Practical course using the R software

Introduction to phylogenetics using R

Thibaut Jombart (tjombart@imperial.ac.uk)

Imperial College London — MSc "*Modern Epidemiology*"

Abstract

This practical aims to illustrate the basics of phylogenetic reconstruction using R, with an emphasis on how the methods work, how their results can be interpreted, and the relative advantages and limitations of the methods. Three main classes of phylogenetic approaches are introduced, namely distance-based, maximum parsimony, and maximum likelihood methods. We also illustrate how to assess the reliability of individual nodes using bootstrap. Methods are illustrated using a toy dataset of seasonal influenza isolates sampled in the US from 1993 to 2008.

Contents

1 Introduction

1.1 Phylogenetics in a nutshell

The reconstruction of evolutionary relationships of a set of organisms can be a tricky task, and has led to the development of a variety of methods over the last decades, implemented in an even larger number of software. However, these methods can be classified in three main categories:

- **distance-based methods**: compute a matrix of pairwise genetic distances between the studied taxa, and summarize it using a hierarchical clustering algorithm such as UPGMA or Neighbour-Joining (Supplementary Figure S1). *Advantages*: fast (the fastest) and flexible (different genetic distances allow to account for different features of DNA sequence evolution). *Limitations*: no model comparison (can't test for the 'best' tree, or the 'best' model of evolution); may be inaccurate and highly dependent on the distance and clustering algorithm chosen.
- **maximum parsimony**: seeks the tree with the smallest number of overall genetic changes between the taxa. This is achieved by changing randomly the topology of the tree until parsimony is no longer improved (Supplementary Figure S2). *Advantages*: intuitive interpretation (assumes that the simplest scenario is the most likely), usually accurate when the amount of genetic changes is small. *Limitations*: computer-intensive, simplistic model of evolution, no model comparison, inaccurate when substantial evolution takes place.
- **likelihood-based method**: based on a model of sequence evolution which allows to compute a likelihood, that is, the probability of observing the data given the model and a set of parameters. There are two main branches of likelihood-based method: maximum likelihood and Bayesian methods. The first seeks the 'best' tree and parameter values, i.e. the one maximizing the likelihood. The second derives samples of tree topologies and model parameters which are the most consistent with the likelihood and possible prior knowledge about the tree/parameters (Supplementary Figure S3). *Advantages*: flexible (any model of evolution can be used), usually accurate, model selection possible, measure of uncertainty (in Bayesian approaches). *Limitations*: computer-intensive, model selection possibly cumbersome.

The R software implements one of the largest selection of phylogenetic methods, including all of the above except for Bayesian reconstruction.

1.2 Required packages

This practical requires a working version of R [6] greater than or equal to 2.12.1. It uses the following packages: *stats* implements basic hierarchical clustering routines, *ade4* [1] and *adegenet* [3] are here used essentially for their graphics, *ape* [5] is the core package for phylogenetics, and *phangorn* [7] implements parsimony and likelihood-based methods. Make sure that the dependencies are installed as well when installing the packages:

```
> install.packages("adegenet", dep=TRUE)
> install.packages("phangorn", dep=TRUE)
```

Then load the packages using:

```
> library(stats)
> library(ade4)
> library(ape)
> library(adegenet)
> library(phangorn)
```

Some graphical functions used in this practical are also only part of the devel version of *adegenet*, and may not be present in the installed version of the package. To make sure these functions are available, source the patch online:

```
> source("http://adegenet.r-forge.r-project.org/files/patches/auxil.R")
```

or install the devel version using:

```
> install.packages("adegenet",repos="http://r-forge.r-project.org")
```

1.3 The data

The data used in this practical are DNA sequences of seasonal influenza (H3N2) downloaded from Genbank (`http://www.ncbi.nlm.nih.gov/genbank/`). Alignments have been realized beforehand using standard tools (Clustalw2 for basic alignment and Jalview for refining the results). We selected 80 isolates genotyped for the hemagglutinin (HA) segment sampled in the US from 1993 to 2008. The dataset consists of two files: i) `usflu.fasta`, a file containing aligned DNA sequences and ii) `usflu.annot.csv`, a comma-separated file containing useful annotations of the sequences. Both files are available online from the *adegenet* website:
- DNA sequences: `http://adegenet.r-forge.r-project.org/files/usflu.fasta`
- annotations: `http://adegenet.r-forge.r-project.org/files/usflu.annot.csv`

To read the DNA sequences into R, we use `read.dna` from the *ape* package:

```
> dna <- read.dna(file="http://adegenet.r-forge.r-project.org/files/usflu.fasta",
+                 format="fasta")
> dna

80 DNA sequences in binary format stored in a matrix.

All sequences of same length: 1701

Labels: CY013200 CY013781 CY012128 CY013613 CY012160 CY012272 ...

Base composition:
    a     c     g     t
0.335 0.200 0.225 0.239

> class(dna)

[1] "DNAbin"
```

Sequences are stored as `DNAbin` objects, an efficient representation of DNA/RNA sequences which use bytes (as opposed to character strings) to code nucleotides. For instance, the first 10 nucleotides of the first 5 isolates:

```
> as.character(dna)[1:5,1:10]

         [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
CY013200 "a"  "t"  "g"  "a"  "a"  "g"  "a"  "c"  "t"  "a"
CY013781 "a"  "t"  "g"  "a"  "a"  "g"  "a"  "c"  "t"  "a"
CY012128 "a"  "t"  "g"  "a"  "a"  "g"  "a"  "c"  "t"  "a"
CY013613 "a"  "t"  "g"  "a"  "a"  "g"  "a"  "c"  "t"  "a"
CY012160 "a"  "t"  "g"  "a"  "a"  "g"  "a"  "c"  "t"  "a"
```

are actually coded as `raw` bytes:

```
> unclass(dna)[1:5,1:10]
```

```
         [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
CY013200   88   18   48   88   88   48   88   28   18    88
CY013781   88   18   48   88   88   48   88   28   18    88
CY012128   88   18   48   88   88   48   88   28   18    88
CY013613   88   18   48   88   88   48   88   28   18    88
CY012160   88   18   48   88   88   48   88   28   18    88
```

```
> typeof(unclass(dna)[1:5,1:10])
```

```
[1] "raw"
```

This results in significant savings in terms of memory required to represent the data:

```
> object.size(as.character(dna))/object.size(dna)
```

```
7.71879054549557 bytes
```

While this dataset is very small, such compression can become essential for larger genomes (bacterial genomes can be several millions of nucleotides long). Note that for even larger datasets, more efficient data reduction can be achieved using the bit-level coding of polymorphic sites implemented in *adegenet* [**?**].

The annotation file is read in R using the standard procedure:

```
> annot <- read.csv("http://adegenet.r-forge.r-project.org/files/usflu.annot.csv",
+                   header=TRUE, row.names=1)
> head(annot)
```

```
  accession year                        misc
1  CY013200 1993 (A/New York/783/1993(H3N2))
2  CY013781 1993 (A/New York/802/1993(H3N2))
3  CY012128 1993 (A/New York/758/1993(H3N2))
4  CY013613 1993 (A/New York/766/1993(H3N2))
5  CY012160 1993 (A/New York/762/1993(H3N2))
6  CY012272 1994 (A/New York/729/1994(H3N2))
```

`accession` contains the Genbank accession numbers, which are unique sequence identifiers; `year` is the year of collection of the isolates; `misc` contains other possibly useful information. Before going further, we check that isolates are identical in both files (accession numbers are used as labels for the sequences):

```
> dim(dna)
```

```
[1]   80 1701
```

```
> dim(annot)
```

```
[1] 80  3
```

```
> all(annot$accession==rownames(dna))
```

```
[1] TRUE
```

```
> table(annot$year)
```

```
1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008
   5    5    5    5    5    5    5    5    5    5    5    5    5    5    5    5
```

Good! The data we will analyse are 80 isolates (5 per year) typed for the same 1701 nucleotides.

2 Distance-based phylogenies

Distance-based phylogenetic reconstruction consists in i) computing pairwise genetic distances between individuals (here, isolates), ii) representing these distances using a tree (Figure S1), and iii) evaluating the relevance of this representation.

