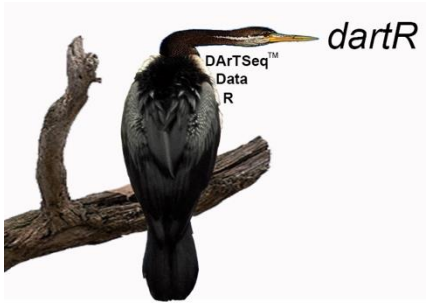
UNIVERSITY OF CANBERRA

SNP Analysis using dartR



Phylogenetics

Version 1

Copies of the latest version of this tutorial are available from:

The Institute for Applied Ecology
University of Canberra ACT 2601
Australia

Email: georges@aerg.canberra.edu.au

dartR is a collaboration between the University of Canberra, CSIRO and Diversity Arrays Technology, and is supported with funding from the ACT Priority Investment Program, CSIRO and the University of Canberra.

Contents

Introduction

Phylogenetics

Phylogenetics is the study of the evolutionary relationships among entities whether they be species, lineages within species, genes or proteins. The objective of phylogenetic analysis is to recover the evolutionary history of a target group of organisms.

In the absence of fossil material, or as a complement to the information contained in fossils, phylogenetic relationships are inferred from data associated with heritable traits such DNA sequences, amino acid sequences or morphological attributes. These data are collected from individuals collected from a snapshot in time, the present.

From the patterns so revealed, a defensible phylogenetic history can be inferred. As such, the fodder of phylogenetics is a matrix of individuals by traits (e.g. a genetic locus), with the body of the matrix containing the character state for each individual (e.g. DNA sequence).

The end product of a phylogenetic analysis is a phylogenetic tree which represents the inferred evolutionary relationships – the pattern of ancestry and descent in the form of a bifurcating tree.

Unfortunately, the evolutionary history of a group of organisms is not uniquely defined by the data we can collect from the extant terminal entities or fossils. The challenge is to extract the phylogenetic signal in the context of a substantial body of noise. This noise arises from homoplasy, that is, arises by chance under drift or by convergence under selection.

The phylogenetic tree that emerges from the analysis as the best representation of the evolutionary history of the target group can be the one most parsimoniously consistent with available data, or the one most likely given the available data, or the best guess given the available data and some additional relevant prior knowledge.

In this sense, a phylogenetic tree should be viewed as a phylogenetic hypothesis, one to be tested in the light of new evidence as it comes to hand. This mindset encourages collection of additional data to address areas of contention rather than focussing on confirming areas with unqualified support.

Phylogenetics and SNPs

SNP markers have traditionally been applied in the domain of population genetics but are increasingly being applied to address phylogenetic hypotheses. SNPs have their strengths and their limitations, especially if applied to resolve deep phylogenies.

There is nevertheless utility in the application of phylogenetic analyses for lineages within species or sets of closely related species where the focus may be on identifying lineage diversity or on contributing to decisions on species delimitation. This tutorial has been written on the assumption that the focus is on shallow levels of divergence to complement what can be learned from a population genetics perspective.

So what are the strengths of SNPs?

The obvious value in SNPs is that they are genome-wide and can be considered randomly distributed across the genome. In the case of DArT SNPs, they are sparse and, provided attention is paid to multiple SNPs occurring on the one sequence tag, are unlikely to be linked. There are exceptions to this where there are substantial structural variants (e.g large polymorphic inversions), but these can be identified and accommodated in analyses.

SNPs are less expensive and time-consuming to produce than other approaches to generating phylogenetic data. This means that many more individuals can be included in scoring the data allowing more comprehensive evaluation of within taxon variation and providing a very productive connection between lineage analyses and population genetics. Multiple individuals can be scored from each site across a comprehensive range of sites that capture variation within species or lineages.

So what are the limitations of SNPs?

There is a possibility of ascertainment biases. Were one to establish the SNP panel in a pilot study (as was customary with microsatellites), then rare variants and their associated SNPs are likely to be lost, favouring the inclusion of SNPs that have alleles at intermediate frequencies. This can be managed by combining the development of the SNP panel with the analysis of the full dataset, as is done by DArT.

A second limitation is that missing data is not random. There are two sources of missing data in a SNP dataset. The first includes missing sequence tags that are present in the genome but absent in the dataset because of low read depth. These are random. The second source of missing data arises from mutations in the restriction enzyme sites used to generate the sequence tags – null alleles. Null alleles arise from heritable variation, and so are differentially retained across individuals. The biases that arise from non-random missing data can be managed by filtering heavily on call rate (for both loci and individuals) and, where appropriate, by imputation. This however can favour of SNPs at sites with low mutation rates with attendant ascertainment bias.

A third limitation is that the pipelines for generating SNPs are very selective, in that the focus is on identifying SNPs that are essentially biallelic (single point mutations are not inherently biallelic, so again risking ascertainment bias against sites with low mutation rates) and of course polymorphic. Invariant sequence tags do not contain a SNP and so are discarded in the process. This will have implications not so much for tree topology, but certainly for estimation of branch lengths, dating of nodes and comparability across studies.

A fourth limitation of SNPs is that the sequence tags are very short. In the case of DArT, sequence tags stripped of adaptors range in length from 20 bp to 69 bp. These typically have only one variable site. This together with the sheer number of SNP loci precludes some analysis options such as Austal III and SNAPP that are appropriate for handling gene-tree/species-tree discordance.

The common practice of concatenating the sequence tags for each individual and running a Maximum Likelihood analysis ignores the challenge of gene-tree/species-tree discordance (independent lineage sorting) by "averaging" as does distance analysis. Both SNAPPER and SVDquartets manage these issues appropriately and computational time is unaffected by the number of SNP loci scored.

The bottom line is that there are many considerations governing a phylogenetic analysis using SNP, there is no clear consensus on the most appropriate and practical methods to use in the literature, and everyone will have their own opinion and leanings. Resolving this is well beyond the scope of this session in the workshop. We simply present the options and how to execute them and leave you to weave through the complexities of how to craft a defensible workflow. The review by Leache' and Oaks (2017, Annual Review of Ecology, Evolution and Systematics 48:69-84) is a good place to start.

This Tutorial

In this tutorial, we explore options for generating phylogenetic trees from SNP data and associated SilicoDArT data. We conveniently sidestep the issues of what is and what is not a species and focus on the evolutionary relationships between lineages (and their descendant clades).

Such lineages need to be diagnosable to provide a basis for considering them to be on independent evolutionary trajectories and not subject to horizontal exchange of alleles. This is a requirement enforced by the bifurcating character of phylogenetic trees. Where there is a history of such horizontal exchange, a network is arguably a better model than a bifurcating tree. Similarly, where the individuals that contribute to the data matrix are part of an interbreeding population, the techniques applied fall more into the domain of population genetics or genealogy.

The tutorial is structured to first introduce the type of data that is generated during a SNP study using the services of Diversity Arrays Technology Pty Ltd (DArT). We then introduce different approaches to phylogenetic analysis first with distance analyses using Phylip, then Maximum likelihood analyses using IQ-Tree, SVDquartets, and then Bayesian analysis using Snapper in BEAST 2. We finish with the application of parsimony approaches to the binary SilicoDArT data.

Session 1: Datasets

SNP genotypes

The characteristics of SNP data are described in the tutorial on *Data Input and Data Structures*.

Briefly, a representation of the genome of an organism is generated using two restriction enzymes to generate short reproducible fragments of DNA (sequence tags) which are size selected as part of the process of sequencing on an Illumina short read platform. The resultant sequence tags generated by DArT range in length from 20 to 75 base pairs (69 once adapters are removed).

These sequence tags are screened for multiple copy variants. Homologous single copy sequence tags are examined for single nucleotide polymorphisms (SNPs). The polymorphic loci are scored as 0 for homozygous reference (the most frequent allele), 2 for homozygous alternate, or 1 for heterozygous for each individual at each locus when stored as an {adegenet} genlight object in dartR. Loci that cannot be called as scored as missing.

SNP data are of the form shown here, that is, The data can be represented in a table of SNP bases (A, T, C or G), with two states for each individual at each locus in a diploid organisms.

	Ind 01	Ind 02	Ind 03	Ind 04	Ind 05	Ind 06	Ind 07	Ind 08	Ind 09	Ind 10
Locus 1	A/A	A/A	A/A	A/A	A/G	A/A	A/A	A/A	A/A	-/-
Locus 2	C/C	C/C	C/C	C/C	C/C	C/C	C/T	C/C	C/C	C/C
Locus 3	C/G	G/G	G/G	G/G	G/G	C/C	C/C	C/C	C/C	C/C
Locus 4	A/A	A/T	A/A	A/T	T/T	A/A	A/A	A/A	A/A	A/A
Locus 5	A/A	A/A	A/A	A/A	-/-	A/G	A/A	A/A	A/A	A/A
Locus 6	C/C	C/C	C/C	C/C	C/C	C/C	C/T	C/C	C/C	C/C
Locus 7	C/G	G/G	G/G	G/G	G/G	C/C	C/C	C/C	C/C	C/C
Locus 8	A/A	A/T	A/A	A/T	T/T	A/A	A/A	A/A	A/A	A/A
Locus 9	A/A	A/A	A/A	A/A	A/A	A/A	A/A	A/A	A/A	A/A
Locus 10	C/C	C/C	C/C	C/C	C/C	C/C	C/T	C/C	C/C	C/C
Locus 11	C/G	G/G	G/G	G/G	G/G	C/C	C/C	C/C	C/C	C/C

Alternatively, because the data are biallelic, it is computationally convenient to code the data as 0 for homozyogotes for one allele, 1 for heterozygotes, and 2 for homozygotes of the other allele.

The reference allele is arbitrarily taken to be the most common allele, so 0 is the score for homozygous reference, and 2 is the score for homozygous alternate or SNP state. NA indicates that the SNP could not be scored.

