phangorn 3.0

Klaus Schliep[1], Iris Bardel-Kahr[2], Elske van der Pol[1], Barnabas Daru[3], Leonardo de Oliveira Martins[8], Kristin Winchell[4], Robert Kourist[1], Emmanuel Paradis[5], and Liam J. Revell[6, 7]

[1]Graz University of Technology, Graz, Austria

[2]University of Graz, Graz, Austria

[3]Department of Biology, Stanford University, Stanford, CA, USA

[4]Department of Biology, New York University, New York, NY, USA

[5]SEM, University of Montpellier, CNRS, IRD, Montpellier, France

[6]Department of Biology, University of Massachusetts Boston, Boston, MA, USA

[7]Facultad de Ciencias, Universidad Católica de la Santísima Concepción, Concepción, Chile

[8]University of Liverpool, Faculty of Science and Engineering, Liverpool, UK

Corresponding author:

Klaus Schliep[1]

Email address: `klaus.schliep@tugraz.at`

ABSTRACT

Phylogentic trees are often the first step in many evolutionary analyses. The *phangorn* package (Schliep 2011) has become one of the most important packages to infer phylogentic trees in R. *phangorn* contains a variety of functions to infer phylogenetic trees from distances, with maximum parsimony or with maximum likelihood. It contains functions for reconstructing ancestral sequences, tree comparison and to visualize conflicting signals. We describe some significant features of and recent updates to the *phangorn* R package and use the opportunity to illustrate several popular workflows of the software.

1 INTRODUCTION

A phylogenetic tree shows relationship among evolutionary related objects in form of a connected graph without cycles. Objects are nowadays most often biological sequences, but can be also morphological measurements, languages, manuscripts.

The development of phylogenetics has been tightly linked to the use of computers since the late 1950's

(Felsenstein 2004). During many decades, the computations needed for phylogenetic inference were done with closed programs. The rise of free and open-source software in the late 1990's stimulated the development of a radically new approach based on the open collaborations and code sharing through the adoption of general public licencing. R, which was created in 1993, appeared an ideal choice to implement phylogenetic methods in the philosophy of sharing and collaborating.

When the development of *phangorn* started no phylogentic inference methods were available in R apart from distance based methods. *phangorn* introduced the possibility to infer phylogenies with Maximum Parsimony or Maximum Likelihood directly in R. Felsenstein (2004) and Yang (2014) offer more background, and mathematical detail over the methods we describe. We not only introduced new methods, but recently improved the popular distance base method UPGMA by adding tree rearrangements to improve the tree. UPGMA is in other fields better known as hierarchical clustering or average linkage. A strength of R is that there are the capabilities to manipulate data and visualisation. This allows to explorative analysis which otherwise needs several different programs.

Additionally R offers a huge ecosystem devoted to phylogenetic comparative methods which make use of phylogenies like *phytools* (Revell 2012; Revell 2024), *geiger* (Pennell et al. 2014), *ape* or *vegan* (Oksanen et al. 2022) and many more.

Nowadays there exist several R packages which allow to call specialized phylogenetic inference software like babette* (Bilderbeek and Etienne 2018) to BEAST2 (Bouckaert et al. 2014), *Revticulate* (Charpentier and Wright 2022) to RevBayes (Höhna et al. 2016) or *Rphylip* (Revell and Chamberlain 2014) to phylip (Felsenstein 2013), *ips* (Heibl et al. 2019) to several programs like on RAxML (Stamatakis 2014) and MrBayes (Ronquist and Huelsenbeck 2003).

The original article describing *phangorn* (Schliep 2011) was written over a decade ago and is in need of an update. With this article we want to take the opportunity to not only offer a placeholder for new citations, but take the opportunity to showcase some common workflows, several we improved and streamlined recently.

2 PHYLOGENETIC RECONSTRUCTION

2.1 The `phyDat` format

First we want to use the opportunity to introduce the `phyDat` object for storing alignments. Aligned sequences have a matrix form (see table 1). The `phyDat` format is similar to a `factor` and allows to store categorial data. Whereas in a `data.frame` each column or variable might have different categories, an alignment shares all the categories for all positions. Currently phangorn differentiates between four types of data, “DNA” to store nucleotide data, “AA” for amino acids, “CODON” for codon data and

Table 1. Excerpt from the original mites data set showing the first 6 species and 10 variables. See main text for additional details.

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
S._alpinus	2	2	3	2	4	4	0	2	0	0
S._arenocolus	2	2	3	2	4	7	0	2	0	0
S._ianus	2	2	3	2	4	7	0	2	0	0
S._minutus	2	2	3	2	4	7	0	2	0	0
S._pannonicus	2	2	3	2	4	7	0	2	0	0
S._pictus	3	2	1	0	4	6	0	2	0	0

finally "USER" for any kind of discrete / categorial data. There is a convenience function `dna2codon` to translate nucleotides to codons.

With the help of a contrast matrix we define ambiguous states (Felsenstein 2004). Most phylogenetic software code gaps, usually coded by "-", as ambiguous states assuming characters might not have been sequenced. On the other hand one might want to treat gaps as a state on its own right resulting from an insertion or deletions. `phangorn` provides a convenience function `gap_as_state` to switch easily from coding gaps as ambiguous to coding gap as an state for either nucleotide sequences or amino acids.

Now we start with a parsimony analysis using categorical morphological data. While it is most common nowadays to construct phylogenies from nucleotide sequences or amino acids, we have seen the data format in phangorn is quite flexible allowing any kind of discrete data.

2.2 Parsimony example with morphological data

A hurdle for morphological analysis is reading in the data and afterwards coding these properly. Here we profit from the data processing capabilities of the R environment. The dataset we are using contains morphological data for 12 mite species (Schäffer et al. 2010), with 79 encoded characters, see table 1 for a subset of the date. When reading in the *.csv* file, `row.names = 1` uses the first column (species) as row names. To get a `phyDat` object we are using for the analysis, we have to convert the data.frame into a matrix with `as.matrix`.

```r
library(phangorn)
fdir <- system.file("extdata", package = "phangorn")
mm <- read.csv(file.path(fdir, "mites.csv"), row.names = 1)
mm_pd <- phyDat(as.matrix(mm), type = "USER", levels = 0:7)
```

Now that we have our data, we can start the analyses. Maximum parsimony (MP) tries to find the tree with the least number of substitutions explaining the data. We use the function `pratchet` which a heuristic search of the tree space. It implements the parsimony ratchet (Nixon 1999) and uses subtree pruning and rearrangements (SPR) as a tree rearrangement.

```r
mm_tree <- pratchet(mm_pd, minit = 1000, maxit = 10000, all = TRUE,
                    trace = 0)
mm_tree
```

```
## 23 phylogenetic trees
```

Here we specified a few additional arguments. In this manuscript we frequently will set the argument `trace=0` or `pml.control(trace=0)`. This is done to avoid printing out progress of the current parameter estimates, so that the output in the document and insider R are as close as possible. With `all=TRUE` we get all trees with lowest parsimony score which were visited during the search in a `multiPhylo` object. The `minit` and `maxit` set the number of iterations for the ratchet. Since we set a minimum of 1000 iterations, we already have at least 1000 bootstrap samples for edge support as a byproduct of the parsimony ratchet. For larger trees this might takes some time and one might want to reduce the number of iterations. The trees returned by `pratchet` contain no edge weights. At last we assign edge lengths, but have to keep in mind that often there will be not a unique way to assign edge lengths.

```r
mm_tree <- acctran(mm_tree, mm_pd)
```

The first of the most parsimonious trees is shown in figure 1 a) with bootstrap values, which are stored at the nodes. Finally we can export these trees using the *ape* function `write.tree` to save in Newick or `write.nexus` to store in Nexus format (Maddison et al. 1997).

2.3 Branch and bound

In the case of our mites-dataset with 12 sequences, it's also possible to apply a branch and bound algorithm (Hendy and Penny 1982). This algorithm guarantees to find all most parsimonious trees, but in the worst case the algorithm has to evaluate every tree and it can very time consuming. If characters are supporting a specific tree, especially if characters are homoplasy free, this can be very effective. With bigger datasets (more than 20 taxonomic units) it is definitely recommended to use `pratchet`.

```r
mm_bab <- bab(mm_pd, trace = 0) |> acctran(mm_pd)
mm_bab
```

```
## 37 phylogenetic trees
```

In this case we found even one tree more than in the original article (Schäffer et al. 2010). We build a consensus network (Holland et al. 2004) containing all splits which are shared in at least 20% (eight trees) of all most parsimonious trees.

