corHMM Dredge: Automatic model discovery

James D. Boyko

corHMM Dredge is a combination of regularization and parameter sharing optimization to optimize models of discrete character evolution. Technical details can be found in Boyko (2024). This vignette will focus on a simple use case.

Start by loading the data:

```
library(corHMM)
```

```
## Loading required package: ape

## Loading required package: nloptr

## Loading required package: GenSA
```

```
data(primates)
phy <- multi2di(primates[[1]])
phy$edge.length <- phy$edge.length + 1e-7
data <- primates[[2]]
```

Now we run corHMMDredge. The function essentially attempts to fit various model structures automatically. It is able to drop parameters and equate parameters depending on the parameter estimates from previous fits. There is also a regularization term, which penalizes fast transition rates and encourages low rates. This also encourages a sparser transition matrix, encouraging dredge to drop model parameters. Dredge will keep running until the models are no longer improving (based on something like AIC).

We can set various hyper parameters but the most important things to note are the max.rate.cat, the pen.type, and the lambda. For this run we are only going to be looking at a max.rate.cat of 1, the penalty type will be l1, and lambda will be set to 1. l1 as the regularization type means that high rates will be penalized and setting lambda to 1 means that it will be a full penalization. Now lambda is not like most parameters in our model in the sense that it cannot be optimized in the standard likelihood search. Instead, we use cross validation which will be run after we fit a set of dredge models.

```
dredge_fits <- corHMMDredge(phy = phy, data = data, max.rate.cat = 1, pen.type = "l1", root.p = "maddfi
```

```
## Begining dredge...
##
## AIC: 101.5139
## Mapping matrix:
##     0|0 1|0 0|1 1|1
## 0|0  NA   3   5  NA
## 1|0   1  NA  NA   7
## 0|1   2  NA  NA   8
## 1|1  NA   4   6  NA
##
##  * Continuing dredge *
##
## AIC: 93.51388
## Mapping matrix:
```

```
##     0|0 1|0 0|1 1|1
## 0|0  NA  NA   2  NA
## 1|0  NA  NA  NA  NA
## 0|1   1  NA  NA   4
## 1|1  NA  NA   3  NA
##
##  * * Continuing dredge * *
##
## AIC: 91.5828
## Mapping matrix:
##     0|0 1|0 0|1 1|1
## 0|0  NA  NA   3  NA
## 1|0  NA  NA  NA  NA
## 0|1   1  NA  NA   2
## 1|1  NA  NA   3  NA
##
##  * * * Continuing dredge * * *
##
## AIC: 90.11056
## Mapping matrix:
##     0|0 1|0 0|1 1|1
## 0|0  NA  NA   2  NA
## 1|0  NA  NA  NA  NA
## 0|1   1  NA  NA   2
## 1|1  NA  NA   2  NA
##
##  * * * * Continuing dredge * * * *
##
## AIC: 92.11887
## Mapping matrix:
##     0|0 1|0 0|1 1|1
## 0|0  NA  NA   1  NA
## 1|0  NA  NA  NA  NA
## 0|1   1  NA  NA   1
## 1|1  NA  NA   1  NA
##
## Done.
```

```
model_table <- getModelTable(dredge_fits)
```

As the dredge runs, it will print an AIC value and the mapping matrix. The AIC value is what dredge uses to determine if it should continue fitting models. The mapping matrix are the model structures that dredge tried based on previous estimates. You can see that as the dredge continues the models become simpler and it only stopped when the AIC difference was greater than 2.

