [1] -0.306388
```

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.