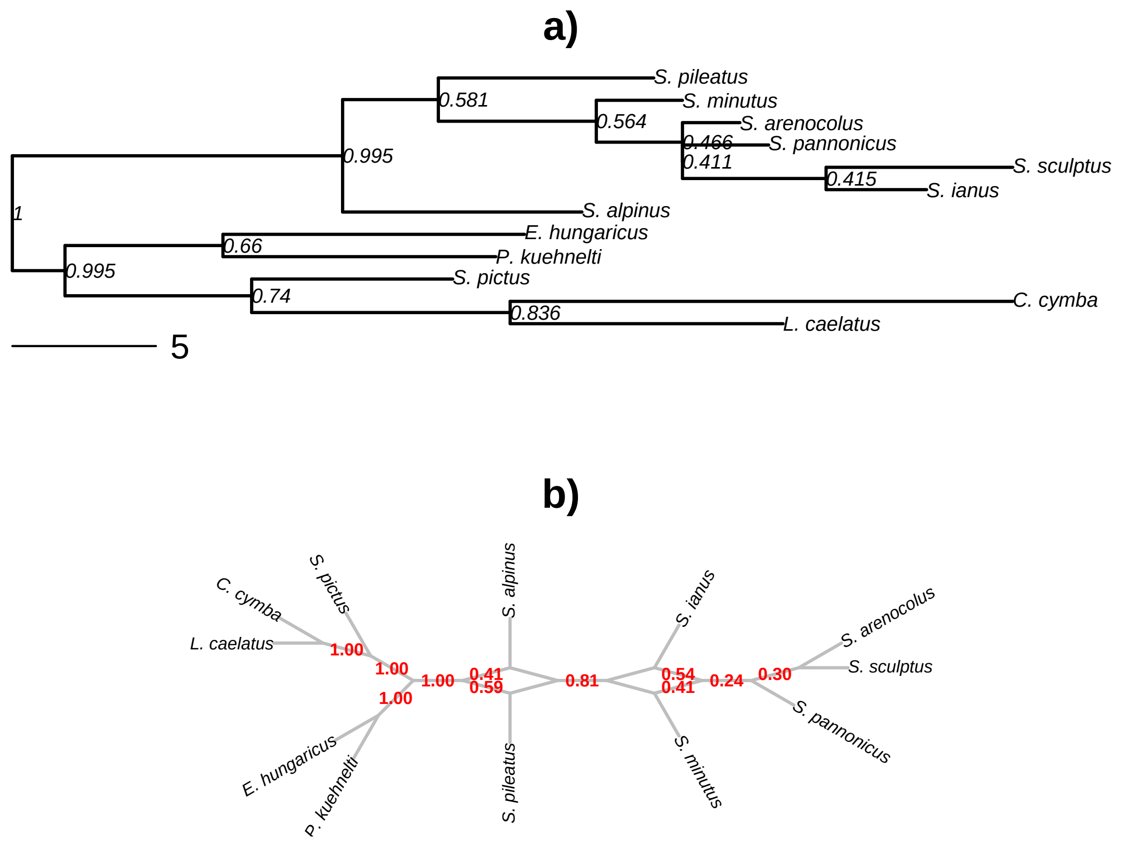

Figure 1. a) One of the maximum parsimony trees with bootstrap support and edge length assign using ACCTRAN. b) Consensus network of the 37 most parsimonious trees. See main text for additional details.

```
cnet <- consensusNet(mm_bab, p=0.2)
```

Figure 1 b) shows the consensus network as a splits graph (Dress and Huson 2004). The following lines of code have been used to plot the parsimony tree and the consensus network.

```
plot(midpoint(mm_tree[[1]]), show.node.label=TRUE, y.lim=c(-0.5, 12),
     cex=.6, main="a)", edge.width=1.5)
add.scale.bar(y=0, x=0)


plot(cnet, direction="axial", show.edge=TRUE, col.edge= "red",
     edge.color = "grey", digits=2, font.edge.label=2,
     cex=0.5, use.edge.length = FALSE, edge.width=1.5, main="b)")
```

The package *tanggle* (Schliep et al. 2022) provides an alternative to plot split graphs and explicit networks extending the *ggtree* package (Yu et al. 2017) in an “grammar of graphics” frame work (Wilkinson 1999; Wickham 2016).

Table 2. The 10 best models according to the BIC

Model	df	logLik	AIC	AICc	BIC
GTR+G(4)+I	101	-44599.13	89400.26	89406.96	90012.76
TIM2+G(4)+I	99	-44607.51	89413.03	89419.46	90013.40
TPM2u+G(4)+I	98	-44704.48	89604.96	89611.26	90199.27
TVM+G(4)+I	100	-44697.03	89594.07	89600.63	90200.50
TIM2+G(4)	98	-44738.40	89672.79	89679.09	90267.10
GTR+G(4)	100	-44730.65	89661.29	89667.86	90267.72
TPM2u+G(4)	97	-44828.00	89850.00	89856.18	90438.24
TVM+G(4)	99	-44820.85	89839.71	89846.14	90440.07
SYM+G(4)+I	98	-44837.02	89870.04	89876.34	90464.34
TIM1+G(4)+I	99	-44857.00	89912.00	89918.43	90512.37

Additionally the functions `read.nexus.networx` and `write.nexus.networx` can import and export split networks as nexus files together with some annotation to exchange with *Splitstree* (Huson and Bryant 2006) and other software.

2.4 Classical Maximum likelihood analysis

For molecular sequence data maximum likelihood (ML) and Bayesian MCMC inference is nowadays more common than maximum parsimony. Maximum likelihood inference for phylogentic trees offers a full statistical framework and estimates branch length, tree topology and substitution parameters (Felsenstein 1981, 2004). For large trees ($>$ 1000 taxa) *iqtree* (Nguyen et al. 2015) or *RAxML* (Stamatakis 2014) might be considerably faster.

As input we need aligned nucleotide or amino acid sequences and we can read in these directly with several popular file formats like fasta, phylip or nexus. In case of DNA or AA data there is no need for a transformation.

```
dna <- read.phyDat("Laurasiatherian.fas", format="fasta")
```

As a first we select the best fitting transition model. For this we call the function `modelTest` to compare different nucleotide or protein models with the AIC, AICc or BIC. This is similar to popular programs ModelTest and ProtTest (Posada and Crandall 1998; Abascal et al. 2005; Posada 2008). By default all available nucleotide or amino acid models are compared, but one can restrict the models tested.

```
mt <- modelTest(dna, control = pml.control(trace = 0))
```

Table 2 shows the best 10 models according to the BIC (Schwarz 1978). In this case the best of the tested models according to the BIC is GTR with invariant sites and a (discrete) gamma rate variation.

To speed up computations some the thresholds for the optimization in `modelTest` are not as strict set as in the functions `pml_bb` or `optim.pml` below. Most important no tree rearrangements are performed, which usually is the most time consuming part of the optimizing process. Furthermore the function `modelTest` can optimize models in parallel. As `modelTest` computes and optimizes a lot of models it would be a waste of computer time not to save these results.

The function `modelTest` not only returns the table but also stores the associated trees and model parameter which can used to head start the analysis. These are saved as a `call` together with the optimized trees in an environment and the function `as.pml` evaluates this call to get a `pml` object back to use for further optimization or analysis. This can either be done for a specific model (e.g. `as.pml(mt, "BIC")`), or for a specific criterion (e.g. `as.pml(mt, "HKY+G(4)+I")`).

After we found the best fitting model we can start optimizing all parameters including performing tree rearrangements. For this we use the function `pml_bb`, which internally calls the `optim.pml`. The function `pml_bb` has one mandatory argument `x`, but this can input can have different classes:

1. A data set an object of class `phyDat`, or from `AAbin`, `DNAbin` from the *ape* package and in this case we provide also model term e.g. “HKY+G(4)”. Additionally one can also supply a starting tree. If no tree is supplied than a starting tree is computed.

2. An object of class `modelTest`. in this case it extracts the model parameters according to the best model according to the BIC (AICc or AIC). So the optimization starts with already some parameters optimized.

3. An object of class `pml`. This can be the output of analysis from 1. or 2. It is often a good idea to estimate a tree using NNI rearrangements before doing extensive topology search using “stochastic” or “ratchet”. So one can have a look on an reasonable tree and also can get a feeling how the full analysis might take. The functions `pml.control` and `ratchet.control` allow to control the thresholds and iterations for the optimisation.

Next we showcase `pml_bb` providing a `modelTest` object we computed before.

```r
fit_mt <- pml_bb(mt, control = pml.control(trace = 0))
fit_mt
```

```
## model: GTR+G(4)+I
## loglikelihood: -44565.74
## unconstrained loglikelihood: -17300.92
## Proportion of invariant sites: 0.2923959
## Model of rate heterogeneity: Discrete gamma model
## Number of rate categories: 4
## Shape parameter: 0.5957749
##        Rate Proportion
## 1 0.0000000  0.2923959
## 2 0.0738235  0.1769010
## 3 0.4379787  0.1769010
## 4 1.2424293  0.1769010
## 5 3.8986475  0.1769010
##
## Rate matrix:
##           a          c          g         t
## a  0.000000  3.6246452 14.0126676  3.883554
## c  3.624645  0.0000000  0.4369491 25.428692
## g 14.012668  0.4369491  0.0000000  1.000000
## t  3.883554 25.4286925  1.0000000  0.000000
##
## Base frequencies:
##         a         c         g         t
## 0.3321866 0.1990791 0.2040652 0.2646691
```

The function `pml_bb` here takes an object returned by `modelTest` and performs now additional to optimizing the parameters of the model also tree rearrangements. At the time of writing there are three options for the argument `rearrangements` available: Nearest neighbor interchange (`"NNI"`) performs local tree rearrangements. This is usually pretty fast but only guarantees to find a local optimum. The option `"stochastic"` implements stochastic rearrangement algorithm following in (Nguyen et al. 2015) and `"ratchet"` the ML ratchet (Nixon 1999; Vos 2003). This might take quite a substantial amount of time, but often will result in better trees as they are able to escape local optima. And as a byproduct both these options return a bootstrap sample. With the default case of `rearrangements="stochastic"` "ultrafast bootstrapping" (Minh et al. 2013) and for `rearrangements="ratchet"` the classic bootstrap (Felsenstein 1985; Penny and Hendy 1985) are returned in the slot `bs`.