To determine which model to use after dredge, we do not do any model averaging. These models are not associated with biological hypotheses (though you can still make biological interpretations) and so the multimodel approach is not appropriate (I make arguments for this in Boyko (2024)). Instead, we will just find which model had the best AIC and use that!

```
model_table <- getModelTable(dredge_fits)
print(model_table)
```

```
##   np     lnLik       AIC      dAIC       AICwt
## 1  8 -42.75694 101.51388 11.403323 0.001644679
## 2  4 -42.75694  93.51388  3.403323 0.089796408
## 3  3 -42.79140  91.58280  1.472245 0.235823607
## 4  2 -43.05528  90.11056  0.000000 0.492358244
## 5  1 -45.05944  92.11887  2.008314 0.180377062
```

In this case, it's the 4th model that has the best AIC value and that's the one we will proceed with.

```
dredge_model <- dredge_fits[[which.min(model_table$dAIC)]]
print(dredge_model$index.mat)
```

```
##     0|0 1|0 0|1 1|1
## 0|0  NA  NA   2  NA
## 1|0  NA  NA  NA  NA
## 0|1   1  NA  NA   2
## 1|1  NA  NA   2  NA
```

Above I have printed out the optimized model structure. That's all good, but now we have to optimize lambda. We will use k-fold cross validation with phylogenetic sampling (again described in detail in Boyko (2024)). We have to specify the corHMM model, the number of folds, and which lambda values we want to evaluate. These scores are a measure of divergence from the known tip state. So lower values are better.

```
k_fold_res <- kFoldCrossValidation(dredge_model, k = 5, lambdas = c(0,0.25,0.5,0.75,1))
```

```
## Evaluating lambda = 0
## Fold 0 Score: 0.01443587
## Fold 1 Score: 0.03165594
## Fold 2 Score: 0.0284769
## Fold 3 Score: 0.03303103
## Fold 4 Score: 0.04880152
## Average Cross-Validation Score: 0.03128025
## Evaluating lambda = 0.25
## Fold 0 Score: 0.01421111
## Fold 1 Score: 0.03143803
## Fold 2 Score: 0.02646486
## Fold 3 Score: 0.03273638
## Fold 4 Score: 0.04873756
## Average Cross-Validation Score: 0.03071759
## Evaluating lambda = 0.5
## Fold 0 Score: 0.01403961
## Fold 1 Score: 0.03125159
## Fold 2 Score: 0.02414567
## Fold 3 Score: 0.03252268
## Fold 4 Score: 0.04868174
## Average Cross-Validation Score: 0.03012826
## Evaluating lambda = 0.75
## Fold 0 Score: 0.01391095
## Fold 1 Score: 0.03109008
## Fold 2 Score: 0.02253771
## Fold 3 Score: 0.03237341
## Fold 4 Score: 0.04863398
## Average Cross-Validation Score: 0.02970923
## Evaluating lambda = 1
## Fold 0 Score: 0.01381558
## Fold 1 Score: 0.03094863
## Fold 2 Score: 0.02152674
## Fold 3 Score: 0.03227638
## Fold 4 Score: 0.04859148
## Average Cross-Validation Score: 0.02943176
```

As the cross validation runs, I print out the scores for each fold per lambda, and then give an average score for each lambda. A more convenient way to look at this is in table form.

```r
cv_table <- corHMM:::getCVTable(k_fold_res)
print(cv_table)
```

```
## $score_table
##          Lambda:0 Lambda:0.25 Lambda:0.5 Lambda:0.75   Lambda:1
## Fold:0 0.01443587  0.01421111 0.01403961  0.01391095 0.01381558
## Fold:1 0.03165594  0.03143803 0.03125159  0.03109008 0.03094863
## Fold:2 0.02847690  0.02646486 0.02414567  0.02253771 0.02152674
## Fold:3 0.03303103  0.03273638 0.03252268  0.03237341 0.03227638
## Fold:4 0.04880152  0.04873756 0.04868174  0.04863398 0.04859148
##
## $avg_scores
##    Lambda:0 Lambda:0.25  Lambda:0.5 Lambda:0.75    Lambda:1
##  0.03128025  0.03071759  0.03012826  0.02970923  0.02943176
```

So here we can see the lowest error is for a Lambda of 1, which means we will proceed from here with lambda of 1.

