rphenoscate: Tutorial 2

Diego S. Porto, Sergei Tarasov, Caleb Charpentier, and SCATE team

August, 2022

In this second tutorial, we will be using again *rphenoscape* and *rphenoscate* to perform semantic-aware evolutionary analyses of morphological data. But this time, we are going to use a user-provided data set and an external ontology.

In this case, we are going to use a Hymenoptera (e.g., wasps, bees) data set including information not only from absences/presences of anatomical entities, but also their qualities (e.g., shape, structure, composition, etc.). Once again, we will set up appropriate models accounting for trait dependencies based on prior knowledge available from an anatomy ontology, in this case the Hymenoptera Anatomy Ontology (HAO) from Yoder et al. (2010): 10.1371/journal.pone.0015991.

STEP 1. Loading the packages.

First, let's load *rphenoscate*.

```
library("rphenoscate")
```

Let's load some other packages that might be useful as well. In particular, *ontologyIndex* allows us to work with external ontologies provided in .OBO format, the common standard used in the Open Biological and Biomedical Ontology Foundry (https://obofoundry.org/). *ontoFAST* allows us to perform semi-automatic annotations of phylogenetic character statements with anatomy ontology terms.

```
library("rphenoscape")
library("ontologyIndex")
library("ontoFAST")
library("TreeTools")
library("phytools")
```

STEP 2. Assembling the data set.

For this tutorial, we are going to use the data set from Porto and Almeida (2021):doi.org/10.1093/isd/ixab008, which comprises of 289 characters for 53 taxa of apid bees (Hymenoptera: Apoidea: Apidae).

```
# Import NEXUS file.
nex <- TreeTools::ReadCharacters(file = "./data/matrix.nex")

# Extract character states.
chars <- attributes(nex)$state.labels
names(chars) <- colnames(nex)

# Organize matrix and data labels.
mat <- as.data.frame(cbind(rownames(nex), nex))
colnames(mat)[1] <- "taxon"
```

And stochastic character mapping will be performed using the phylogeny available from Porto and Almeida (2021). Note that the original tree is not a dated phylogeny. For demonstrative purposes only, this tree was forced to ultrametric using the function 'force.ultrametric' from phytools. Therefore, branch lengths are not the result of a formal dating analysis.

```
# Import tree file.
tree <- read.tree("./data/tree.tre")
```

In the current form, this data set is simply a standard phylogenetic character matrix in NEXUS format. However, with functions from *ontoFAST* it is possible to parse through character statements and extract candidate matches to terms from a user-provided anatomy ontology. For that, we need first to import the HAO .OBO file in R using functions from *ontologyIndex*.

```
HAO <- get_OBO("./data/HAO.obo", extract_tags = "everything",
               propagate_relationships = c("BFO:0000050", "is_a"))
```

And then perform the semi-automatic term annotation.

```
# Pre-organize the ontology.
onto <- HAO
onto$parsed_synonyms <- syn_extract(HAO)
id_characters <- paste("CHAR:", c(1:(dim(mat)[2] - 1)), sep = "")
name_characters <- colnames(mat)[-1]
names(name_characters) <- id_characters
onto$name_characters <- name_characters
onto$id_characters <- id_characters

# Run ontoFAST.
auto_annot <- annot_all_chars(onto)
```

```
## [1] "Doing automatic annotation of characters with ontology terms..."
```

```
# Extract candidate terms.
x <- auto_annot
names(x) <- name_characters
```

Although this process is semi-automatic, we still need to refine the annotations. For that, we can use a filter function from *rphenoscate*. Based on expert knowledge, we can set two types of filters: strict and general. A strict filter removes all specific terms from a given set. In this case, a series of unnecessary terms referring to general anatomical regions (e.g. 'margin', 'depression') or general entities from the insect anatomy (e.g., 'sclerite', 'tergite', 'sternite'). A general filter removes all terms that contain a given word or expression (e.g., 'length', 'distance'). Here we are trying to get only terms referring to particular anatomical entities (e.g., 'antenna', 'head', 'maxilla', 'wing'), so this is why we need the filtering step.

