Phylogenetics in R
package phangorn

Klaus Schliep

UMPC, MNHN

March 18, 2010

Phylogenetic packages in R

This talk is mainly about the two packages *ape* and *phangorn*.
There are many other phylogenetic packages on CRAN (some are for very specific tasks) e.g.:

- *phylobase* (nice plot functions), *apTreeshape*, *geiger*, *ouch*, *ade4*

an overview over many packages is given at:
`http://www.cran.r-project.org/web/views/Phylogenetics.html`
For handling biological data:

- *seqinr*
- *ShortRead* (bioconductor)
- many bioconductor packages for meta-data, annotations etc.

Overview of R-packages for phylogenetics

phylogeny reconstruction:

- ape (NJ, fastme)
- phangorn (ML, MP, Network methods, Hadamard)
- Hierarchical clustering `hclust` in package *stats*
  `upgma` is just a wrapper around `hclust`

phangorn

Loading data

`read.dna` in *ape* reads in nucleotide data (phylip and fasta), `read.aa` amino acids and `read.nexus.data` nexus files. The files are either of class DNAbin or a list

```
> data <- read.dna("data.phy")
> data <- read.dna("data.fas", format = "fasta")
> data <- read.nexus.data("data.nex")
> data <- as.phyDat(data)
> data <- read.phyDat(data, format = "phylip",
+     type = "DNA")
```

`read.phyDat` is a wrapper around the other function and transforms object into class `phyDat`. Nexus files come in lot of different dialects. Splitstree has a quite good nexus parser, so importing into and exporting from Splitstree often helps to make them readable to other software.

Loading trees

ape also offers to functions to read in trees:

- `read.tree` for reading trees in Newick format
- `read.nexus` for reading trees in Nexus format

There are also some functions to convert between different tree formats in R, e.g. `hclust`.

Distance methods

There are many different distance based methods available nj, fastme.bal and fastme.ols in *ape* and upgma, wpgma in *phangorn*(based on code from hclust)

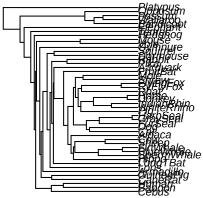
```
> library(phangorn)
> library(multicore)
> data(Laurasiatherian)
> dm = dist.dna(as.DNAbin(Laurasiatherian),
+     model = "JC69")
> treeUPGMA = upgma(dm)
> treeNJ = nj(dm)
> treeFME = fastme.bal(dm)
```

Plotting trees

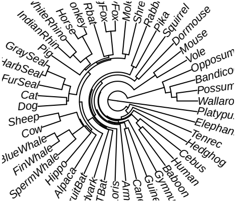
We can plot these trees

```
> par(mfrow = c(2, 2), mar = c(2, 2, 4,
+     2))
> plot(treeUPGMA)
> title("UPGMA")
> plot(treeUPGMA, type = "fan")
> title("UPGMA (fan)")
> plot(treeNJ, type = "unrooted", main = "NJ")
> title("NJ")
> plot(treeFME, type = "unrooted", main = "fastME")
> title("fastME")
```

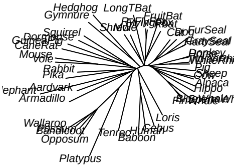
UPGMA



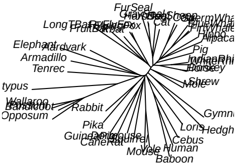
UPGMA (fan)



NJ



fastME



Plotting trees

Plotting trees in R has some advantages, make set up favorite set up once and reuse it. `plot.phylo` offers a big variety of styles

> *args(plot.phylo)*

```
function (x, type = "phylogram", use.edge.length = TRUE, node
    show.tip.label = TRUE, show.node.label = FALSE, edge.colo
    edge.width = 1, edge.lty = 1, font = 3, cex = par("cex"),
    adj = NULL, srt = 0, no.margin = FALSE, root.edge = FALSE
    label.offset = 0, underscore = FALSE, x.lim = NULL, y.lim
    direction = "rightwards", lab4ut = "horizontal", tip.colo
    ...)
NULL
```

phylobase offers also nice plotting with annotations, but less variety yet.