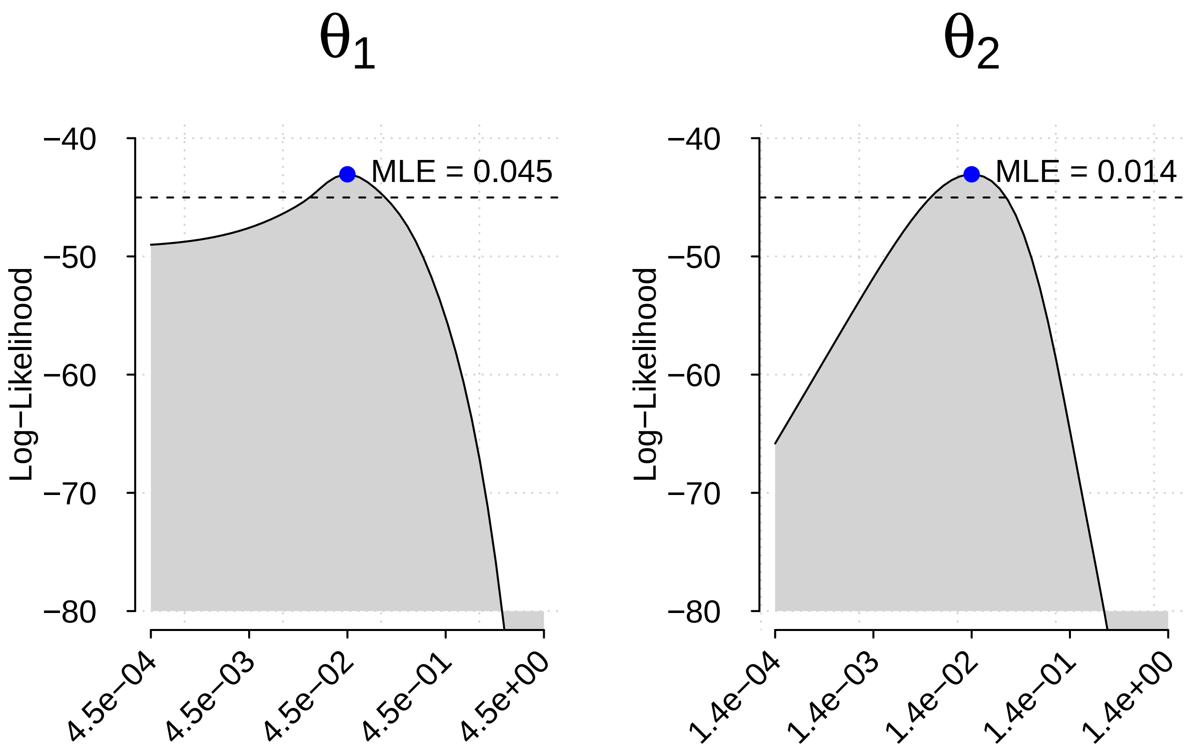
I have also implemented a profile likelihood function so that users can examine the shape of the likelihood surface and find ridges if they are there. For range factor we are specifying how far away from the MLE we are going to examine. So if the MLE is 1 and the range factor is 100 then the profile likelihood will search between 1/100 to 1*100. The n_points is how many points between that range the profile likeihood will examine.

```r
profile_results_dredge <- get_batch_profile_lik(dredge_model, dredge = TRUE,
  range_factor = 100, n_points = 50, verbose = TRUE, ncores = 10)
```

```
##
##  1 of 2 ...
##  2 of 2 ...
```

It prints out which parameter it's examining. The model dredge found had two parameters, hence when we plot our profile likelihood we will only see two graphs.

```r
plot_batch_profile_lik(profile_results_dredge, , ylim = c(-80, -40), xlab = "", label_cex = 1, cex.main
```



The dashed line is the 95% confidence interval and the MLE is plotted with a blue dot. These are nice likelihood surfaces with no indiciation of ridges!