2.1 Computing genetic distances

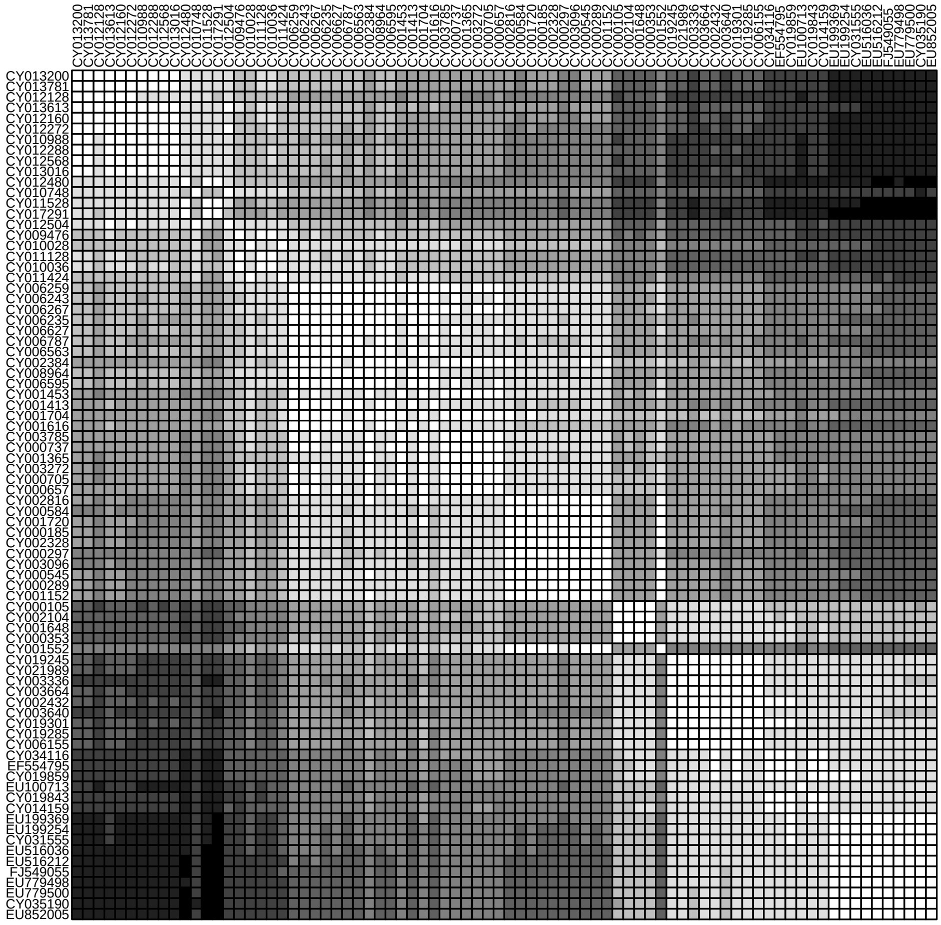
We first compute genetic distances using *ape*'s `dist.dna`, which proposes no less than 15 different genetic distances (see `?dist.dna` for details). Here, we use Tamura and Nei 1993's model [8] which allows for different rates of transitions and transversions, heterogeneous base frequencies, and between-site variation of the substitution rate.

```
> D <- dist.dna(dna, model="TN93")
> class(D)

[1] "dist"

> length(D)

[1] 3160
```

`D` is an object of class `dist` which contains the distances between every pairs of sequences.

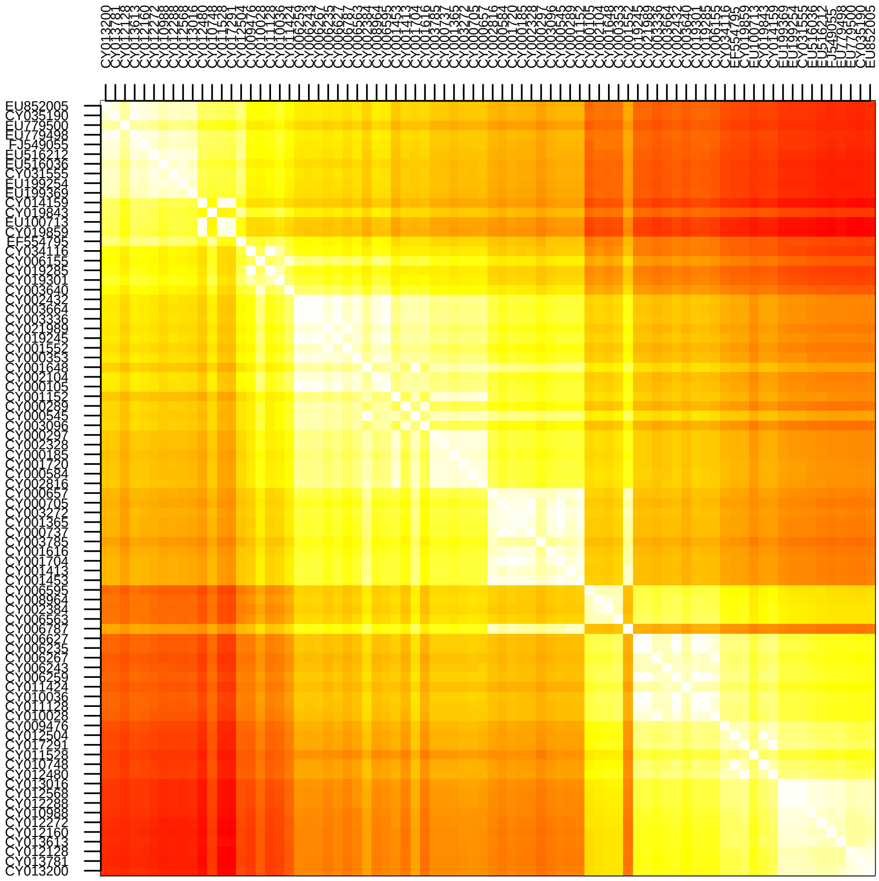
Now that genetic distances between isolates have been computed, we need to visualize this information. There are $n(n-1)/2$ distances for n sequences; here, $n = 80$ so that the genetic relationships between the sampled isolates are described by $80 \times 79/2 = 3160$ pairwise distances. Most of the time, summarising such information is not entirely trivial. The simplest approach is plotting directly the matrix of pairwise distances:

```
> temp <- as.data.frame(as.matrix(D))
> table.paint(temp, cleg=0, clabel.row=.5, clabel.col=.5)
```



Darker shades of grey represent greater distances. Note that to use `image` to produce similar plots, data need to be transformed first; for instance:

```
> temp <- t(as.matrix(D))
> temp <- temp[,ncol(temp):1]

> par(mar=c(1,5,5,1))
> image(x=1:80, y=1:80, temp, col=rev(heat.colors(100)), xaxt="n", yaxt="n",
+       xlab="",ylab="")
> axis(side=2, at=1:80, lab=rownames(dna), las=2, cex.axis=.5)
> axis(side=3, at=1:80, lab=rownames(dna), las=3, cex.axis=.5)
```



(see `image.plot` in the package *fields* for similar plots with a legend).

Since the data are roughly ordered by year, we can already see some genetic structure appearing, but this is admittedly not the most satisfying or informative approach, and tells us little about the evolutionary relationships between our isolates.

2.2 Building trees

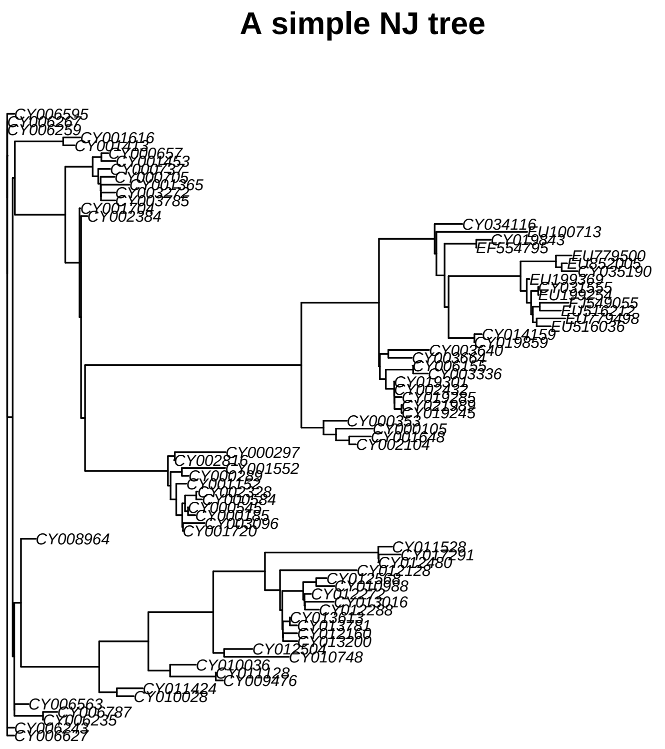
We use trees to get a better representation of the genetic distances between individuals. It is important, however, to bear in mind that the obtained trees are not necessarily efficient representations of the original distances, and information can —and likely will— be lost in the process.

A wide array of algorithms for constructing trees from a distance matrix are available in R, including:

- ⋆ `nj` (*ape* package): the classical Neighbor-Joining algorithm.
- ⋆ `bionj` (*ape*): an improved version of Neighbor-Joining.
- ⋆ `fastme.bal` and `fastme.ols` (*ape*): minimum evolution algorithms.
- ⋆ `hclust` (*stats*): classical hierarchical clustering algorithms including single linkage, complete linkage, UPGMA, and others.