Maximum Parsimony

`parsimony` returns the parsimony score.

```
> trees = structure(list(treeFME, treeNJ,
+     treeUPGMA), class = "multiPhylo")
> parsimony(trees, Laurasiatherian)

[1]  9751  9776 10015
```

These functions are vectorized and can also take `multiPhylo` objects.
phangorn contains the possibility to search for the better parsimony trees:

```
> trB <- optim.parsimony(treeFME, Laurasiatherian)

> parsimony(trB, Laurasiatherian)

[1] 9731
```

Searching is so far slow (in comparison to Paup*) and only NNI moves are implemented. To find a lower bound of the pscore - use Min-Max Squeeze (Holland et al. 2005).

ML

We can compute the likelihood given the data:

```
> fit <- pml(treeNJ, Laurasiatherian)
> fit <- update(fit, k = 4, inv = 0.2)
```

The function `optim.pml` is used to optimize the different parameter.

```
> fit2 <- optim.pml(fit, optNni = TRUE,
+       optGamma = TRUE, optInv = TRUE, model = "GTR")
```

Maximum Likelihood

The function `pml` returns an object of class `pml`. The design differs from most phylogeny packages, but closer to R functions like `lm` or `glm` There exist several generic functions for further analysis of these objects:

```
> methods(class = "pml")
```

```
[1] anova.pml*  logLik.pml* plot.pml*
[4] print.pml*  update.pml* vcov.pml*

   Non-visible functions are asterisked
```

and other generic functions like `AIC` work on these objects.

modeltest

Before running the analysis we should have checked which model to use:

```
> mT = modelTest(treeFME, Laurasiatherian,
+       c("JC", "GTR"))
```

```
> mT
```

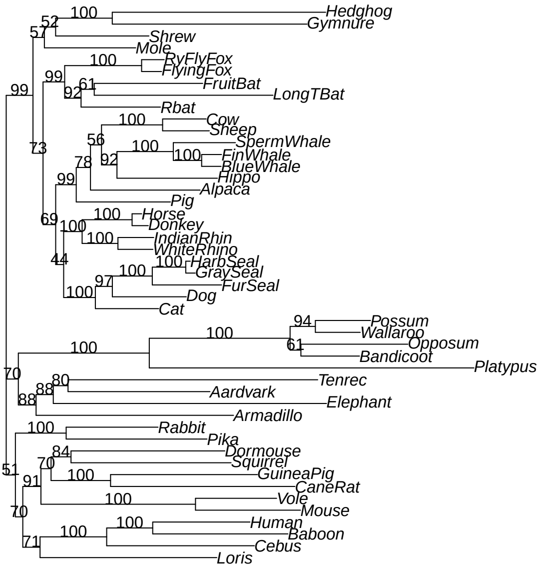
	Model	df	logLik	AIC	BIC
1	JC	91	-58142.27	116466.54	117018.40
2	JC+I	92	-55277.59	110739.18	111297.09
3	JC+G	94	-53136.57	106461.14	107031.18
4	JC+G+I	95	-53775.30	107740.61	108316.72
5	GTR	99	-54907.98	110013.96	110614.33
6	GTR+I	100	-51957.25	104114.51	104720.94
7	GTR+G	102	-49039.82	98283.65	98902.21
8	GTR+G+I	103	-48671.55	97549.09	98173.72

Bootstrap

It is also possible to produce bootstrap samples. The function `bootstrap.pml` makes use of the *multicore* package (under Linux and without GUI interface).

```
> bs <- bootstrap.pml(fit2, bs = 100, optNni = TRUE)
> plotBS(fit2$tree, bs)
```