For demonstrative purposes, let's get a sample of 20 character from the original data set.

```
#char.sample <- c(1,2,19,20,24,26,42,50,51,54,63,64,68,80,87,204,232,233,239,241)
char.sample <- c(1,2,10,19,20,26,37,42,50,51,54,63,64,68,80,87,232,233,239,241)
u <- x[char.sample]
```

Then, let's set the filters.

```
s.terms <- c("margin", "edge","ridge", "carina", "spine", "spur", "angle",
             "depression", "sclerite", "tergite", "sternite")
g.terms <- c("length", "distance", "diameter")
```

And finally, let's process the ontoFAST list of candidate terms to obtain the final term annotations. Note that the filter function will still require the input of an expert to select the preferred terms from the filtered set (i.e., it is an interactive step). For simplicity, let's just import the already filtered final set of terms.

```r
#w <- process.ontofast(u, HAO, s.filter = T, g.filter = T, s.terms = s.terms, g.terms = g.terms)
# For demonstrative purposes, let's just load data with the expert based selection of terms.
w <- readRDS("./data/terms.RDS")
```

Then, let's subset and organize the original data set based on our sample of 20 characters.

```r
# Subset original data set.
m <- mat[,c(1, char.sample + 1)]

# Subset character state information.
state.data <- chars[char.sample]

# Relabel character statements using the final set of ontology terms.
names(w) <- NULL
rownames(m) <- NULL
colnames(m) <- c("taxon", unlist(w))
```

One key difference between standard phylogenetic character matrices, such as this one, and a synthetic absence/presence matrix produced by OntoTrace, such as the one used in the previous tutorial, is that in the former, multiple characters statements can refer to the same anatomical entity, but not necessarily imply dependencies. For example, one character statement may refer to the shape of the 'antenna' and another one to its color; but in principle, there is no prior reason to assume ontological dependency between these two transformational character statements. Nonetheless, multiple characters referring to the same anatomical entity, with or without dependencies, can be represented as a single amalgamated character (see Tarasov 2019: doi.org/10.1093/sysbio/syz005 and Tarasov 2021: doi.org/10.1101/2021.04.26.441495 for in-depth discussion). Therefore, let's amalgamate all characters referring to the same anatomical entities.

```r
# Build initial data set.
td <- list()
td$phy <- tree
td$dat <- m
# OBS.: treeplyr::make.treedata does not work properly with duplicated columns!

# Get the dependency matrix using rphenoscape.
dep.mat <- auto_dep_matrix(td, tax.col = T)

# Amalgamate dependent characters.
amal.deps <- amalgamate_deps_gen(td, dep.mat, mode = "check", state.data = state.data)
```

```
## Using automatic qualitative type of intial vector phi.
## The intial vector phi is phi=c(1,1)
##
## Using automatic qualitative type of intial vector phi.
## The intial vector phi is phi=c(1,1)
##
## Using automatic qualitative type of intial vector phi.
## The intial vector phi is phi=c(1,1,1,1,1)
```

