Lab 10: Diversification Analysis

Today we are going to use both *R* and *Mesquite* to simulate random trees and to analyze diversification. We will ask several questions. What are the diversification parameters? Are these trees significantly unbalanced or balanced? What clades are unexpectedly diverse? Is diversification associated with a given character?

We will use both programs, because each program can do things that the other can not. *Mesquite* has many more methods for simulating trees than *R* does, including simulations in which diversification parameters are dependent on either continuous or discrete characters. We will hold off on simulating trees in *R* until next time. *Mesquite* can also test for a correlation between a discrete character and a branching pattern. On the other hand, *R* can calculate measures of imbalance, identify particularly diverse clades, and test for a correlation between a continuous character and branching patterns.

Random Trees in Mesquite

Mesquite has several different methods for simulating random trees. We will explore a few relevant ones here. For each set of random trees we will use a new project. To create a new project use the following process.

File>New
Name the project ***something.nex*** and hit **Save**
We will create trees with **40 taxa** so we want to create a taxa block with **40 taxa**. This is only necessary for a few of our methods, but we'll just do it every time

Simulating Random Trees from the Birth Death Process

The first thing we'll do is create a bunch of random trees using the good old birth-death process. Under this model taxa have a constant probability of reproducing, λ, and a constant probability of going extinct, μ.

Create a new project called **"BDtrees.nex".**
Now simulate the trees:
Taxa&Trees>Make New Trees Block From>Simulated Trees>Birth/Death Process Trees
There λ and μ values are fine.
Make **100** trees.
Don't do it on a separate thread.

Page through your trees. Look at the branch lengths while you do it. As you can see all the trees have 40 taxa just like our taxa block. Store these trees (**Tree>Store Tree)**, and save the file. Then you can shut the project down, we'll use these again soon.

Simulating Random Trees from Other Processes

Create a new project called **"RandomTrees.nex"**
Go back to the **simulated trees** menu (**Taxa&Trees>Make New Trees Block From**

>Simulated Trees) and peruse the options. We will use the coalescence simulators in a later lab. Today I'll describe the others:

Uniform Speciation (Yule): The same as the BD trees, except that there is no extinction. The topologies will be the same as the BD trees, but the branch lengths may vary slightly. The math is much simpler for the Yule process than for the BD process, and the difficulty in accurately calculating μ, makes Yule estimations almost as good, despite the obviously incorrect assumption of no extinction.

Uniform Speciation with Sampling: Same as Yule, except a tree is simulated with more taxa than the final tree. Branches are randomly pruned to produce a tree of the desired size.

Equiprobable Trees: Every labeled tree topology is considered equally likely.

Make 100 Equiprobable Trees, store the trees, save the file, and shut down the project.

Simulating Trees with Continuous Evolving Diversification

Up until now all of our models have had no variation in macroevolutionary forces. *Mesquite* has two models of character evolution in which the rate of diversification evolves. In the first model the speciation rate evolves along the tree by Brownian Motion. There is no extinction. Thus taxa with a high speciation rate should be related to other taxa with a high speciation rate and should be found in more diverse clades. We would expect this process to create an imbalanced phylogeny.

Open a new file called **"MacroTrees1.nex"**
Then simulate some trees:
Taxa&Trees>Trees & Diversification Characters
Then select **Evolving Speciation Rate (Continuous)**.
Stick with **0.1** as the rate of evolution for this character. We'll make **100** trees.
The random seed they choose is fine.

When it's done double click on the tree icon to see your trees. Do these trees look more imbalanced to you? Don't worry we'll get a chance to test that. Store your trees, save the file and exit the project.