	Ind01	Ind02	Ind03	Ind04	Ind05	Ind06	Ind07	Ind08	Ind09	Ind10
Locus 1	0	0	0	0	1	0	0	0	0	NA
Locus 2	0	0	0	0	0	0	1	0	0	0
Locus 3	1	2	2	2	2	0	0	0	0	0
Locus 4	0	1	0	1	2	0	0	0	0	0
Locus 5	0	0	0	0	NA	1	0	0	0	0
Locus 6	0	0	0	0	0	0	1	0	0	0
Locus 7	1	2	2	2	2	0	0	0	0	0
Locus 8	0	1	0	1	2	0	0	0	0	0
Locus 9	0	0	0	0	0	0	0	0	0	0
Locus 10	0	0	0	0	0	0	1	0	0	0
Locus 11	1	2	2	2	2	0	0	0	0	0

This is the form the data are stored in in dartR, though note that it departs from the coding arrangement used by DArT PL.

SNP Sequence Tags

Sequence tags can contain multiple SNPs and, when trimmed of adaptors (yielding 20- 69 bp), these sequence tags serve as data in their own right. Phylogenetic analysis can be applied directly to the SNP scores (individuals as entities, loci as attributes, SNP call as state) or to the sequence tags after they have been trimmed of adaptor sequence.

SilicoDArT data

SilicoDArT (sequence tag presence-absence) data involves scoring loci as "called" [1] or "not called" [0]. They are called because if the two restriction enzymes (used in the accompanying DArTSeq or ddRAD) find their mark, the corresponding sequence tags are amplified and sequenced, and the locus is scored as 1 for that locus.

If, however, there is a mutation at one or both of the restriction enzyme sites, then the restriction enzyme(s) does not find its mark, the corresponding sequence tag in that individual is not amplified or sequenced (null allele), or if it is amplified from a different start site, is no longer considered homologous during pre-processing. The individual is scored as 0 for that locus.

Specific preprocessing options are used to separate true null alleles (sequence tag absences) from sequence tags present in the target genome but missed by the sequencing because of low read depth. First, the sequence tags are filtered more stringently on read depth than for SNPs, typically with a threshold of 8x or more. Second, the distributions of the sequence tag counts are examined for bimodality (clustering on either 0 or 1) and those loci showing a continuum of counts rather than two clusters are discarded. Hemizygous loci are scored as missing (-). Finally, the scores are tested for reproducibility using technical replicates, providing the analyst with the opportunity to filter loci with poor reproducibility in the SilicoDArT scores. In this way, a reliable set of dominant markers are obtained, to complement the SNP dataset generated from the same individuals.

SilicoDArT Sequence Tags

Note that although the SilicoDArT and SNP genotypes are derived from the same samples and representation, the SilicoDArT markers include sequence tags that do not have a SNP, that is, sequence tags that are invariant across all individuals. Phylogenetic analysis can be applied to the SilicoDArT markers directly using parsimony which focusses on shared character states across individuals (apomorphies).

Accessing the Data

The SilicoDArT dataset used in this tutorial is `Example_12.1_SilicoDArT.Rdata` which can be read into RStudio using

```
gl.load("Example_12.1_SilicoDArT.Rdata")
```

or

```
readRDS("Example_12.1_SilicoDArT.Rdata")
```

Worked Example: Exploring the SNP Dataset

Reading in the SNP data

The SNP dataset used in this tutorial is Example_SNP.Rdata which can be read into RStudio using gl.load("Example_SNP.Rdata") or readRDS("Example_SNP.Rdata").

```
gl.set.verbosity(3)
gl <- readRDS("Example_12.1_SNP.Rdata")
```

Note that this dataset has already been filtered on call rate, reproducibility and read depth.

Examining the Contents

```
gl
```

```
/// 110 genotypes,  10,718 binary SNPs, size: 73.6 Mb
14009 (1.19 %) missing data
// Basic content
  @gen: list of 110 SNPbin
  @ploidy: ploidy of each individual  (range: 2-2)
// Optional content
  @ind.names:  110 individual labels
  @loc.names:  10718 locus labels
  @loc.all:  10718 alleles
  @position: integer storing positions of the SNPs
  @pop: population of each individual (group size range: 10-10)
  @other: a list containing: ind.metrics  loc.metrics  history  loc.metrics.flags  verbose
```

Simply typing the name of the genlight object provides a substantial amount of information. We can see that there are 1,027 individuals scored for 13,789 SNP loci, all but 1.37% having been successfully called.

We could have used the adegenet accessors to pull this information, for example,

```
nLoc(gl)
```

```
[1] 10,718
```

```
nInd(gl)
```

```
[1] 110
```

```
nPop(gl)
```

```
[1] 11
```

and can in addition, list the individual names and population names

```
indNames(gl)[1:10]
```

```
[1] "HBS_100900" "HBS_100922" "HBS_100923" "HBS_100745" "UC_0161"    "AA033579"
"AA033609"   "UC_1060"    "HBS_101029" "HBS_101024" .....
```

```
popNames(gl)
```

```
[1] "signata"    "farnorth"   "krefftii"   "nigra"      "gunabarra"  "emmottii"   "macquarii"
"subglobosa" "worrelli"   "tanybaraga" "victoriae"
```

Samples sizes can be obtained using

```
table(pop(gl))
```

```
   signata   farnorth   krefftii      nigra  gunabarra   emmottii  macquarii subglobosa
        10         10         10         10         10         10         10         10
  worrelli tanybaraga  victoriae
        10         10         10
```

Examining Selected Locus Metrics

You can interrogate the locus metrics dataframe associated with the genotype data by accessing various variable in the loc.metrics dataframe associated with the genlight object `gl`.

```
gl@other$loc.metrics$TrimmedSequence[1:10]
```

```
 [1] TGCAGGAGGGAATAAGGCCCACGTTGCAGGGAGACTGACTGGGCATG
 [2] GCAGTGGTTGGGGAGCTTCATGTACATGTTTCCGCCCATTCCTCTTCTTCACAAAAGTCCTGCGGAAA
 [3] TGCAGATGTCTGGGTGCCCAGTAGACTCAGGTCTTGAGATTTTAGCCAACAGTACCACTAGGCGGTGCA
 [4] TGCAGCAAGCAGGAGCTGGGACTGCATG
 [5] TGCAGGATTTGGGATGGGTGACTCAGTAGGGGGCATG
 [6] TGCAGCAGGGAGCCCGCTAAGGGCAAAAGCGGCTCTGCTTTGGGCGTGAGGAGAGAGAGTCGGCTAGGC
 [7] TGCAGCTGCTCAGTATTTCTGATGTCCTAAAAAGCCAGCTAGACTGCCTGATGGACTGCCCTGCTTGAT
 [8] TGCAGTGCCACTGACAGTATGTCTGCACCAGAGCTAATTAAGCTGGCATG
 [9] TGCAGAAATTATGCAATTATGTCTCCTTCCCCCAGCTAGAATGCATG
[10] TGCAGAGGAATGAAATCTTACTTTGCAAATCTTCATCAGATGAGGCTCATGCACTTTTTCCTCAATACT
```

Here we are checking to see if the data contain the sequences trimmed of adaptors which we might want to use for phylogenetic analysis. Note that the sequence tags are of variable length and indeed range from 20 bp to 69 bp. We can examine the distribution of tag lengths using

```
gl.report.taglength(gl)
```

```
Starting gl.report.taglength
  Processing genlight object with SNP data
  Reporting Tag Length
  No. of loci = 13789
  No. of individuals = 1027
    Minimum :  20
    1st quantile :  42
    Median :  64
    Mean : 55.86489
    3r quantile : 69
    Maximum : 69
    Missing Rate Overall : 0.01
Completed: gl.report.taglength
```

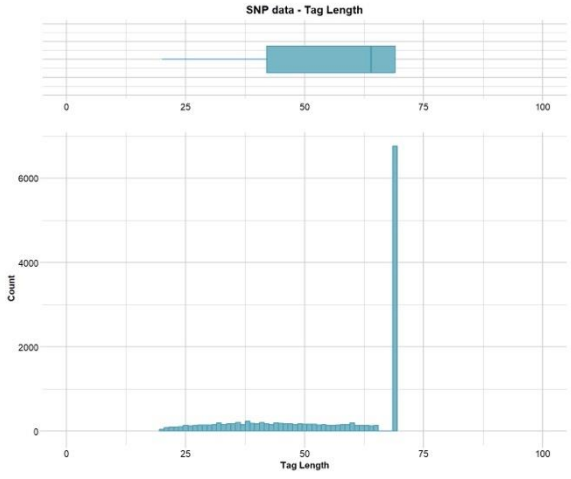


Figure 1. Distribution of sequence tag lengths, which range from 20 to 69 arising from the sequencing protocols of DArT.

This information is important because some phylogenetic techniques are brutal in their treatment of missing data. To minimize the impact of this, it might be necessary to filter loci on tag length prior to such analyses.

Another locus metric that is of interest to us is the SNP position in the sequence tag, incase we want to look at the details of the transition or transversion mutation.

```
gl@other$loc.metrics$SnpPosition[1:10]
```

or

```
gl@position[1:10]
```

```
 [1] 30 13 21 19 49 33 28 27 48 24
```

The mutations in each sequence tag can be examined with

```
gl@other$loc.metrics$SNP[1:10]
```

```
 [1] 30:C>G 13:T>C 21:T>C 19:A>T 49:A>G 33:C>T 28:T>A 27:A>G 48:T>A 24:G>A
```

Note that a sequence tag can have more than one SNP mutation, and that each of these SNP markers is represented in the dataset as a separate line entry. All but one SNP mutation can be removed from the dataset using `gl.filter.secondaries(gl)`. This was done in preparing this example dataset.

Base frequencies and transition/transversion (Ts/Tv) ratios can be obtained with

```
gl.report.bases(gl)
```

```
Processing genlight object with SNP data
Counting the bases
Counting Transitions and Transversions
Average trimmed sequence length: 55.6 ( 20 to 69 )
Total number of trimmed sequences: 10718
Base frequencies (%)
  A: 26.6
  G: 25.13
  T: 26.37
  C: 21.9
Transitions  : 61.74
Transversions: 38.26
tv/ts ratio: 1.6135
```

Where have we come?

The above Session was designed to give you a very brief overview to the characteristics of a SNP dataset as implemented in the dartR package.