Here, we go for the standard:

```
> tre <- nj(D)
> class(tre)

[1] "phylo"

> tre

Phylogenetic tree with 80 tips and 78 internal nodes.

Tip labels:
        CY013200, CY013781, CY012128, CY013613, CY012160, CY012272, ...

Unrooted; includes branch lengths.

> plot(tre, cex=.6)
> title("A simple NJ tree")
```

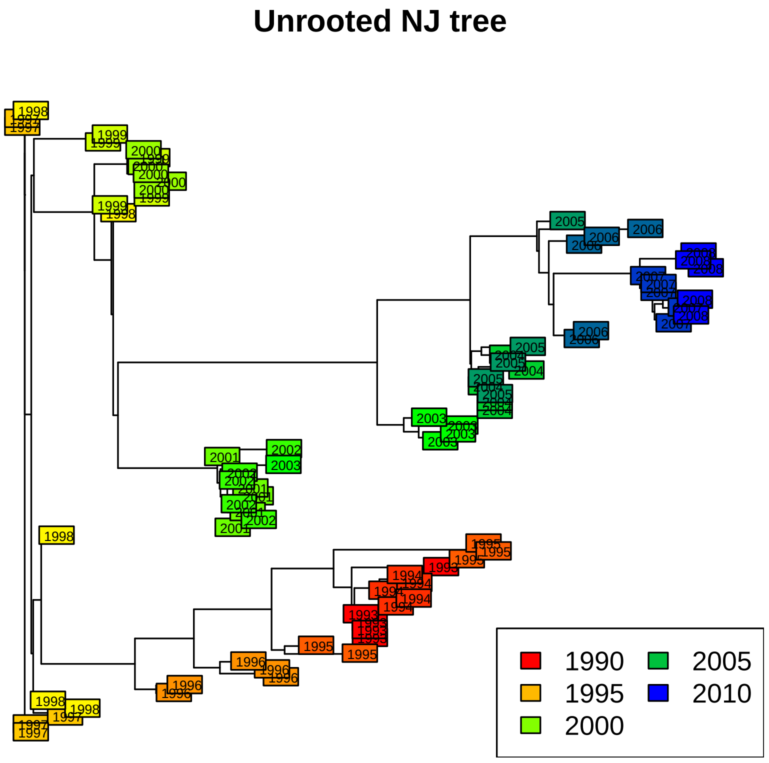


Trees created in the package *ape* are instances of the class `phylo`. See `?read.tree` for a description of this class.

2.3 Plotting trees

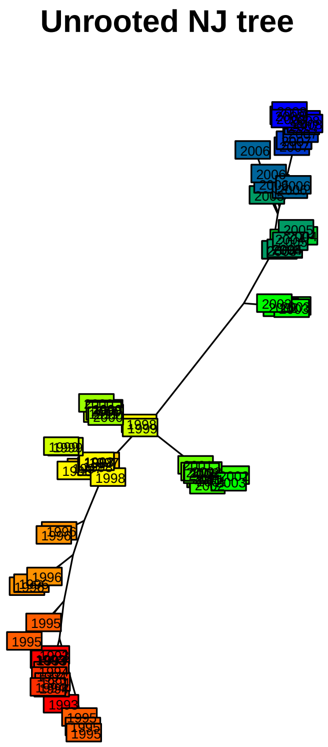
The plotting method offers many possibilities for plotting trees; see `?plot.phylo` for more details. Functions such as `tiplabels`, `nodelabels`, `edgelabels` and `axisPhylo` can also be useful to annotate trees. For instance, we may simply represent years using different colors (red=ancient; blue=recent):

```
> plot(tre, show.tip=FALSE)
> title("Unrooted NJ tree")
> myPal <- colorRampPalette(c("red","yellow","green","blue"))
> tiplabels(annot$year, bg=num2col(annot$year, col.pal=myPal), cex=.5)
> temp <- pretty(1993:2008, 5)
> legend("bottomright", fill=num2col(temp, col.pal=myPal), leg=temp, ncol=2)
```



This illustrates a common mistake when interpreting phylogenetic trees. In the above figures, we tend to assume that the left-side of the phylogeny is ‘ancestral’, while the right-side is ‘recent’. This is wrong —as suggested by the colors— unless the phylogeny is actually rooted, i.e. some external taxa has been used to define what is the most ‘ancient’ split in the tree. The present tree is not rooted, and should be better represented as such:

```
> plot(tre, type="unrooted", show.tip=FALSE)
> title("Unrooted NJ tree")
> tiplabels(annot$year, bg=num2col(annot$year, col.pal=myPal), cex=.5)
```

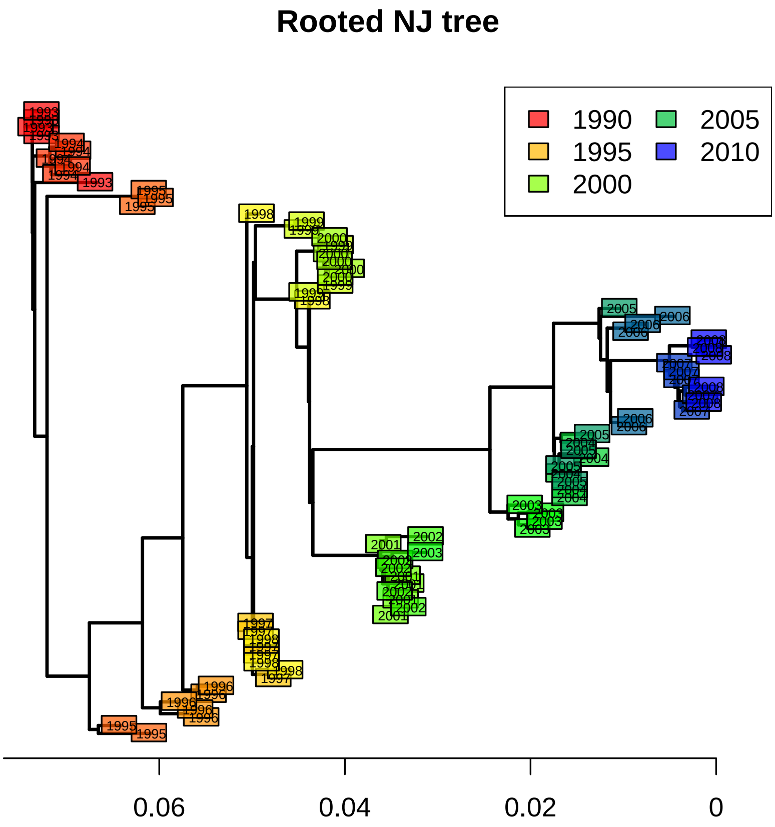


In the present case, a sensible rooting would be any of the most ancient isolates (from 1993). We can take the first one:

```
> head(annot)

  accession year                       misc
1  CY013200 1993 (A/New York/783/1993(H3N2))
2  CY013781 1993 (A/New York/802/1993(H3N2))
3  CY012128 1993 (A/New York/758/1993(H3N2))
4  CY013613 1993 (A/New York/766/1993(H3N2))
5  CY012160 1993 (A/New York/762/1993(H3N2))
6  CY012272 1994 (A/New York/729/1994(H3N2))


> tre2 <- root(tre, out=1)
```

and plot the result:

```
> plot(tre2, show.tip=FALSE, edge.width=2)
> title("Rooted NJ tree")
> tiplabels(annot$year, bg=transp(num2col(annot$year, col.pal=myPal),.7), cex=.5,
+           fg="transparent")
> axisPhylo()
> temp <- pretty(1993:2008, 5)
> legend("topright", fill=transp(num2col(temp, col.pal=myPal),.7), leg=temp, ncol=2)
```



