The `adephylo` package

Thibaut Jombart

December 15, 2008

Contents

1 Introduction

This document describes the `adephylo` package for the R software. `adephylo` aims at implementing exploratory methods for the analysis of phylogenetic comparative data, i.e. biological traits measured for taxa whose phylogeny is also provided. Procedures implemented in this package rely on exploratory data analysis. They include data visualization and manipulation, tests for phylogenetic autocorrelation, multivariate analysis, computation of phylogenetic proximities and distances, and modelling phylogenetic signal using orthonormal bases.

These methods can be used to visualize, test, remove or investigate the phylogenetic signal in comparative data. The purpose of this document is to provide a general view a the main functionalities of `adephylo`, and to show how this package can be used along with `ape`, `phylobase` and `ade4` to analyse comparative data.

2 First steps into adephylo

2.1 Data representation: why we are not reinventing the weel

Data representation can be defined as the way data are stored in a software (R, in our case). Technically, they are classes of objects containing the information. In the case of phylogeny, and comparative data, very efficient data representation are already defined in other packages. Hence, it made much more sense using directly objects from these classes.

Phylogenies are best represented in Emmanuel Paradis's ape package (http://ape.mpl.ird.fr/), as the class phylo. Note that as ape is by far the largest package dedicated to phylogeny, using the phylo class assures a good interoperability of data. This class is defined in an online document: http://ape.mpl.ird.fr/misc/FormatTreeR_28July2008.pdf.

However, data that are to be analyzed in adephylo do not only contain trees, but also traits associated to the tips of a tree. The package phylobase (http://r-forge.r-project.org/projects/phylobase/) is a collaborative effort designed to the handling of such data. Its representation of phylogenies is very similar to that of ape: the class phylo4 basically is an extension of phylo class into formal (S4) class. More interestingly, the S4 class phylo4d can be used to store a tree and data associated to tips, internal nodes, or even edges of a tree. Classes of phylobase are described in a vignette of the package, accessible by typing:

```
> vignette("phylobase", package = "phylobase")
```

As trees and comparative data are already handled by ape and phylobase, no particular data format shall be defined in adephylo. In particular, we are no longer using phylog objects, which were used to represent phylogenies in ade4. This class is now deprecated, but all previous functionalities available for phylog objects have been re-implemented and – in some cases – improved in adephylo.

2.2 Installing the package

What is tricky here is that a vignette is basically available once the package is installed. Assuming you got this document before installing the package, here are some clues about installing adephylo.

First of all, adephylo depends on other packages, being methods, ape, phylobase, and ade4. These dependencies are mandatory, that is, you actually need to have these packages installed (with or without their dependencies) before using adephylo. Also, it is better to make sure you are using the latest versions of these packages. This can be achieved by typing update.packages,

or (better for ade4 and phylobase) by installing devel versions from R-Forge (http://r-forge.r-project.org/). In all cases, the latest version of adephylo can be found from http://r-forge.r-project.org/R/?group_id=303.

When loading the package, dependencies are also loaded:

```
> library(adephylo)
```

Note that possibly conflicting, deprecated functions or datasets from ade4 are masked by adephylo. In case the converse would occur (i.e. deprecated function masking a function of adephylo), one can refer to the 'good' version of a function by adding the prefix adephylo:: to the function, without space. Hence, it is possibly to coerce the version of a masked function, using a kludge like:

```
> cat("\n=== Old - deprecated- version ===\n")

=== Old - deprecated- version ===

> orthogram <- ade4::orthogram
> args(orthogram)

function (x, orthobas = NULL, neig = NULL, phylog = NULL, nrepet = 999,
    posinega = 0, tol = 1e-07, na.action = c("fail", "mean"),
    cdot = 1.5, cfont.main = 1.5, lwd = 2, nclass, high.scores = 0,
    alter = c("greater", "less", "two-sided"))
NULL

> cat("\n=== New version === \n")

=== New version ===

> orthogram <- adephylo::orthogram
> args(orthogram)

function (x, tre = NULL, orthobas = NULL, prox = NULL, nrepet = 999,
    posinega = 0, tol = 1e-07, cdot = 1.5, cfont.main = 1.5,
    lwd = 2, nclass, high.scores = 0, alter = c("greater", "less",
        "two-sided"))
NULL
```

Luckily, this should not be required as long as one is not playing with loading and unloading ade4 once adephylo is loaded.

