An introduction to *pez*

WD Pearse, MW Cadotte, J Cavender-Bares, AR Ives,
C Tucker, S Walker, & MR Helmus

June 6, 2018

Contents

1 Preamble

You can install *pez* by typing `install.packages("pez")`, and get a listing of the functions in the package by typing `library(help=pez)`. If you find any bugs, or have any feature requests for the package, please use the online tracker. Indeed, please contribute to the package using at its GitHub site—help is always welcome! If you need help, please email the user mailing list, not one of the developers.

While ***pez*** contains much novel code, it relies heavily on the *R* ecosystem. Much of the community phylogenetic metric functions are wrappers around existing code (detailed in the help files for each function); notably *caper* (**?**) and *picante* (**?**) but many others as well. Please cite the authors of these packages in your publications so that their hard-work is rewarded!

The functions within *pez* are grouped into families; thus, while there is no `pez.metric` function, there is a help-file with this title that describes over thirty different functions. Looking for the help file for each of these functions will take you to combined, 'overview' help-file.

Often, *pez* functions will return warnings (and sometimes error messages!) because certain metrics or methods are not appropriate for the kinds of data you are working with. Indeed, you will see such warnings in this vignette. These are not *bugs*; one simply cannot calculate some metrics without fully resolved phylogenies, and some models do not make sense without certain kinds of data.

2 Data formats in *pez*

pez functions work with `comparative.comm` objects (*comparative community ecology*). These are designed to help keep phylogenies, community data matrices, species trait data, and environmental data all in the same place in a format that makes it easy to work with them. They're much less scary than they sound!

Below we load *pez*, some example data that comes with it, and then make a `comparative.comm` object. You can examine the phylogeny (`tree`), community data (`comm`), and trait data (`data`) that went into making dataset for yourself, although all the data types are explained in more detail below. A phylogeny and a community matrix are all you need to make a `comparative.comm` object; everything else is optional. Below we use the `?` dataset to show *pez*'s features.

```
library(pez)
data(laja)
data <- comparative.comm(invert.tree, river.sites, invert.traits, river.env)
```

pez is conservative; if you give it trait data for only half of the species in your community data, the `comparative.comm` object will only contain data on those species that have both trait data and community data. The same goes for the phylogeny, and for sites with environmental data. *pez* will warn you about the loss of species or traits when you print the object to screen, and while it's making the comparative.comm object (unless you set the argument `warn=FALSE`).

You can also subset your comparative.comm object to exclude certain species or sites, in much the same way you can a `data.frame`. Dropping a site that contained the only instance of a species will remove it from dataset, and *pez* will not warn you about this unless you specify `[,,warn=TRUE]`.

```
site.subset <- data[1:5,]
spp.subset <- data[,1:3]
```

pez makes it easier to work with and manipulate datasets. The functions `species` and `sites` are safe ways of manipulating all the parts of your data at the same time. For example:

```
species(data)[1:2]

## [1] "Acari"         "Erpetogomphus"
```

```
species(data)[1:2] <- c("new", "names")
sites(data)[1:2] <- c("newer", "names")
data <- data[, colSums(data$comm) > 5]
traits(data)$new.trait <- rep("nonsense", nrow(traits(data)))
traits(data)$new.trait <- NULL
```

The final example above showed you can work with the internal components of a `comparative.comm` to get things done quicker, in this case removing all species that were only recorded five times or fewer in the dataset. The help entry for `cc.manip` contains more examples, and `plot.comparative.comm` is a quick plotting tool. We have also provided the `comm`, `phy`(/`tree`), `traits`, and `env` wrappers to examine and manipulate the community matrix, phylogeny, trait data, and environmental data, slots in your `comparative.comm` objects (*e.g.*, `phy(data)` would return your phylogeny). We give some examples above, and you can find more in the help file for `cc.manip`. By using these wrappers, as opposed to interacting directly with the slots in your `comparative.comm` object, you ensure your species and sites remain consistent across your data, and make your code a little easier to read. Internally, your trait data are stored in the `data` slot; this means that `comparative.comm` is compatible with the `caper` package. Anything you can do in that (*e.g.*, PGLS regression, comparative modelling and simulation) can be done using you data in *pez* without modification.