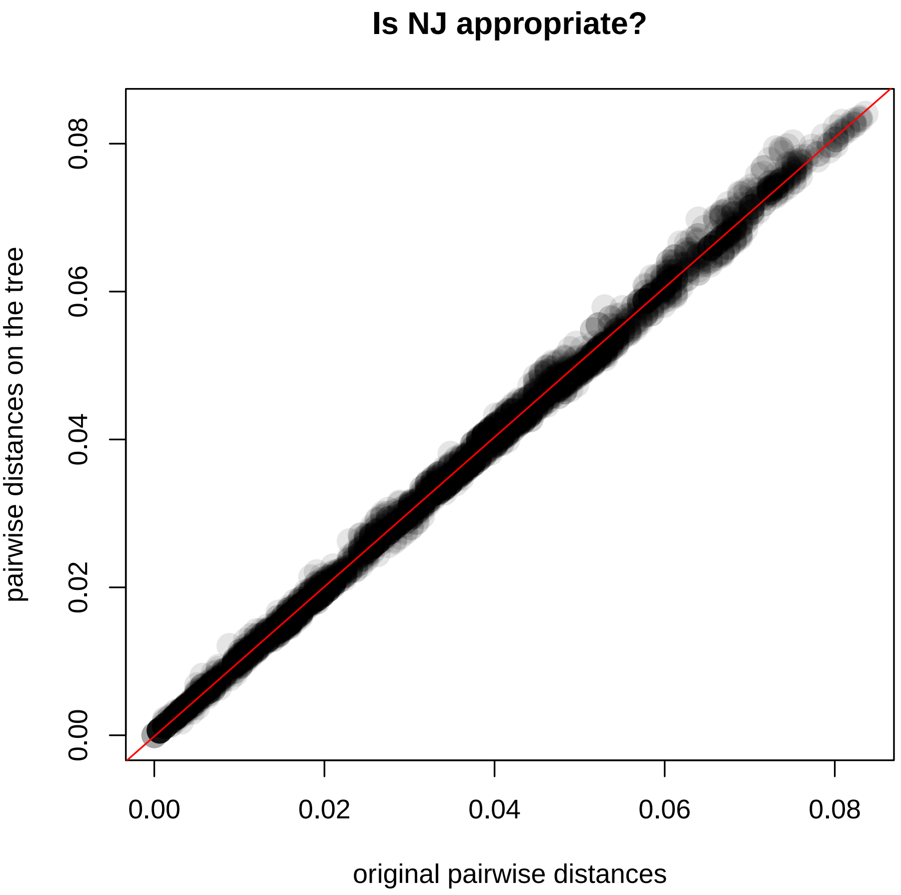
Now, the horizontal axis can globally be interpreted as temporal evolution; however, it is not uncommon that isolates from consecutive years cluster together, suggesting that the turnover of strains from one season to another is somehow smooth.

2.4 Assessing the quality of a phylogeny

Many genetic distances and hierarchical clustering algorithms can be used to build trees; not all of them are appropriate for a given dataset. Genetic distances rely on hypotheses about the evolution of DNA sequences which should be taken into account. For instance, the mere proportion of differing nucleotides between sequences (`model='raw'` in `dist.dna`) is easy to interprete, but only makes sense if all substitutions are equally frequent. In practice, simple yet flexible models such as that of Tamura and Nei (1993, [8]) are probably fair choices. At the very least, the genetic distance used should allow different rates for transitions ($a \leftrightarrow g$, $c \leftrightarrow t$) and transversions (other changes).

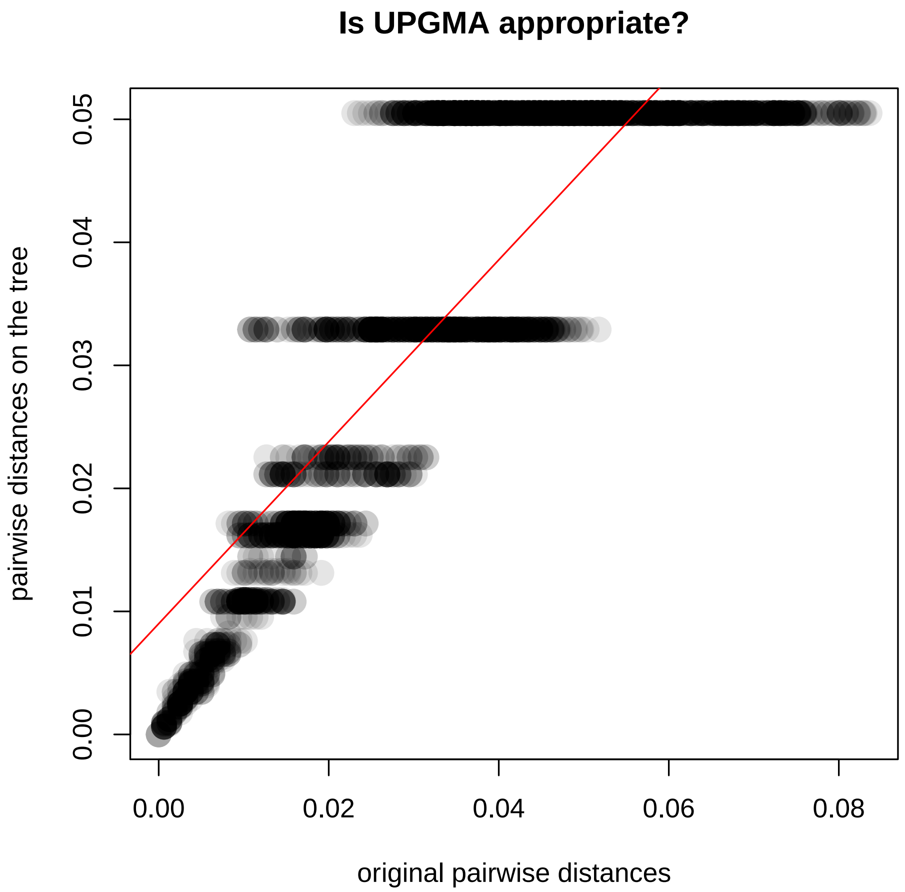
Once one has chosen an appropriate genetic distance and built a tree using this distance, an essential yet most often overlooked question is whether this tree actually is a good representation of the original distance matrix. This is easily investigated using simple biplots and correlation indices. The function `cophenetic` is used to compute distances between the tips of the tree. Note that more distances are available in the *adephylo* package (see `distTips` function).

```
> x <- as.vector(D)
> y <- as.vector(as.dist(cophenetic(tre2)))
> plot(x, y, xlab="original pairwise distances", ylab="pairwise distances on the tree",
+      main="Is NJ appropriate?", pch=20, col=transp("black",.1), cex=3)
> abline(lm(y~x), col="red")
> cor(x,y)^2

[1] 0.9975154
```



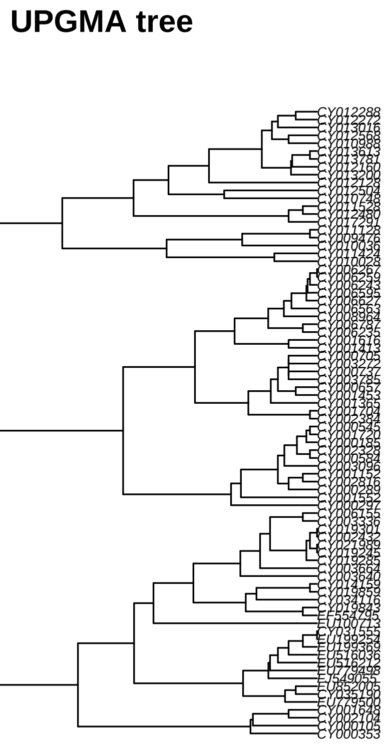
As it turns out, our Neighbor-Joining tree (tre2) is a very good representation of the chosen genetic distances. Things would have been different had we chosen, for instance, UPGMA:

```
> tre3 <- as.phylo(hclust(D,method="average"))
> y <- as.vector(as.dist(cophenetic(tre3)))
> plot(x, y, xlab="original pairwise distances", ylab="pairwise distances on the tree",
+      main="Is UPGMA appropriate?", pch=20, col=transp("black",.1), cex=3)
> abline(lm(y~x), col="red")
> cor(x,y)^2

[1] 0.7393009
```



In this case, UPGMA is a poor choice. Why is this? A first explanation is that UPGMA forces ultrametry (all the tips are equidistant to the root):

```
> plot(tre3, cex=.5)
> title("UPGMA tree")
```



The underlying assumption is that all lineages have undergone the same amount of evolution, which is obviously not the case in seasonal influenza sampled over 16 years.