Having completed this Session, you should now be familiar the following concepts.

- The means by which SNP genotypes are stored in a genlight object in preparation for phylogenetic analysis.
- The distinction between SNP genotype datasets and SilicoDArT datasets.
- Methods for interrogating the contents of a genlight object, including the associated locus metrics of relevance to phylogenetics.
- How to access and scrutinize the tag sequences, the position of the SNPs in each sequence tag, the mutational change for each SNP and to calculate base frequencies and ts/tv ratios.

Session 2: Distance Phylogenetics

Distance methods work with a matrix of genetic distances calculated for individuals or populations. In the case of SNP markers, such a distance matrix could be calculated directly from the SNP scores.

There are a number of considerations that complicate a phylogenetic analysis based on distances.

- What distance measure to select, taking into account different rates of mutation at different sites (e.g rates of transition mutations as opposed to transversion mutations).
- If and how to manage missing sequence in the aligned set of sequence tags, given that the sequence tags range in length from 20 to 69 (for DArT data).
- What algorithm is appropriate for extracting the best supported tree (Neighbour-joining, Fitch-Margoliash, Minimum Evolution).
- Choosing an outgroup to provide the phylogeny with a defensible root.

Model Selection

There is a vast body of theory directed at selecting an appropriate substitution model of evolution, most electing to optimize the model to the limited data available for analyses (that is, the statistical sample). An alternative is to look to the literature and select the substitution model that has the greatest consensus across a range of studies (arguably optimized for the statistical population from which the statistical sample is drawn).

Options for substitution model selection are provided in the documentation supporting the {ape} package in R. We leave model selection to another day (refer to software options like ModelTest – Posada and Crandall, 2001). Dr Google and Wikipedia (under Computational Phylogenetics) provide some good general advice. Some methods such as IQ-Tree have inbuilt options to estimate and choose the most appropriate model.

In this tutorial, we will use here an evolutionary model that is satisfactory for most purposes, the K80 model devised by Kimura (1980), sometimes referred to as Kimura's 2-parameters distance which adopts different rates transitions (A <-> G, C <-> T) and transversions (A <-> C, A <-> T, C <-> G, G <-> T) and takes into account variable base (A,C,G,T) frequencies.

Missing Sequence (NNNNN)

The method used to convert the trimmed sequences in the dartR genlight object to a form amenable to distance analysis is

```
gl2fasta(gl,outfile="Example1.fas",outpath = getwd())
```

This produces a fastA file of concatenated aligned sequence tags padded where necessary with NNNNNs (Figure 2). Because the sequence tags can vary in length considerably, the substantial addition of Ns can cause problems for distance methods. Some will delete all point information where there is an N for one individual. Filtering on tag length to minimize the impact of this might be considered.

```
gl <- gl.filter.taglength(gl, lower=40)
```

Alternatively, the impact of missing data can be minimized by using pairwise calculations where the distance algorithm allows this.

We will be disregarding this issue for the sake of the exercise.

Tree-building Algorithm

Neighbour-joining

A common method of building a phylogenetic tree from distances uses the Neighbour-joining algorithm (Saitou and Nei, 1987). It is a fast agglomerative clustering method which starts with the pair of taxa with the smallest distance between them, joins them, and then progressively adds nearest neighbours. Thus, the optimization is local not global and the final solution is influenced by the order of similarities between taxa.

Fitch-Margoliash

The Fitch–Margoliash method (F-M) matches the tree with the input distances using weighted least squares, where more similar distances are given greater weight than less similar distances. Thus, the optimization is global not local. Exhaustive searches for the best tree are computationally limited as the number of taxa increases, so heuristic search methods are used to find the best tree.

Both neighbour-joining and Fitch-Margoliash methods are good for examining relationships among your terminal individuals or groups as a prelude to a more sophisticated analysis.

Outgroup Selection

Depending on the application, the phylogeny may need to be rooted. This is conventionally done by choosing an outgroup taxon, one that unambiguously lies outside the clade comprising the taxa of primary interest.

This can be problematic in SNP studies because they work primarily with closely related taxa, the relationships among all of them being of primary interest and no clear taxon that unambiguously lies outside the focal taxa based on a priori considerations. Typically, any clear candidate for an outgroup taxon belongs to a sister species and its inclusion greatly increases the number of null alleles with marginal benefit in terms of numbers of phylogenetically informative SNPs (those with alleles shared between outgroup and some ingroup taxa).

Alternative methods of rooting the tree based on branch length (mid-point rooting) or a priori knowledge of the taxa may be required.

Note that whether the tree is rooted or not rooted does not influence the topology of the tree arising from any of the analyses.

Implementation in dartR

The R package {dartR} is essentially a suite of R scripts, managed on CRAN, that provides a link between SNP data generated by DArT (or using ddRAD) and a range

of analyses. Where capability is already available to the community, dartR will provide wrappers for existing R scripts developed by other sources, or will provide a script to export data in a form that can be used by other programs ourside the R environment.

Distance phylogeny in dartR is implemented by drawing upon the capabilities of R package {ape} and program Phylip. The dartR functions to use are

```
d <- gl.dist.phylo(gl,subst.model="F81")
```

```
gl.tree.fitch(D=d,x=gl,bstrap=100)
```

which together allow you to specify the substitution model, the algorithm for tree building, elect to filter out tag lengths below a given threshold and specify an outgroup taxon.

A neighbour joining tree can be generated with

```
gl.tree.nj(D)
```

but bootstrapping is not available as an option.

Worked Example: Distance Trees

Preparation

To start you need to assign the working directory to the path that contains the datafile `Example_SNP.Rdata`. Also make a note of the directory that holds the Phylip executables, for example, `"D:/workspace/R/phylip-3.695/exe"`.

Read in the data using

```
gl <- gl.load("Example_SNP.Rdata")
```

```
  Starting gl.load
    Processing genlight object with SNP data
    Loaded object of type SNP from Example_SNP.Rdata
  Completed: gl.load
```

Consider if you want to filter on taglength, say

```
gl <- gl.filter.taglength(gl,lower=69, verbose=3)
```

```
  Starting gl.filter.taglength
    Processing genlight object with SNP data
  Initial no. of loci = 10718
    Removing loci with taglength < 69 and > 69
    No. of loci deleted = 5552
    Summary of filtered dataset
      Sequence Tag Length between 69 and 69
      No. of loci: 5166
      No. of individuals: 110
      No. of populations:  11
  Completed: gl.filter.taglength
```

Compute distance matrix

Select a substitution model, in this case F81 (Felsenstein (1981) which is a generalized the Jukes–Cantor model that distinguishes different substitution rates for transitions and transversions and relaxes the assumption of equal base frequencies.

```
dd <-gl.dist.phylo(gl,by.pop=TRUE,
     subst.model="F81",verbose=3)
```

```
Starting gl.dist.phylo
  Processing genlight object with SNP data
  Converting sequence tags to input format for package {ape}
  Calculating distances between individuals
    Substitution model: F81
    Pairwise missing value deletion: TRUE
  Calculating average distances between populations
Completed: gl.dist.phylo
```

which yields the following distance matrix

```
as.matrix(dd)
```

	signata	farnorth	krefftii	nigra	gunabarra	emmottii	macquarii	subglobosa	worrelli	tanybaraga	victoriae
signata	0.0000000000	0.0006919932	0.0004548163	0.0005397176	0.0006250878	0.0007282959	0.0003583931	0.0009670177	0.0009446626	0.00392077	0.0010532297
farnorth	0.0006919932	0.0000000000	0.0005146501	0.0007409745	0.0007403894	0.0007559918	0.0004397814	0.0010380667	0.0010100847	0.00464554	0.0011336961
krefftii	0.0004548163	0.0005146501	0.0000000000	0.0004218141	0.0005222082	0.0006000946	0.0002408085	0.0008466333	0.0008225648	0.00273425	0.0009371353
nigra	0.0005397176	0.0007409745	0.0004218141	0.0000000000	0.0006857328	0.0007798456	0.0004086651	0.0010103591	0.0009906497	0.00443126	0.0011063716
gunabarra	0.0006250878	0.0007403894	0.0005222082	0.0006857328	0.0000000000	0.0007707118	0.0004025354	0.0010166697	0.0009886344	0.00437870	0.0011034903
emmottii	0.0007282959	0.0007559918	0.0006000946	0.0007798456	0.0007707118	0.0000000000	0.0004561135	0.0010720188	0.0010356809	0.00488784	0.0011515267
macquarii	0.0003583931	0.0004397814	0.0002408085	0.0004086651	0.0004025354	0.0004561135	0.0000000000	0.0006936837	0.0006671292	0.00116147	0.0007792588
subglobosa	0.0009670177	0.0010380667	0.0008466333	0.0010103591	0.0010166697	0.0010720188	0.0006936837	0.0000000000	0.0010886388	0.00169187	0.0013695994
worrelli	0.0009446626	0.0010100847	0.0008225648	0.0009906497	0.0009886344	0.0010356809	0.0006671292	0.0010886388	0.0000000000	0.00159929	0.0013509831
tanybaraga	0.0013920773	0.0014645541	0.0012734249	0.0014431256	0.0014378698	0.0014887840	0.0011161471	0.0016901872	0.0015999292	0.00000000	0.0017347977
victoriae	0.0010532297	0.0011336961	0.0009371353	0.0011063716	0.0011034903	0.0011515267	0.0007792588	0.0013695994	0.0013509831	0.00173479	0.0000000000

Neighbour-joining tree (local optimization)

For a quick look, produce a neighbour joining tree

```
gl.tree.nj(x=gl,dist=dd)
```