3 Community phylogenetic metrics

pez splits community phylogenetic metrics into four functions according to the scheme outlined by `?`: `pez.shape`, `pez.evenness`, `pez.dispersion`, and `pez.dissimilarity`. Shape metrics measure the structure of a community phylogeny, while evenness metrics additionally incorporate species abundances. Dispersion metrics examine whether phylogenetic biodiversity in an assemblage differs from the expectation of random assembly from a given set of species. Finally, dissimilarity measures the pairwise difference in phylogenetic biodiversity between assemblages.

You can calculate all metrics within a class at the same time (which is what we recommend), or you can pick a particular one. The intention is to make it easy to work with different community phylogenetic metrics, since each captures a different part of the structure of your data. Working with `shape`, `evenness`, and `dispersion` metrics is exactly the same, so below we only show `shape`.

```
shape.output <- pez.shape(data)

## Warning in pez.shape(data):  Cannot compute Colless' index with non-binary
tree
## Warning in x(data, dist = dist, abundance.weighted = FALSE, which.eigen =
which.eigen):  This won't work for abundance weighted (yet) - Will, look up
the weighting package
## Warning in x(data, dist = dist, abundance.weighted = FALSE, which.eigen =
which.eigen):  This won't work for abundance weighted (yet) - Will, look up
the weighting package

dim(shape.output)

## [1] 11 19

shape.output[1:3,1:3]

##       pd.pd pd.pd.ivs       psv
## CA 6426.567 -53.51115 0.6618014
## FC 4325.100  11.79561 0.7201516
## LA 4791.700 -63.29775 0.6660466
```