We can also have a look at the phylogeny using the generic function `plot` (figure 2). A generic function is a function whose behavior depends on the class of the input object. In this case `plot` calls the the function `plot.pml`.

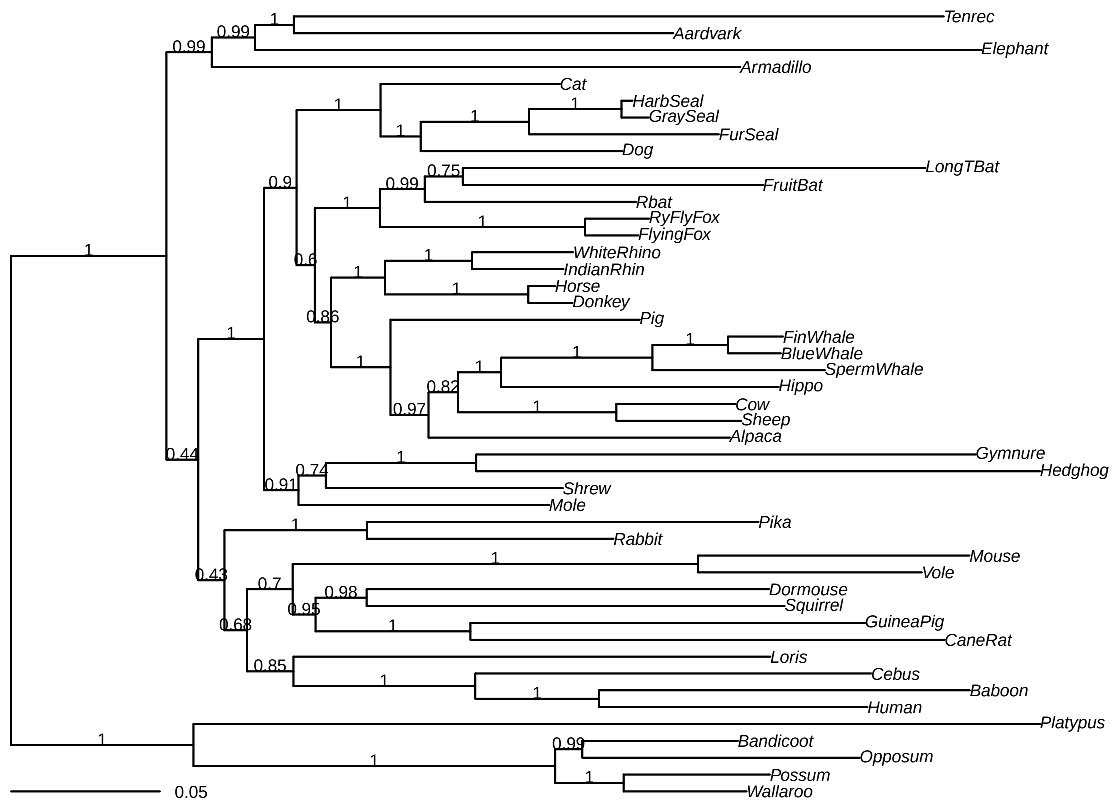

Figure 2. ML tree with approximate bootstrap support. The unrooted tree is midpoint rooted. See main text for additional details.

```
plot(fit_mt, cex=.5)
```

The generic `plot` function for `pml` objects is just a convenience function based on `plot` for `phylo` objects from the *ape* package. In figure 2 the edge length are the expected number of substitution per site, as we cannot distinguish between mutation rate and time. If some bootstrap was implicitly performed support values are shown.

2.5 Inference of rooted trees

Zuckerkandl and Pauling proposed the molecular clock in the 1960s (Zuckerkandl and Pauling 1965). Most programs for ML inference return unrooted trees. Rooted phylogenies can tell us about the ordering of the divergence times. *phangorn* offers the estimate an ultrametric trees (Paradis et al. 2023), that means all tips have the same distance to the root. In this case we still would need information about some internal nodes from fossils record to separate time and mutation rates. For fast evolving organisms or viruses we can estimate mutation rates and divergence times without the need of fossil calibration. In this case we only need molecular sequences and their associated sampling times. Most implementations for maximum likelihood analysis return unrooted trees. The `pml_bb` function offers the opportunity to estimate also ultrametric and tipdated phylogenies under a strict clock. The only other ML program we are aware of inferring tipdated phylogenies is *treetime* (Sagulenko et al. 2018). However *treetime* is very limited when it comes to tree rearrangement. There are several Bayesian phylogenetic tools (e.g. *BEAST* (Drummond and Rambaut 2007)) which allow to estimate rooted phylogenies and additional do not rely on a strict molecular clock, but use relaxed clock models.

First we show how to estimate ultrametric trees. One of the oldest methods is to infer ultrametric trees is UPGMA or average linkage clustering (Sokal and Michener 1958; Sneath and Sokal 1962; Murtagh 1985). *phangorn* provides an implementation which improves the underlying the minimum evolution criteria of UPGMA trees using efficient NNI rearrangements. The trees are also used as starting trees for the ML estimates.

```
dm <- dist.ml(dna, model = "F81", pairwise=TRUE)
tree_upgma <- upgma(dm, NNI=FALSE)
tree_upgma_nni <- upgma(dm)
fit_ultrametric <- pml_bb(dna, model="F81", method="ultrametric",
          rearrangement = "NNI", control=pml.control(trace = 0))
```

To fit a tipdated phylogeny we import an alignment and additionally a table containing the labels and the sampling date of the sequences. We than transform the two columns to an named vector to enable subsetting and ensure that the dates match.

```
tmp <- read.csv("../data/H3N2.csv")
H3N2 <- read.phyDat("../data/H3N2.fas", format="fasta")
dates <- setNames(tmp$numdate_given, tmp$accession)
head(dates)
```

```
## KF789866 CY148382 GQ895004 CY115546 CY001279 CY009150
## 2013.405 2012.838 2009.482 2009.523 2000.134 2000.682
```

phangorn can estimate tipdated phylogenies assuming a strict clock using from distances and using maximum likelihood. First we infer a tree with sUPGMA (Drummond and Rodrigo 2000), which extends the ultra-metric method UPGMA (Sokal and Michener 1958) to tipdated data.

```
dm <- dist.ml(H3N2, "F81")
tree_supgma <- supgma(dm, dates)
```

`supgma` takes as input a distance matrix and the vector with the tipdates and return a tree, where the tip dates are constraint. Next we estimate the tipdated phylogeny using maximum likelihood. For this we use again the function `pml_bb`, but need to specify the argument method as “tipdated” and provide the tip dates. A warning at the start, the optimization of a rooted phylogeny takes considerable more time than for an unrooted phylogeny.

```
fit_td <- pml_bb(H3N2, model="HKY+I" , method="tipdated",
                 tip.dates=dates, rearrangement="NNI",
                 control=pml.control(trace = 0))
fit_td
```

```
## model: HKY+I
## loglikelihood: -3117.857
## unconstrained loglikelihood: -2883.911
## Proportion of invariant sites: 0.6865497
##
## Rate: 0.002543586
##
## Rate matrix:
##         a       c       g       t
## a 0.00000 1.00000 9.86663 1.00000
## c 1.00000 0.00000 1.00000 9.86663
## g 9.86663 1.00000 0.00000 1.00000
## t 1.00000 9.86663 1.00000 0.00000
##
## Base frequencies:
##         a         c         g         t
## 0.3097759 0.1928617 0.2376819 0.2596805
##
## Rate: 0.002543586
```

In figure 3 a) we can see a comparison of the ML and sUPGMA tree we computed adopting the function `densiTree`. The classical use of the `densiTree` function we will see further down in figure 3 c).

If no starting tree is supplied `pml_bb` first estimates a tipdated phylogeny based on sUPGMA (Drummond and Rodrigo 2000). A major difference between tip dated, ultrametric and unrooted trees is that the number of parameter to estimate edge weights differs. For unrooted trees we have one parameter to estimate the length of each edge. In case of an binary tree this is $2n-3$, where n is the number of tips. For ultrametric trees the number of internal nodes, for a binary tree this is $n-1$. In case of tipdated phylogenies with a strict clock we have one additional parameter for the mutation rate.

