Special features of phangorn (Version 1.1-2)

Klaus P. Schliep*

September 20, 2010

1 Introduction

This document illustrates some of the *phangorn* some specialised features which are useful but maybe not as wellknown or just not (yet) described elsewhere. This is mainly interesting for someone who wants to explore different models or set up some simulation studies. We show how to construct data objects for different character states other than nucleotides or amino acids or how to set up different models to estimate transition rate.

2 User defined data formats

The vignette *Trees* describes in detail how to estimate phylogenies from nucleotide or amino acids.

To better understand how to define our own data type it is useful to know a bit about the internal representation of `phyDat` objects. The internal representation of `phyDat` object is very similar to `factor` objects.

As an example we will show here several possibilities to define nucleotide data with gaps defined as a fifth state. When the number of gaps is low and the are missing at random this may be not important.

Let assume we have given a matrix where each row contains a character vector of a taxonomixcal unit:

```
library(phangorn)
data = matrix(c("r", "a", "y", "g", "g", "a", "c", "-", "c",
     "t", "c", "g", "a", "a", "t", "g", "g", "a", "t", "-", "c",
     "t", "c", "a", "a", "a", "t", "-", "g", "a", "c", "c", "c",
     "t", "?", "g"), dimnames = list(c("t1", "t2", "t3"), NULL),
     nrow = 3, byrow = TRUE)
data
```

*mailto:kschliep@snv.jussieu.fr

```
   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
t1 "r"  "a"  "y"  "g"  "g"  "a"  "c"  "-"  "c"  "t"   "c"   "g"
t2 "a"  "a"  "t"  "g"  "g"  "a"  "t"  "-"  "c"  "t"   "c"   "a"
t3 "a"  "a"  "t"  "-"  "g"  "a"  "c"  "c"  "c"  "t"   "?"   "g"
```

Normally we would transform this matrix into an phyDat object and gaps are handled as ambiguious character like "?".

```
gapsdata1 = phyDat(data)
gapsdata1
3 sequences with 12 character and 11 different site patterns.
The states are a c g t
```

Now we will define a "USER" defined object and have to supply a vector levels of the character states for the new data, in our case the for nucleotide states and the gap. Additional we can define ambigious states which can be any of the states.

```
gapsdata2 = phyDat(data, type = "USER", levels = c("a", "c",
     "g", "t", "-"), ambiguity = c("?", "n"))
gapsdata2
3 sequences with 10 character and 9 different site patterns.
The states are a c g t -
```

This is not yet what we wanted as two sites of our alignment, which contain the abigious characters "r" and "y", got deleted. To define ambigious characters like "r" and "y" explicitly we have to supply a contrast matrix similar to contrasts for factors.

```
contrast = matrix(data = c(1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
     1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0,
     0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1), ncol = 5, byrow = TRUE)
dimnames(contrast) = list(c("a", "c", "g", "t", "r", "y", "-",
     "n", "?"), c("a", "c", "g", "t", "-"))
contrast
  a c g t -
a 1 0 0 0 0
c 0 1 0 0 0
g 0 0 1 0 0
t 0 0 0 1 0
r 1 0 1 0 0
y 0 1 0 1 0
- 0 0 0 0 1
n 1 1 1 1 0
? 1 1 1 1 1
```

```
gapsdata3 = phyDat(data, type = "USER", contrast = contrast)
gapsdata3
3 sequences with 12 character and 11 different site patterns.
The states are a c g t -
```

Here we defined "n" as a state which can be any nucleotide but not a gap "-" and "?" can be any state including a gap.

These data can be used in all functions available in *phangorn* to compute distance matrices or perform parsimony and maximum likelihood analysis.

3 Estimation of non-standard transition rate matrices

In the last section˜2 we described how to set up user defined data formats. Now we describe how to estimate transition matrices with pml.

Again for nucleotide data the most common models can be called directly in the optim.pml function (e.g. "JC69", "HKY", "GTR" to name a few). Table 2 lists all the available nucleotide models, which can estimated directly in `optim.pml`. For amino acids several transition matrices are available ("WAG", "JTT", "Dayhoff" and "LG") or can be estimated with optim.pml, e.g. Mathews et al. (2010) [1] used this function to estimate a phytochrome amino acid transition matrix.