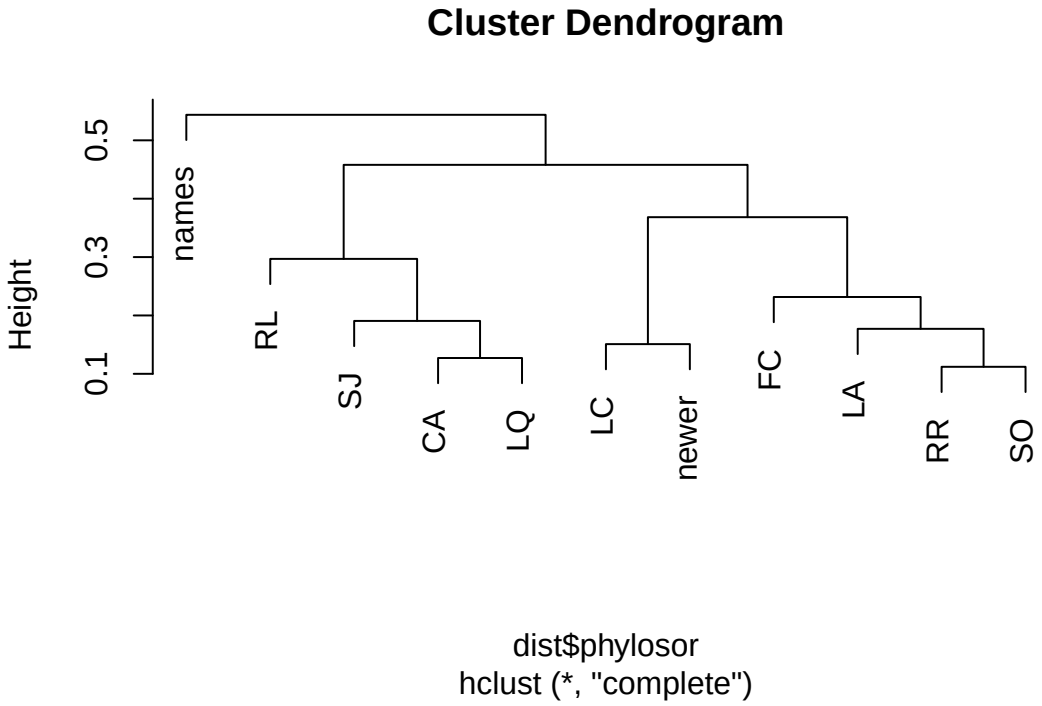
Both `shape` and `evenness` metrics, by default, only calculate the `all-quick` metrics; specifying `metric='all'` will calculate slower metrics such as Pagel's λ. These can take a *very* long time to calculate for large datasets! You can also calculate these metrics using functional traits, a square-rooted phylogeny (following `?`), or any kind of distance matrix you can put together. The argument `traitgram` can be used to set a distance matrix that mixes explanatory power from phylogeny and traits, following `?` (see below), and you can compare the output of different traitgram values. Not all metrics can meaningfully be calculated using external distance matrices, traitgrams, or square-rooted phylogenies, however, and such metrics will not be calculated (and no errors will be displayed).

```
sqrt <- pez.shape(data, sqrt.phy=TRUE)

## Warning in x(data, dist = dist, abundance.weighted = FALSE, which.eigen =
which.eigen):  This won't work for abundance weighted (yet) - Will, look up
the weighting package
## Warning in x(data, dist = dist, abundance.weighted = FALSE, which.eigen =
which.eigen):  This won't work for abundance weighted (yet) - Will, look up
the weighting package
```

```
traits <- pez.shape(data, traitgram=1) #traits alone

## Warning in x(data, dist = dist, abundance.weighted = FALSE, which.eigen =
which.eigen):  This won't work for abundance weighted (yet) - Will, look up
the weighting package
## Warning in x(data, dist = dist, abundance.weighted = FALSE, which.eigen =
which.eigen):  This won't work for abundance weighted (yet) - Will, look up
the weighting package

traits <- pez.shape(data, traitgram=c(0,0.5))#phylogeny and both

## Warning in x(data, dist = dist, abundance.weighted = FALSE, which.eigen =
which.eigen):  This won't work for abundance weighted (yet) - Will, look up
the weighting package
## Warning in x(data, dist = dist, abundance.weighted = FALSE, which.eigen =
which.eigen):  This won't work for abundance weighted (yet) - Will, look up
the weighting package
## Warning in x(data, dist = dist, abundance.weighted = FALSE, which.eigen =
which.eigen):  This won't work for abundance weighted (yet) - Will, look up
the weighting package
## Warning in x(data, dist = dist, abundance.weighted = FALSE, which.eigen =
which.eigen):  This won't work for abundance weighted (yet) - Will, look up
the weighting package

traits <- pez.shape(data, ext.dist=as.dist(cophenetic(phy(data))))

## Warning in x(data, dist = dist, abundance.weighted = FALSE, which.eigen =
which.eigen):  This won't work for abundance weighted (yet) - Will, look up
the weighting package
## Warning in x(data, dist = dist, abundance.weighted = FALSE, which.eigen =
which.eigen):  This won't work for abundance weighted (yet) - Will, look up
the weighting package
```

`dissimilarity` works slightly differently, because it returns a list of distance matrices that describe your community data. *phylosor* (**?**) is reported as a dissimilarity in *pez*: it's not the fraction of shared branch lengths, but 1- the fraction of shared branch length. This is not how it is in other packages, but remember: the function is called `dissimilarity`!

```
dist <- pez.dissimilarity(data, "phylosor")
plot(hclust(dist$phylosor))
```



It is possible to calculate any arbitrary combination of metrics using `generic.metrics`, and compare those metrics with values derived from null distributions (`generic.null`). You can get a list of the metrics you can calculate by looking at `?pez.metrics`; there are many, but they all follow the same naming (`.name`) and argument conventions. Note that you can also pass arguments (external distance matrices, etc.) to all the metrics you calculate.

```
metrics <- generic.metrics(data, c(.mpd,.pse,.ses.mpd))
#null.comparisons <- generic.null(data, c(.mpd,.pse))
metrics <- generic.metrics(data, c(.mpd,.mntd), dist=as.dist(cophenetic(phy(data))))
```

4 Eco-evolutionary regression

Calculating metric values is useful, but often we want to make statistical models. *pez* features a set of regression techniques, based on **?**, **?**, and **?**, which are described in the helpfiles for `eco.xxx.regression` and `fingerprint.regression`.