After computing the ML tree we now can explicitly run an bootstrap analysis (Felsenstein 1985; Penny and Hendy 1985, 1986). During the optimization of each bootstrap sample we only change the tree topology and edge lengths. We could also optimize additional parameters, but this would increase the running time.

```
bs <- bootstrap.pml(fit_td, rearrangement="NNI",
                    bs = 100, control=pml.control(trace = 0))
```

In figure 3 we highlight different ways uncertainty in the topology and in divergence times. In figure 3 a) a summary of describing the bootstrap tree is presented in form of support values and boxplots are used to represent the divergence times.

```
par(mar=c(2,3,2,1)) #mfrow=c(2.2)
layout(matrix(c(1,1,1,2,3,4), 2, 3, byrow = TRUE))
densiTree(c(tree_upgma, tree_upgma_nni, fit_ultrametric$tree),
          type="phylogram", width=0.8, col=c("blue", "red", "green"),
          jitter=list(amount=0.25, random=FALSE), cex=0.6, alpha=1,
          main="a)", direction="right")
legend("topleft", c("UPGMA", "UPGMA NNI", "ML"), lty=1, lwd=2, bty="n",
       col=c("blue", "red", "green"))

densiTree(c(tree_supgma, fit_td$tree), type="phylogram", width=2,
     col=c("blue", "red"), jitter=list(amount=.2, random=FALSE),
     cex=.6, alpha=1, tip.dates=dates,  ylim=c(-3, 20), main="b)",
     direction="down")
legend("topleft", c("sUPGMA", "ML"), lty=1, lwd=2, bty="n",
       col=c("blue", "red"))

plot(fit_td, align.tip.label=TRUE, main="c)",
     direction="down", cex=.6, y.lim=c(-3, 20), x.lim=c(-0,20))
add_boxplot(fit_td$tree, bs, boxwex=.5, cex=2, outline = FALSE)
add_support(fit_td$tree, bs, adj = c(0.5, -0.5), cex = 0.7,
           method=c("FBP", "TBE"), scale=FALSE)

densiTree(bs, type="phylogram", width=2, tip.dates=dates, main="d)",
          direction="down", cex=.6,
          ylim=c(-3, 20), show.consensus = FALSE)
```

An alternative representation is a densiTree (Bouckaert 2010) figure 3 b). A nice property of this approach is that we present all the information available. All trees are plotted on top of each other, whereas otherwise a summary of this information is presented in form of support values and a box-and whisker plots for the divergence times.

Currently there are three flavors of support values available. Support values can be based on splits in case of unrooted trees (classical bootstrap), on clades for rooted trees and transfer bootstrap (Lemoine et al. 2018). The transfer bootstrap might be more appropriate for phylogenies with many taxa (Zaharias et al. 2023). In case of these small dataset all support values are often very similar.

There is often some confusion about what the units of the edge lengths in phylogenetic tree represent. For maximum likelihood trees the reported edge lengths are the expected number of changes per site (like) figure 2), and for parsimony trees (e.g. figure 1 a) the (total) number of substitutions. With tip dated trees we can distinguish the (mutation) rate and time and edge length in figure 3 are here proportional to time in years. In our case the rate is estimated to be around 0.0025 mutations per site per year. We know from theory that the expected number of substitutions should always be larger than the observed number of substitutions, which is the parsimony score.

```
# expected number of mutations
sum(fit_td$tree$edge.length) * fit_td$rate * 1407
```

```
## [1] 185.9422
```

```
# observed number of mutations
parsimony(fit_td$tree, H3N2)
```

```
## [1] 180
```

And it is the case. When the number expected number of substitutions is considerably higher than the parsimony one has to be cautious and might need further investigation. It is a sign of saturation of mutations or that the model might not be appropriate.

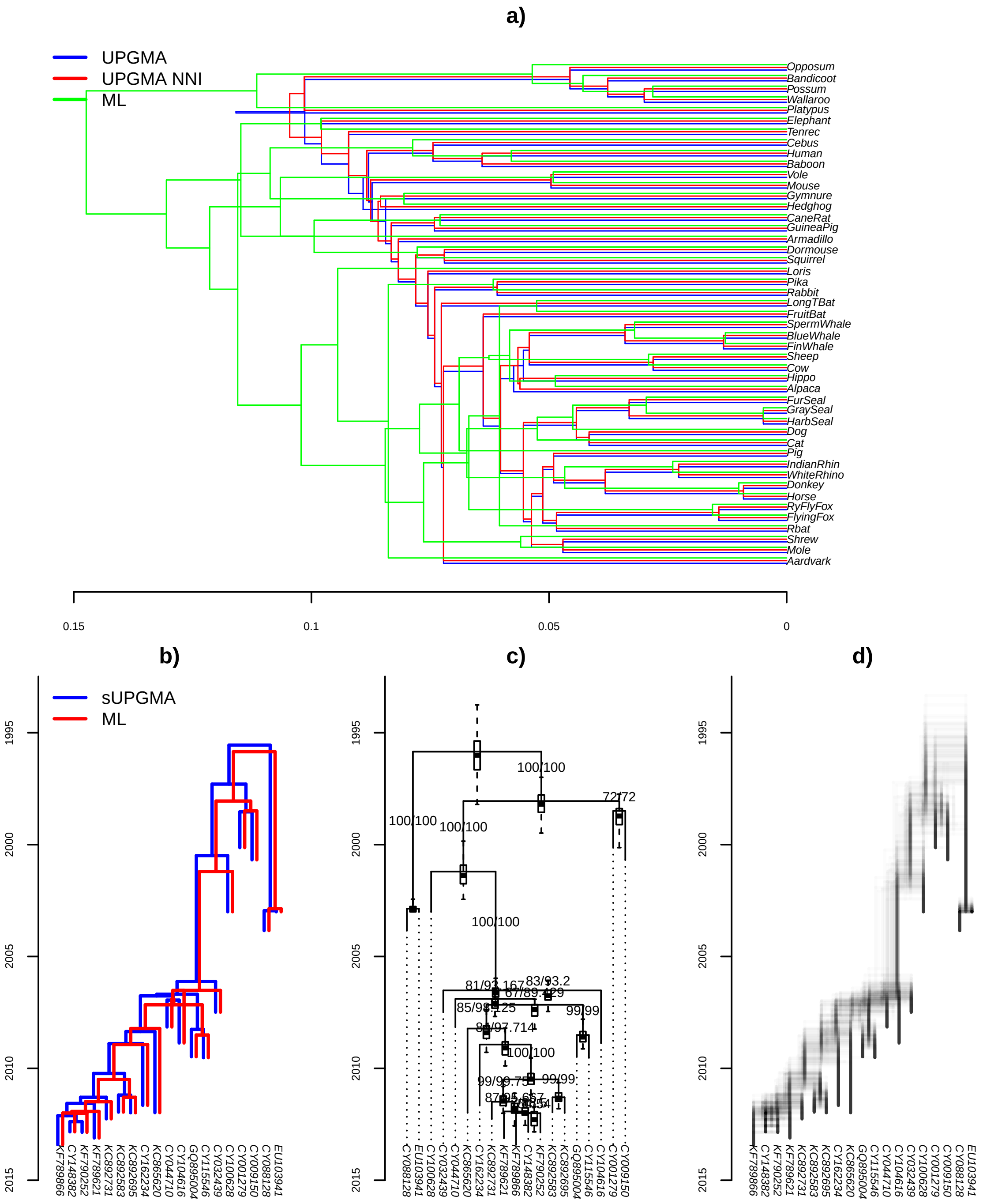

Figure 3. a) Ultrametric trees UPGMA with and without NNI rearrangement and maximum likelihood tree assuming a strict molecular clock. b) Tipdated sUPGMA (blue) and maximum likelihood tree (red) asuming strict clock model. c) ML tree with bootstrap support and uncertainty of the divergence times represented by boxplots. d) DensiTree for the same data. See main text for additional details.

3 ANCESTRAL SEQUENCE RECONSTRUCTION

Often we are not only interested in the tree (topology) but also in the ancestral sequence reconstruction (ASR) for specific nodes. ASR is becoming a common tool in biotechnology (Spence et al. 2021; Thomson et al. 2022; Nicoll et al. 2023). For example ASR of enzymes are often used as a starting point to engineer more heat tolerant enzymes or improve the reactions (Wheeler et al. 2016).

We first will load in an alignment and clean up that alignment. Furthermore we assume that gaps are a state represent absence of any nucleotide and not an ambiguous state.

```
aa <- read.phyDat("../data/GRASPTutorial_mafft.fasta",
                  format="fasta", type="AA")
names(aa) <- gsub( '\\s.*$', "", names(aa))
aa <- gap_as_state(aa)
```

In the following we will infer the ASR with maximum likelihood, but *phangorn* additionally offers the possibility to perform ASR with maximum parsimony. We first infer the ML tree and the associated parameter. In absence of knowledge of an out group we midpoint root the tree.

```
fit <- pml_bb(aa, model="LG+G(4)", rearrangement="NNI",
              control=pml.control(trace=0))
fit <- update(fit, tree=midpoint(fit$tree))
```

Finally we can perform ancestral sequence reconstruction. This object contains the tree, the original alignment and the marginal reconstruction. The tree needs to have unique node labels. This will ensure that the node labels on the tree correspond to labels of the character sequences and can be identified later on. If no node labels are present or these are not unique labels are created and added to the tree. This returns an object of class `ancestral`.

```
anc_ml <- anc_pml(fit)
```

An object of class `ancestral` contains several slots, the original alignment and the phylogenetic tree with node labels the ancestral reconstruction is based on. Additionally it contains an object with probabilities of each state for each internal node and site as the result of a marginal reconstruction. And the most likely state for each node as the result of a joint reconstruction (Pupko et al. 2000). When a model with rate variation is used the most likely state based on the marginal reconstruction is used. These objects can be exported in a format similar to iqtree and read in again to plot the reconstruction (see figure 4).

