Continuous character models

Brian C. O'Meara

21 January, 2021

First get packages we need

```
install.packages("yearn")
```

```
## Warning: unable to access index for repository http://cran.cnr.berkeley.edu/src/contrib:
##   cannot open URL 'http://cran.cnr.berkeley.edu/src/contrib/PACKAGES'
```

```
## Warning: package 'yearn' is not available for this version of R
##
## A version of this package for your version of R might be available elsewhere,
## see the ideas at
## https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages
```

```
## Warning: unable to access index for repository http://cran.cnr.berkeley.edu/bin/macosx/contrib/4.0:
##   cannot open URL 'http://cran.cnr.berkeley.edu/bin/macosx/contrib/4.0/PACKAGES'
```

```
library(ape) #utility fns
library(geiger) #utilty fns
library(OUwie)
```

Now get the tree and data. For these exercises, knowing uncertainty in your measurements can also be important. (remember for homework to change `eval=FALSE` to `eval=TRUE`).

```
tree <- read.tree("____PATH_TO_TREE_OR_SOME_OTHER_WAY_OF_GETTING_A_TREE____")
continuous.data <- read.csv(file="____PATH_TO_DATA_OR_SOME_OTHER_WAY_OF_GETTING_TRAITS____", stringsAsF
```

A function to clean data, make sure taxon names match between tree and data, etc.

```
CleanData <- function(phy, data) {
    #treedata() in Geiger is probably my favorite function in R.
}
```

A function to plot data. Look at `phytools::contMap()`. This is all part of checking: do your data all seem sensible? **LOOK AT IT**.

```
VisualizeData <- function(phy, data) {
    #Important here is to LOOK at your data before running it. Any weird values? Does it all make sense
}
```

First, start basic. What is the rate of evolution of your trait on the tree?

```
BM1 <- geiger::fitContinuous(tree, cleaned.continuous, model="BM")
print(paste("The rate of evolution is", _____, "in units of", _______))
```

Important: What are the rates of evolution? In what units?

```
OU1 <- fitContinuous(tree, cleaned.continuous, model="OU")
par(mfcol(c(1,2)))
```

```
plot(tree, show.tip.label=FALSE)
ou.tree <- rescale(tree, model="OU", ___alpha____)
plot(ou.tree)
```

How are the trees different?

Compare trees

```
AIC.BM1 <- ________FIGURE_OUT_HOW_TO_DO_THIS_____
AIC.OU1 <- ________FIGURE_OUT_HOW_TO_DO_THIS_____
delta.AIC.BM1 <- ________FIGURE_OUT_HOW_TO_DO_THIS_____
delta.AIC.OU1 <- ________FIGURE_OUT_HOW_TO_DO_THIS_____
```

##OUwie runs##

This takes longer than you may be used to.

We're a bit obsessive about doing multiple starts and in general performing a thorough numerical search. It took you 3+ years to get the data, may as well take an extra five minutes to get an accurate answer

First, we need to assign regimes. The way we do this is with ancestral state estimation of a discrete trait. We can do this using ace() in ape, or similar functions in corHMM or diversitree. Use only one discrete char.

```
one.discrete.char <- _____________
reconstruction.info <- ace(one.discrete.char, tree, type="discrete", method="ML", CI=TRUE)
best.states <- colnames(reconstruction.info$lik.anc)[apply(reconstruction.info$lik.anc, 1, which.max)]
```

Now add these labels to your tree.

```
labeled.tree <- _________________
nodeBased.OUMV <- OUwie(tree, cleaned.continuous,model="OUMV", simmap.tree=FALSE, diagn=FALSE)
print(nodeBased.OUMV)
```

What do the numbers mean?