The functions described in `eco.xxx.regression` focus on relating the co-occurrence of species to species' phylogenetic (`phy`), trait (`trait`), and environmental tolerances (`env`). The environmental tolerances are based on Pianka's distance and derived from your `$env` data, while the trait distances can be based on any distance metric you can define. These are useful to explore your data, but also because the trait results are used in the *fingerprint regression* described below.

```
phy <- eco.phy.regression(data, permute=10)
trait <- eco.trait.regression(data, permute=10, method="quantile", tau=c(0.25,0.5,0.7))
trait <- eco.trait.regression(data, altogether=FALSE)
```

In the last line above, we calculated separate regressions for each trait in our dataset (returning a `eco.xxx.regression.list` object). While this isn't particularly thrilling in this dataset where we only have two traits, such a regression forms the basis of the `fingerprint.regression`. In this, we will regress the association between species co-occurrence and trait similarity for each trait against the phylogenetic conservatism of each trait. Which is a mouthful, but the papers describing it (**??**) go into more detail. *pez* does things slightly differently to these original papers, in that it uses measures of phylogenetic 'signal' instead of Mantel tests (`phy.signal`), and provides more distance matrix and regression model options for the link between co-occurrence and trait similarity.

Figure **??** may make things clearer. On the horizontal axis we move from where there is a positive correlation between co-occurrence and each trait's similarity (left) to a negative correlation (right). On the vertical axis, traits are arranged according to whether they show trait conservation (top) or lability/lack of phylogenetic inertia (bottom). Remember: each of your traits makes up one data-point in this space, but in figure **??** we have made a cartoon of a single trait in different quadrants of the graph to make things clearer. If communities are not just assembled *but have also evolved* under limiting similarity, traits should tend to lie in the blue circle. Above the blue circle, traits have evolved under niche conservatism and habitat filtering is taking place across those traits. By creating a regression such as this with your data, you are directly relating the present-day ecology of species to their evolutionary history.

```
model <- fingerprint.regression(data, eco.permute=10)
```

Once you've performed a *fingerprint regression* (the name is new to *pez*), you can examine the coefficients of each of the `$eco` and `$evo` slots in your model, which correspond to the two axes in figure **??**. Make sure you check which traits are where

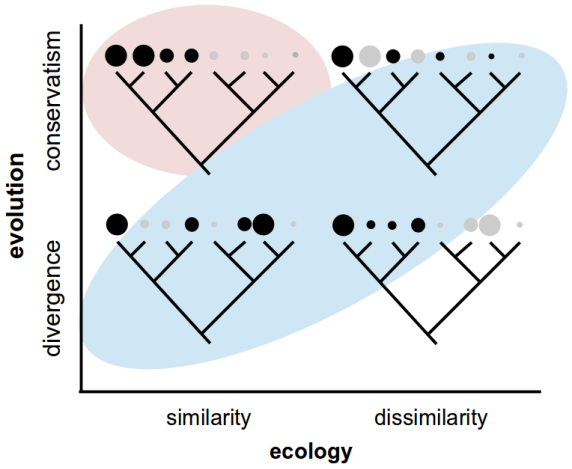


Figure 1: Overview of a fingerprint regression. A hypothetical trait is shown in each quadrant; the size of the circles represents the numerical value of the trait, and colour of the circles represents hypothetical communities. Areas of interest on the diagram are coloured to mirror the discussion in the text. The quadrants are based on the framework of (Webb2002), although the axes are continuous following the approach used by (Cavender-Bares2004). Trait patterns in the bottom left quadrant can be interpreted as environmental filtering on labile (or phylogenetically convergent) traits. Trait patterns in the top right quadrant may be interpreted as indicative of limiting similarity of phylogenetically conserved traits: these patterns emerge when functionally and phylogenetically similar species co-occur less than expected. Finally, trait patterns in the bottom right quadrant indicate that functionally similar but phylogenetically distantly related species co-occur less than expected. Such a pattern may indicate limiting similarity of distantly related species but may be difficult to interpret.

in your graph; not all traits are independent, either ecologically or evolutionarily, which could lead to bias. Examine the `$eco` and `$evo` slots in your output to see what it plotted where; you can plot these names by provide these two values and `traits(data)` to the function `text`. In our example here the results are not particularly interesting as we have only two traits!