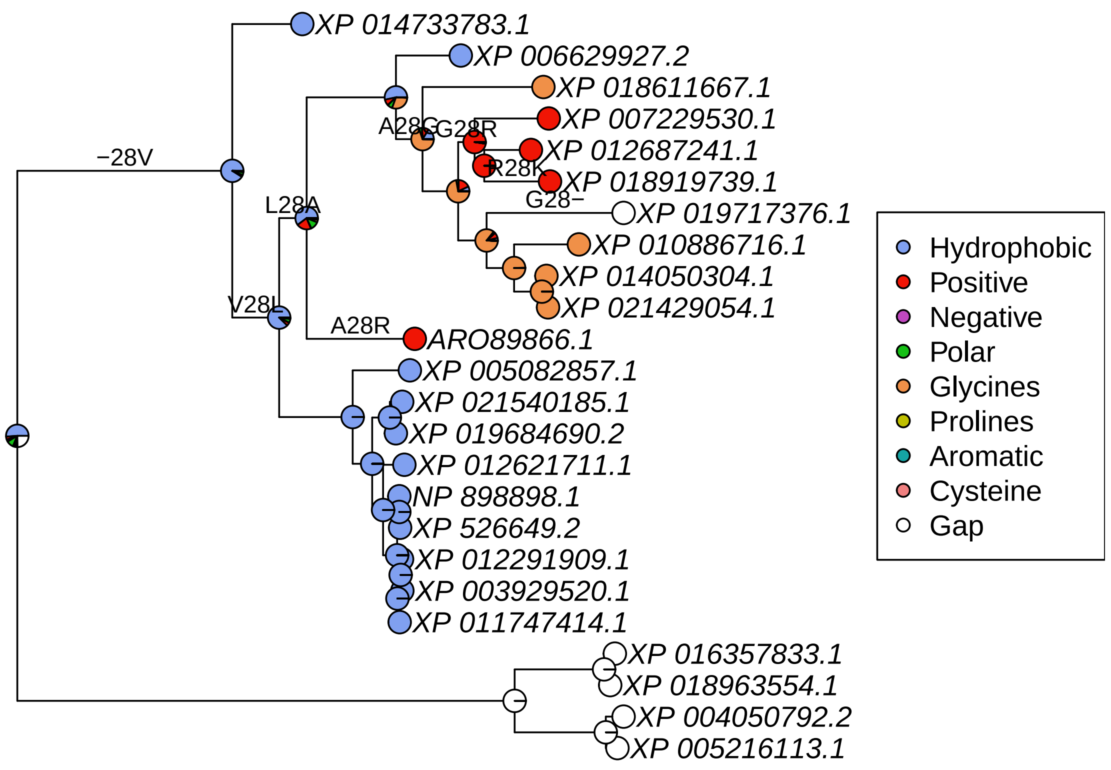

Figure 4. Maximum likelihood tree with the marginal reconstruction for the 28th character shown at each node. The mutations for this site are shown in their most likely position based on the joint reconstruction. See main text for additional details.

Figure 4 show the phylogenetic tree the proportion of each state shown as a pie diagram for each node with the ancestral reconstruction reconstruction for the 28th site (column) of the alignment.

```
plotAnc(anc_ml, 28, scheme="Clustal", pos="right", x.lim=c(0, 3.2))
add_mutations(anc_ml, pos=28, adj=c(0.5, -0.3), cex=.8)
```

4 EXPLORING TREES

The result of a phylogentic analysis is often a sample of trees, not just a single tree as we have seen already above. For example the samples of trees can represent gene trees, trees from a bootstrap or MCMC sample. *phangorn* in combination with *ape* offer several functions to process samples of trees and exploring these highlighting agreement and differences between them.

4.1 Processing trees

Before we run tree comparisons we often need to process the trees. Many comparison are based on support values, but we have to remember that support values are properties of splits and clades, even though support values stored at node or edge labels. There can be several edges which code for the same split, e.g. edges connecting a node of degree 2 or in more general graph structures like in splits networks as in figure 1.

- To root the trees with an out-group or at an node the *ape* function `root` is very flexible. Midpoint rooting is available through the function `midpoint`, if we just want to have a nicer looking tree. (Czech et al. 2017) highlighted that re-rooting might lead to wrong assignment of support values if these values are stored at the nodes. The function `root` in *ape* and `midpoint` can take care of this. However if the we support values on rooted trees which are based on clade probabilities, we always need to reassign support values as these might have changed.

- Another transformation is pruning trees to a certain set of tips. For example we might only show the tree for which we have data collected for phylogentic comparative methods. *ape* provides the generic functions function `drop.tip` and `keep.tip` to prune the tree. It is important to perform pruning before adding support values as these values will likely change and in case of so-called "rogue" taxa support values will increase dramatically.

- Remove short edges close to zero: Short edges can lead to spurious results for support values. Therefor we want remove edges which are close to zero. Many phylogenetic tree reconstruction methods return binary trees, even though some edges are zero or close to zero so a multifurcation is more appropriate. By default the shortest branch length returned by `pml_bb` is set by default is 1.0×10^{-8}. This default can be changed with the argument control and the helper function `pml.control`. To make things worse some phylogenetic reconstruction methods might depends on the input order of taxa, so that we would always see only one site pattern. To avoid this problem we can use the function `di2multi`. If the trees are ultrametric or tipdated setting the argument `tip2root = TRUE`, ensures that is also the case after removing the edges. So for rooted bootstrap trees we use a line like `di2multi(tree, tol=1e-7, tip2root = TRUE)`.

All the functions (`root`, `midpoint`, `drop.tip`, `keep.tip`, `di2multi`) mentioned in the paragraph above are generic. This means so they will work on a single tree, but also list of trees, an object of class `multiPhylo`. To promote reproducible research and support FAIR principle (Wilkinson et al. 2016; Jacobsen et al. 2020) it should be best practice to provide bootstrap or MCMC sample from the analysis and not only supplying a "best" tree with associated information like support values. This allows transform the trees (re-rooting, pruning) and assign afterwards support values.

In the following we taken 100 trees from a MCMC sample from (Upham et al. 2019) and prune these trees down to 60 tips.

```r
trees <- read.tree("../data/trees/Upham_trees_100.tre")
trees <- .compressTipLabel(trees)
# shorten names
short_nam <- attr(trees, "TipLabel") |> abbreviateGenus()
attr(trees, "TipLabel") <- short_nam
keep <- sample(short_nam, 40) # sample 40 names
trees <- keep.tip(trees, keep)
trees <- di2multi(trees, tol=0.5, tip2root = TRUE)
```

4.2 Consensus trees and networks

The next task is to compute a consensus tree. *Ape* provides the function `consensus` to compute strict (argument p=1) and majority consensus trees (argument p=0.5). These consensus trees might not be binary, but contain multifurcations. *Phangorn* extends these with the functions `maxCladeCred`, `allCompat` and `consensusNet`, which resolves multifurcations and in case of `consensusNet` can show alternative splits:

- allCompat: a greedy algorithm adding compatible splits or clades in order of their support to a majority consensus tree.
- consensusNet (Holland et al. 2004): extending the majority consensus to allow adding splits with a threshold smaller than 0.5.
- maxCladeCred: each internal split or clade gets a score as the fraction of seeing this split/ clade in the sample. The tree with the highest product of the scores presents the maximum clade credibility tree. So it is a tree from the sample, which is not guaranteed for the other consensus trees. However there might be several trees with the same score.

4.3 Assigning edge length

These Consensus trees do not have edge length associated with them. The exception is maximum clade credibility tree as it is one tree from the sample. Generally there are two options to assign edge lengths to a tree.

- We can extract and summarize the edge length or the distance from the root for rooted trees for all the bipartitions (edges) or clades (nodes) which are shared of the consensus tree and the sample of trees. For unrooted trees the functions `add_edgelength` computes the assign them derived from the splits or found in the sampled trees. If the tree is derive by any of the consensus tree methods there should be at least one split / clade also in the sample. If we choose an arbitrary tree this is not guaranteed.

- We can use an alignment when available as compute edge length for a given tree using a distance based method (`nnls.tree`), maximum parsimony (`acctran`) or maximum likelihood (`optim.pml`).

```
strict_consensus <- consensus(trees, rooted=TRUE) |>
                          add_edge_length(trees)
majority_consensus <- consensus(trees, p=.5, rooted=TRUE) |>
                          add_edge_length(trees)
all_compat <- allCompat(trees, rooted=TRUE) |> add_edge_length(trees)
max_clade_cred <- maxCladeCred(trees)
```

Figure 5 shows different consensus trees.

```
par(mar = c(1,1,2,1), mfrow=c(2,2))
plot(strict_consensus, cex=0.5)
mtext("a)", adj=0.1,line=0.5,cex=1.1)
plot(majority_consensus, cex=0.5)
mtext("b)", adj=0.1,line=0.5,cex=1.1)
plot(all_compat, cex=0.5)
mtext("c)", adj=0.1,line=0.5,cex=1.1)
plot(max_clade_cred, cex=0.5)
mtext("d)", adj=0.1,line=0.5,cex=1.1)
```

4.4 Tree distances

Having a set of trees we also are interested comparing these trees. *phangorn* contains several functions to compute distances between trees. The symmetric or Robinson-Foulds (Robinson and Foulds 1979, 1981) and its weighted version, the Kuhner-Felsenstein distance (Kuhner and Felsenstein 1994), the path distance (Steel and Penny 1993) and the approximate SPR - distance (Oliveira Martins 2008; De Oliveira Martins et al. 2016).