We will now show how to estimate a rate matrix with different transition (α) and transversion ratio (β) and a fixed rate to the gap state (γ) - a kind of Kimura two-parameter model (JC69) for nucleotide data with gaps as fifth state (see table ˜1).

	a	c	g	t	-
a					
c	β				
g	α	β			
t	β	α	β		
-	γ	γ	γ	γ	

Table 1: Rate matrix K to optimise.

The parameters subs accepts a vector of consecutive integers and one (or more) element has to be zero (these gets the reference rate of 1).

```
tree = unroot(rtree(3))
fit = pml(tree, gapsdata3)
fit = optim.pml(fit, optQ = TRUE, subs = c(1, 0, 1, 2, 1, 0,
     2, 1, 2, 2), control = pml.control(trace = 0))
fit
```

```
 loglikelihood: -33.03651

unconstrained loglikelihood: -28.43259

Rate matrix:
             a            c            g            t         -
a 0.000000e+00 1.382019e-08 1.000000e+00 1.382019e-08 0.6898839
c 1.382019e-08 0.000000e+00 1.382019e-08 1.000000e+00 0.6898839
g 1.000000e+00 1.382019e-08 0.000000e+00 1.382019e-08 0.6898839
t 1.382019e-08 1.000000e+00 1.382019e-08 0.000000e+00 0.6898839
- 6.898839e-01 6.898839e-01 6.898839e-01 6.898839e-01 0.0000000

Base frequencies:
0.2 0.2 0.2 0.2 0.2
```

Here are some conventions how the models are estimated:

If a model is supplied the base frequencies bf and rate matrix Q are optimised according to the model (nucleotides) or the adequate rate matrix and frequencies are chosen (for amino acids). If optQ=TRUE and neither a model or subs are supplied than a symmetric (optBf=FALSE) or reversible model (optBf=TRUE, i.e. the GTR for nucleotides) is estimated. This can be slow if the there are many character states, e.g. for amino acids.

4 Generating trees

phangorn has a few functions to generate trees, which may are interesting for simulation studies. `allTrees` computes all possible bifurcating tree topologies either rooted or unrooted for up to 10 taxa (one has to keep in mind that the number of trees is growing exponentially).

```
trees = allTrees(5)


par(mfrow = c(3, 5), mar = c(2, 2, 2, 2) - 2)
for (i in 1:15) plot(trees[[i]], cex = 1, type = "u")
```

`rNNI` and `rSPR` generate trees which are a defined number of NNI (nearest neighbor interchange) or SPR (subtree pruning and regrafting) away.

model	optQ	optBf	subs
JC	FALSE	FALSE	$c(0,0,0,0,0,0)$
F81	FALSE	TRUE	$c(0,0,0,0,0,0)$
K80	TRUE	FALSE	$c(0,1,0,0,1,0)$
HKY	TRUE	TRUE	$c(0,1,0,0,1,0)$
TrNe	TRUE	FALSE	$c(0,1,0,0,2,0)$
TrN	TRUE	TRUE	$c(0,1,0,0,2,0)$
TPM1	TRUE	FALSE	$c(0,1,2,2,1,0)$
K81	TRUE	FALSE	$c(0,1,2,2,1,0)$
TPM1u	TRUE	TRUE	$c(0,1,2,2,1,0)$
TPM2	TRUE	FALSE	$c(1,2,1,0,2,0)$
TPM2u	TRUE	TRUE	$c(1,2,1,0,2,0)$
TPM3	TRUE	FALSE	$c(1,2,0,1,2,0)$
TPM3u	TRUE	TRUE	$c(1,2,0,1,2,0)$
TIM1e	TRUE	FALSE	$c(0,1,2,2,3,0)$
TIM1	TRUE	TRUE	$c(0,1,2,2,3,0)$
TIM2e	TRUE	FALSE	$c(1,2,1,0,3,0)$
TIM2	TRUE	TRUE	$c(1,2,1,0,3,0)$
TIM3e	TRUE	FALSE	$c(1,2,0,1,3,0)$
TIM3	TRUE	TRUE	$c(1,2,0,1,3,0)$
TVMe	TRUE	FALSE	$c(1,2,3,4,2,0)$
TVM	TRUE	TRUE	$c(1,2,3,4,2,0)$
SYM	TRUE	FALSE	$c(1,2,3,4,5,0)$
GTR	TRUE	TRUE	$c(1,2,3,4,5,0)$

Table 2: DNA models available in phangorn and how they are defined.