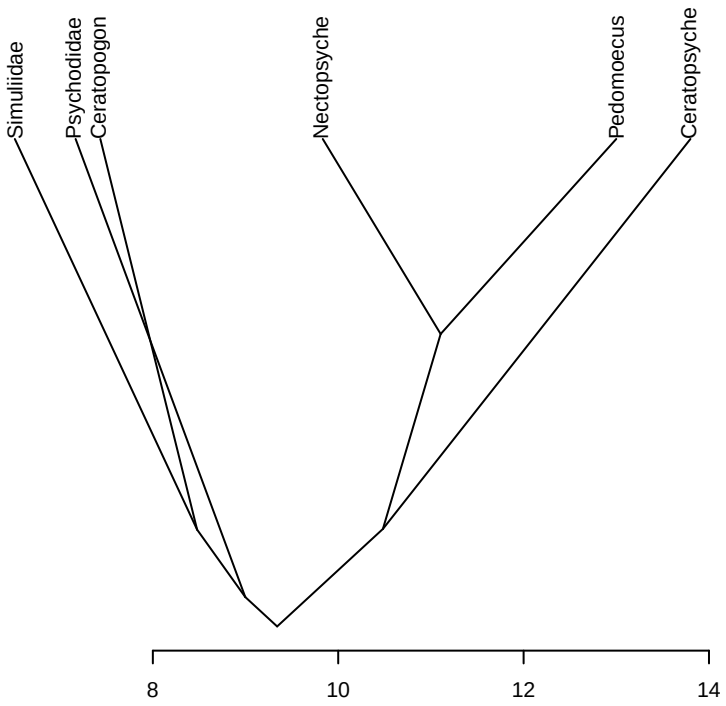
It is worth pausing to consider the interpretation of various outcomes. A positive relationship would indicate a combination of environmental filtering on functional traits with multiple independent origins and limiting similarity of phylogenetically conserved functional traits. Note that a non-significant fingerprint regression does not indicate that the data are not interpretable. For example, traits concentrated in the top left quadrant (red circle) could most likely be interpreted in terms of niche conservatism because species with phylogenetically and functionally similar traits are found in the same habitats or communities. A negative relationship, in contrast, would be difficult to interpret because it would indicate a combination of niche conservatism (environmental filtering on conserved functional traits) and trait patterns of dissimilar labile traits within communities that could be generated any number of ways or result from stochastic processes. A further complication, of course, is that different measures of phylogenetic signal mean very different things; ensure that you understand how to interpret the metric you are using!

An important caveat is that multiple processes may contribute to observed patterns making interpretation difficult (**?**). However, examination of which traits fall out in different quadrants can help provide functional interpretations of community patterns. For example, if disease resistance traits are highly conserved and occur in the top right quadrant where close relatives do not co-occur in communities, this would help implicate Janzen-Connell mechanisms as contributing to the community structure; testing such a hypothesis with experimental methods would be an important next step. Alternatively, if fire tolerance traits were found in the bottom left quadrant, a plausible interpretation would be that there had been repeated evolutionary origins of fire tolerance traits, and environmental filtering (by fire) operated on these phylogenetically convergent traits. Both of these scenarios could occur simultaneously and both would help explain an overdispersed (or even) pattern of community phylogenetic structure. In contrast, if fire tolerance traits appeared in the top left quadrant, one would implicate niche conservatism for fire-tolerance traits, and it would be reasonable to assume that fire acted as an environmental filter causing fire tolerant species to assemble in fire-dominated habitats and fire-intolerant species to assemble in habitats lacking fire.