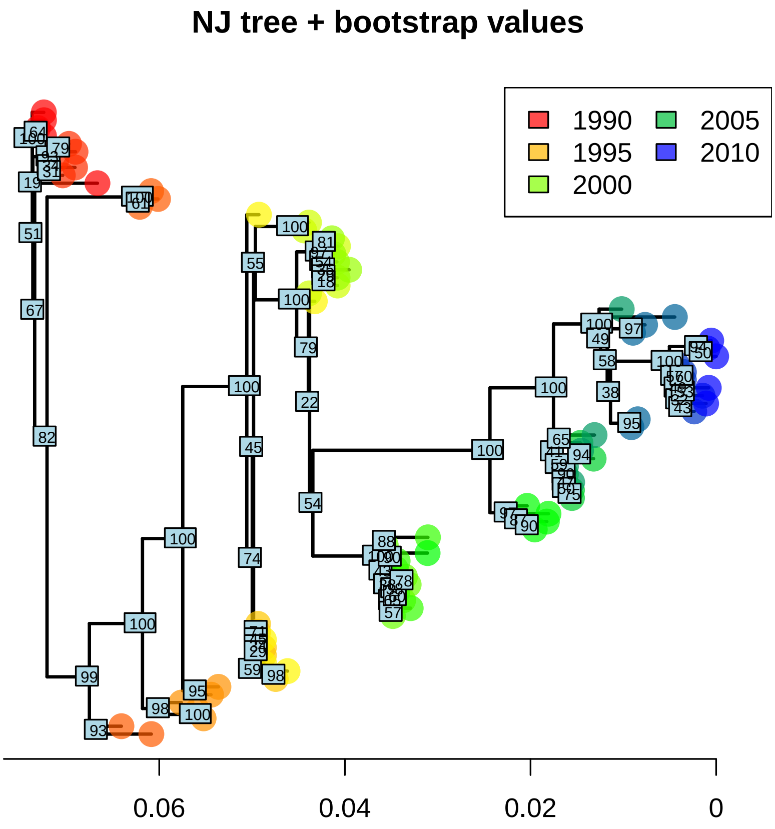
Another validation of phylogenetic trees, much more commonly used, is bootstrap. Bootstrapping a phylogeny consists in sampling the nucleotides with replacement, rebuilding the phylogeny, and checking if the original nodes are present in the bootstrapped trees (Supplementary Figure S4). In practice, this procedure is repeated a large number of times (e.g. 100, 1000), depending on how computer-intensive the phylogenetic reconstruction is. The underlying idea is to assess the variability in the obtained topology which results from conducting the analyses on a random sample the genome. Note that the assumption that the analysed sequences represent a random sample of the genome is often dubious. For instance, this is not the case in our toy dataset, since HA segment has a different rate of evolution and experiences different selective pressures from other segments of the influenza genome. We nonetheless illustrate the procedure, implemented by `boot.phylo`:

```
> myBoots <- boot.phylo(tre2, dna, function(e) root(nj(dist.dna(e, model = "TN93")),1))
> myBoots
```

The output gives the number of times each node was identified in bootstrapped analyses (the order is the same as in the original object). It is easily represented using `nodelabels`:

```
> plot(tre2, show.tip=FALSE, edge.width=2)
> title("NJ tree + bootstrap values")
> tiplabels(frame="none", pch=20, col=transp(num2col(annot$year, col.pal=myPal),.7),
+           cex=3, fg="transparent")
```

```
> axisPhylo()
> temp <- pretty(1993:2008, 5)
> legend("topright", fill=transp(num2col(temp, col.pal=myPal),.7), leg=temp, ncol=2)
> nodelabels(myBoots, cex=.6)
```

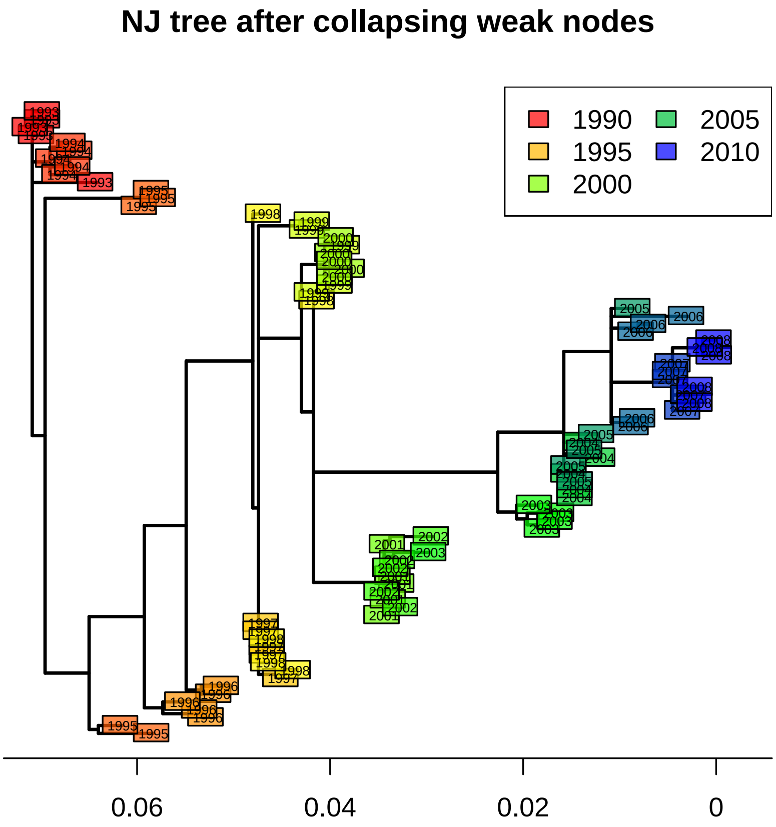


As we can see, some nodes are very poorly supported. One common practice is to collapse these nodes into multifurcations. There is no dedicated method for this in *ape*, but one simple workaround consists in setting the corresponding edges to a length of zero (here, with bootstrap $< 70\%$), and then collapsing the small branches:

```
> temp <- tre2
> N <- length(tre2$tip.label)
> toCollapse <- match(which(myBoots<70)+N, temp$edge[,2])
> temp$edge.length[toCollapse] <- 0
> tre3 <- di2multi(temp, tol=0.00001)
```

The new tree might be slightly less informative, but more robust than the previous one:

```
> plot(tre3, show.tip=FALSE, edge.width=2)
> title("NJ tree after collapsing weak nodes")
> tiplabels(annot$year, bg=transp(num2col(annot$year, col.pal=myPal),.7), cex=.5,
+           fg="transparent")
> axisPhylo()
> temp <- pretty(1993:2008, 5)
> legend("topright", fill=transp(num2col(temp, col.pal=myPal),.7), leg=temp, ncol=2)
```


NJ tree after collapsing weak nodes

3 Maximum parsimony phylogenies

3.1 Introduction

Phylogenetic reconstruction based on parsimony seeks trees which minimize the total number of changes (substitutions) from ancestors to descendents. While a number of criticisms can be made to this approach, it is a simple way to infer phylogenies for data which display low divergence (i.e. most taxa differ from each other by only a few nucleotides, and the overall substitution rate is low).

In practice, there is often no way to perform an exhaustive search amongst all possible trees to find the most parsimonious one, and heuristic algorithms are used to browse the space of possible trees (Figure S2). The strategy is fairly simple: i) initialize the algorithm using a tree and ii) make small changes to the tree and retain those leading to better parsimony, until the parsimony score stops improving.

3.2 Implementation

Parsimony-based phylogenetic reconstruction is implemented in the package *phangorn*. It requires a tree (in *ape*'s format, i.e. a `phylo` object) and the original DNA sequences in *phangorn*'s own format, `phyDat`. We convert the data and generate a tree to initialize the method:

```
> dna2 <- as.phyDat(dna)
> class(dna2)

[1] "phyDat"

> dna2

80 sequences with 1701 character and 269 different site patterns.
The states are a c g t


> tre.ini <- nj(dist.dna(dna,model="raw"))
> tre.ini

Phylogenetic tree with 80 tips and 78 internal nodes.

Tip labels:
        CY013200, CY013781, CY012128, CY013613, CY012160, CY012272, ...

Unrooted; includes branch lengths.
```

The parsimony of a given tree is given by:

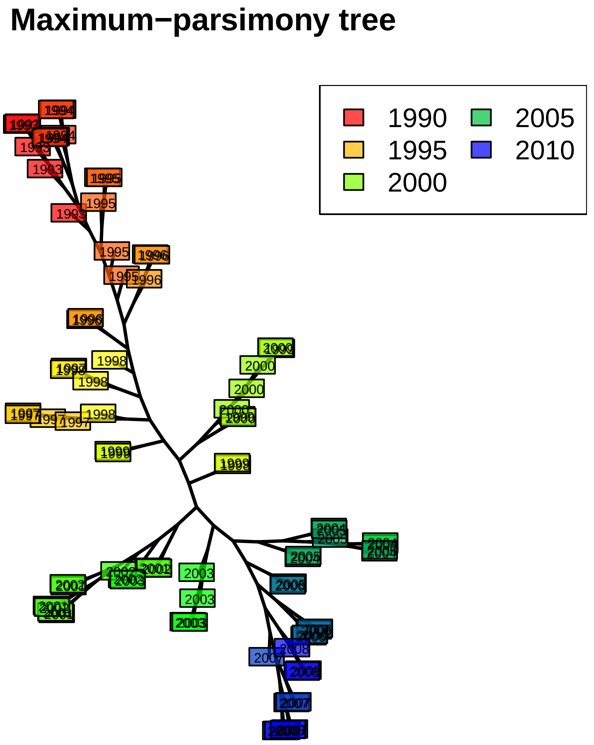
```
> parsimony(tre.ini, dna2)

[1] 422
```

Then, optimization of the parsimony is achieved by:

```
> tre.pars <- optim.parsimony(tre.ini, dna2)

Final p-score 420 after  2 nni operations


> tre.pars

Phylogenetic tree with 80 tips and 78 internal nodes.

Tip labels:
        CY013200, CY013781, CY012128, CY013613, CY012160, CY012272, ...

Unrooted; no branch lengths.
```

Here, the final result is very close to the original tree. The obtained tree is unrooted and does not have branch lengths, but it can be plotted as previously:

```
> plot(tre.pars, type="unr", show.tip=FALSE, edge.width=2)
> title("Maximum-parsimony tree")
> tiplabels(annot$year, bg=transp(num2col(annot$year, col.pal=myPal),.7), cex=.5,
+           fg="transparent")
> temp <- pretty(1993:2008, 5)
> legend("topright", fill=transp(num2col(temp, col.pal=myPal),.7), leg=temp, ncol=2,
+        bg=transp("white"))
```



In this case, parsimony gives fairly consistent results with other approaches, which is only to be expected whenever the amount of divergence between the sequences is fairly low, as is the case in our data.