Open a new file called **"MacroTrees3.nex"** Make another 100 trees, but this time have the rate evolve at **0.3.** Check out your trees (More unbalanced still?), store them save them, shut down the project

Simulating Trees Dependent on Binary Characters

The second method of evolving macroevolutionary forces relies on a discrete character. A two state character evolves on a tree; the rates of speciation and extinction depend on the state of this character. Let's simulate some trees using this method:

First. create a new file called **"Bintree.nex".**
Then simulate some trees:
Taxa&Trees>Trees & Diversification Characters
Then select **BISSE Trees & Characters**. We'll make **10** trees.
The random seed they choose is fine.

A box will open with the parameters we will use for our simulation. The first two parameters are the rates of switching between states 0 and 1. After that you have the speciation and extinction rates for each of the character states. To make this interesting let's try to balance speciation rates

against microevolutionary processes.

Change **Rate 1-> 0** to **0.0200** and change **Rate of Speciation 1** to **0.13**. In this way microevolution will favor state 0, and macroevolution will favor state 1.

The next option is the desired tree size, that is how many taxa you want in your simulated tree. The *Mesquite* manual says that you can set this parameter yourself, but on my computer this parameter was fixed to the number of taxa in our taxa block.

Continue to tree size is a very important parameter for our simulation. The problem with doing simulations using such a complicated process is that you can not pick a random set of branch lengths that will correspond to the model parameters you have set. Instead you have to just run the simulation for a set period of time or until you get a certain number of taxa. We want to get 40 taxa; if we wait till we get 40 taxa and then keep running the model, there is a possibility that we will get 40 taxa again later on. So do you stop the first time you get 40 taxa, the second, the tenth (of course there may not be ten times)? No we simulate until we have many more taxa than 40, and then go back and pick one of the previous times when we had 40 taxa in our tree. Set this number to **60** and hit **OK**.

A character state window with ten characters and a tree list window with ten trees will appear. Click on the tree icon to see the actual tree. Now make the tree **square**, make the **branches proportional to branch length**, and **trace** the **Character History** using **Parsimony.** Page through the trees and the characters. Each tree matches up with the character of the same number, so page through them together. Characters should not be mapped on trees with different numbers. That's cool. Do your trees have interesting shapes and interesting distributions of characters? Did you get about half your taxa in state 1? Does it look like state 1 had a higher diversification rate?

Pick a tree with nice mix of taxa in each state. Do the rest of your analyses on this tree. This is not proper statistical methodology, but it should make the rest of the lab better.

Examining Birth-Death Parameters in Mesquite

Uncorrelated Birth Death Parameters in Mesquite

First we'll just calculate the parameters of the Birth-Death model for this tree using Maximum Likelihood.

Analysis>Diversification(Char. Indep.)

A box will appear to the right of the tree as it does the analysis. It will end up giving maximum likelihood values for λ and μ. How do these numbers correspond to the values you actually used? If λ and μ aren't so good, how about *r*, the difference between them?

When your done go to:

Diversification>Close Diversification Analysis

Nonparametric Test For Correlation Between Diversification and a Bimary Character

We know that this tree did have a higher speciation rate for character state 1. Let's see if our statistical tests show that.

Analysis>Character Associated Diversification

For the first test we'll do **Sister Diversification**.

A **Diversification Analysis** box will appear to the right of your tree. From the menu bar select **Diversification(Ch)>Choose Character**; select the correct character to go with your tree. Make sure that the character and the tree match up in your Analysis box.

This analysis picks pairs of sister clades in which all the taxa of one clade have state 0 and all the taxa in the other clade have state 1. It then looks to see how often the clade with one of those states is larger and compares that to a two-tailed binomial distribution with probability 0.5. I bet you guys could do this one by hand. I don't know about your tree and character, but I got a pretty insignificant p-value for mine.

ML Analysis of Correlation Between Diversification and a Binary Character

Another option is to do a maximum likelihood analysis of the data on your tree using a model in which λ and μ depend on the state of the character.

Diversification(Ch)>Diversification Measure> BISSE Speciation/Extinction Likelihood.