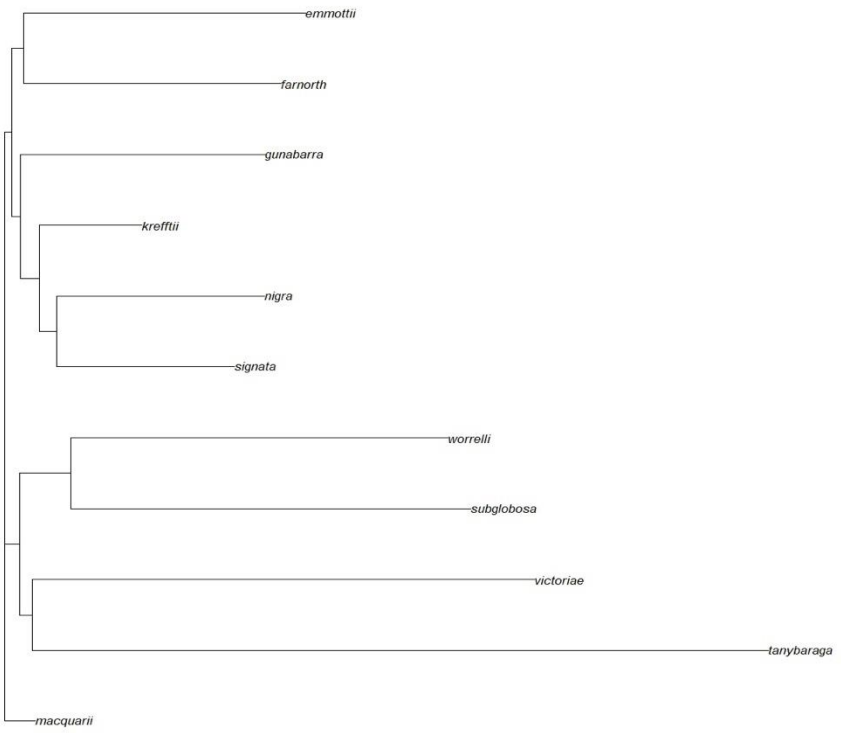


Figure 2. Neighbour-joining tree showing relationships among the lineages defined in the SNP genlight object `gl`. Distances adjusted for substitution rates using model `F81` (Felsenstein, 1981).

Fitch-Margoliash Tree (global optimization)

```
gl.tree.fitch(D=dd,x=gl,
phylip.path="D:/workspace/R/phylip-3.695/exe",
outgroup="macquarii",
verbose=3)
```

```
Starting gl.tree.fitch
  Processing a distance matrix
  Transferred fitch executables to tempdir
  Transferred consense executables to tempdir
  Running fitch from the Phylip executables

Newick format tree file:
((emmottii:0.00041,(farnorth:0.00037,((signata:0.00027,(krefftii:0.00013,
nigra:0.00029):0.00002):0.00002,gunabarra:0.00034):0.00001):0.00001):0.00001,
(victoriae:0.00072,(tanybaraga:0.00104,(worrelli:0.00053,
subglobosa:0.00056):0.00006):0.00002):0.00002,macquarii:0.00004);

Plotting a Fitch-Margoliash Distance Phylogeny
Completed: gl.tree.fitch
```

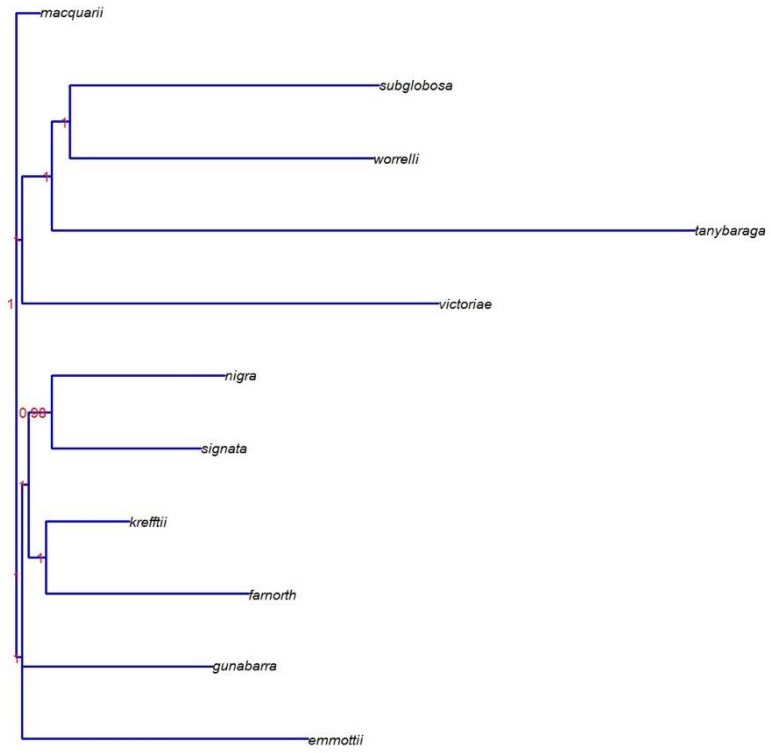


Figure 3. Fitch-Margoliash tree showing relationships among the lineages defined in the SNP genlight object `gl`. Distances adjusted for substitution rates using model `F81` (Felsenstein, 1981). Bootstrap support based on 100 replications. Note that the topology is very similar but not identical to that of the Neighbour-joining tree.

Bootstrapping

We now bootstrap the Fitch-Margoliash tree which is quite computationally expensive.

```
gl.tree.fitch(D=dd,x=gl,
phylip.path="D:/workspace/R/phylip-3.695/exe",
outgroup="macquarii",
bstrap=100,
verbose=3)
```

The results are shown in Figure 3. Bootstrap values are in red.

Where have we come?

The above Session was designed to give you an overview on the application of distance methods in phylogeny using a SNP dataset as implemented in the dartR package.

Having completed this Session, you should now be familiar the following concepts.

- The considerations to make in constructing a distance matrix from a SNP dataset, including decisions on the nucleotide substitution model and the mode of deletion of missing values.
- Familiarity with two options for phylogenetic tree construction, the Neighbour-joining method and the Fitch-Margoliash method.
- Options for the analysis and mode of display.

Session 3: Maximum Likelihood

A popular way to construct phylogenies from SNP data is to retain multiple SNP in sequence tags, concatenate the sequence tags with standard ambiguous codon codes for the variable sites, and apply maximum likelihood methods to generate a phylogeny.

This analysis can be conducted using

```
gl2fasta(gl,method=1,outpath=getwd(),
outfile="ml.phy")
```

```
  Starting gl2fasta
    Processing genlight object with SNP data
    Assigning ambiguity codes to heterozygote SNPs, concatenating trimmed sequence
    Removing loci for which SNP position is outside the length of the trimmed sequences
  Generating haplotypes ... This may take some time
  Completed: gl2fasta
```

which will create a fastA file in a format that can be passed directly to IQ-TREE version 1.6

```
./iqtree2.exe -s ml.phy -m MFP+asc -B 1000 -bnni
```

assuming you have installed IQ-Tree and placed `ml.phy` in the executable directory. The resultant tree is shown in Figure 4.

```
tree <- ape::read.tree("ml.phy.treefile")
plot(tree,cex=0.35)
```

This file can be manipulated for better presentation in packages like Mega 11 which is an excellent tool for preparing trees for publication.

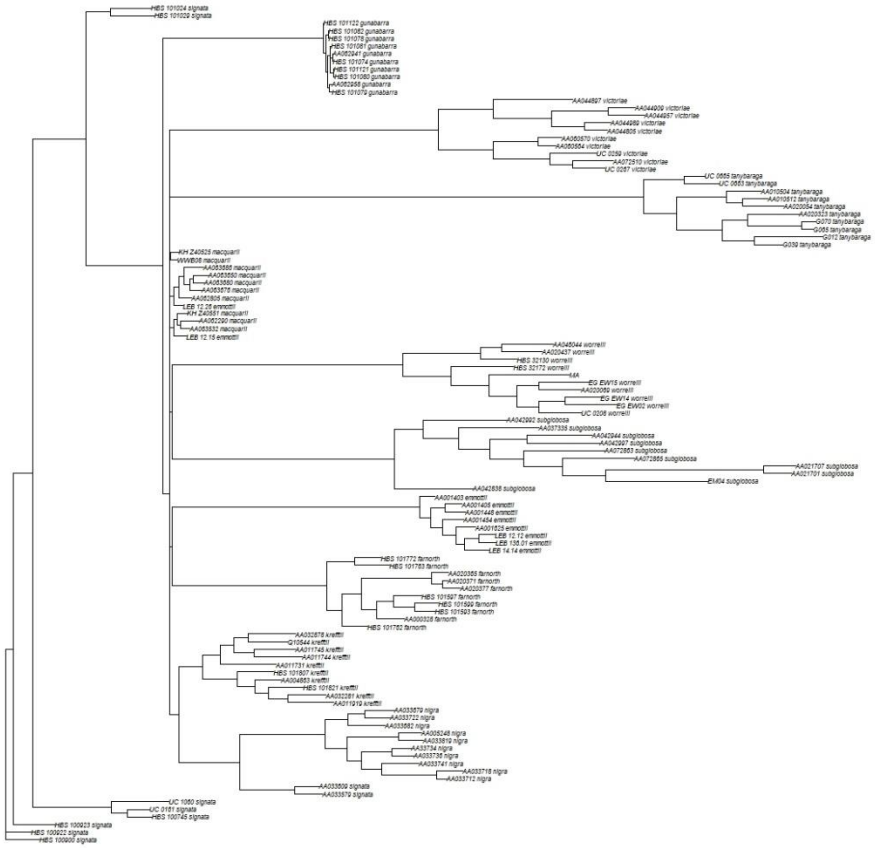


Figure 4. Maximum Likelihood tree showing relationships among the individuals in the SNP genlight object `gl`. Substitution model has been estimated from the data and corrected for ascertainment bias.