5 Functional phylogenetic distances and traitgrams

Taxa differ both functionally and phylogenetically, a fact that is clearly illustrated using traitgrams (**??**). A traitgram of a `comparative.comm` object can be made using the `traitgram.cc` (a wrapper for the `traitgram` function in `picante`). Here is a traitgram for a particular assemblage of species against the `length` trait,

```
assemblage <- c("Nerophilus", "Hydroptila", "Psorophora",
                "Simuliidae", "Psychodidae", "Ceratopogon",
                "Nectopsyche", "Pedomoecus", "Ceratopsyche")
dataAssemblage <- data[, species(data) %in% assemblage]
traitgram.cc(dataAssemblage, "length")
```



The traitgram plots the phylogenetic time on the y-axis and `length` on the x-axis. Note that some taxa are very distantly related but nevertheless have converged to very similar trait values (e.g. Psorophora and Simuliidae), whereas others are closely

related but functionally very dissimilar (e.g. Nerophilus and Nectopsyche).

? argued that distances in this 'traitgram space' might provide a better indication of ecological differences between taxa, than if either function or phylogenetic data were used in isolation. If FD_{ij} and PD_{ij} are functional and phylogenetic distances between species i and j, the functional-phylogenetic distance between i and j is (?),

$$\left((1-a)FD_{ij}^{p} + aPD_{ij}^{p}\right)^{1/p} \tag{1}$$

where a is the phylogenetic weighting parameter and p is the exponent for the p-norm combination of phylogenetic and functional distances (e.g. $p = 2$ gives a Euclidean combination). This distance matrix can be computed using the `funct.phylo.dist` function. For example, for $a = 0.5$, and $p = 2$, we have,

```
fpd.data <- funct.phylo.dist(data, phyloWeight = 0.5, p = 2)
```

One use of these distance matrices is in community randomisation tests (?), an example of which can be computed using the following code:

```
ses.mfpd.data <- .ses.mpd(data, dist=fpd.data)
head(ses.mfpd.data)[,c("ntaxa", "mpd.obs", "mpd.obs.p")]

##       ntaxa   mpd.obs mpd.obs.p
## CA       18 0.5553092 0.1718282
## FC       10 0.5802422 0.4745255
## LA       12 0.5525134 0.2427572
## LC       11 0.6463215 0.8831169
## LQ       20 0.5535377 0.1188811
## names     7 0.6683667 0.8531469
```

6 Phylogenetic Generalised Linear Mixed Models

Phylogenetic Generalised Linear Models (PGLMMs) are powerful tools that permit detailed tests of what structures ecological communities. They were originally intended to examine phylogenetic patterns in community composition, sensitivity to environmental gradients (and variation in species' sensitivity based on trait data), and species co-occurrence (?). They were later extended to model interaction networks (?), and both of these methods are flexibly implemented in *pez* such that they can take any kind of correlation structure, whether it be among species or among

sites (*e.g.*, spatial auto-correlation). Both species occurrence and abundance can be modelled, depending on the error `family` specified.

What follows is a simple example of how to simulate, and then analyse, the most basic kind of PGLMM. The help file for PGLMM is quite thorough; the first section provides a helpful reference for how to quickly fit different kinds of models once you are familiar with the approach. What follows is a greatly condensed version of the second section; the help file contains much more information of why and how each step of the simulation works.

```
# Basic parameters
nspp <- 15; nsite <- 10
# Fixed effects
beta0 <- beta1 <- 0
# Random effects' magnitudes
sd.B0 <- sd.B1 <- 1

# Generate environmental site variable
X <- matrix(1:nsite, nrow=1, ncol=nsite)
X <- (X - mean(X))/sd(X)

# Simulate phylogeny
phy <- compute.brlen(rtree(nspp), method = "Grafen", power = 0.5)
# Standardise phy. covariance matrix
Vphy <- vcv(phy); Vphy <- Vphy/(det(Vphy)^(1/nspp))

# Generate phylogenetic signal in parameters
iD <- t(chol(Vphy))
b0 <- beta0 + iD %*% rnorm(nspp, sd = sd.B0)
b1 <- beta1 + iD %*% rnorm(nspp, sd = sd.B1)

#Simulate presences
y <- matrix(outer(b0, array(1, dim = c(1, nsite))), nrow = nspp, ncol = nsite) + matrix(out
e <- rnorm(nspp * nsite, sd = 0)
y <- y + matrix(e, nrow = nspp, ncol = nsite)
y <- matrix(y, nrow = nspp * nsite, ncol = 1)
Y <- rbinom(n = length(y), size = 1, prob = exp(y)/(1 + exp(y)))
Y <- matrix(Y, nrow = nspp, ncol = nsite)

#Neat up the data to show structure
rownames(Y) <- 1:nspp; colnames(Y) <- 1:nsite
```