A panel will pop up showing all the parameters in your model. This panel allows you to constrain every parameter. You can constrain a parameter to a specific value by typing that value into the first box, or you can use the pull down menu in order to constrain a parameter to be equal to another parameter. For now, we'll just leave all the parameters unconstrained. Hit **OK**.

The box to the right of the tree will now show the outcome of the maximum likelihood analysis. It may take a second for the analysis to complete. At this point what you are really interested in is the maximum likelihood calculations of the parameter values. Are these values close to the parameter values we used for our simulation? What if you calculate diversification ($=\lambda-\mu$), is that closer to your simulation value?

ML Test for Correlation Between Diversification and a Binary Character

What we really want to do is test the hypothesis that diversification is associated with our character. To do that we'll use a Likelihood Ratio Test. *Mesquite* will do this test for you, but you have to set it up. Select:

Diversification(Ch)>Diversification Measure> Ln Likelihood Difference.

The parameter box will appear again. This is for our unconstrained model, so just leave everything as it is and hit **OK**. Another parameter box will appear, this is for our constrained model. Here we want to force λ_1 to equal λ_0 and μ_1 to equal μ_0. Use the pull down menus for all 4 parameters to set them equal to each other, then hit OK.

This may take a minute.

The box on the right will now show the parameter values from our unconstrained model, the log likelihoods from both models and the difference between those likelihoods. Mine was really small, hopefully yours is a little larger. You have two choices for generating a p-value. The first and much simpler option is to compare twice this difference to a chi-squared distribution. Make that calculation in *R*. How many degrees of freedom should you use?

The second and more accurate way of generating a p-value is to simulate a bunch of uncorrelated characters on this tree and compare the LRT from those tests to the LRT from our character. We'll save that for a later lab.

Question 1: How would you test the hypothesis that both μs are significantly different from 0? Give a description of the exact commands you would use in *Mesquite*. Use the same notation for pull down

menus as I do.

Shut down *Mesquite* when you're done. We're moving on to *R*.

Evaluating the Birth-Death Process in *R*

The first thing that we're going to do is examine all those simulated trees that we made in *R*. So open *R*, load *ape* and open all those *nexus* files:

```
bd.tree<-read.nexus("...\BDtrees.nex")
mac.tree1<-read.nexus("...\MacroTrees1.nex")
mac.tree2<-read.nexus("...\MacroTrees3.nex")
rand.tree<-read.nexus("...\RandomTrees.nex")
```

What we have here now are actually objects of class *multiPhylo*, not *phylo*. That just means that it is a *list*, where each element of the list is a tree of class *phylo*. We index different elements of a list using double brackets. We can now treat those individual trees, just like we did before. For example we could plot a tree:

```
plot(bd.tree[[1]])
```

ape has a number of methods for examining the Birth-Death process. Like *Mesquite* it can calculate λ and μ for a given tree by maximum likelihood. It can also test for a correlation between λ and a continuous character. We will look briefly at both.

Calculating Birth, Death and Diversification Rates

Calculating the BD parameters is easy:

```
birthdeath(mac1.tree[[1]])
```

Hmm, did you get an error? I got an error the first time I used this function. Hopefully it worked for you. Just in case it is working, I'll describe the output. *N* is the number of taxa in the tree; *dev* is the *-2log(likelihood)*; *para* is the estimate of the parameters. This function does not produce estimates of λ and μ, but instead estimates $a=\mu/\lambda$ and $r=\lambda-\mu$. *se* is the standard error of those estimates, and *CI* is the 95% confidence interval.

We can also calculate the parameters from the Yule process, which is the BD process in which we assume $\mu=0$.

```
yule(mac1.tree[[1]])
```

That worked. I think the output here is pretty self explanatory. Let's look at the distributions of λ for all our trees.

```
temp<-function(a) yule(a)$lambda
hist(sapply(mac1.tree,temp))
```

That's an interesting spread. I wish we had a better idea of the λ we used, so we could make the comparison. I bet that the variation is almost completely from the total tree depth. That is to say, trees with greater distance from root to tip have smaller λs.