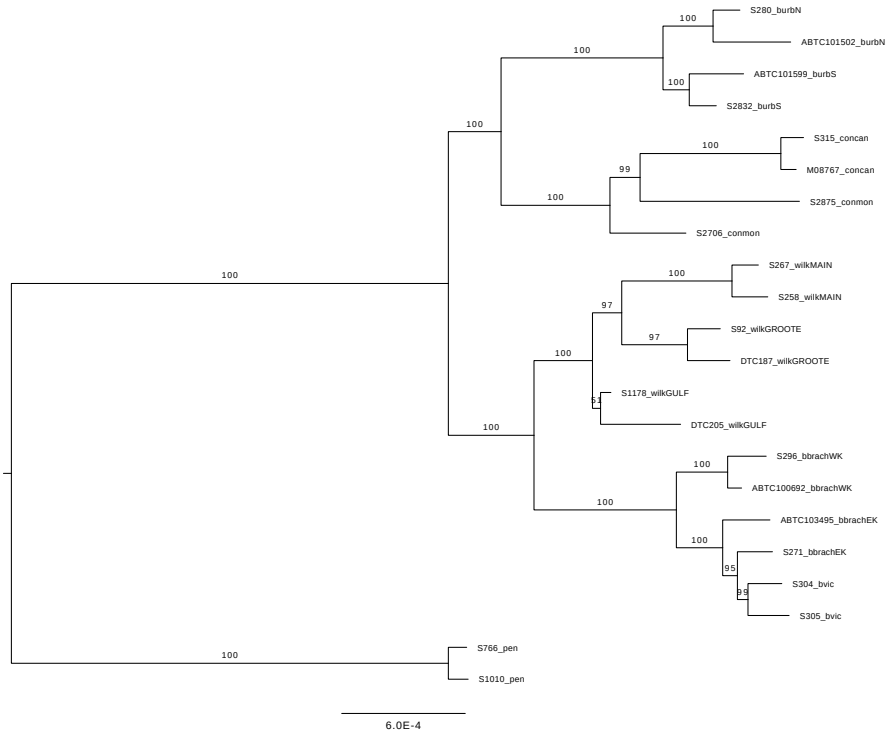
IQ-TREE can implement a range of substitution models. The `-m` switch is the option to specify the model for the analysis. If you do not know which model is appropriate for your data, you can use ModelFinder to determine the best-fit model. `-m MFP` stands for ModelFinder Plus, which tells IQ-TREE to perform ModelFinder and the remaining analysis using the selected model. ModelFinder computes the loglikelihoods of an initial parsimony tree for many different models and the Akaike information criterion (AIC), corrected Akaike information criterion (AICc), and the Bayesian information criterion (BIC). Then ModelFinder chooses the model that minimizes the BIC score (you can also change to AIC or AICc by adding the option -AIC or -AICc, respectively). With `-m MFP` you will get an additional output file `ml.phy.model` which outputs the log-likelihoods for all models.

Note: for SNP data that generally doesn't contain invariant sites, you can explicitly tell the model to include ascertainment bias correction `(+asc)`.

Another example for a quicker analysis is:

```
./iqtree2.exe -s brachyotis_tutorial_IQtree.phy -m MFP+asc -B
     1000 -bnni
```

This is an analysis of some rock-wallaby nucleotide sequence data from *ca* 1000 nuclear exons. There are multiple lineages across five species (e.g., burbidgei = burb and there is a North -N and south -S lineage). The `.contree` file was imported into FigTree and penicillata used as the outgroup to generate the tree here. Bootstrap support values are located on the branches of the tree. See below for details on bootstrap analysis.



Note: sometimes you only want the best-fit model and not a tree reconstruction, so you can achieve this by

```
./iqtree2.exe -s brachyotis_tutorial_IQtree.phy -m MF
```

Alternatively, you can use the best-fit model from the initial `-m MFP` run in downstream analyses instead of running MFP every run.

Running bootstrap analysis

IQtree uses an ultrafast bootstrap approximation (UFBoot) (Minh et al., 2013; Hoang et al., 2018) to overcome computational challenges from running nonparametric bootstrap, enabling computationally achievable analyses. The `-B` flag specifies the number of bootstrap replicates (`-B 1000` : 1000 is minimum number recommended). The output tree will show branch support values in percentage.

Note: you can still estimate bootstrap support using the standard nonparametric bootstrap option `-b` flag. In this scenario, 100 replicates are recommended as a minimum (`-b 100`).

Reducing model violation impact with UFBoot

There is a chance that you can overestimate branch supports with UFBoot due to severe model violations. To reduce this risk, you can run `-bnni` flag which optimises

each bootstrap tree using a hill-climbing nearest neighbour interchange (NNI) search.

Multi-core CPU analysis

You can utilise multiple CPU cores to speed up IQtree analysis. Using the `-T` flag you can set the number of CPUs, or you can use `-T AUTO` to determine the best number of cores. Setting a higher number of CPUs is only efficient for long alignments. You can set an upper limit to the number of CPUs being used by the `-ntmax` flag (e.g., `-ntmax 8` for 8 CPUs).

Concordance Factors

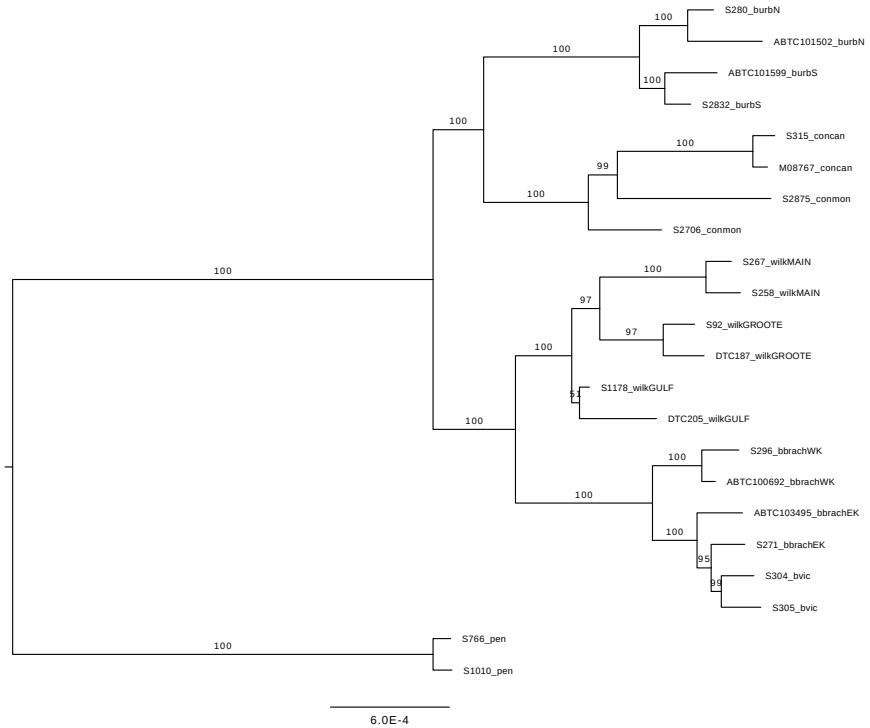
IQtree enables assessment of genealogical concordance via the gene concordance factor (gCF) or the site concordance factor (sCF). For SNP datasets you can use the sCF – the percentage of decisive alignment sites supporting a branch in the ML tree. sCF gives an overview of the underlying disagreement between sites and can be informative when thinking about incomplete lineage sorting.

To estimate sCF you first have to generate a species tree (ML tree). Given the treefile and the original alignment file you can calculate sCF for each branch of the tree.

```
./iqtree2.exe -t brachyotis_tutorial_IQtree.phy.treefile -s
    brachyotis_tutorial_IQtree.phy --scfl 100 -T AUTO -pre
    scf
```

The `--scfl` flag specifies the number of quartets for computing the sCF. It is recommended to have a minimum of 100 quartets for stable sCF values. If you want reproducible sCF values you need to add the `-seed` flag and a value (e.g., `-seed 12345`), otherwise there may be some variance in the sCF values due to randomness. The flag `-pre scf` adds the prefix scf to your output files.

The output `.scf.treefile` was loaded into FigTree and again, the penicillata individuals set as the outgroup. Below is an example of the output tree showing the site concordance factors on the branches of the tree.



Where have we come?

The above Session was designed to give you an overview of the theory and analyses used to estimate maximum likelihood trees in IQTree2.

Having completed this Session, you should now be familiar the following concepts.

- Familiarity with maximum likelihood analyses.
- Estimating ML trees from SNP data and accounting for uncertainty in the models.
- Estimating bootstrap support for a phylogeny.
- Assigning different nucleotide models to analyses.
- Evaluating the site concordance factors for support of clades of the tree.

Session 3: SVDquartets

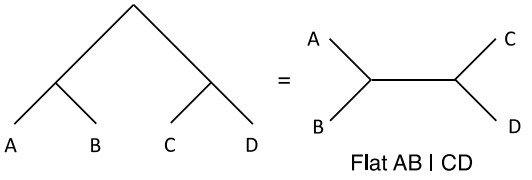
SVDquartets is a software package used to infer a phylogenetic species/lineage tree from genetic sequence data. It introduces a new theoretical framework which uses the entire dataset but makes analysis computationally feasible for large-scale data. To achieve this outcome, it uses a coalescent based quartet assembly method. It assumes each site in the data is independent and it infers the relationships among quartets of taxa. It estimates these quartet relationships for all combinations in the data.

To achieve these goals, the program evaluates site pattern probabilities.

Taxon Sequence	
(A) Red Kangaroo	*TAGG T ACCCGATGCCGTA*
(B) Swamp Wallaby	*TAGG A ACCCGTTGCCGTA*
(C) Eastern Grey Kangaroo	*TACG T TCCGGAAGCGGAT*

In this example, the probability of the pattern TAT

To simplify the data, it then collapses the matrix when patterns are identical using a singular value decomposition (SVD). Below is an example of calculating probabilities of site patterns. In this case no two columns of site patterns are identical If they were, these columns (and site patterns) would be collapsed to reduce the matrix rank by one.