This complements several other packages which also offer tree distance measures distances e.g. quartet distance (Estabrook et al. 1985) in the package Quartet (Smith 2019), geodesic distances (Billera et al. 2001) in the package *distory* (Chakerian and Holmes 2020).

We compute all pairwise distances for the approximate SPR-distance and the Kuhner-Felsenstein distance from a vector consisting of the tip dated ML tree and the bootstrapped trees.

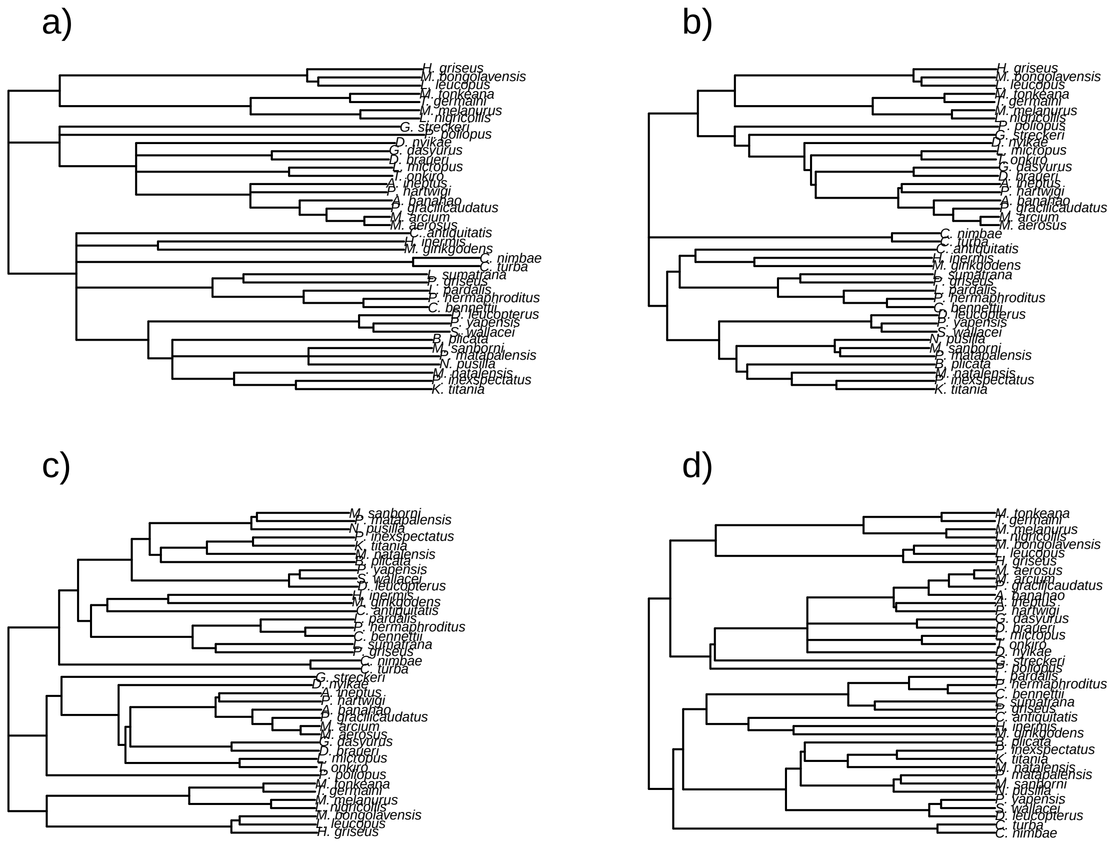

Figure 5. Different consensus trees. a) Strict consensus, b) Majority Consensus, c) Majority consensus tree with compatible splits, d) Maximum clade credibility tree, e) Consensus network. See main text for additional details.

```
trees  <- c(bs, fit_td$tree)
dm_spr <- SPR.dist(trees)
dm_kf <- KF.dist(trees)
dm_rf <- RF.dist(trees)
dm_pd <- path.dist(trees)
```

Figure 6 shows the projection of these distance using multidimensional scaling (MDS)(Gower 1966), where we highlight the ML tree in red. The Kuhner-Felsenstein distance takes edge length into account, whereas the SPR-distance only takes the branching patterns. For the Kuhner-Felsenstein all bootstrap trees are scattered around the ML tree. For the SPR-distance, which only takes the branching pattern into account, one could visually identify two cluster.

```
xy_spr <- cmdscale(dm_spr)
xy_rf <- cmdscale(dm_rf)
xy_kf  <- cmdscale(dm_kf)
xy_pd  <- cmdscale(dm_pd)
col <- c(rep("black", nrow(xy_spr)-1), "red")
pch <- c(rep(1, nrow(xy_spr)-1), 19)
par(mfrow=c(2, 2), mar=c(2, 2, 3, 1))
plot(xy_spr, asp=1, xlab="Coordinate 1", ylab="Coordinate 2",
     col=col, pch=pch)
mtext("a)", adj=0,line=0.5,cex=1.1)
plot(xy_kf, asp=1, xlab="Coordinate 1", ylab="Coordinate 2",
     col=col, pch=pch)
mtext("b)", adj=-0, line=0.5, cex=1.1)
plot(xy_rf, asp=1, xlab="Coordinate 1", ylab="Coordinate 2",
     col=col, pch=pch)
mtext("c)", adj=-0,line=0.5,cex=1.1)
plot(xy_pd, asp=1, xlab="Coordinate 1", ylab="Coordinate 2",
     col=col, pch=pch)
mtext("d)", adj=-0, line=0.5, cex=1.1)
```

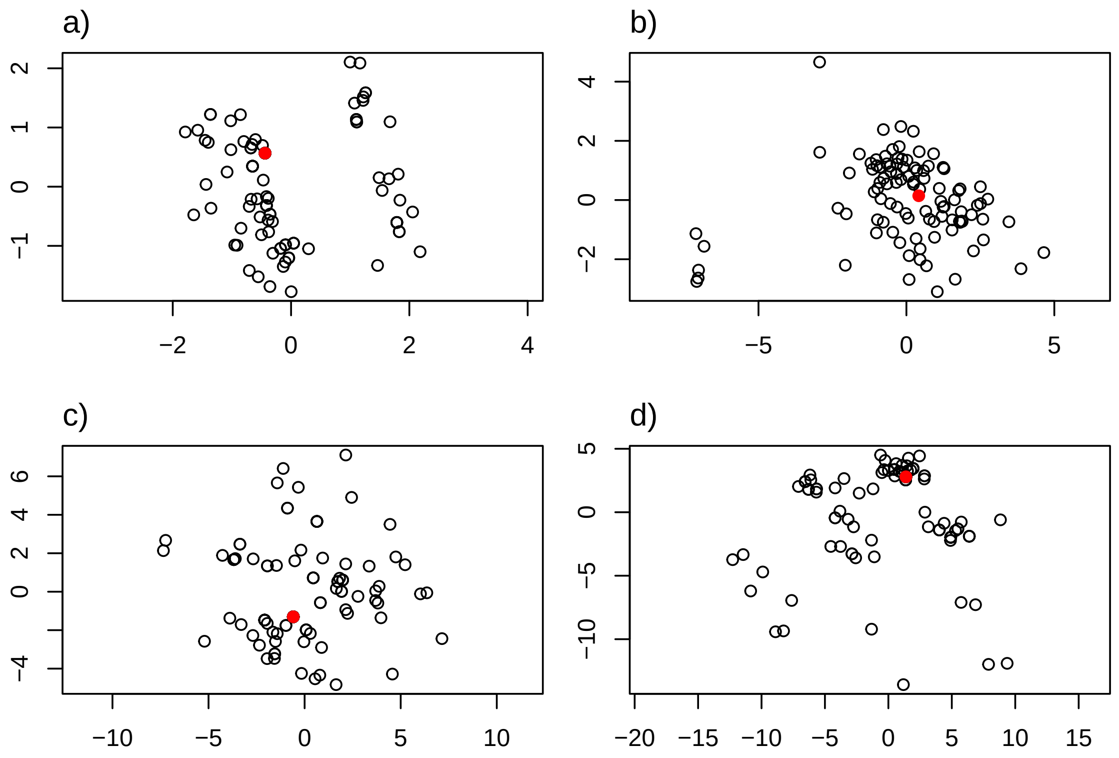

Figure 6. Multidimensional scaling based on the a) approximate SPR distance, the b) Kuhner-Felsenstein distance, c) the Robinson-Foulds and d) the path distance. See main text for additional details.

5 RELATIONSHIP OF *PHANGORN* TO OTHER PACKAGES AND SOFTWARE

The *phangorn* package has grown to become (along with *ape*(Paradis and Schliep 2019), *phytools* (Revell 2024) and *geiger*) among the most important core packages for phylogenetic analysis in the R environment. Nowadays around 50 packages on CRAN or Bioconductor depend on the *phangorn* package. As of the time of writing, the original publication describing *phangorn* (Schliep 2011) had been cited more than 3,000 times on *Google Scholar* and continues to be cited over 400 times per year.

From the start *phangorn* has been relying heavily on object classes and methods of the core R phylogenetics package, *ape* (Paradis et al. 2004; Paradis and Schliep 2019). *phangorn* has its own class *phyDat* for storing alignments, but there are many convenience functions to easily transform to the classes `AAbin` and `DNAbin` (*ape*) or to `data.frames` for categorical data. Several frequently used functions in *phangorn* (`bab`, `pml_bb`, `dist.ml`, `pratchet`) accept the classes `AAbin` and `DNAbin` from *ape* directly as input. Furthermore alignments can exchanged `alignment` with the package *Biostrings* (Pagès et al. 2022) and connect to the Bioconductor world.