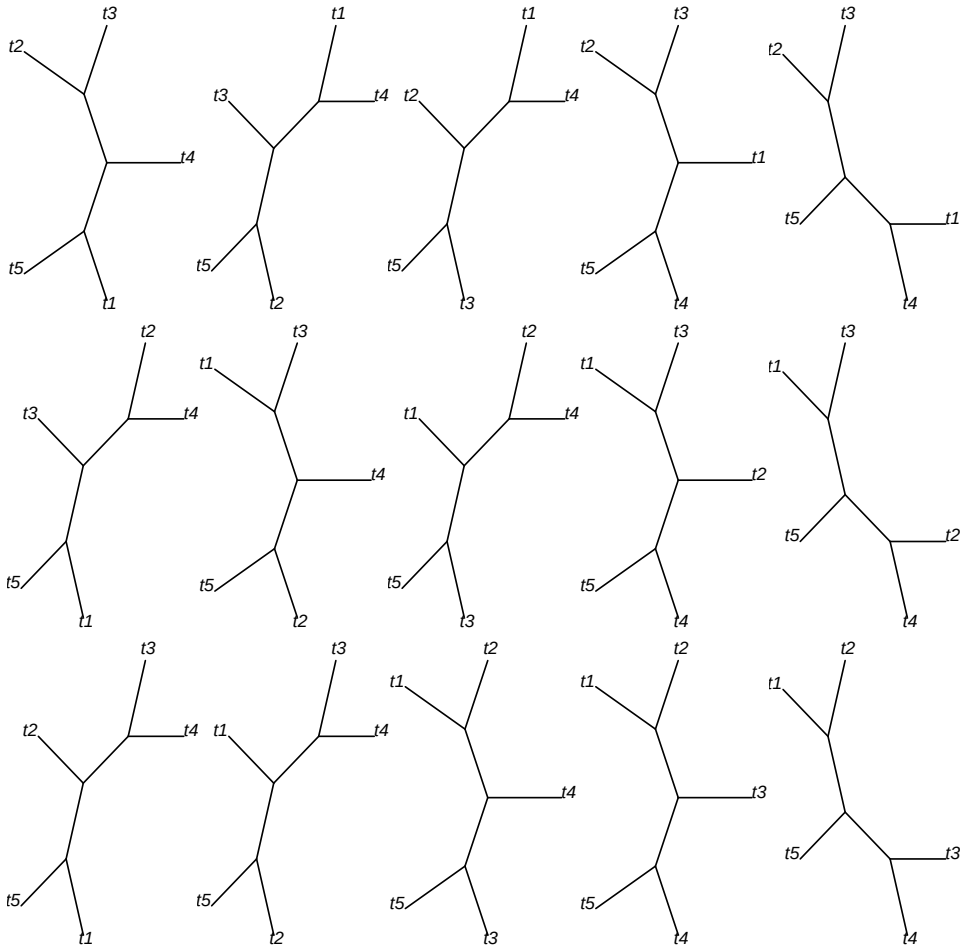

Figure 1: all (15) unrooted trees with 5 taxa

References

[1] S.˜Mathews, M.D. Clements, and M.A. Beilstein. A duplicate gene rooting of seed plants and the phylogenetic position of flowering plants. *Phil. Trans. R. Soc. B*, 365:383–395, 2010.

[2] Emmanuel Paradis. *Analysis of Phylogenetics and Evolution with R*. Springer, New York, 2006.

5 Session Information

The version number of R and packages loaded for generating the vignette were:

- R version 2.12.0 Under development (unstable) (2010-06-02 r52179), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.utf8`, `LC_NUMERIC=C`, `LC_TIME=en_US.utf8`, `LC_COLLATE=C`, `LC_MONETARY=C`, `LC_MESSAGES=en_US.utf8`, `LC_PAPER=en_US.utf8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.utf8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: ape˜2.5-3, multicore˜0.1-4, phangorn˜1.1-2, quadprog˜1.5-3
- Loaded via a namespace (and not attached): gee˜4.13-15, grid˜2.12.0, lattice˜0.19-11, nlme˜3.1-96, tools˜2.12.0