Tree with bootstrap values



Cheat Sheet

Putting things together here is a script for a standard ML analysis:

```
library(multicore)
library(phangorn)
dat = read.phyDat("myfile")
dm = dist.ml(dat)
tree = fastme.bal(dm)
(mT = modelTest(tree, dat))
fit = pml(tree, dat, k = 4, inv = 0.2)
fit = optim.pml(fit, optNni = TRUE, optGamma = TRUE,
     optInv = TRUE, model = "GTR")
bs = bootstrap.pml(fit, bs = 100, optNni = TRUE)
plotBS(fit$tree, bs)
```

One tree can't rule them all

Comparing trees

treedist returns several tree distance measures, RF.dist is a fast and more memory efficient implementation of the Robinson-Foulds distance for big trees (10.000 taxa)

```
> tree1 = unroot(rtree(100))
> tree2 = unroot(rtree(100))
> treedist(tree1, tree2)
     symmetric.difference
               194.000000
   branch.score.difference
                 9.308153
          path.difference
               442.497458
quadratic.path.difference
               224.744427
> RF.dist(tree1, tree2)
[1] 194
```

An alternative is dist.topo in *ape*.

Partition Models

Partition Models are frequently used to adjust for differences in codon positions. In the next example we allow different rates for different genes.

```
> data(yeast)
> dm.y <- dist.logDet(yeast)
> tree.y <- NJ(dm.y)
> fits <- pml(tree.y, yeast)
> fits <- optim.pml(fits)
> weight = xtabs(~index + genes, attr(yeast,
+     "index"))

> fit.part <- pmlPart(edge ~ rate, fits,
+     weight = weight[, 1:10])
```

Partition Models

We can compare the different partitions with the Shimodaira-Hasegawa test.

```
> set.seed(123)
> sh.p <- SH.test(fit.part)
> sh.p[1:9, ]
      Partition Trees      ln L Diff ln L p-value
 [1,]         1     2 -10142.89  2.155272  0.8793
 [2,]         1     3 -10188.93 48.190589  0.0374
 [3,]         1     4 -10173.94 33.202811  0.0906
 [4,]         1     5 -10191.95 51.217693  0.0312
 [5,]         1     6 -10170.99 30.255435  0.1171
 [6,]         1     7 -10192.16 51.427521  0.0367
 [7,]         1     8 -10157.74 16.998760  0.2305
 [8,]         1     9 -10198.69 57.956076  0.0201
 [9,]         1    10 -10179.33 38.588812  0.0651
```

We observe that the first gene does not differ significantly from the other.

Clustering genes

If number of partitions is too high to justify a different rate for each gene, the `pmlCluster` clusters groups genes together which are similar.

```
> set.seed(111)
> fit.cluster <- pmlCluster(edge ~ rate,
+     fits, weight = weight, p = 4)
```

Clustering genes

```
> set.seed(321)
> sh.c <- SH.test(fit.cluster)
> sh.c
      Partition Trees      ln L Diff ln L p-value
 [1,]         1     2 -193624.9 1923.0837  0e+00
 [2,]         1     3 -192675.5  973.6675  0e+00
 [3,]         1     4 -192205.5  503.6238  0e+00
 [4,]         2     1 -165790.1 1505.8416  0e+00
 [5,]         2     3 -168746.2 4461.8649  0e+00
 [6,]         2     4 -164647.5  363.2420  0e+00
 [7,]         3     1 -168918.4  908.3175  0e+00
 [8,]         3     2 -173249.8 5239.7492  0e+00
 [9,]         3     4 -170676.9 2666.7813  0e+00
[10,]         4     1 -208419.1  512.5676  0e+00
[11,]         4     2 -208380.8  474.2584  6e-04
[12,]         4     3 -210897.2 2990.6184  0e+00
```

Now we cannot reject the hypothesis that all clusters differ.