Taxon	*Sequence*
(A)	*TAGGTACCCGTTGCCGTA*
(B)	*TAGGTACCCGTTGCCGCA*
(C)	*TACGTACCGGTAGCGGCT*
(D)	*TTCCATCCGCTACGGTCA*

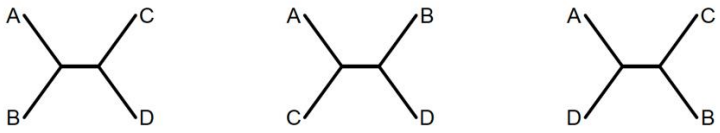
Flat AB | CD*(P)* =

	[TT]	[TA]	[AA]	[CC]
[TT]	P_{TTTT}	P_{TTTA}	P_{TTAA}	P_{TTCC}
[TA]	P_{TATT}	P_{TATA}	P_{TAAA}	P_{TACC}
[AA]	P_{AATT}	P_{AATA}	P_{AAAA}	P_{AACC}
[CC]	P_{CCTT}	P_{CCTA}	P_{CCAA}	P_{CCCC}

Flat AB | CD*(P)* =

	[TT]	[TA]	[AA]	[CC]
[TT]	2	1	1	P_{TTCC}
[TA]	P_{TATT}	P_{TATA}	P_{TAAA}	P_{TACC}
[AA]	P_{AATT}	1	P_{AAAA}	P_{AACC}
[CC]	P_{CCTT}	P_{CCTA}	P_{CCAA}	2

Main idea: use the observed site pattern distribution to provide information about which of the three possible splits for a set of four taxa is the true split.



Compute a score for each split in a given quartet of taxa and choose the split with the best (lowest) score.

Below are examples of how to generate a species/lineage tree with bootstrap support and how to estimate divergence times for a species tree. Prior to this analysis you should:

- Calculate the number of populations/lineages (OTUs).
- Execute `gl2svdquartets()` to generate the nexus input file for PAUP.
- Install PAUP.
- Run paup executable file.

We will start the tutorial from already having the nexus file ready.

Dataset: 11 taxa, 22 individuals, 1550 genes, 749814 base pairs.

Lineage	Number individuals
Petrogale brachyotis brachyotis East Kimberley	2
Petrogale brachyotis brachyotis West Kimberley	2
Petrogale brachyotis victoriae	2
Petrogale burbidgei North	2
Petrogale burbidgei South	2
Petrogale concinna canescens	2
Petrogale concinna monastria	2
Petrogale wilkinsi MAIN	2
Petrogale wilkinsi GROOTE	2
Petrogale wilkinsi GULF	2
Petrogale penicillata (outgroup)	2

Type svdq ?; to see all options for svdquartets analysis (see below).

```
paup> svdq ?;

Usage: svdQuartets [options...];

Available options:

Keyword ------- Option type ---------------------------------------- Current setting ---------
evalQuartets    all|random                                           random
nquartets       <real-value>                                         100000
preferAllQ      no|yes                                               yes
taxpartition    <taxpartition-name>|none                             none
treeInf         QFM|curTrees|none                                    QFM
seed            <int> or <int:int:int:int>                           0
bootstrap       no|standard|multilocus                               no
loci            <charpartition-name>                                 (none)
nreps           <integer-value>                                      100
nthreads        auto|<number-of-threads>                             2
mrpFile         <MRP-outfile-name>                                   none
qfile           <quartet-file-name>                                  none
qformat         qmc|qfm                                              qmc
replace         no|yes                                              *no
showScores      no|yes                                               no
showSV          no|yes                                               no
treefile        <filename-for-treefile>                              none
treemodel       mscoalescent|shared                                  mscoalescent
forceRank       <integer-value>                                      0
keepQuartets    no|yes                                               no
ambigs          missing|distribute                                   distribute
plus2           no|yes                                               no

The following options are for experimental research and are not generally safe/useful:

Keyword ------- Option type ---------------------------------------- Current setting ---------
includeInAll    no|<taxon-id>|<partition-subset-name>               *none
qweights        none|reciprocal|exponential                          none
wtlambda        <real-value>                                         1000
lakeInvar       no|yes                                               no
frobDFile       <filename-for-Frobenius-dists>                       none
                                                                    *Option is nonpersistent
```

The first step is to get the data into PAUP*. This file contains data for 2 individuals from each taxon, so we use a taxon-partition to assign each individual tip to a species/OTU. This definition is already at the bottom of the nexus file and you do not need to type it here. It looks like this:

```
taxpartitions OTUs=
bbrachEK: 1-2,
     bbrachWK: 3-4,
     bvic: 5-6,
     burbN: 7-8,
     burbS: 9-10,
     concan: 11-12,
     conmon: 13-14,
     wilk: 15-16,
     wilkGROOTE: 17-18,
     wilkGULF: 19-20,
     pen: 21-22;
```

You can then specify the outgroup. From your unix prompt, launch PAUP*. The command will be something like:

```
./paup4a168_osx
```

Execute the file *brachyotis_tutorial_SVD.nex* by typing the following at the PAUP* prompt:

```
exe brachyotis_tutorial_SVD.nex
```

Note that you may need to include the full pathname to the file. Set up a logfile for the analysis:

```
log file=brachyotis_SVD.log
```

Move the pen individuals to the outgroup by typing:

```
outgroup 21-22
```

The outgroup samples are number 11 and 12 in the taxon partition.

Part 1: Estimating a phylogenetic tree

We are now ready for our first run of SVDQuartets. For the first run, you can obtain a phylogenetic tree of all individuals to assess if they form monophyletic lineages. You can find information about all of the options associated with SVDQuartets in PAUP* by typing `help svdq;` at the command line.

Run SVDquartets using the command:

```
svdq taxpartition = none showScores=yes seed=1234
     treeFile=brach_SVD.tre;
```

Then root the phylogeny with the outgroup:

```
rootTrees rootMethod=outgroup
```

Save the tree

```
savetrees file=brach_SVD_rooted.tre
```

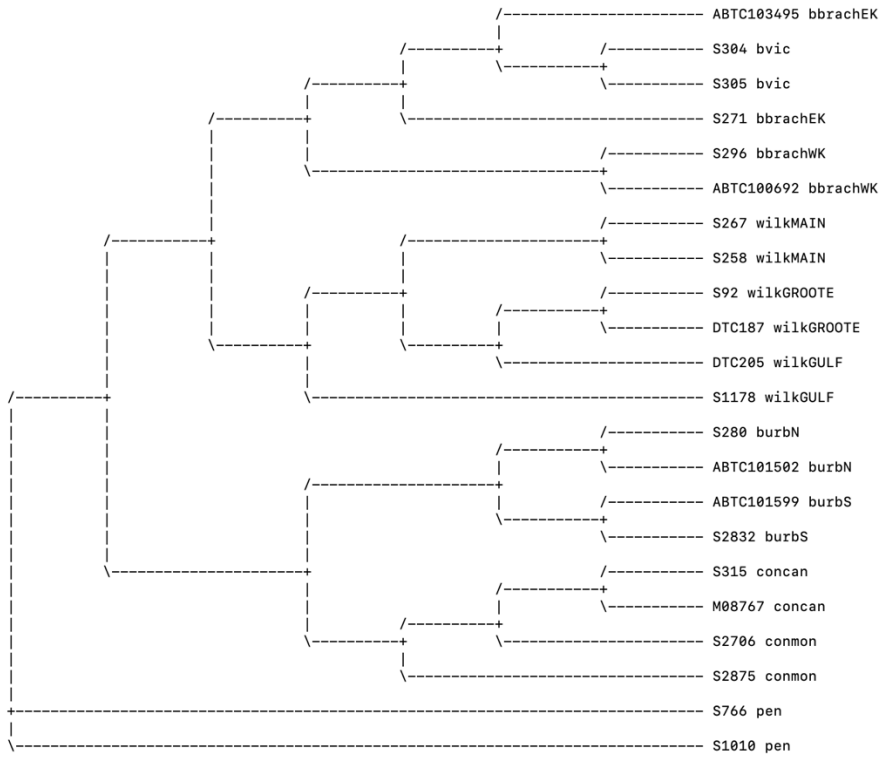
You should get the following output in your terminal. A list of SVD scores for each quartet and a phylogenetic tree of individuals. You should see in the output the scores for all three possible quartets for all of the possible sets of four taxa for these data.

```
SVD scores for each quartet

  quartet    ----split----        score
-----------------------------------------
        1     1  2 |  3  4   1.8281e-06
              1  3 |  2  4   2.3952e-05
              1  4 |  2  3   2.3869e-05
        2     1  2 |  3  5   9.3960e-06
              1  3 |  2  5   4.0127e-06
              1  5 |  2  3   8.6807e-06
        3     1  2 |  3  6   9.7673e-06
              1  3 |  2  6   4.4987e-06
              1  6 |  2  3   1.1045e-05
        4     1  2 |  3  7   1.1316e-05
              1  3 |  2  7   1.7177e-05
              1  7 |  2  3   1.5422e-05
        5     1  2 |  3  8   9.6282e-06
              1  3 |  2  8   1.7488e-05
              1  8 |  2  3   1.5547e-05
        6     1  2 |  3  9   1.1211e-05
              1  3 |  2  9   1.9668e-05
              1  9 |  2  3   1.7083e-05
        7     1  2 |  3 10   9.2679e-06
              1  3 |  2 10   1.5580e-05
              1 10 |  2  3   1.3362e-05
        8     1  2 |  3 11   1.0145e-05
              1  3 |  2 11   1.9123e-05
              1 11 |  2  3   1.6987e-05
        9     1  2 |  3 12   9.7229e-06
              1  3 |  2 12   1.8964e-05
              1 12 |  2  3   1.7108e-05
       10     1  2 |  3 13   1.8840e-05
              1  3 |  2 13   1.8076e-05
              1 13 |  2  3   1.1219e-05

Quartet assembly completed:
  Total weight of incompatible quartets = 693 (9.474%)
  Total weight of compatible quartets   = 6622 (90.526%)
  Time used for QFM = 0.01 sec (CPU time = 0.00 sec)

Tree from SVDQuartets analysis (also stored to tree buffer):
```



Part 2: Estimating a species tree topology

We can now estimate a species tree from the same dataset. To do this, you need to assign individuals to taxa. We will select the option to output quartet scores (you can ignore this step if you don't want to see quartet scores):

Run SVDquartets using the command:

```
svdq taxpartition = OTUs showScores=yes seed=1234
     treeFile=brach_SVD_OTUs.tre;
```

Then root the phylogeny with the outgroup:

```
rootTrees rootMethod=outgroup
```

Save the tree

```
savetrees file=brach_SVD_OTUs_rooted.tre
```

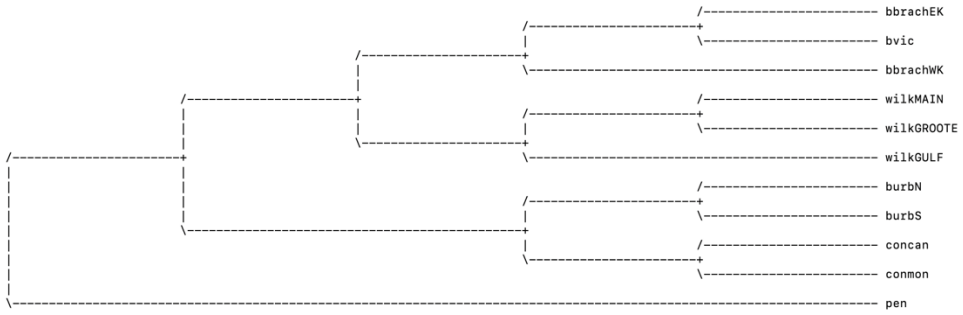
This analysis evaluates all possible quartets, and outputs the species tree inferred by SVDQuartets. In the output, you get the species tree and just above it there will be information regarding the analysis (e.g., the proportion of inferred quartets that are compatible with the species tree). This analysis should have run quickly.