2.3 Getting started

All the material of the package is summarized in a manpage accessible by typing:

```
> `?`(adephylo)
```

Note that a html version may be preferred to browse easily the content of adephylo; this is accessible by typing:

```
> help("adephylo", package = "adephylo", html = TRUE)
```

To revert help back to text mode, simply type:

```
> options(htmlhelp = FALSE)
```

2.4 Putting data into shape

While this is not the purpose of this document to go through the details of `phylo`, `phylo4` and `phylo4d` objects, we shall show briefly how these objects can be obtained.

2.4.1 Making a `phylo` object

The simplest way of turning a tree into a `phylo` object is using ape's function `read.tree`. This function reads a tree with the Newick (or 'parentetic') format, from a file (default, argument `file`) of from a character string (argument `text`).

```
> data(ungulates)
> ungulates$tre

[1] "((Antilocapra_americana,((Gorgon_taurinus,Oryx_leucoryx)W1,(Taurotragus_livingstoni,Tautragus_oryx)W2,(Gazel

> myTree <- read.tree(text = ungulates$tre)
> myTree

Phylogenetic tree with 18 tips and 13 internal nodes.

Tip labels:
        Antilocapra_americana, Gorgon_taurinus, Oryx_leucoryx, Taurotragus_livingstoni, Tautragus_oryx, Gazella_t
        Node labels:
        Root, W11, W10, W1, W2, W7,...

Rooted; no branch lengths.

> plot(myTree, main = "ape's plotting of a tree")
```

It is easy to convert `ade4`'s `phylog` objects to a `phylo`, as `phylog` objects store the Newick format of the tree in the `$tre` component.

Note that `phylo` trees can also be constructed from alignements (see `read.GenBankdist.dna`, `read.dna`, `dist.dna`, `nj`, `bionj`, and `mlphylo`, all in `ape`), or even simulated (for instance, see `rtree`).

Also note that, if needed, conversion can be done back and forward with `phylo4` trees:

```
> temp <- as(myTree, "phylo4")
> class(temp)

[1] "phylo4"
attr(,"package")
[1] "phylobase"

> temp <- as(temp, "phylo")
> class(temp)

[1] "phylo"

> all.equal(temp, myTree)

[1] TRUE
```

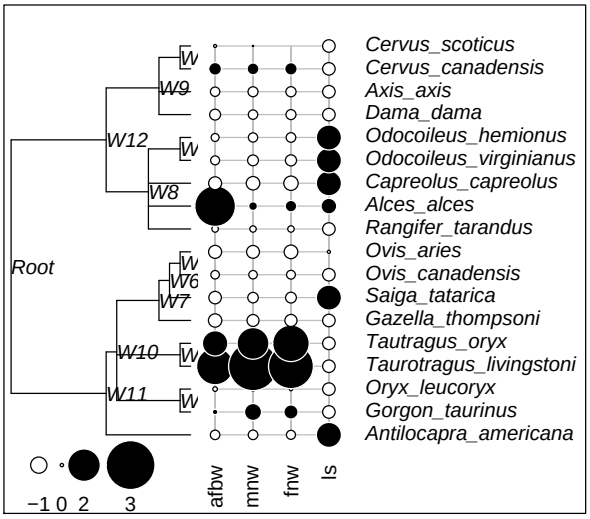
2.4.2 Making a `phylo4d` object

`phylo4d` objects are S4 objects, and are thus created in a particular way. The most immediate way of creating a `phylo4d` object is to call to the constructor, also named `phylo4d`. This is a function that takes two arguments: a tree (`phylo` or `phylo4` format) and a data.frame containing data, for tips by default (see `?phylo4d` for more information). Here is an example:

```
> ung <- phylo4d(myTree, ungulates$tab)
> class(ung)

[1] "phylo4d"
attr(,"package")
[1] "phylobase"

> table.phylo4d(ung)
```



Note that the constructor checks the consistency of the names used for the tips of the tree and for the rows of the data.frame. Inconsistencies issue an error. To override this behaviour, one can specify `use.tip.names=FALSE`. However, this can be tricky: often, mismatches between names can indicate that data are not sorted adequately; moreover, object created with such mismatches will often be invalid objects, and may issue errors in further analyses.