Apart from the *ape* package *phangorn* currently depends on a number of other packages. First of all *phangorn* depends on many of the R core packages (R Core Team 2020). Furthermore on the packages *digest* (Dirk Eddelbuettel with contributions by Antoine Lucas et al. 2022), *fastmatch* (Urbanek 2021), *generics* (Wickham et al. 2022), *igraph* (Csardi and Nepusz 2006), *Matrix* (Bates et al. 2023), *quadprog* (Berwin A. Turlach with contributions by Andreas Weingessel and Moler 2019), and *Rcpp* (Eddelbuettel and François 2011; Eddelbuettel 2013; Eddelbuettel and Balamuta 2018). This relationship evolve with the package development and might change.

Ancestral reconstructions from iqtree or networks from Splitstree can be imported and visualized.

6 CONCLUSIONS

More than a decade has passed since the original article describing *phangorn* was published (Schliep 2011). Since that time, the *phangorn* package has both evolved into one of the core function libraries of the R phylogenetics ecosystem, and expanded in size an scope. We decided the literature reference for *phangorn* was sorely in need of updating. In creating one, we were determined to make something that could serve as more than a placeholder to capture citations of the *phangorn* package. We hope that this will help guide new *phangorn* users towards interesting analytical tools, as well as perhaps inspire experienced *phangorn* and R phylogenetics researchers to generate new types of questions and data that will in turn help motivate and participate in the continued development of the *phangorn* package into the future.

7 SOFTWARE AND DATA AVAILABILITY

The *phangorn* is free and open source, and can be downloaded from its CRAN (`https://CRAN.R-project.org/package=phangorn`) or the development version from GitHub (`https://github.com/KlausVigo/phangorn`) pages. Binaries for the development version are kindly supplied on the r-universe pages (`https://klausvigo.r-universe.dev/phangorn#`).

This article was written in Rmarkdown (Xie et al. 2018, 2020; Allaire et al. 2023), and developed with the help of both *bookdown* (Xie 2016, 2023) and the posit Rstudio IDE (RStudio Team 2023).

The underlying markdown code and data files necessary to reproduce the analyses of this article are available at `https://github.com/KlausVigo/phangorn-v3/`.

REFERENCES

Abascal, F., R. Zardoya, and D. Posada. 2005. ProtTest: Selection of best-fit models of protein evolution. Bioinformatics 21:2104–2105.

Allaire, J., Y. Xie, C. Dervieux, J. McPherson, J. Luraschi, K. Ushey, A. Atkins, H. Wickham, J. Cheng, W. Chang, and R. Iannone. 2023. Rmarkdown: Dynamic documents for r.

Bates, D., M. Maechler, and M. Jagan. 2023. Matrix: Sparse and dense matrix classes and methods.

Berwin A. Turlach with contributions by Andreas Weingessel, and, and C. Moler. 2019. Quadprog: Functions to solve quadratic programming problems.

Bilderbeek, R. J., and R. S. Etienne. 2018. Babette: BEAUti 2, BEAST 2 and tracer for r. Methods in Ecology and Evolution. Wiley Online Library.

Billera, L. J., S. P. Holmes, and K. Vogtmann. 2001. Geometry of the space of phylogenetic trees. Advances in Applied Mathematics 27:733–767.

Bouckaert, R. R. 2010. DensiTree: Making sense of sets of phylogenetic trees. Bioinformatics 26:1372–1373.

Bouckaert, R., J. Heled, D. Kühnert, T. Vaughan, C.-H. Wu, D. Xie, M. A. Suchard, A. Rambaut, and A. J. Drummond. 2014. BEAST 2: A software platform for bayesian evolutionary analysis. PLoS computational biology 10:e1003537. Public Library of Science San Francisco, USA.

Chakerian, J., and S. Holmes. 2020. Distory: Distance between phylogenetic histories.

Charpentier, C. P., and A. M. Wright. 2022. Revticulate: An r framework for interaction with RevBayes. Methods in Ecology and Evolution 13:1177–1184.

Csardi, G., and T. Nepusz. 2006. The igraph software package for complex network research. InterJournal Complex Systems:1695.

Czech, L., J. Huerta-Cepas, and A. Stamatakis. 2017. A Critical Review on the Use of Support Values in Tree Viewers and Bioinformatics Toolkits. Molecular Biology and Evolution 34:1535–1542.

De Oliveira Martins, L., D. Mallo, and D. Posada. 2016. A bayesian supertree model for genome-wide species tree reconstruction. Systematic Biology 65:397–416.

Dirk Eddelbuettel with contributions by Antoine Lucas, and, J. Tuszynski, H. Bengtsson, S. Urbanek, M. Frasca, B. Lewis, M. Stokely, H. Muehleisen, D. Murdoch, J. Hester, W. Wu, Q. Kou, T. Onkelinx, M. Lang, V. Simko, K. Hornik, R. Neal, K. Bell, M. de Queljoe, I. Suruceanu, B. Denney, D. Schumacher, and and Winston Chang. 2022. Digest: Create compact hash digests of r objects.

Dress, A. W. M., and D. H. Huson. 2004. Constructing splits graphs. IEEE/ACM Transactions on Computational Biology and Bioinformatics 1:109–115.

Drummond, A. J., and A. Rambaut. 2007. BEAST: Bayesian evolutionary analysis by sampling trees. BMC evolutionary biology 7:1–8. BioMed Central.

Drummond, A., and A. G. Rodrigo. 2000. Reconstructing genealogies of serial samples under the assumption of a molecular clock using serial-sample UPGMA. Molecular Biology and Evolution 17:1807–1815. Society for Molecular Biology; Evolution.

Eddelbuettel, D. 2013. Seamless R and C++ integration with Rcpp. Springer, New York.

Eddelbuettel, D., and J. J. Balamuta. 2018. Extending R with C++: A Brief Introduction to Rcpp. The American Statistician 72:28–36.

Eddelbuettel, D., and R. François. 2011. Rcpp: Seamless R and C++ integration. Journal of Statistical Software 40:1–18.

Estabrook, G. F., F. McMorris, and C. A. Meacham. 1985. Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. Systematic Zoology 34:193–200. Society of Systematic Zoology.

Felsenstein, J. 1985. Confidence limits on phylogenies. An approach using the bootstrap. Evolution 39:783–791.

Felsenstein, J. 1981. Evolutionary trees from DNA sequences: A maximum likelihood approach. Journal of Molecular Evolution 17:368–376.

Felsenstein, J. 2004. Inferring phylogenies. Sinauer Associates, Sunderland.

Felsenstein, J. 2013. PHYLIP (phylogeny inference package), version 3.695. Joseph Felsenstein.

Gower, J. C. 1966. Some distance properties of latent root and vector methods used in multivariate analysis. Biometrika 53:325.

Heibl, C., N. Cusimano, and F.-S. Krah. 2019. Ips: Interfaces to phylogenetic software in r.

Hendy, M. D., and D. Penny. 1982. Branch and bound algorithms to determine minimal evolutionary trees. Math. Biosc. 59:277–290.

Höhna, S., M. J. Landis, T. A. Heath, B. Boussau, N. Lartillot, B. R. Moore, J. P. Huelsenbeck, and F. Ronquist. 2016. RevBayes: Bayesian phylogenetic inference using graphical models and an interactive model-specification language. Systematic biology 65:726–736. Oxford University Press.

Holland, B. R., K. T. Huber, V. Moulton, and P. J. Lockhart. 2004. Using consensus networks to visualize contradictory evidence for species phylogeny. Molecular Biology and Evolution 21:1459–1461.

Huson, D. H., and D. Bryant. 2006. Application of phylogenetic networks in evolutionary studies. Molecular Biology and Evolution 23:254–267.

Jacobsen, A., R. de Miranda Azevedo, N. Juty, D. Batista, S. Coles, R. Cornet, M. Courtot, M. Crosas, M. Dumontier, C. T. Evelo, C. Goble, G. Guizzardi, K. K. Hansen, A. Hasnain, K. Hettne, J. Heringa, R. W. W. Hooft, M. Imming, K. G. Jeffery, R. Kaliyaperumal, M. G. Kersloot, C. R. Kirkpatrick, T. Kuhn, I. Labastida, B. Magagna, P. McQuilton, N. Meyers, A. Montesanti, M. van Reisen, P. Rocca-Serra, R. Pergl, S.-A. Sansone, L. O. B. da Silva Santos, J. Schneider, G. Strawn, M. Thompson, A. Waagmeester, T. Weigel, M. D. Wilkinson, E. L. Willighagen, P. Wittenburg, M. Roos, B. Mons, and E. Schultes. 2020. FAIR Principles: Interpretations and Implementation Considerations. Data Intelligence 2:10–29.

Kuhner, M. K., and J. Felsenstein. 1994. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. Molecular Biology and Evolution 11:459–468.

Lemoine, F., J.-B. D. Entfellner, E. Wilkinson, D. Correia, M. D. Felipe, T. De Oliveira, and O. Gascuel. 2018. Renewing felsenstein's phylogenetic bootstrap in the era of big data. Nature 556:452–456.