Partition Models

- The partition models is quite general. One can easily specify which parameters get optimized for each partition and which for all together.
- For given trees (gene/bootstrap/MCMC etc.) estimated from sequence data one can estimate rates for changing their ecological niches (alpine, coastal environment etc.). In this case the trees not even need to have the same taxon set.
- If to many partitions exist `pmlCluster` can group them in clusters with similar trees/parameters.

Hadamard conjugation

Hadamard conjugation is a analytical tool to analyze relations between observed sequence patters and edge weights.

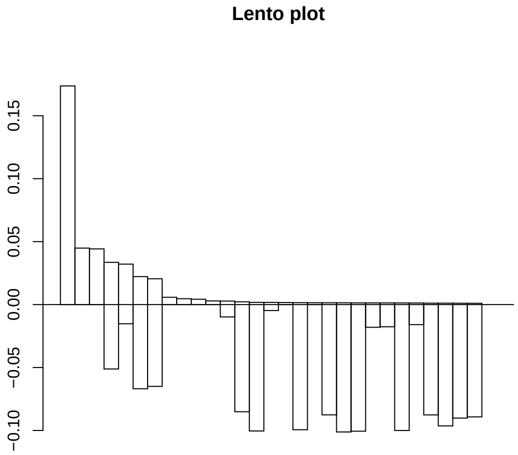
```
> data(yeast)
> dat = as.character(yeast)
> dat[dat == "a" | dat == "g"] = "r"
> dat[dat == "c" | dat == "t"] = "y"
> dat = phyDat(dat, type = "USER", levels = c("r",
+     "y"))
> sp = h2st(dat)
> write.nexus.splits(sp, file = "splits_for_SP_SpectroNet.nex

> lento(sp)
```

Conflicting splits can be represented by an lento plot or split graphs (via SpectroNet or SplitsTree).
Problem: works only for up to 24 taxa.

Lentoplot

The lento plot offers a nice possibility to illustrate these conflicting signals:



Simulating trees and sequences

User defined data formats

The data format `phyDat` is very general and it is easy to construct user defined data formats. For example following would generate data where gaps "-" are coded as a fifth state.

```
> dat = phyDat(dat, "USER", levels = c("a",
+     "c", "g", "t", "-"))
```

This data can used with all the parsimony or maximum likelihood methods.

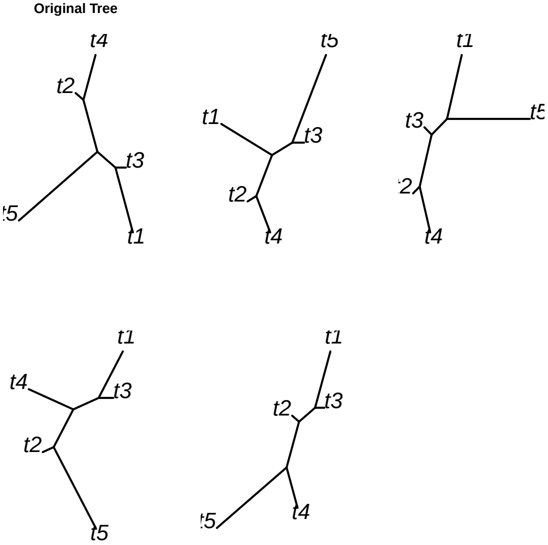
Simulating trees

ape contains some functions to simulate trees:

```
> tree5 = unroot(rtree(5))
> treeNNI = nni(tree5)
```

`nni` in *phangorn* generates all trees which are one Nearest Neighbor Interchange away.

Distribution of parsimony scores



Simulating trees

With the function `allTrees` in *phangorn* constructs all possible trees (up to 10 taxa), what can be interesting for simulation studies.