Data are stored inside the slot `@tip.data` of the object. They can be accessed either via this slot (in our example, `ung@tip.data`), or using the function `tdata`:

```
> x <- tdata(ung)
> head(x)
```

```
                          afbw   mnw   fnw  ls
Antilocapra_americana    50586  3832  3908 1.8
Gorgon_taurinus         165000 18600 14500 1.0
Oryx_leucoryx            87700  6840  6490 1.0
Taurotragus_livingstoni 405000 36300 28500 1.0
Tautragus_oryx          316000 26800 24700 1.0
Gazella_thompsoni        21300  2500  2500 1.0
```

3 Exploratory data analysis

3.1 Quantifying and testing phylogenetic signal

In this document, the terms 'phylogenetic signal' and 'phylogenetic autocorrelation' are used interchangeably. They refer to the fact that observations of traits are not independent in closely related taxa. Several procedures are implemented by `adephylo` to measure and test phylogenetic autocorrelation.

3.1.1 Moran's I

The function `moran.idx` computes Moran's I, the most widely-used autocorrelation measure. It can also provide additionnal information (argument `addInfo`), being the null value of I (i.e., the expected value in absence of phylogenetic autocorrelation), and the range of variation of I. It requires the degree of relatedness of tips on the phylogeny to be modelled by a matrix of phylogenetic proximities. Such a matrix can be obtained using different methods implemented by the function `proxTips`.

```
> W <- proxTips(myTree, met = "Abouheif")
> moran.idx(tdata(ung)$afbw, W)
```

```
[1] 0.1132682
```

```
> moran.idx(ung$tip.data[, 1], W, addInfo = TRUE)
```

```
[1] 0.1132682
attr(,"I0")
[1] -0.05882353
attr(,"Imin")
[1] -0.5217391
attr(,"Imax")
[1] 1.000699
```

From here, it is quite straightforward to build a non-parametric test based on Moran's I. For instance (taken from `?moran.idx`):

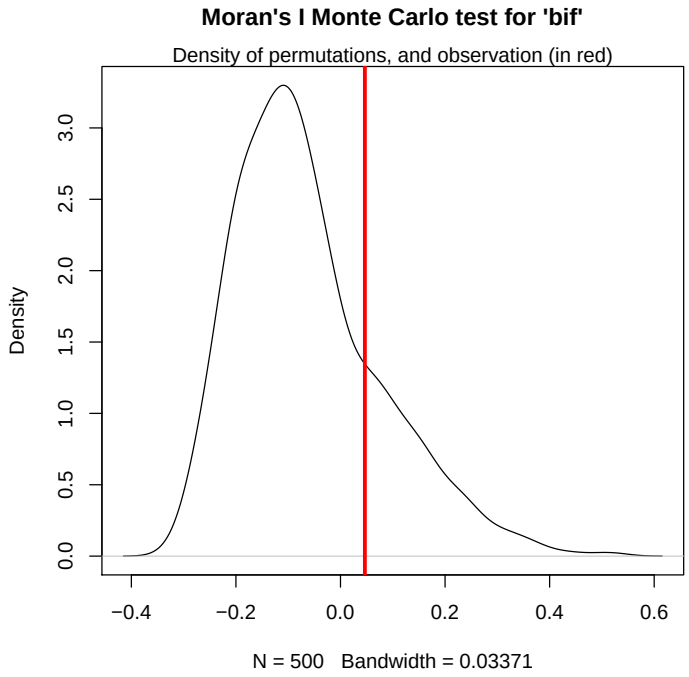
```
> afbw <- tdata(ung)$afbw
> sim <- replicate(499, moran.idx(sample(afbw), W))
> sim <- c(sim, moran.idx(afbw, W))
> cat("\n=== p-value (right-tail) === \n")
```

```
=== p-value (right-tail) ===


> pval <- mean(sim >= sim[1])
> pval


[1] 0.912


> plot(density(sim), main = "Moran's I Monte Carlo test for 'bif'")
> mtext("Density of permutations, and observation (in red)")
> abline(v = sim[1], col = "red", lwd = 3)
```



Here, afbw is likely not phylogenetically autocorrelated.

3.1.2 Abouheif's test

The test of Abouheif (see reference in `?abouheif.moran`) is designed to test the existence of phylogenetic signal. In fact, it has been shown that this test amounts to a Moran's I test with a particular proximity matrix (again, see references in the manpage). The implementation in `abouheif.moran` proposes different phylogenetic proximities, using by default the original one.