Arguably the most important thing for you to understand about the above is what `Y` represents: it is a matrix where each species has its own row, and each site its own column (see the final line). When we take the transpose of it, we have what we need to create a `comparative.comm` object.

We will now transform that data such that it is in a 'long format'—each element in the community matrix will become a row in a `data.frame` and we will perform a regression (a PGLMM, to be precise!) on that data. This long format is important because PGLMM is sufficiently flexible that it can be fitted to data that don't fit into the `comparative.comm` format. An example is interaction-network data, of which there is an example in the help file. Calling `as.data.frame` on a `comparative.comm` object will convert your data into this long-format structure for you.

```
# Transform data into 'long' format
# - Occurrence data
YY <- matrix(Y, nrow = nspp * nsite, ncol = 1)
# - Environmental (site) data
XX <- matrix(kronecker(X, matrix(1, nrow = nspp, ncol = 1)), nrow = nspp * nsite, ncol = 1)
site <- matrix(kronecker(1:nsite, matrix(1, nrow = nspp, ncol = 1)), nrow = nspp * nsite, n
sp <- matrix(kronecker(matrix(1, nrow = nsite, ncol = 1), 1:nspp), nrow = nspp * nsite, nco
# - Make data.frame with all data
dat <- data.frame(Y = YY, X = XX, site = as.factor(site), sp = as.factor(sp))

# Setup random effects
# - 1: random intercept - species independent
re.1 <- list(1, sp = dat$sp, covar = diag(nspp))
# - 2: random intercept - species phylogenetically covary
re.2 <- list(1, sp = dat$sp, covar = Vphy)
# - 3: random slope - species independent
re.3 <- list(dat$X, sp = dat$sp, covar = diag(nspp))
# - 4: random intercept - species covary
re.4 <- list(dat$X, sp = dat$sp, covar = Vphy)
# (Random effect for site)
re.site <- list(1, site = dat$site, covar = diag(nsite))

# Fit model!
model <- communityPGLMM(Y ~ X, data = dat, family = "binomial", sp = dat$sp, site = dat$sit
```

There are three steps: transforming the data (as discussed above), setting up the random effects, and finally fitting the model. Fitting the model is comparatively easy; make sure you specify your random effects, and if you're modelling presence/absence

data set the `family` to `binomial` as you might in any GLM.

The random effects describe how likely species are to occur in the sites based on a regression of the environmental variable we simulated in the first step. There are random effects for the intercept of the regression (split into independent and phylogenetically shared variance for each species) and the slope of their relationship (also with phylogenetic and non-phylogenetic components). There is also a random effect for site-level variation. The first section of the help file (`?pglmm`) gives examples of how to set up other kinds of hypotheses involving species traits. These map onto the original **?** paper that describes PGLMMs.

Your choice of random effects determines the hypotheses you are testing with your data—PGLMMs are attractive mostly because of this flexibility. If you are confused over what random effects to fit, go back to the original papers (**??**), and maybe check a more recent (but basic) overview (**?**). Powerful methods mean you get to make powerful choices about what you analyse: this is a good thing!

7 Simulation

A good simulation is one that does exactly what you want it to do, and *pez* provides a number of simulation functions that may be useful to you as (1) tools, or (2) starting points for your own simulations.

`scape` allows you to repeat the analysis of **?**, simulating the assembly of species across a landscape given phylogenetically structured assembly. The parameters are complex, but they can generate some useful expected distributions, and give you a feel for regional assembly.

Alternatively, you can model the evolution of species and, at the same time, their assembly through a community. The only problem here is that the models are much simpler, but hopefully they are tune-able to your liking! Explore the `sim.meta` and `sim.phy` functions to find out more.