As a bonus I want to show what would have happened if you just ran default corHMM without considering alternative model structures. This is not as uncommon as you may think. There are plenty of cases where the model fitting is really a nuisance parameter and people are just interested in an ancestral state reconstruciton or something along those lines. So let's run an ARD default model.

```
corhmm_model <- corHMM(phy, data, 1, model = "ARD", root.p = "maddfitz", collapse = FALSE, nstarts = 10
```

```
## State distribution in data:
## States:  0|0 1|0 0|1 1|1
## Counts:  29  0   10  21
## Beginning thorough optimization search -- performing 10 random restarts
## Finished. Inferring ancestral states using marginal reconstruction.
```

If we look at the model results, there is nothing untoward. The likelihood value seems fine, the parameter estimates are okay, so we could probably be pretty happy continuing with this for any downstream analysis. Right?

```
print(corhmm_model$loglik)
```

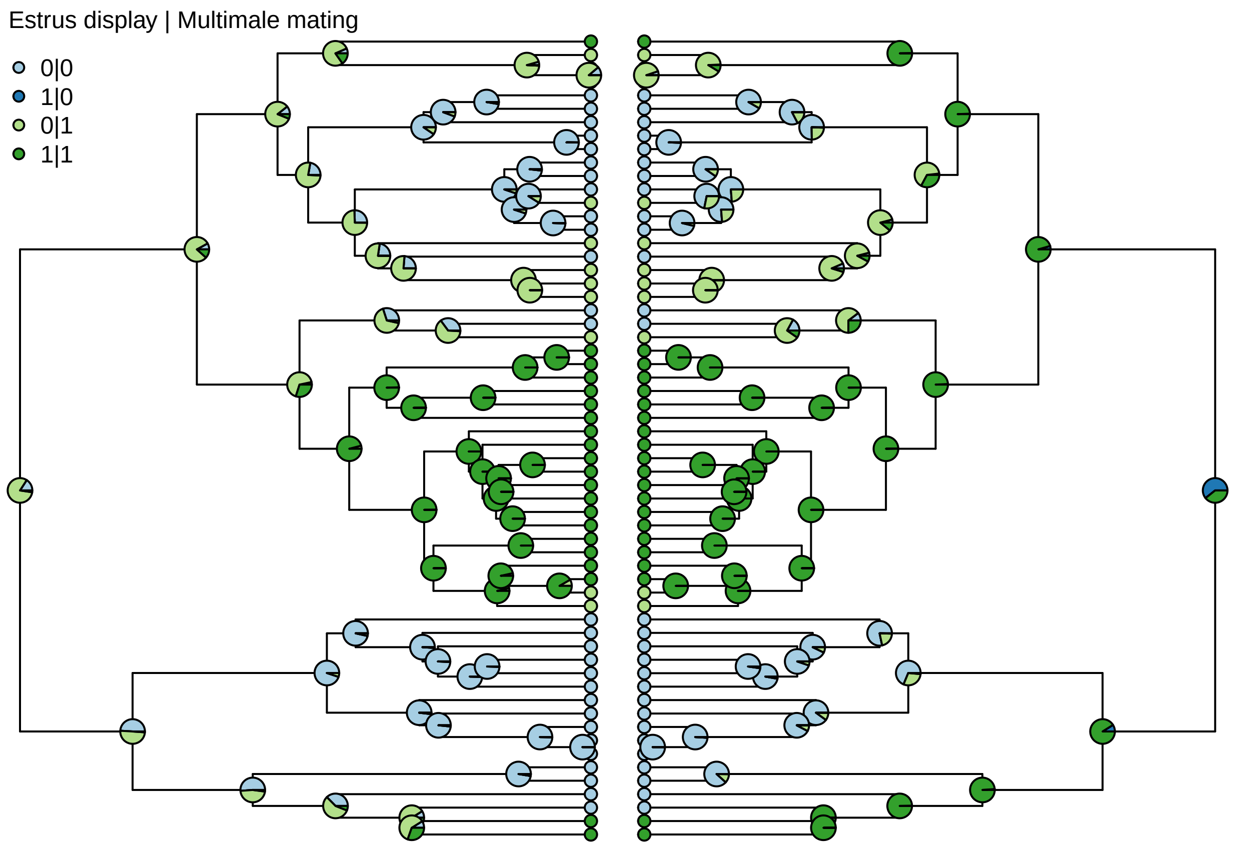
```
## [1] -41.32702
```

```
print(corhmm_model$solution)
```

```
##              0|0   1|0         0|1         1|1
## 0|0           NA 1e-09 0.007673625          NA
## 1|0 0.000000001    NA          NA 0.183115793
## 0|1 0.083599086    NA          NA 0.000000001
## 1|1           NA 1e-09 0.026797328          NA
```