Now run all OUwie models:

```
models <- c("BM1","BMS","OU1","OUM","OUMV","OUMA","OUMVA")
results <- lapply(models, RunSingleOUwieModel, phy=tree, data=trait)

AICc.values<-sapply(results, "[[", "AICc")
names(AICc.values)<-models
AICc.values<-AICc.values-min(AICc.values)


print(AICc.values) #The best model is the one with smallest AICc score

best<-results[[which.min(AICc.values)]] #store for later

print(best) #prints info on best model
```

We get SE for the optima (see nodeBased.OUMV$theta) but not for the other parameters. Let's see how hard they are to estimate. First, look at ?OUwie.fixed to see how to calculate likelihood at a single point.

```
?OUwie.fixed
```

Next, keep all parameters but alpha at their maximum likelihood estimates (better would be to fix just alpha and let the others optimize given this constraint, but this is harder to program for this class). Try a range of alpha values and plot the likelihood against this.

```
alpha.values<-seq(from= _____ , to= _____ , length.out=50)
```

Keep it simple (and slow) and do a for loop:

```
likelihood.values <- rep(NA, length(alpha.values))
for (iteration in sequence(length(alpha.values))) {
    likelihood.values[iteration] <- OUwie.fixed(tree, trait, model="OUMV", alpha=rep(alpha.values[itera
}

plot(x= _____ , y= _____, xlab="_____", ylab="_____", type="l",
points(x=best$solution[1,1], y=best$loglik, pch=16, col="red")
text(x=best$solution[1,1], y=best$loglik, "unconstrained best", pos=4, col="red")
```

A rule of thumb for confidence for likelihood is all points two log likelihood units worse than the best value. Draw a dotted line on the plot to show this

```
abline(h=_____, lty="dotted") #Two log-likelihood
```

Now, let's try looking at both theta parameters at once, keeping the other parameters at their MLEs

```
require("akima")
nreps<-400
theta1.points<-c(best$theta[1,1], rnorm(nreps-1, best$theta[1,1], 5*best$theta[1,2])) #center on optima
theta2.points<-c(best$theta[2,1], rnorm(nreps-1, best$theta[2,1], 5*best$theta[2,2])) #center on optima
likelihood.values<-rep(NA,nreps)

for (iteration in sequence(nreps)) {
    likelihood.values[iteration] <- OUwie.fixed(tree, trait, model="OUMV", alpha=best$solution[1,], sig
}
```

Think of how long that took to do 400 iterations. Now remember how long the search took (longer).

```
likelihood.differences<-(-(likelihood.values-max(likelihood.values)))
```

We are interpolating here: contour wants a nice grid. But by centering our simulations on the MLE values, we made sure to sample most thoroughly there

```
interpolated.points<-interp(x=theta1.points, y=theta2.points, z= likelihood.differences, linear=FALSE,

contour(interpolated.points, xlim=range(c(theta1.points, theta2.points)),ylim=range(c(theta1.points, th

points(x=best$theta[1,1], y=best$theta[2,1], col="red", pch=16)

points(x=trait$X[which(trait$Reg==1)],y=rep(min(c(theta1.points, theta2.points)), length(which(trait$Re
points(y=trait$X[which(trait$Reg==2)],x=rep(min(c(theta1.points, theta2.points)), length(which(trait$Re
```

The below only works if the discrete trait rate is low, so you have a good chance of estimating where the state is. If it evolves quickly, hard to estimate where the regimes are, so some in regime 1 are incorrectly mapped in regime 2 vice versa. This makes the models more similar than they should be. See Revell 2013, DOI:10.1093/sysbio/sys084 for an exploration of this effect.

```
library(phytools)
trait.ordered<-data.frame(trait[,2], trait[,2],row.names=trait[,1])
trait.ordered<- trait.ordered[tree$tip.label,]
z<-trait.ordered[,1]
names(z)<-rownames(trait.ordered)
tree.mapped<-make.simmap(tree,z,model="ER",nsim=1)
leg<-c("black","red")
```

```
names(leg)<-c(1,2)
plotSimmap(tree.mapped,leg,pts=FALSE,ftype="off", lwd=1)

simmapBased<-OUwie(tree.mapped,trait,model="OUMV", simmap.tree=TRUE, diagn=FALSE)
print(simmapBased)
print(best)
```

How does this compare to our best model from above? Should they be directly comparable?