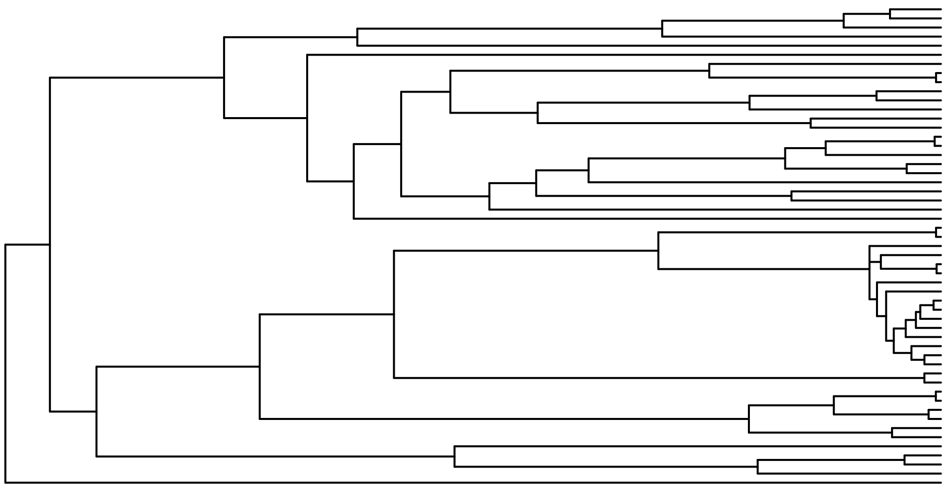
In this case, we are building a 'dependency matrix' to first amalgamate characters annotated with the same anatomy term. Some characters annotated will be actually independent from each other (e.g., 'shape' and 'color' of a structure) whereas others will be dependent (Vogt 2017: property-instantiation: e.g., 'absence/presence' and some quality of a structure). These will require different types of evolutionary models (i.e., SMM-ind and EDql; see Tarasov 2021 for more details). Finally, other types of dependencies among different anatomical entities (i.e., parthood-relations) can also be present, as demonstrated in the previous tutorial.

Then let's recode the amalgamated characters.

```r
td.comb <- recode_traits_gen(td = td, amal.deps = amal.deps)
```

Let's just visualize the tree by plotting it.

```r
plot.phylo(td$phy, show.tip.label = F)
```



STEP 3. Check data set for dependencies across anatomical entities.

As done in the previous tutorial, now we need to check for possible dependencies across the different anatomical entities represented by one or more (i.e., amalgamated) characters from the previous step. Once again, this can be achieved by using a function from *rphenoscate*, but in this case with an external ontology instead of Phenoscape KB.

```r
# Get the dependency matrix using rphenoscape.
dep.mat2 <- dep_matrix(td.comb, HAO, tax.col = F)
```

Checking the dependency matrix we can see that in this case no further dependencies were found (i.e., all lower diagonal values equal 0) so we can proceed to the next step without recoding the characters.

```r
sum(dep.mat2[lower.tri(dep.mat2)])
```

```
## [1] 0
```

STEP 4. Fitting models of trait evolution.

For simplicity, once again, we are going to fit models using *corHMM* through the wrapper function ‘amalgamate_fits_corHMM’ from *rphenoscate*.

```
corhmm.fits <- amalgamated_fits_corHMM(td.comb, amal.deps)
```

STEP 5. Sampling histories of trait evolution.

Finally, let’s use ‘amalgamated_simmaps_corHMM’ from *rphenoscate* to perform stochastic mapping.

```
stmaps <- amalgamated_simmaps_corHMM(corhmm.fits, nSim = 100)
names(stmaps) <- colnames(td.comb$dat)
```

And then plot some samples of character histories from different traits.

```
par(mfrow = c(2,2), mar = c(0.1,0.1,5.0,0.1))
plotSimmap(stmaps[[3]][[10]], ftype = "off")
```

```
## no colors provided. using the following legend:
##         1         2         3
##   "black" "#DF536B" "#61D04F"
```

```
title(main = names(stmaps)[3], font.main = 2, cex.main = 0.75, line = -0.3)
plotSimmap(stmaps[[4]][[10]], ftype = "off")
```

```
## no colors provided. using the following legend:
##         1         2
##   "black" "#DF536B"
```