Calculating Birth, Death Rates with Unresolved Phylogeny

Often you want to make a calculation of the diversification parameters, but you don't know the entire phylogeny. If your phylogeny is completely unresolved, then the ML calculation is trivial. If t is the time from the last common ancestor to the present, and N is the number of taxa, then $\mu=0$ and $\lambda=log(N/2)/t$.

A more common situation is where you know the phylogeny at the base of the tree, but the subtaxa at the tips are unresolved. a*pe* can make this calculation; you need a vector giving the number of species at the tips. I'm going to use an example tree from the *laser* package, because I think it will make the method more clear than our random trees will.

All the packages have several data sets stored in them already; let's call one up and plot the tree. This is a tree of skinks with taxon diversity data (Rabosky, et al. Proc.Roy. Soc. Lond. Ser. B 274:2915-2923). First load the *laser* package:

```
data(skinktree)
plot(skinktree)
```

That's a cool tree. Now let's get the diversity data:

```
data(skinkdiversity)
skinkdiversity
```

Anyone know anything about skinks? Does that look right? We need to make the order of the two vectors match:

```
temp<-match(rownames(skinkdiversity),skinktree$tip.label)
skink.tips<-skinkdiversity[temp,1]

plot(skinktree,show.tip.label=FALSE,x.lim=c(0,30))
tiplabels(skink.tips,frame="n",adj=c(0,0.5))
```

The idea here is that the evolution of all those species happened since the last internal node. So that the number of species grew along those long terminal branches. Of course you are assuming that these orders are in fact monophyletic.

Finally, let's do the analysis.

```
bd.ext(skinktree,skink.tips)
```

There you go; it estimated μ as 0, and gave you an estimate of λ. I wonder if this result is any different from just assuming you don't know the topology at all.

Examining Balance in *R*

apTreeshape

What we really want to do is examine these trees for imbalance; for that we will use another *R* package, *apTreeshape* by Nicolas Bortolussi, Eric Durand, Michael Blum, and Olivier Francois Unfortunately, I could not download this package from *CRAN*. If you could, great. If you could not,

open the *R* script ***treeshape.part.R***. This is a file, I put together of several scripts from the *apTreeshape* package. However, this is a far from exhaustive version of the package; it only has the functions we will be using here. The actual package has a number of other useful functions.

apTreeshape uses a class of trees called *treeshape* instead of the *phylo* class that we have used up until now. We must convert a tree to class *treeshape* in order to examine its shape.

bd.tshape<-as.treeshape.phylo(bd.tree[[1]])

Imbalance Indices

Colless' Index is the sum of the absolute values of the differences between daughter clade sizes at each node. Thus it will clearly be high for imbalanced clades and small for highly balanced clades. In fact for a fully imbalanced tree it will be *(n-1)(n-2)/2*, and it will be 0 for a completely balanced tree, although only trees with 2^n taxa can be completely balanced.

To calculate Colless' index:

colless(bd.tshape)

apTreeshape can also normalize Colless' Index for a tree with *n* taxa. It can do this on either the basis of the *Yule* or *PDA* distribution of tree shapes. We already discussed the *Yule* distribution and the *PDA* is the same as *Equiprobable Trees* in *Mesquite*. This can be an extremely useful feature, if you need to compare trees with different number of taxa. I think I've made clear how I feel about Equiprobable trees, but let's normalize with the Yule:

colless(bd.tshape,norm="yule")

apTreeshape can also analyze imbalance using Sackin's Index. Sackin's Index is calculated as the sum of the number of nodes between each tip and the root. In this way it is also larger for more imbalanced clades. It will take the value *(n+2)(n-2)/2* for a completely imbalanced clade, and $n*log_2 n$ for a completely balanced one. a*pTreeshape* can also calculate a normalized or an unnormalized value for this index:

sackin(bd.tshape)
sackin(bd.tshape,norm="yule")

Neither of these indices rely on branch lengths. They are both solely measures of topology.