So let's use this for our downstream ASR and compare it to the model corHMMDredge found.

```
piecolors <- RColorBrewer::brewer.pal(4, "Paired")
par(mar=c(.1,.1,.1,.1), mfrow=c(1,2))
plot(dredge_model$phy, show.tip.label = FALSE)
tiplabels(pie = dredge_model$tip.states, cex=0.5, piecol = piecolors)
nodelabels(pie = dredge_model$states, piecol = piecolors)
legend("topleft", legend = colnames(dredge_model$states),
  pch=21, pt.bg = piecolors, cex = 0.75, bty="n", title = "")
text(x = -1, y = 61.5, label = "Estrus display | Multimale mating",
  cex = 0.75, adj=0)
plot(corhmm_model$phy, show.tip.label = FALSE, direction = "leftwards")
tiplabels(pie = corhmm_model$tip.states, cex=0.5, piecol = piecolors)
nodelabels(pie = corhmm_model$states, piecol = piecolors)
```



These are pretty different… The left shows the corHMM Dredge result and the right is the standard ARD model. I go into some of the differences in Boyko (2024), but these are biologically meaningful in the sense that it changes the interpretation drastically. This hopefully will convince you of the importance of searching for a good model structure. I don't even necessarily mean using Dredge, I mean it is worth considering many possible models not just the default settings. Dredge is just there to make our lives easier in cases where the biology is maybe not clear, or we're just interested in find the absolute best model structure. Let's take this further and examine the ARD model a bit more.

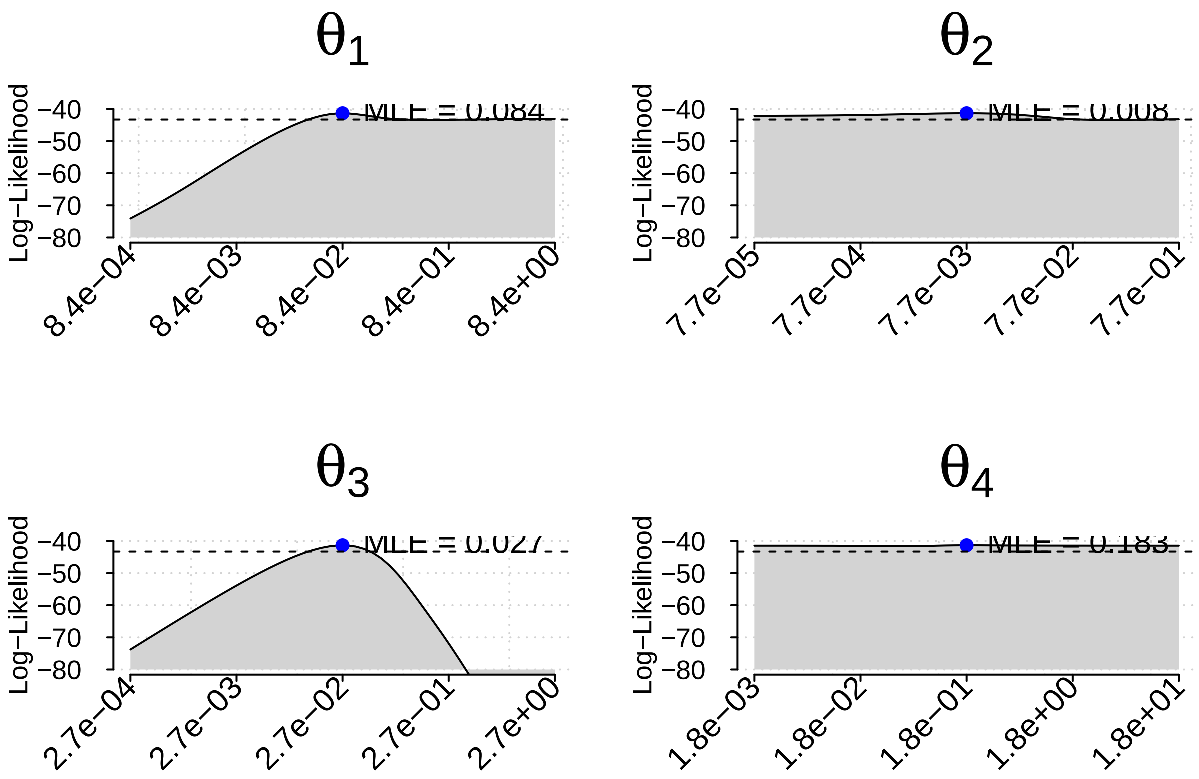
We will start with the profile likeihood. Here I'm just going to drop all of those parameter estimates which were super low, since they aren't doing much and it will be easier to visualize the output

```
to_drop <- corhmm_model$index.mat[which(corhmm_model$solution < 1e-8)]
corhmm_model$index.mat <- dropStateMatPars(corhmm_model$index.mat, to_drop)
profile_results_corhmm <- get_batch_profile_lik(corhmm_model, dredge = FALSE,
  range_factor = 100, n_points = 50, verbose = TRUE, ncores = 10)
```

```
##
##  1 of 4 ...
##  2 of 4 ...
##  3 of 4 ...
##  4 of 4 ...
```