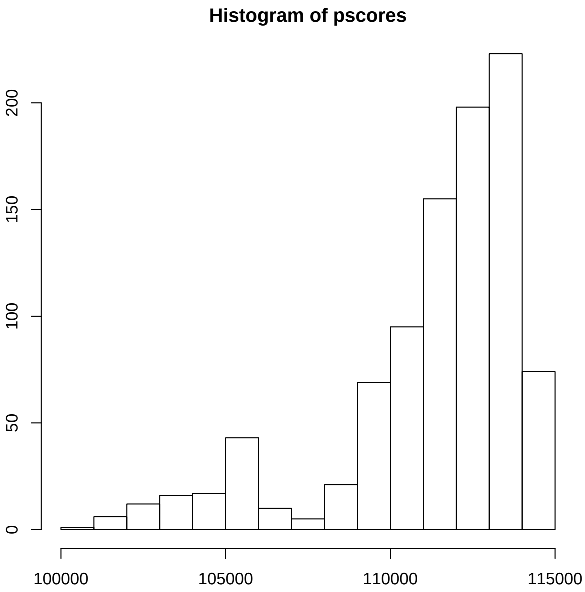
```
> trees = allTrees(7, tip = names(yeast)[-8])
> length(trees)
```

```
[1] 945
```

```
> pscores = parsimony(trees, yeast)
```

```
> plot(hist(pscores))
```

Distribution of parsimony scores



Simulating sequences

```
> tree3 = read.tree(text = "((a:.3, b:.3):.4, c:.7);")
> dat3 = simSeq(tree3, l = 9)
> as.character(dat3)

  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
a "g"  "g"  "a"  "g"  "a"  "a"  "t"  "a"  "g"
b "c"  "a"  "a"  "g"  "g"  "a"  "t"  "a"  "a"
c "c"  "t"  "t"  "c"  "g"  "g"  "t"  "a"  "g"
```

`simSeq` can be used to produce parametric bootstrap samples:

```
> dat0 = simSeq(fit$tree, Q = fit$Q, bf = fit$bf)
```

Phylogenetic analysis with R is possible:

+ large number of functions available

Phylogenetic analysis with R is possible:

- \+ large number of functions available
- \+ very general framework and easy to extend

Phylogenetic analysis with R is possible:

- + large number of functions available
- + very general framework and easy to extend
- + fast to prototype new models

Phylogenetic analysis with R is possible:

- + large number of functions available
- + very general framework and easy to extend
- + fast to prototype new models
- - for big trees when speed is essential: RAXML, Garli

Phylogenetic analysis with R is possible:

+ large number of functions available
+ very general framework and easy to extend
+ fast to prototype new models
- for big trees when speed is essential: RAXML, Garli
- no Bayesian analysis (yet)

In case of help, suggestions, bugs, help with special models (mixtures / partitions), new feature requests etc. feel tree to contact me:

klaus.schliep@snv.jussieu.fr

Acknowledgements

- The NZ crowd, especially my supervisors:
  Mike Hendy, Barbara Holland, David Penny & Peter Waddell

Acknowledgements

- The NZ crowd, especially my supervisors: Mike Hendy, Barbara Holland, David Penny & Peter Waddell
- Philippe Lopez, Eric Bapteste & Nicolas Vidal

Acknowledgements

- The NZ crowd, especially my supervisors:
  Mike Hendy, Barbara Holland, David Penny & Peter Waddell
- Philippe Lopez, Eric Bapteste & Nicolas Vidal
- Emmanuel Paradis (for the great ape package)

Acknowledgements

- The NZ crowd, especially my supervisors: Mike Hendy, Barbara Holland, David Penny & Peter Waddell
- Philippe Lopez, Eric Bapteste & Nicolas Vidal
- Emmanuel Paradis (for the great ape package)
- Julie for volunteering me

Acknowledgements

- The NZ crowd, especially my supervisors:
  Mike Hendy, Barbara Holland, David Penny & Peter Waddell
- Philippe Lopez, Eric Bapteste & Nicolas Vidal
- Emmanuel Paradis (for the great ape package)
- Julie for volunteering me
- and all of you for listening


Fin