Handling Multiple Trees

We want to compare imbalance for all our simulated trees. Unfortunately all these *apTreeshape* functions only work on single trees. To deal with this we will have to use *lapply* to apply a function to every element in our list, and return a list of the results:

bd.ts<-lapply(bd.tree,as.treeshape.phylo)

That was pretty straight forward, now we'll use that to convert all our trees:

mac1.ts<-lapply(mac.tree1,as.treeshape.phylo)
mac3.ts<-lapply(mac.tree3, as.treeshape.phylo)
rand.ts<-lapply(rand.tree,as.treeshape.phylo)

It is just as easy to calculate indices for every tree in the list:

bd.col<-sapply(bd.ts,colless)

Pretty similar, except *sapply* makes a vector instead of a list. We can also pass other commands to *colless*, such as *norm="yule"*, as follows:

bd.col.norm<-sapply(bd.ts,colless,norm="yule")

Make the following objects using the appropriate function and list of trees: **mac1.col, mac3.col, rand.col, bd.sack, mac1.sack, mac3.sack,** and **rand.sack**.

Comparing Balance from Different Models

Let's look at the distribution of tree shapes for birth-death trees.

mean(bd.col)
hist(bd.col)

This is a good null distribution for balance if you are trying to show that a clade is more imbalanced than expected. How about one of our sets of trees with evolving speciation rates?

mean(mac1.col)
hist(mac1.col)

As you can see, the trees with evolving speciation are much more imbalanced.

Let's build a matrix of the indices for each set of trees, and a factor describing the simulation method for each tree, so that we can make a box plot.

comb.col<-cbind(rand.col,bd.col,mac1.col,mac3.col)
comb.method<-c("Rand", "BD", "Evolve 0.1", "Evolve 0.3")
comb.method<-matrix(comb.method,nrow=100,ncol=4,byrow=TRUE)
comb.method<-as.factor(comb.method)
plot(comb.method,comb.col)

Interesting, the Equiprobable Trees were the most imbalanced and the birth-death trees were the least. There did not seam to be much difference between the trees with low and high rates for the evolution of speciation rate. I would have expected a bigger difference, although the higher rate does seam to have bigger variance.

Let's test to see if the difference between these methods are significant. First we'll just do an ANOVA.

compare.anova<-aov(comb.col~comb.method)
summary(compare.anova)

That's a pretty good result. Now let's do a Tukey test to see which groups are significantly different from each other.

TukeyHSD(compare.anova)

So, everything is way different, except the two trees with evolving rates. You know, I am really surprised that the tree with the faster evolving rate is not more imbalanced. I wonder if I made a mistake at some point.

There are two assumptions to this test. That the distributions are normal and the variances are equal. We can use **qqnorm** and **qqline** to check this first assumption, and everything looks good. You can test for equal variances with:

bartlett.test(comb.col~comb.method)

OK, so they don't have equal variances; we'll use the nonparametric Kruskal-Wallis test.

kruskal.test(comb.col~comb.method)

Still significant. I don't know how to make *R* compare the groups nonparametrically.

Comparing trees to a Null Distribution

As I stated before, our Birth-Death trees make the best null distribution for imbalance, as they represent a constant branching process. Let's compare some of our other trees to this null distribution in order to generate a p-value.

greater<-sum(bd.sack>mac1.sack[1])

greater is now the number of trees from our null distribution, which had a Sacken's Index greater than the first tree simulated with λ evolving at 0.1. To calculate a p-value from which to reject the null distribution:

(greater+1)/(length(bd.sack)+1)

We added one to both the numerator and denominator to account for the tree we are testing. You probably rejected the null distribution, thus implying that your tree is significantly more imbalanced than expected. If your test was actually two-tailed then your p-value would be twice that. Remember that you only did 100 simulations, so if you calculated a p of 1/101=0.0099 your actual p-value is probably much lower than that.