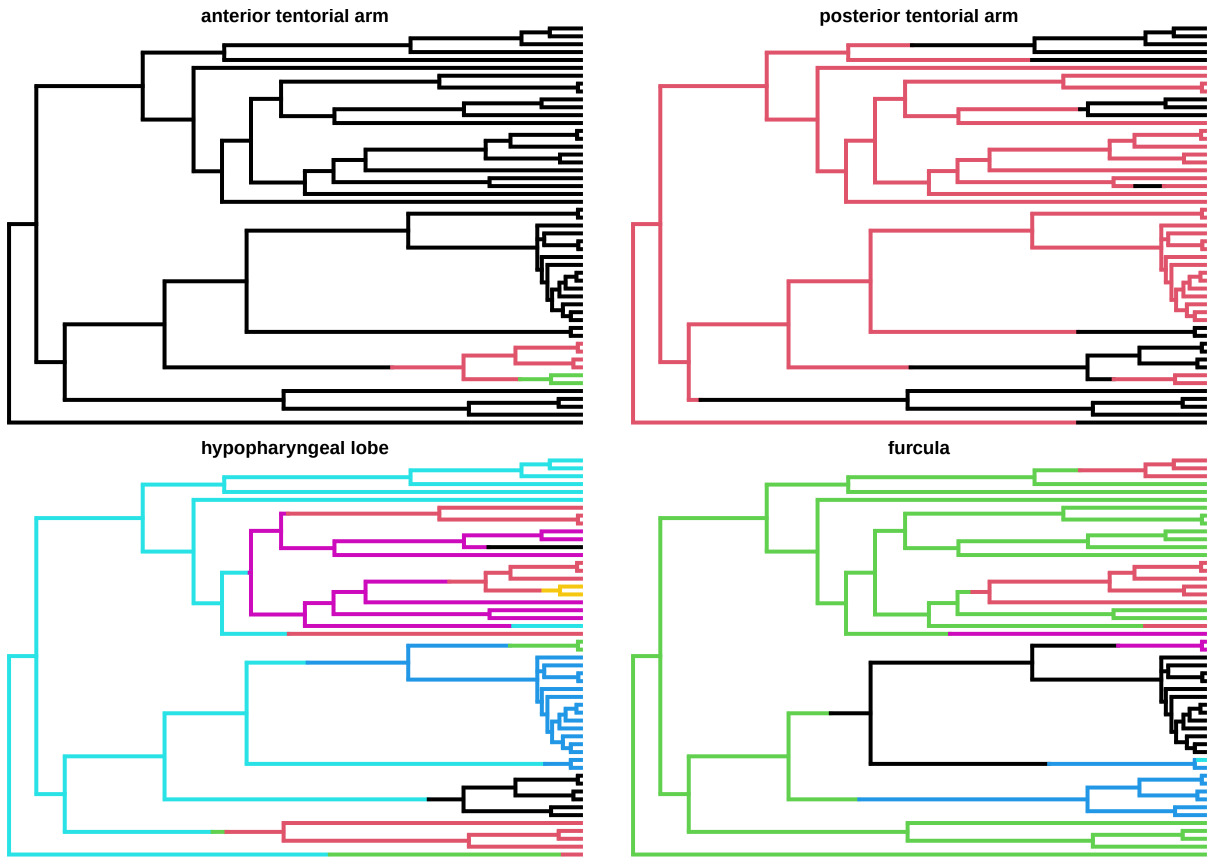
```
title(main = names(stmaps)[4], font.main = 2, cex.main = 0.75, line = -0.3)
plotSimmap(stmaps[[5]][[10]], ftype = "off")
```

```
## no colors provided. using the following legend:
##         1         2         3         4         5         6         7
##   "black" "#DF536B" "#61D04F" "#2297E6" "#28E2E5" "#CD0BBC" "#F5C710"
```

```
title(main = names(stmaps)[5], font.main = 2, cex.main = 0.75, line = -0.3)
plotSimmap(stmaps[[14]][[10]], ftype = "off")
```

```
## no colors provided. using the following legend:
##         1         2         3         4         5         6
##   "black" "#DF536B" "#61D04F" "#2297E6" "#28E2E5" "#CD0BBC"
```