You should see in the output the scores for all three possible quartets for all of the possible sets of four taxa for these data when one taxon is selected from each species. Note: the smaller the score, the more supported that quartet is, compared to the other two possibilities. If you look at this output you will notice how some sets of four taxa provide stronger evidence for a particular quartet relationship than others.

```
Assembling quartets...
[##################################################] 100.0%

Quartet assembly completed:
  Total weight of incompatible quartets = 37.3125 (11.307%)
  Total weight of compatible quartets   = 292.6875 (88.693%)
  Time used for QFM = 0.00 sec (CPU time = 0.00 sec)

Tree from SVDQuartets analysis (also stored to tree buffer):
```




```
Time used for SVDQuartet analysis = 0.65 sec (CPU time = 0.65 sec)
```

Next, we'll run a bootstrap analysis. The command for running the analysis with 100 bootstrap replicates and exhaustive quartet sampling is given below. You can change the random number seed to whatever you like. The analysis should run quickly, but if you are running on your own computer and it is slow, you may wish to reduce the number of bootstrap replicates and/or the number of quartets evaluated on each replicate for the sake of this exercise. Note, it is recommended you run at least 1000 bootstrap replicates in your own analyses – we are minimising run time here.

```
svdq taxpartition=OTUs showScores=yes seed=1234568 bootstrap
     nreps=100 treeFile=brach_SVD_OTUs_bs.tre;
```

You should see a species phylogeny with values of bootstrap support on each brach of the phylogeny. You will get a frequency table at the end as well.

```
Bootstrap 50% majority-rule consensus tree
   (plus other groups compatible with this tree)

                                                                                               /----------------------- bbrachEK(1)
                                                                       /----------100----------+
                                                                       |                       \----------------------- bvic(3)
                                                /---------100----------+
                                                |                      \----------------------------------------------- bbrachWK(2)
                                                |
                        /----------100----------+                                              /----------------------- wilkMAIN(8)
                        |                       |                      /----------97-----------+
                        |                       |                      |                       \----------------------- wilkGROOTE(9)
                        |                       \---------100----------+
/-----------------------+                                              \----------------------------------------------- wilkGULF(10)
|                       |
|                       |                                                                      /----------------------- burbN(4)
|                       |                                              /----------100----------+
|                       |                                              |                       \----------------------- burbS(5)
|                       \--------------------100---------------------+
|                                                                      |                       /----------------------- concan(6)
|                                                                      \----------100----------+
|                                                                                              \----------------------- conmon(7)
|
\---------------------------------------------------------------------------------------------------------------------- pen(11)

Bipartitions found in one or more trees and frequency of occurrence (bootstrap support values):

         11
12345678901          Freq
-------------------------
.......***.           100
.....**....           100
...**......           100
...****....           100
...****...*           100
...********           100
.*.********           100
.......**..            97
```

You can save your tree to a file for graphing with other programs, e.g., R, FigTree, etc. Note that the SVDQuartets estimates an unrooted tree! You may wish to root the tree before plotting it (here, we'll use the outgroup to root the tree).

To root the tree, use the command:

```
rootTrees rootMethod=outgroup;
```

To save the tree to a file, use the command:

```
savetrees file=brach_SVD_OTUs_bs_rooted.tre;
```

Part 3: Estimating speciation times

PAUP* can provide estimates of the times of speciation events along a given species tree topology. Note that this method is distinct from SVDQuartets -- it can be applied to any species tree of interest. In this portion of the tutorial, we will obtain estimates of the speciation times for another brachyotis dataset (more individuals to help with estimating times).

The first step is to provide PAUP* with the species tree for which you would like to estimate the speciation times. Since we just estimated a tree with SVDQuartets, we can use that species tree. However, so that you will see how to analyse any tree you might be interested in, the command to specify the species tree is given below. Note that the dataset must already have been loaded into PAUP*, as well as the taxon partition (also a locus partition if you are using multilocus data). Here we will continue with the dataset from Parts 1-2 and provide the species tree output from Part 2.

The species tree can be specified with the following command:

```
speciestree taxpartition=OTUs
     (pen,(((burbS,burbN),(concan,conmon)),(((bbrachEK,bvic),b
     brachWK),((wilkGROOTE,wilk),wilkGULF))));
```

A basic command to estimate the speciation times is given below.

```
Set outwidth=140;
      qage patProb=exactJC taxpartition=brachyotis_OTUs
      bootstrap=standard treefile=brach_SVD_OTUs_DD.tre;
```

To view the full set of options available in the qAge command, type: `help qage;` If you'd like to see a short summary of available options, you can type: `qage ?;`

```
Usage: qAge [options...];

Available options:

Keyword ------- Option type ------------------------------------- Current setting ----------
patProb         exactJC|expBL                                       exactJC
modelName       <model-name>|none                                   none
theta           <real-value>|estimate                               estimate
tprior          (<mean> <CV>)                                       (0.003 1)
rprior          (<mean> <CV>)                                       (1 1)
rptype          timesHeight|absolute                                timesHeight
taxpartition    <taxpartition-name>                                 none
tree            <integer-value>                                     1
showTree        no|yes                                              yes
showBrlens      no|yes                                              yes
outUnits        coalescent|substitutions|both                       coalescent
treefile        <filename-for-treefile>                             none
replace         no|yes                                             *no
append          no|yes                                             *no
bootstrap       no|standard|multilocus                              no
loci            <charpartition-name>                                (none)
nreps           <integer-value>                                     100
bseed           <int> or <int:int:int:int>                          0
ambigs          missing|distribute                                  distribute
optJC           LBFGS|Praxis                                        LBFGS
optExpBL        LBFGS|Praxis                                        Praxis

Advanced/esoteric options:

Keyword ------- Option type ------------------------------------- Current setting ----------
tradeoff        speed|memory                                        speed
dbout           <integer-value>                                     0
rmseDetail      no|yes                                             *no
method          joint|moments                                       joint
numDv           central|forward                                     forward
transform       no|yes                                              yes
poolJC          no|yes                                              yes
factrBfgs       <real-value>                                        10000
pgtolBfgs       <real-value>                                        1e-05
mBfgs           <integer-value>                                     5
tolPraxis       <real-value>                                        0.0001
                                                                   *Option is nonpersistent
```

In the previous command to estimate the speciation times the taxon partition is defined. In this example you use a standard bootstrap procedure. Alternatively, you could set a locus partition “loci=lociset” and use this locus partition to carry out a bootstrap procedure that uses locus-level information and is invoked with "bootstrap=multilocus". The treefile command specifies the name of the file to which the tree with branch lengths should be saved. The "patProb=exactJC" tells PAUP* to use the formulas from Chifman and Kubatko (2015) to carry out calculations. In this case we use the JC69 model. This is appropriate when this model is a reasonable fit to the data. Below is an example of how to use PAUP* to estimate speciation times when this is not the case.

In the output you should see a tree with the nodes numbered, and below that a table that gives branch lengths and node ages, both with corresponding 95% confidence intervals (confidence intervals are obtained by bootstrapping). The default is to return the estimates in coalescent units. If substitution units are preferred, this can be requested by adding "outUnits=substitutions" or "outUnits=both" to the above command. Above the tree and the estimated lengths, you will see other information about the estimation procedure. Of particular interest are the estimated theta (together with a 95% confidence interval; recall that qAge assumes a single theta for the entire tree) and the composite MAP score. This score is the negative of the log posterior density and can be compared across trees provided that the same prior distributions have been specified. Smaller values indicate a higher composite likelihood.

```
(Unnormalized) composite MAP score = 21878866.08
Estimated theta = 0.003480286 (CI = 0.001376174 - 0.003530580)

Total number of function evaluations for theta initialization = 500
Total number of function/gradient evaluations for L-BFGS = 4379
Total number of function evaluations = 4879

Time used for startup overhead = 0.81 sec
Time used for qAge = 00:01:00 (CPU time = 00:01:00.9)

Species-tree phylogram:

                                                                                                                                  / bbrachEK
                                                                                                                                 23
                                                                                                                                  + bvic
                                                                                                                                 24
                                                                                                                                  + bbrachWK
                                                                                                                                  |
                                                                                                                                 27 wilkMAIN
                                                                                                                                 25
                                                                                                                                  + wilkGROOTE
                                                                                                                                 26
/--------------------------------------------------------------------------------------------------------------------------------31 wilkGULF
|                                                                                                                                 |
|                                                                                                                                 + burbN
|                                                                                                                                28
|                                                                                                                                 + burbS
32                                                                                                                               30
|                                                                                                                                 + concan
|                                                                                                                                29
|                                                                                                                                 \ conmon
|
\---------------------------------------------------------------------------------------------------------------------------------- pen
```

Estimated node ages and branch lengths in coalescent units (2N generations)