Another interesting question is how much power does this statistic have to detect a λ evolving at 0.1 if we reject the null hypothesis at a significance level of 0.05. The first thing to do is identify what is the cut off for a one tailed p=0.05 from our null distribution. We want the 96th smallest value, because it is the 5th largest and 0.05*100=5.

cutoff.05<-sort(bd.sack)[96]

We then want to ask what is the probability of our other distribution generating a value greater than our cutoff:

sum(mac1.sack>cutoff.05)/length(mac1.sack)

That number should be pretty high. It is the power of our test statistic to reject the null distribution in favor of our alternative distribution.

Question 2: What power does Sacken's index have to reject the null hypothesis for our equiprobable trees?

Examining the Diversification Process

Identifying Diverse Nodes

Maybe it's not good enough just to recognize that a tree is particularly imbalanced. You may want to identify which nodes in the tree are particularly diverse. Load the *geiger* package. This package has a test based on Purvis, et al. 2004. In O.R.P. Bininda-Emonds (ed.), Phylogenetic Supertrees, pp. 487-533. In this test a diversification rate is estimated for the entire tree by ML. They then calculate the probability of each clade being as large as it is or larger given that diversification rate. If this value is significant, then those nodes are considered exceptionally diverse. Note that we are going to use the *phylo* objects again.

rc(mac.tree1[[4]],plot.bonf=TRUE)

The first thing that you will see is a table with two columns. The rows are the different clades of the tree, numbered according to their basal nodes. The first column is the probability of getting a clade that diverse and the second column is that probability with the Bonferroni correction for multiple tests. A plot appears with significant clades marked by an asterisk (enlarge the plot to see it better). *plot.bonf =TRUE* makes it plot nodes that are significant after the correction, without that it would have plotted nodes that are significant without the correction as well. Look at some other trees. Do you see many significant nodes? I'd like to see this statistic compared to a null distribution, rather than accepting these probabilities as p-values.

ML Test for Correlation between Diversification and a Continuous Character

Or maybe you would like to see if you can test for a correlation between a particular character and diversification. We already saw how *Mesquite* can do it for a discrete character. *ape* has a function that allows you test for a relationship between multiple continuous characters evolving by Brownian motion and the speciation rate. It is important to understand the assumptions of this model (Paradis, E. (2005) Evolution, 59, 1–12.)

1) The traits in question evolves by BM. We already know how to test this using *diagnostics*.
2) $\mu=0$. This assumption is definitely wrong, but it has little effect on the outcome of the test.
3) The reconstructed trait values at the nodes, which you assign, are correct. This obviously depends on your confidence in the reconstruction.
4) Traits change linearly between nodes. Well, I guess you have to assume something about the ancestral trait values, and this is the best assumption.
5) There is a linear relationship between $log(\lambda/(1-\lambda))$, and the characters in question. This relationship could obviously be anything and this assumption is awfully restrictive. It would be nice if you could test for other possible relationships, but what are you going to do. This assumption probably has an adverse affect on p-values for most real data.

Let's test for a relationship between one of our trees and the lambda character that evolved with it. First we need to read our characters into R.

Open *MacroTrees1.nex* in a text editor, and edit the nexus file to remove ***everything*** except the data matrix itself. Save this modified file with a new name. Now we can read the data:

mac1.data<-read.table(*filename*,row.names=1)
dim(mac1.data)

This data frame should be 40 by 100. Each column is the character value for the appropriate tree, so that you should not compare trees and characters with different numbers. Now we have to make sure that the taxa names are in the same order in both the tree and the data matrix.

rownames(mac1.data)
mac.tree1[[1]]$tip.label
mac.tree1[[2]]$tip.label
mac.tree1[[3]]$tip.label

That's great; all the taxa are listed in the same order in all the trees and our data set. If they weren't we would have to use the *match* function to rearrange the data to match the tree, like in the ***Continuous Characters Lab***. Furthermore if each tree had a different arrangement of taxa, then we would have to do this separately for each column of our data matrix. Luckily we don't.