4 Maximum likelihood phylogenies

4.1 Introduction

Maximum likelihood phylogenetic reconstruction is somehow similar to parsimony methods in that it browses a space of possible tree topologies looking for the ’best’ tree. However, it offers far more flexibility in that any model of sequence evolution can be taken into account. Given one model of evolution, one can compute the likelihood of a given tree, and therefore optimization procedures can be used to infer both the most likely tree topology and model parameters.

As in distance-based methods, model-based phylogenetic reconstruction requires thinking about which parameters should be included in a model. Usually, all possible substitutions are allowed to have different rates, and the substitution rate is allowed to vary across sites according to a gamma distribution. We refer to this model as $\text{GTR} + \Gamma(4)$ (GTR: global time reversible). More information about phylogenetic models can be found in [4].

4.2 Sorting out the data

Likelihood-based phylogenetic reconstruction is implemented in the package *phangorn*. Like parsimony-based approaches, it requires a tree (in *ape*’s format, i.e. a `phylo` object) and the original DNA sequences in *phangorn*’s own format, `phyDat`. As in the previous section, we convert the data and generate a tree to initialize the method:

```
> dna2 <- as.phyDat(dna)
> class(dna2)

[1] "phyDat"

> dna2


80 sequences with 1701 character and 269 different site patterns.
The states are a c g t


> tre.ini <- nj(dist.dna(dna,model="TN93"))
> tre.ini


Phylogenetic tree with 80 tips and 78 internal nodes.

Tip labels:
        CY013200, CY013781, CY012128, CY013613, CY012160, CY012272, ...

Unrooted; includes branch lengths.
```

To initialize the optimization procedure, we need an initial fit for the model chosen. This is computed using `pml`:

```
> pml(tre.ini, dna2, k=4)

 loglikelihood: NaN

unconstrained loglikelihood: -4736.539
Discrete gamma model
Number of rate categories: 4
Shape parameter: 1

Rate matrix:
  a c g t
a 0 1 1 1
c 1 0 1 1
```

```
g 1 1 0 1
t 1 1 1 0

Base frequencies:
0.25 0.25 0.25 0.25
```

The computed likelihood is `NA`, which is obviously a bit of a problem, but a likely frequent issue. This issue is due to missing data (`NAs`) and ambiguous sites in the original dataset:

```
> table(as.character(dna2))

    -     a     c     g     k     m     r     s     t     w 
  147 45595 27170 30613     1     2     1     1 32549     1 
```

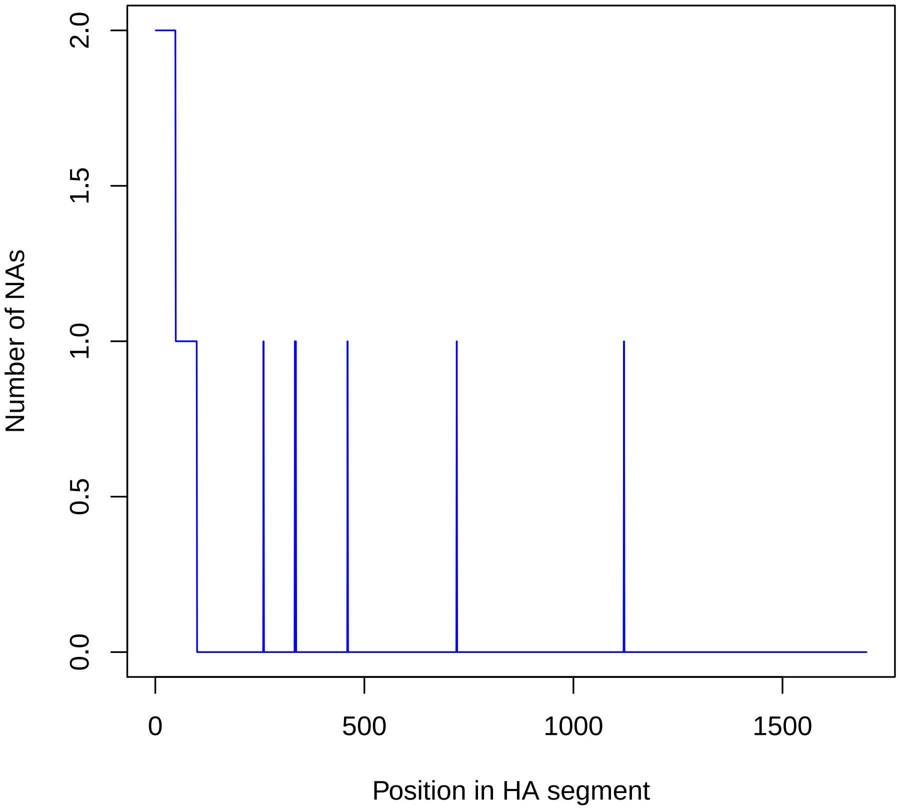
We therefore need to remove missing data before going further.

We first retrieve the position of missing data, i.e. any data differing from 'a', 'g','c' and 't'.

```
> na.posi <- which(apply(as.character(dna),2, function(e) any(!e %in% c("a","t","g","c"))))
```

We can easily plot the number of missing data for each site:

```
> temp <- apply(as.character(dna),2, function(e) sum(!e %in% c("a","t","g","c")))
> plot(temp, type="l", col="blue", xlab="Position in HA segment", ylab="Number of NAs")
```



The begining of the alignment is guilty for most of the missing data, which was only to be expected (extremities of the sequences have variable length).

```
> dna3 <- dna[,-na.posi]
> dna3
```

```
80 DNA sequences in binary format stored in a matrix.

All sequences of same length: 1596

Labels: CY013200 CY013781 CY012128 CY013613 CY012160 CY012272 ...

Base composition:
    a     c     g     t
0.340 0.197 0.226 0.238

> table(as.character(dna3))


    a     c     g     t
43402 25104 28828 30346

> dna4 <- as.phyDat(dna3)
```

The object `dna3` is an alignment of all sequences excluding missing data; `dna4` is its conversion in `phyDat` format.

4.3 Getting a ML tree

We recompute the likelihood of the initial tree using `pml`:

```
> dna4 <- as.phyDat(dna3)
> tre.ini <- nj(dist.dna(dna3,model="TN93"))
> fit.ini <- pml(tre.ini, dna4, k=4)
> fit.ini

 loglikelihood: -5183.648

unconstrained loglikelihood: -4043.367
Discrete gamma model
Number of rate categories: 4
Shape parameter: 1

Rate matrix:
  a c g t
a 0 1 1 1
c 1 0 1 1
g 1 1 0 1
t 1 1 1 0

Base frequencies:
0.25 0.25 0.25 0.25
```

We now have all the information needed for seeking a maximum likelihood solution using `optim.pml`; we specify that we want to optimize tree topology (`optNni=TRUE`), base frequencies (`optBf=TRUE`), the rates of all possible subtitutions (`optQ=TRUE`), and use a gamma distribution to model variation in the substitution rates across sites (`optGamma=TRUE`):

```
> fit <- optim.pml(fit.ini, optNni=TRUE, optBf=TRUE, optQ=TRUE, optGamma=TRUE)

> fit

 loglikelihood: -4915.866

unconstrained loglikelihood: -4043.367
Discrete gamma model
Number of rate categories: 4
Shape parameter: 0.2843531

Rate matrix:
          a          c         g          t
a 0.0000000  2.3831856 8.2953681  0.8555066
c 2.3831856  0.0000000 0.1486213 10.0764253
g 8.2953681  0.1486213 0.0000000  1.0000000
t 0.8555066 10.0764253 1.0000000  0.0000000

Base frequencies:
0.3416519 0.1953526 0.2242948 0.2387007
```

```
> class(fit)

[1] "pml"

> names(fit)

 [1] "logLik"  "inv"     "k"       "shape"   "Q"       "bf"      "rate"   
 [8] "siteLik" "weight"  "g"       "w"       "eig"     "data"    "model"  
[15] "INV"     "ll.0"    "tree"    "lv"      "call"    "df"      "wMix"   
[22] "llMix"  
```