And now we plot…

```r
plot_batch_profile_lik(profile_results_corhmm, ylim = c(-80, -40), xlab = "", label_cex = 1.25, cex.mai
```



Uh oh. If you wondered what I meant by ridge above, there you go. Parameter 2 and 4 show clear ridges and even 1 has a ridge extending far to the right. It looks like there are MANY values within the 95% confidence intervals. And these values are really far apart so we are really uncertain about what the true value is. And if you're wondering if this has downstream consequences, the weird answer is that it depends. The ancestral state reconstruction for any values within the 95% are actually pretty consistent. But the difference in biological interpretation between a rate of 0.01 transition per million years and 100 transitions per million years is pretty massive.

I'll show you what I mean by the ASR not being as influenced as I'd have expected. Parameter 4 has a likelihood ridge extending far to the right. So let's sample one of those points and multiply that rate by 1000. So we're going from a rate of 0.18 to 183!! Surely that will impact the ancestral state reconstruction right?

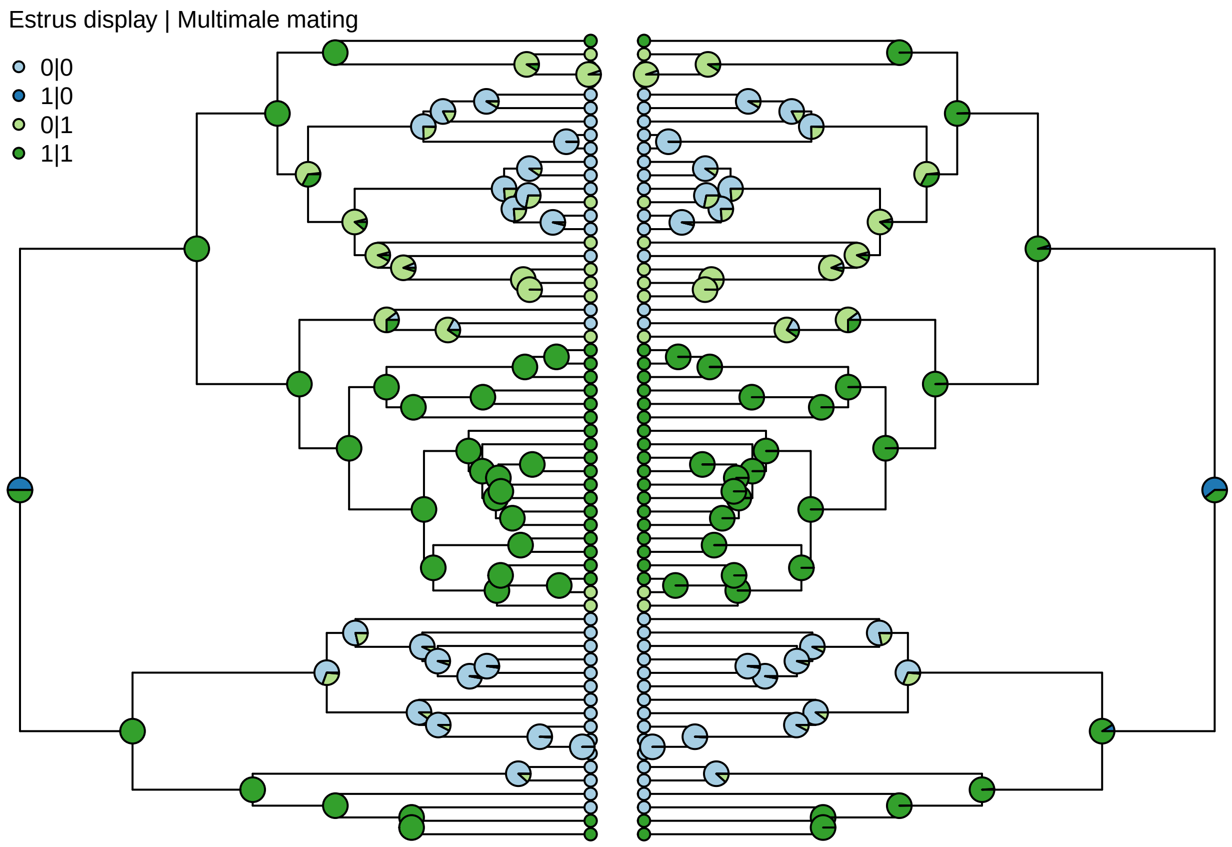
```r
p <- corhmm_model$solution[!is.na(corhmm_model$index.mat)]
p[4] <- p[4]*1000
corhmm_model_3 <- corHMM(phy, data, 1, root.p = "maddfitz", collapse = FALSE, rate.mat = corhmm_model$i