Now let's make sure that the character we will analyze evolved by BM. Run the *diagnostics2* script, and:

diagnostics(mac1.data[,1],mac.tree1[[1]])

I looked through several different trees and one of them looked bad, but the rest looked good. If your diagnostics look bad, then pick a tree character combo that looks good. From now on I'll call that tree ***x***.

Now we can reconstruct ancestral trait values, and create a combined vector of those and our tip values:

mac1.recon.*x*<-c(mac1.data[,*x*],ace(mac1.data[,*x*],mac.tree1[[*x*]])$ace)

and compare those reconstructions to our branching pattern

yule.cov(mac.tree1[[*x*]],~mac1.recon.*x*)

So, I tried multiple trees and did not get good outputs from a few. If yours did not work try a different combination of tree and character.

You will get a log likelihood, and a slope and y intercept for our hypothesized line. You should note that, when $\lambda>1$, $log(\lambda/(1-\lambda))$ decreases as λ increases above 1, so a negative slope implies a positive correlation between our character and the branching pattern. What you really want to know is if this relationship is significantly better than a Yule model that is independent of the character. So, let's run that test.

mac1.yule.*x*<-yule(mac.tree1[[*x*]])

Now you can compare the two models using the LRT:

pchisq(2*(mac1.yule.*x*$loglik-*log likelihood from yule.cov***),1,lower.tail=FALSE)**

You are going to have to copy the log likelihood from your *yule.cov* analysis. Note that *yule*

provides a negative log likelihood and *yule.cov* provides just a log likelihood; so, when I say "**-***log likelihood from yule.cov"* , I mean minus the absolute value of the log likelihood. There is one degree of freedom, because there is one extra parameter, the slope of the line. Was that significant? All mine were. That's not surprising considering that our character was in fact λ.

Comparison of Diversification to a Random Character

I want to see what a random character looks like. Let's simulate a character on our tree:

rand.char.*x*<-evolve.phylo(mac.tree1[[*x*]],c(0,3),1)

This function simulated two characters by BM with variance 1 and value 0 at the root for the first and value 3 for the second . *rand.char* is now an object of class *phylo,* exactly like *mac.tree[[*x*]]*, but with two elements added to it: *$tip.character* is the value of that character at the tips, and *$node.character* is the value at the nodes. We will pretend that we don't know the node values and instead reconstruct node values using *ace*.

rand.char.*x*<-rand.char.*x*$tip.character[,1] *Cause we only care about this vector*
diagnostics(rand.char.*x*,mac.tree1[[*x*]]) *To check our assumptions*
rand.recon.*x*<-c(rand.char.*x*,ace(rand.char.*x*,mac.tree1[[*x*]])$ace)

And now do the two tests:

yule.cov(mac.tree1[[*x*]],~rand.recon.*x*)
yule(mac.tree1[[*x*]])

Now calculate the LRT and use chi squared to get a p-value. I still got a significant log likelihood for my random character. I suspect that this method may not fit the chi-squared quite as well as it claims to. My guess is that λ is correlated with the phylogeny and our random character is correlated with the phylogeny, so that our random character appears to be correlated with λ. Let's try to look at both together:

yule.cov(mac.tree1[[*x*]],~rand.recon.*x*+mac1.recon.*x*)

This has the same likelihood as our truly correlated character and a higher likelihood than our uncorrelated character. Thus this does suggest the correct model. However, it implies that you could have specious correlation of a character, if a phylogeny were highly imbalanced, and you did not have the characters that are truly influencing diversification in your analysis. This method needs more scrutiny. He claims in his original paper that it did a good job of estimating p-values, but I am skeptical.

Question 3. How would you generate a null distribution for the hypothesis that a continuous character is correlated with a given tree?