Maddison, D. R., D. L. Swofford, and W. P. Maddison. 1997. Nexus: An extensible file format for systematic information. Systematic Biology 46:590–621.

Minh, B. Q., M. A. T. Nguyen, and A. von Haeseler. 2013. Ultrafast approximation for phylogenetic bootstrap. Molecular biology and evolution 30:1188–1195. Society for Molecular Biology; Evolution.

Murtagh, F. 1985. Multidimensional clustering algorithms. Compstat lectures.

Nguyen, L.-T., H. A. Schmidt, A. von Haeseler, and B. Q. Minh. 2015. IQ-TREE: A fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. Molecular Biology and Evolution 32:268–274.

Nicoll, C. R., M. Massari, M. W. Fraaije, M. L. Mascotti, and A. Mattevi. 2023. Impact of ancestral sequence reconstruction on mechanistic and structural enzymology. Current Opinion in Structural Biology 82:102669. Elsevier.

Nixon, K. 1999. The parsimony ratchet, a new method for rapid rarsimony analysis. Cladistics 15:407–414.

Oksanen, J., G. L. Simpson, F. G. Blanchet, R. Kindt, P. Legendre, P. R. Minchin, R. B. O'Hara, P. Solymos, M. H. H. Stevens, E. Szoecs, H. Wagner, M. Barbour, M. Bedward, B. Bolker, D. Borcard, G. Carvalho, M. Chirico, M. De Caceres, S. Durand, H. B. A. Evangelista, R. FitzJohn, M. Friendly, B. Furneaux, G. Hannigan, M. O. Hill, L. Lahti, D. McGlinn, M.-H. Ouellette, E. Ribeiro Cunha, T. Smith, A. Stier, C. J. F. Ter Braak, and J. Weedon. 2022. Vegan: Community ecology package.

Oliveira Martins, É. A. K. de, Leonardo AND Leal. 2008. Phylogenetic detection of recombination with a bayesian prior on the distance between trees. PLoS ONE 3:1–13. Public Library of Science.

Pagès, H., P. Aboyoun, R. Gentleman, and S. DebRoy. 2022. Biostrings: Efficient manipulation of biological strings.

Paradis, E., S. Claramunt, J. Brown, and K. Schliep. 2023. Confidence intervals in molecular dating by maximum likelihood. Molecular Phylogenetics and Evolution 178:107652.

Paradis, E., J. Claude, and K. Strimmer. 2004. APE: Analyses of phylogenetics and evolution in R language. Bioinformatics 20:289–290.

Paradis, E., and K. Schliep. 2019. Ape 5.0: An environment for modern phylogenetics and evolutionary analyses in r. Bioinformatics 35:526–528.

Pennell, M. W., J. M. Eastman, G. J. Slater, J. W. Brown, J. C. Uyeda, R. G. Fitzjohn, M. E. Alfaro, and L.

J. Harmon. 2014. Geiger v2.0: An expanded suite of methods for fitting macroevolutionary models to phylogenetic trees. Bioinformatics 30:2216–2218.

Penny, D., and M. D. Hendy. 1986. Estimating the reliability of evolutionary trees. Molecular Biology and Evolution 3:403–417.

Penny, D., and M. D. Hendy. 1985. Testing methods evolutionary tree construction. Cladistics 1:266–278.

Posada, D. 2008. jModelTest: Phylogenetic model averaging. Molecular Biology and Evolution 25:1253–1256.

Posada, D., and K. A. Crandall. 1998. MODELTEST: Testing the model of DNA substitution. Bioinformatics 14:817–818.

Pupko, T., I. Pe, R. Shamir, and D. Graur. 2000. A fast algorithm for joint reconstruction of ancestral amino acid sequences. Molecular Biology and Evolution 17:890–896.

R Core Team. 2020. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.

Revell, L. J. 2024. Phytools 2.0: An updated r ecosystem for phylogenetic comparative methods (and other things). PeerJ 12:e16505. PeerJ Inc.

Revell, L. J. 2012. Phytools: An r package for phylogenetic comparative biology (and other things). Methods in Ecology and Evolution 3:217–223.

Revell, L. J., and S. A. Chamberlain. 2014. Rphylip: An R interface for PHYLIP. Methods in Ecology and Evolution 5:976–981.

Robinson, D. F., and L. R. Foulds. 1979. Combinatorial mathematics VI: Proceedings of the sixth australian conference on combinatorial mathematics, armidale, australia, august 1978. Pp. 119–126 *in* A. F. Horadam and W. D. Wallis, eds. Springer Berlin Heidelberg, Berlin, Heidelberg.

Robinson, D. F., and L. R. Foulds. 1981. Comparison of phylogenetic trees. Mathematical Biosciences 53:131–147.

Ronquist, F., and J. P. Huelsenbeck. 2003. MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics 19:1572–1574. Oxford University Press.

RStudio Team. 2023. RStudio: Integrated development environment for r. Posit, PBC., Boston, MA.

Sagulenko, P., V. Puller, and R. A. Neher. 2018. TreeTime: Maximum-likelihood phylodynamic analysis. Virus evolution 4:vex042. Oxford University Press.

Schäffer, S., T. Pfingstl, S. Koblmüller, K. A. Winkler, C. Sturmbauer, and G. Krisper. 2010. Phylogenetic analysis of european scutovertex mites (acari, oribatida, scutoverticidae) reveals paraphyly and cryptic diversity: A molecular genetic and morphological approach. Molecular Phylogenetics and Evolution 55:677–688.

Schliep, K. P. 2011. Phangorn: Phylogenetic analysis in R. Bioinformatics 27:592–593.

Schliep, K., M. Vidal-Garcia, C. Solis-Lemus, L. Biancani, E. Ada, and L. F. Henao Diaz. 2022. Tanggle: Visualization of phylogenetic networks.

Schwarz, G. 1978. Estimating the dimension of a model. Ann. Statist. 6:461–464. The Institute of Mathematical Statistics.

Smith, M. R. 2019. Quartet: Comparison of phylogenetic trees using quartet and split measures.

Sneath, P. H., and R. R. Sokal. 1962. Numerical taxonomy. Nature 193:855–860. Nature Publishing Group UK.

Sokal, R., and C. Michener. 1958. A statistical method for evaluating systematic relationships. University of Kansas Science Bulletin 38:1409–1438.

Spence, M. A., J. A. Kaczmarski, J. W. Saunders, and C. J. Jackson. 2021. Ancestral sequence reconstruction for protein engineers. Current Opinion in Structural Biology 69:131–141.

Stamatakis, A. 2014. RAxML version 8: A tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics 30:1312–1313. Oxford University Press.

Steel, M. A., and D. Penny. 1993. Distributions of tree comparison metrics—some new results. Systematic biology 42:126–141. Society of Systematic Biologists.

Thomson, R. E. S., S. E. Carrera-Pacheco, and E. M. J. Gillam. 2022. Engineering functional thermostable proteins using ancestral sequence reconstruction. Journal of Biological Chemistry 298:102435.

Upham, N. S., J. A. Esselstyn, and W. Jetz. 2019. Inferring the mammal tree: Species-level sets of phylogenies for questions in ecology, evolution, and conservation. PLoS biology 17:e3000494. Public Library of Science San Francisco, CA USA.

Urbanek, S. 2021. Fastmatch: Fast 'match()' function.

Vos, R. A. 2003. Accelerated likelihood surface exploration: The likelihood ratchet. Systematic Biology 52:368–373.

Wheeler, L. C., S. A. Lim, S. Marqusee, and M. J. Harms. 2016. The thermostability and specificity of ancient proteins. Current opinion in structural biology 38:37–43. Elsevier.

Wickham, H. 2016. ggplot2: Elegant graphics for data analysis. Springer-Verlag New York.

Wickham, H., M. Kuhn, and D. Vaughan. 2022. Generics: Common S3 generics not provided by base r methods related to model fitting.

Wilkinson, L. 1999. The grammar of graphics. 1st ed. Springer, New York.

Wilkinson, M. D., M. Dumontier, Ij. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. G. Gray, P. Groth, C.

Goble, J. S. Grethe, J. Heringa, P. A. C. ’t Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, and B. Mons. 2016. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data 3.

Xie, Y. 2023. Bookdown: Authoring books and technical documents with r markdown.

Xie, Y. 2016. Bookdown: Authoring books and technical documents with R markdown. Chapman; Hall/CRC, Boca Raton, Florida.

Xie, Y., J. J. Allaire, and G. Grolemund. 2018. R markdown: The definitive guide. Chapman; Hall/CRC, Boca Raton, Florida.

Xie, Y., C. Dervieux, and E. Riederer. 2020. R markdown cookbook. Chapman; Hall/CRC, Boca Raton, Florida.

Yang, Z. 2014. Molecular evolution: A statistical approach. Oxford University Press, Oxford.

Yu, G., D. Smith, H. Zhu, Y. Guan, and T. T.-Y. Lam. 2017. Ggtree: An r package for visualization and annotation of phylogenetic trees with their covariates and other associated data. Methods in Ecology and Evolution 8:28–36.

Zaharias, P., F. Lemoine, and O. Gascuel. 2023. Robustness of Felsenstein’s versus Transfer Bootstrap Supports with respect to Taxon Sampling. Systematic Biology syad052.

Zuckerkandl, E., and L. Pauling. 1965. Molecules as documents of evolutionary history. Journal of theoretical biology 8:357–366.