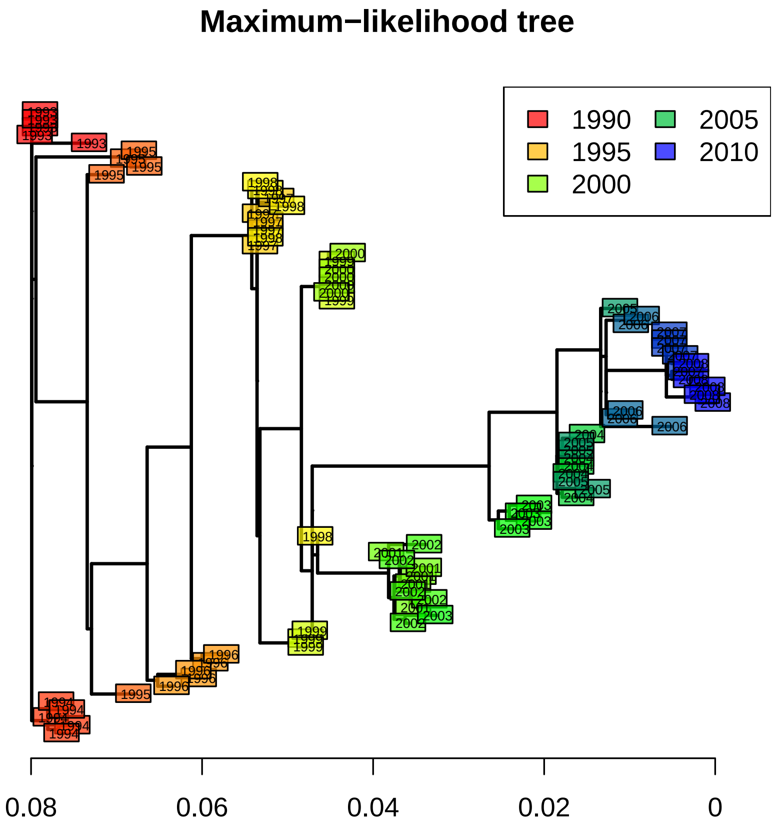
`fit` is a list with class `pml` storing various useful information about the model parameters and the optimal tree (stored in `fit$tree`). In this example, we can see from the output that transitions ($a \leftrightarrow g$ and $c \leftrightarrow t$) are much more frequent than transversions (other changes), which is consistent with biological expectations (transversions induce more drastic changes of chemical properties of the DNA and are more prone to purifying selection). One advantage of using probabilistic models of evolution is that different models can be compared formally. For instance, here, we can verify that the optimized tree is indeed better than the original one using standard likelihood ratio tests and AIC:

```
> anova(fit.ini, fit)

Likelihood Ratio Test Table
  Log lik.  Df Df change Diff log lik. Pr(>|Chi|)    
1  -5183.6 158                                       
2  -4915.9 166         8        535.56  < 2.2e-16 ***
---
Signif. codes:  0

> AIC(fit.ini)

[1] 10683.3

> AIC(fit)

[1] 10163.73
```

Both the ANOVA test (highly significant) and the AIC (lower=better) indicate that the new tree is a better model of the data than the initial one.

We can extract and plot the tree as we did before with other methods:

```
> tre4 <- root(fit$tree,1)
> plot(tre4, show.tip=FALSE, edge.width=2)
> title("Maximum-likelihood tree")
> tiplabels(annot$year, bg=transp(num2col(annot$year, col.pal=myPal),.7), cex=.5,
+           fg="transparent")
> axisPhylo()
> temp <- pretty(1993:2008, 5)
> legend("topright", fill=transp(num2col(temp, col.pal=myPal),.7), leg=temp, ncol=2)
```



This tree is statistically better than the original NJ tree based on Tamura and Nei's distance [8]. However, we can note that it is remarkably similar to the 'robust' version of this distance-based tree (after collapsing weakly supported nodes). The structure of this dataset is fairly simple, and all methods give fairly consistent results. In practice, different methods can lead to different interpretations, and it is often worth exploring different approaches before drawing conclusions on the data.

5 Supplementary figures

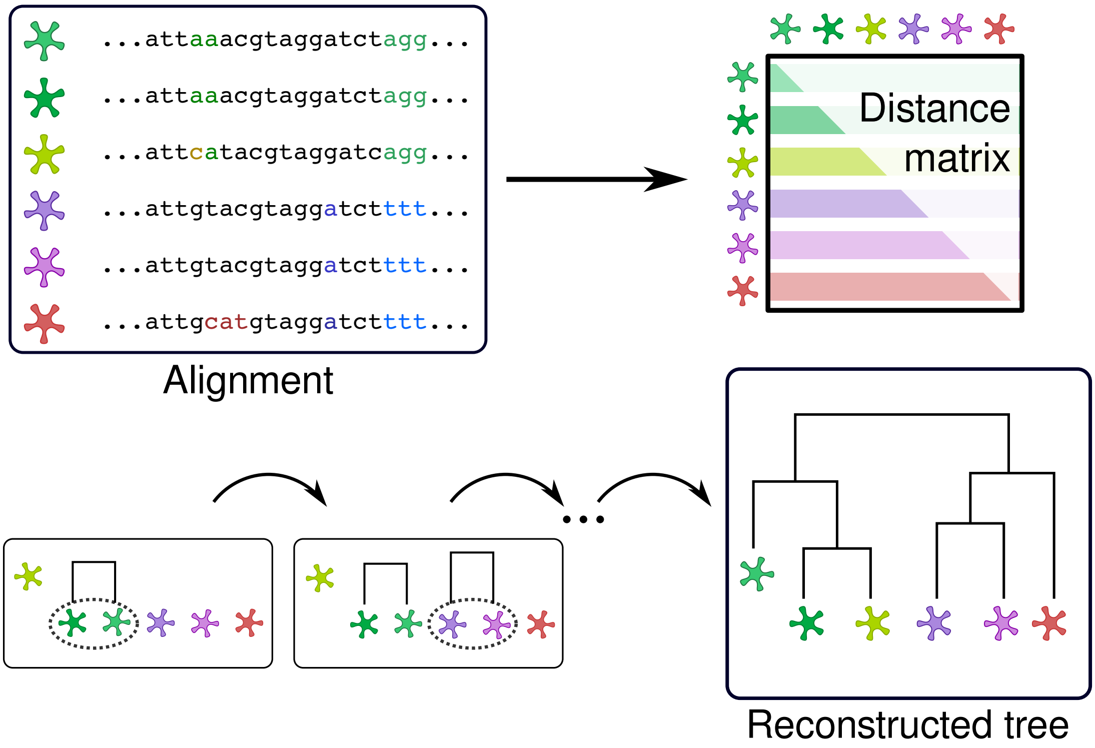

...attaaacgtaggatctagg...
...attaaacgtaggatctagg...
...attcatacgtaggatcagg...
...attgtacgtaggatctttt...
...attgtacgtaggatctttt...
...attgcatgtaggatctttt...
Alignment
Distance
matrix
...
Reconstructed tree

Figure S1: distance-based phylogenetic reconstruction. Pairwise genetic distances between the sampled pathogens (represented as star-like bugs) are computed from aligned DNA sequences. The resulting distance matrix is summarized using a hierarchical clustering algorithm such as UPGMA or NJ, which aggregate taxa until a complete tree is reconstructed.