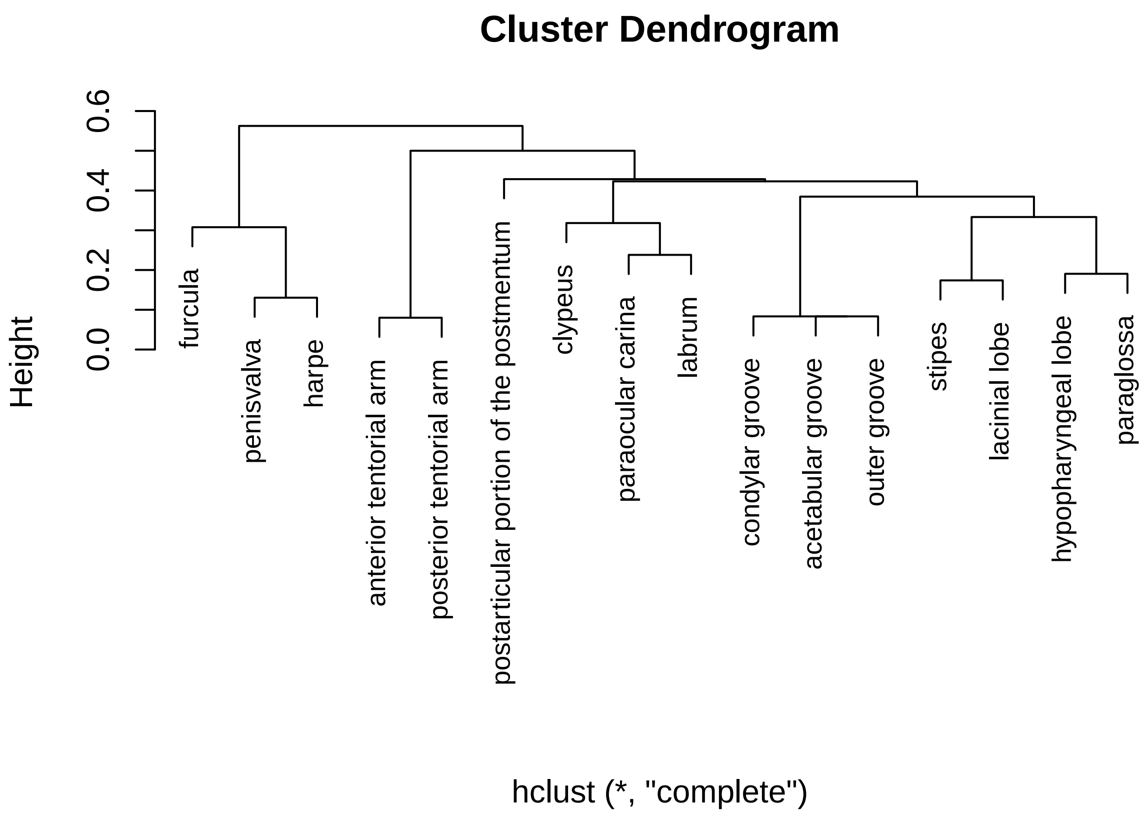
```
title(main = names(stmaps)[14], font.main = 2, cex.main = 0.75, line = -0.3)
```



STEP 6. Exploring semantic properties of the data set.

As the last step, let's build a clustering dendrogram based on the semantic similarity to investigate the relations among ontology terms annotated to the anatomical entities in our data set.

```
plot(makeTraitTree(td.comb, external = T, ONT = HAO, method = "cls"), cex = 0.8, xlab = "")
```


Cluster Dendrogram
Height
0.0
0.2
0.4
0.6
furcula
penisvalva
harpe
anterior tentorial arm
posterior tentorial arm
postarticular portion of the postmentum
clypeus
paraocular carina
labrum
condylar groove
acetabular groove
outer groove
stipes
lacinial lobe
hypopharyngeal lobe
paraglossa
hclust (*, "complete")