```

```
## State distribution in data:
## States:  0|0 1|0 0|1 1|1
## Counts:  29  0   10  21
## Calculating likelihood from a set of fixed parameters
## Finished. Inferring ancestral states using marginal reconstruction.
```

Now we plot.

```r
piecolors <- RColorBrewer::brewer.pal(4, "Paired")
par(mar=c(.1,.1,.1,.1), mfrow=c(1,2))
plot(corhmm_model_3$phy, show.tip.label = FALSE)
tiplabels(pie = corhmm_model_3$tip.states, cex=0.5, piecol = piecolors)
nodelabels(pie = corhmm_model_3$states, piecol = piecolors)
legend("topleft", legend = colnames(corhmm_model_3$states),
  pch=21, pt.bg = piecolors, cex = 0.75, bty="n", title = "")
text(x = -1, y = 61.5, label = "Estrus display | Multimale mating",
  cex = 0.75, adj=0)

plot(corhmm_model$phy, show.tip.label = FALSE, direction = "leftwards")
tiplabels(pie = corhmm_model$tip.states, cex=0.5, piecol = piecolors)
nodelabels(pie = corhmm_model$states, piecol = piecolors)
```



To the left we see the results of the high rates version. And to the right we see the results of the standard fit. There really is not much difference between the two. Despite having a transition rate 1000x faster. This really surprised me, but it's interesting. So it seems that model structure is more important for ASR than the rate estimates.