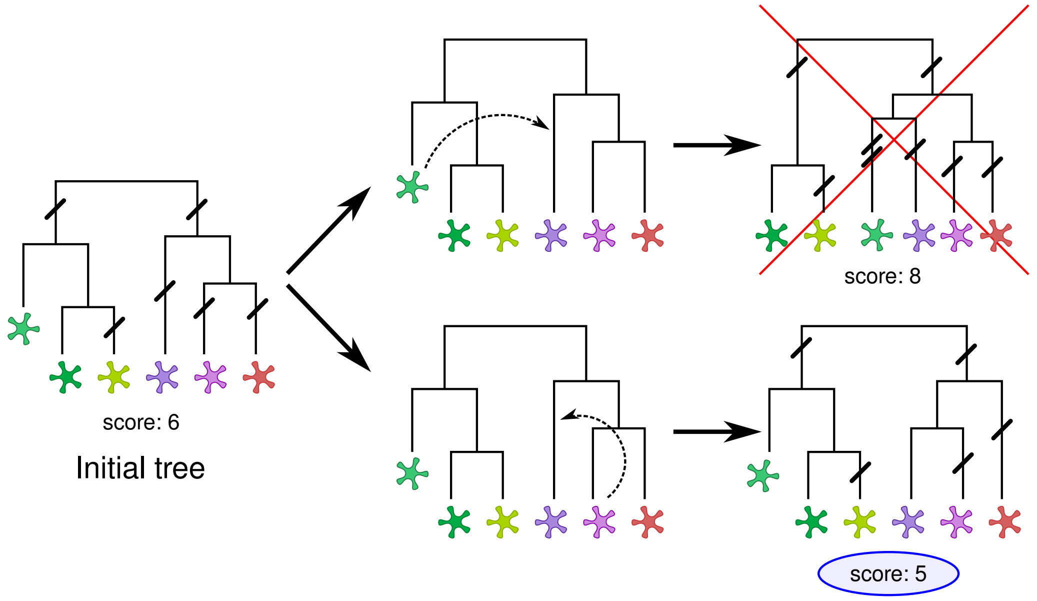


Figure S2: maximum-parsimony phylogenetic reconstruction. The method researches the tree with the smallest total number of genetic changes (thick bars). This is achieved by permuting nodes randomly (dotted arrows) and retaining configurations with improved parsimony; the algorithm stops when the parsimony score can no longer be improved.

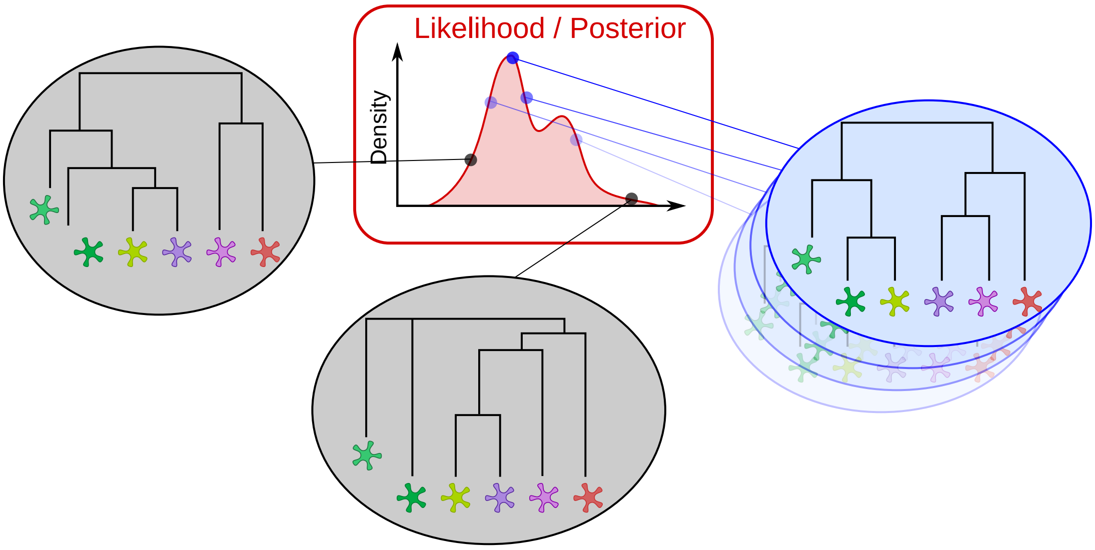


Figure S3: likelihood-based phylogenetic reconstruction. Maximum-likelihood (ML) and Bayesian approaches use heuristic methods to find trees and parameters which are consistent with the data (blue background), while overlooking less plausible topologies (grey background). In ML methods, the tree and parameters maximizing the likelihood function are retained, while Bayesian approaches provide samples of trees and parameters consistent with the posterior distribution.

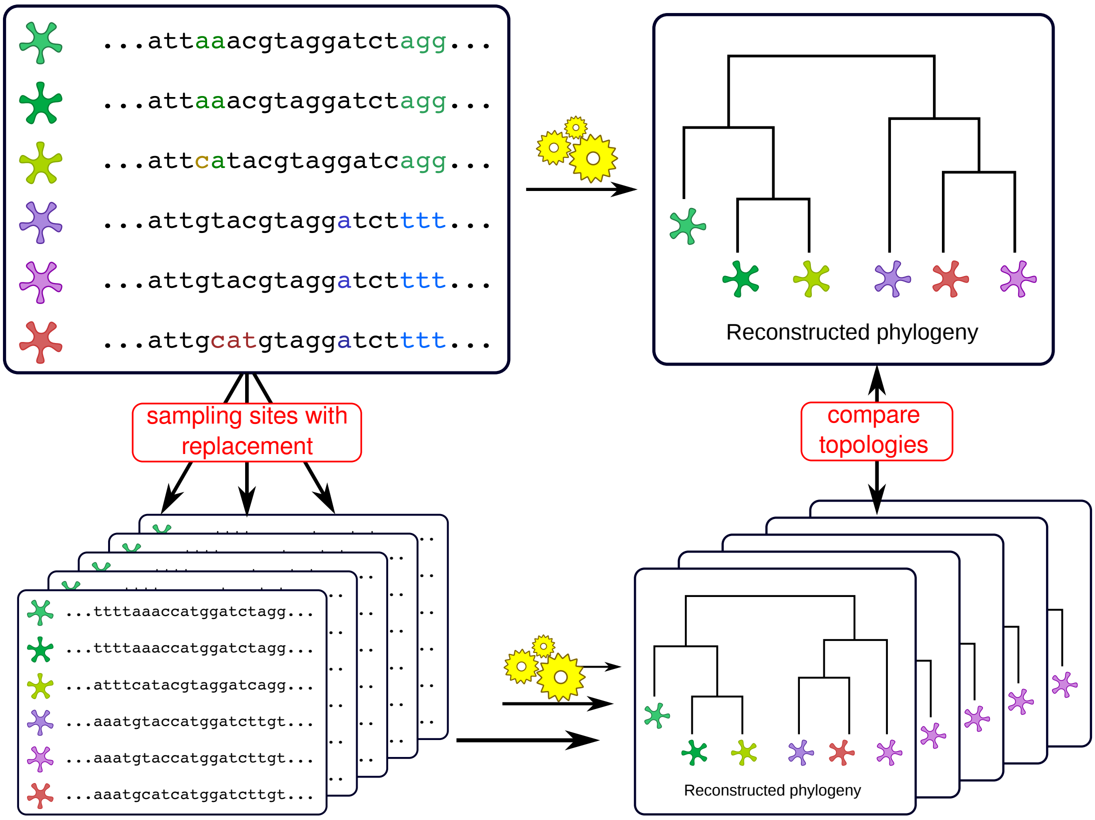

...attaaacgtaggatctagg...
...attaaacgtaggatctagg...
...attcatacgtaggatcagg...
...attgtacgtaggatctttt...
...attgtacgtaggatctttt...
...attgcatgtaggatctttt...
Reconstructed phylogeny
sampling sites with replacement
compare topologies
...ttttaaaccatggatctagg...
...ttttaaaccatggatctagg...
...atttcatacgtaggatcagg...
...aaatgtaccatggatcttgt...
...aaatgtaccatggatcttgt...
...aaatgcatcatggatcttgt...
Reconstructed phylogeny

Figure S4: bootstrapping phylogenies. Bootstrapping a phylogeny consists in comparing a reference tree to a set of trees obtained by re-sampling the data. Each re-sampled dataset yields a phylogeny which can be compared to the reference. If the original dataset is a random sample of the genome, then the variability observed across bootstrapped trees reflects the variability of the genome.

References

[1] S. Dray and A.-B. Dufour. The ade4 package: implementing the duality diagram for ecologists. *Journal of Statistical Software*, 22(4):1–20, 2007.

[2] T. Jombart. adegenet: a R package for the multivariate analysis of genetic markers. *Bioinformatics*, 24:1403–1405, 2008.

[3] T. Jombart and I. Ahmed. adegenet 1.3-1: new tools for the analysis of genome-wide snp data. *Bioinformatics*, 27:3070–3071, 2011.

[4] Scot A Kelchner and Michael A Thomas. Model use in phylogenetics: nine key questions. *Trends Ecol Evol*, 22(2):87–94, Feb 2007.

[5] E. Paradis, J. Claude, and K. Strimmer. APE: analyses of phylogenetics and evolution in R language. *Bioinformatics*, 20:289–290, 2004.

[6] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[7] Klaus Peter Schliep. phangorn: phylogenetic analysis in r. *Bioinformatics*, 27(4):592–593, Feb 2011.

[8] K. Tamura and M. Nei. Estimation of the number of nucleotide substitutions in the control region of mitochondrial dna in humans and chimpanzees. *Mol Biol Evol*, 10(3):512–526, May 1993.