Finally, you can also simulate sets of communities under phylogenetic and/or trait repulsion, using `sim.trait.asm`. These communities are excellent for use as null models to compare with your own data, and as such they take a `comparative.comm` object as an argument to generate communities that match your own data.

References

Ackerly, D. (2009) Conservatism and diversification of plant functional traits: evolutionary rates versus phylogenetic signal. *Proc. Natl Acad. Sci.* **106**, 19699–19706.

Bryant, J.A., Lamanna, C., Morlon, H., Kerkhoff, A.J., Enquist, B.J. & Green, J.L. (2008) Microbes on mountainsides: contrasting elevational patterns of bacterial and plant diversity. *Proceedings of the National Academy of Sciences* **105**, 11505–11511, URL `http://www.pnas.org/content/105/suppl.1/11505.abstract`.

Cadotte, M., Albert, C. & Walker, S.C. (2013) The ecology of differences: assessing community assembly with trait and evolutionary distances. *Ecology Letters* **16**, 1234–1244.

Cavender-Bares, J., Ackerly, D.D., Baum, D.A. & Bazzaz, F.A. (2004) Phylogenetic overdispersion in Floridian oak communities. *The American Naturalist* **163**, 823–43.

Cavender-Bares, J. & Wilczek, A. (2003). Integrating micro- and macroevolutionary processes in community ecology. *Ecology*, **84**, 592-597.

Cavender-Bares, J., A. Keen & B. Miles (2006) Phylogenetic structure of Floridian plant communities depends on taxonomic and spatial scale. *The American Naturalist* **87**, S109–S122.

Cavender-Bares, J., Kozak, K., Fine, P. & Kembel, S. (2009). The merging of community ecology and phylogenetic biology. *Ecology Letters*, **12**, 693-715.

Evans, M.E.K., Smith, S.A., Flynn, R.S. & Donoghue, M.J. (200) Climate, niche evolution, and diversification of the "Bird-cage" evening primroses (Oenothera, sections Anogra and Kleinia). *Am. Nat.* **173**, 225–240.

Helmus, M.R. & Ives, A.R. (2012) Phylogenetic diversity-area curves. *Ecology* **93**, S31–S43.

Helmus, M.R., Mercado-Silva, N., & Zanden, M.J.V. (2014) Subsidies to predators, apparent competition and the phylogenetic structure of prey communities *Oceologia* **173**, 997–1007.

Ives, A.R. & Helmus, M.R. (2011) Generalized linear mixed models for phylogenetic analyses of community structure *Ecological Monographs* **81**, 511–525.

Kembel, S.W., Cowan, P.D., Helmus, M.R., Cornwell, W.K., Morlon, H., Ackerly, D.D., Blomberg, S.P. & Webb, C.O. (2010) Picante: R tools for integrating phylogenies and ecology. *Bioinformatics* **26**, 1463–1464.

Letten, A.D. & Cornwell, W.K. (2014) Trees, branches and (square) roots: why evolutionary relatedness is not linearly related to functional distance. *Methods in Ecology and Evolution*.

Orme, D., Freckleton, R., Thomas, G., Petzoldt, T., Fritz, S., Isaac, N. & Pearse, W.D. (2013) *caper: comparative analyses of phylogenetics and evolution in R*. URL `http://CRAN.R-project.org/package=caper`, r package version 0.5.2.

Pearse, W.D., Cavender-Bares, J., Puvis, A. & Helmus, M.R. (2014) Metrics and models of community phylogenetics. *Modern Phylogenetic Comparative Methods and their Application in Evolutionary Biology—Concepts and Practice* (ed. L.Z. Garamszegi), Springer-Verlag, Berlin, Heidelberg.

Rafferty, N.E. & Ives, A.R. (2013) Phylogenetic trait-based analyses of ecological networks *Ecology* **94**, 2321–2333.

Webb, C.O., Ackerly, D.D., McPeek, M.A. & Donoghue M.J. (2002). Phylogenies and community ecology. *Annual Review of Ecology and Systematics*, **33**, 475-505.