		Branch lengths				Node ages		
Node	Connected to node	length	stderr	95% CI		age	stderr	95% CI
bbrachEK (1)	23	0	1.9639e-08	(0 - 6.3731e-08)		0	-	-
bvic (3)	23	0	1.9639e-08	(0 - 6.3731e-08)		0	-	-
bbrachWK (2)	24	0	0.035953	(0 - 0.094108)		0	-	-
wilkMAIN (8)	25	0	0.212097	(0 - 0.471267)		0	-	-
wilkGROOTE (9)	25	0	0.212097	(0 - 0.471267)		0	-	-
wilkGULF (10)	26	0	0.387146	(0 - 0.839042)		0	-	-
burbN (4)	28	0	0.179368	(0 - 0.408757)		0	-	-
burbS (5)	28	0	0.179368	(0 - 0.408757)		0	-	-
concan (6)	29	0	0.179710	(0 - 0.411461)		0	-	-
conmon (7)	29	0	0.179710	(0 - 0.411461)		0	-	-
pen (11)	32	1e-06	1.231637	(1e-06 - 2.634035)		0	-	-
23	24	0	0.035953	(0 - 0.094108)		0	1.9639e-08	(0 - 6.3731e-08)
24	27	0	0.401367	(0 - 0.864730)		0	0.035953	(0 - 0.094108)
25	26	0	0.175713	(0 - 0.389844)		0	0.212097	(0 - 0.471267)
26	27	0	0.049470	(0 - 0.123210)		0	0.387146	(0 - 0.839042)
27	31	0	0.144621	(0 - 0.329129)		0	0.435532	(0 - 0.928834)
28	30	0	0.299331	(0 - 0.653316)		0	0.179368	(0 - 0.408757)
29	30	0	0.298821	(0 - 0.640806)		0	0.179710	(0 - 0.411461)
30	31	0	0.102282	(0 - 0.239254)		0	0.478066	(0 - 1.030351)
31	32	1e-06	0.652318	(1e-06 - 1.432664)		0	0.579583	(0 - 1.246274)
32	-	-	-	-		1e-06	1.231637	(1e-06 - 2.634035)
Sum		2e-06						

The estimated time in coalescent units can be converted to divergence times in years by the following equation.

```
Divergence time = coalescent units x generation time x 2N
```

where `N = theta / (4 x u)`.

The qAge analysis calculates the average theta for the run and you can substitute in a mutation rate (u) to calculate divergence time using the above equation.

To estimate the speciation times using a more complex substitution models, you change `patProb=exactJC` to `patProb=expBL`. Prior to doing this, however, you must define a model using the `lset` command and provide the name of that model to the qAge command. Below is sample code – to be run on your own as it is more computationally intensive. In the first line, we specify a GTR model, with parameters in the model to be estimated. The second line asks PAUP* to estimate those parameters using our data and the current tree in memory. In the third line, we set the model with the estimated values of the parameters, and we name it "MyGTR". The final line uses qAge to carry out estimation under that model. Note that this analysis is more computationally intensive than the previous one that assumes the JC69 model.

```
lset nst=6 rmatrix=estimate baseFreq=empirical;
     lscores;
     lset nst=6 rmatrix=(1.01 3.49 0.54 1.36 3.20 1.0)
     baseFreq=(0.2533 0.2467 0.2454 0.2547) modelName=MyGTR;
     qage patProb=expBL taxpartition=OTUs bootstrap=standard
     treefile=brachyotis_SVD_DD_GTR.tre modelName=MyGTR
     poolJC=no nreps=100;
```

If you are looking for additional tutorials or information, including comparing between different trees look at **https://phylosolutions.com/tutorials/svdq-qage/svdq-qage-tutorial.html** and **https://molevolworkshop.github.io/faculty/kubatko/pdf/species-trees-tutorial-2023.html** These tutorials include examples of how to run simulations in PAUP* taking into account the number of loci, length of each locus and the scaling factor for the branch lengths.

Note: you can adjust the number of threads the analysis runs on, as well as the evaluation method of SVDQuartets and the number of quartets analysed.

```
svdq nthreads=1 evalq=all
```

Where have we come?

The above Session was designed to give you an overview of the theory and analyses used to estimate species trees in SVDquartets.

Having completed this Session, you should now be familiar the following concepts.

- Familiarity with site pattern probabilities and coalescent based quartet assembly analyses.
- Estimating species/lineage trees from SNP data using the quartet method.
- Estimating bootstrap support for a phylogeny.
- Assigning different nucleotide models to analyses.
- Estimating divergence times given a species tree and sequence data.
- How to convert the divergence estimates to time using theta and mutation rate estimates.
- Options for the analysis and how to seek help.

Session 4: TreeMix

Introduction

TreeMix is a software package designed to infer patterns of population splitting and mixing from genome wide allele frequency data (e.g., population sampling SNP or WGrS datasets). TreeMix estimates a maximum likelihood tree for a set of populations and infers a number of admixture events. It uses a set of allele frequencies from a number of populations/OTUs which are assumed to have been generated from a dataset of unlinked SNPs. It assumes sites are biallelic. If your dataset consists of SNPs in high LD, then you will need to first prune your dataset. Note, TreeMix does not like missing data, so you should also remove sites with missing data.

Below is an example of how to generate a maximum likelihood tree with varying levels of migration to estimate the most likely model of evolution. Prior to this analysis you should:

- Calculate the number of populations/lineages (OTUs).
- Execute `gl2treemix()` to generate the input file for TreeMix.
- Install TreeMix.
- At the end of this tutorial you will generate a bash script to run all the replicates and models for your analyses which you can adapt to your future analyses.

Input file

The input file is a gzipped file. The first row "header" consists of a space-delimited list of names of populations/OTUs. The second and remaining rows contain allele counts at each SNP and assumes the SNPs are in the order they are in the genome. These rows are also space delimited between the allele counts for each population/OTU and comma-delimited between tehe two allele counts within the population. Here is an example:

```
pop1 pop2 pop3 pop4
5,1 1,1 4,0 0,4
3,3 0,2 2,2 0,4
1,5 0,2 2,2 1,3
```

Step 1: Build the ML tree

TreeMix will build a maximum likelihood tree of the populations under the assumption all sites are independent.

```
./treemix -i SAMIG_tutorial_treemix.input.gz -o SAMIG_ML
```

Step 2: Root the ML tree

Before you start assessing migration within the tree you must first set the position of the root. To build the ML tree and set the position of the root, you must identify the name of the outgroup.

```
treemix -i SAMIG_tutorial_treemix.input.gz -root godmani -o
    godmani_root_ML
```

Step 3: Account for linkage disequilibrium by grouping SNPs together

Nearby SNPs may not be independent, so you can group them together in windows of size n SNPs by using the -k flag. You should set the value of n that far exceeds the known extent of linkage disequilibrium in the organism (dependent on SNP density). For example, if you have a low density of SNPs you might set the -k value to 1. If you have SNPs in closer LD then you might use blocks of 1000 SNPs.

```
treemix -i SAMIG_tutorial_treemix.input.gz -root godmani -k 1
     -o godmani_root_k1_ML
```

Step 4: Generate bootstrap replicates for your ML tree

To evaluate the confidence in any given ML tree you can generate bootstrap replicates. This is done over clocks of contiguous SNPs. In the example below you generate 1000 bootstrap replicates by resampling blocks of 1000 SNPs.

```
treemix -i SAMIG_tutorial_treemix.input.gz -root godmani -
     bootstrap 1000 -k 1 -o godmani_root_k1_b1000_ML
```

Step 5: Build the ML tree with migration (-m)

You can start to allow for a number of migration events in the tree using the -m flag. Assign the number of migration events to test using the -m followed by the number of migration events. For example, you can assign two migration events.

```
treemix -i SAMIG_tutorial_treemix.input.gz -root godmani -m 1
     -k 1 -o godmani_root_k1_m1_ML
```

Note: By default TreeMix includes a correction for sample size effects when it calculates the covariance matrix among populations. It can be turned off by the `-noss` flag. Sometimes this can overcorrect (e.g., single individual in a population) – if you are getting many branches with length zero, this might be the problem.

You can incorporate known migration events (`-cor mig` and `-climb`)

You can incorporate known migration events. To do this you will need to prepare a file describing the known events. The input file should have each event on a line, with the first column the source population, the second column the admixed population, and the third column the admixture proportion. For example, you expect *P. godmani* to have 25% of their ancestry from *P. mareeba*.

```
godmani mareeba 0.25
```

You can then build the tree accounting for these events using the `-cor mig` option. It is recommended you perform a round of hill-climbing to optimise the exact migration edges using the `-climb` flag.

```
treemix -i SAMIG_tutorial_treemix.input.gz -core mig known
     events -climb -root godmani -k 1 -o
     godmani_root_k1_core1_ML
```

You can run a small bash script to run replicate analyses across multiple numbers of migration events to estimate the best model for the data. Below is a script estimating the ML tree with 1-20 migration events and for 5 replicates for each migration event. These outputs will have the migration number and replicate number appended to the end of the output file names.

```
for m in {1..5}
     do
     for I in {1..5}
            do
            tremix \
           -I SAMIG_tutorial_treemix.input.gz \
          -o SAMIG.${i}.${m} \
```

```
                -global \
                -m ${m} \
                -k 1
                -root godmain
        done
done
```

Run the bash script by typing `bash SAMIG_treemix.sh` at the command line. Or copy the text in the file and paste into terminal and run it like that. Make sure you're in your working directory as this is where all the output files will be saved. Edit the bash script to change the value of `k`. Change this to `-k 10` and edit the output `-o SAMIG_k10.${i}.${m}`. Save ALL of the outputs from both `-k 1` and `-k 10` runs into a folder. You will need this folder to upload into OptM analyses below.

There is another example file `treemix_lateralis.sh`. Make sure the input files are in the same directory you are in. Again, run at `-k 1` and `-k 10`.

Output files

- **out_ML.cov.gz** The covariance matrix between populations estimated from the data
- **out_ML.covse.gz** The standard errors for each entry in the covariance matrix
- **out_ML.modelcov.gz** The fitted covariance
- **out_ML.treeout.gz** The fitted tree model and migration events
- **out_ML.vertices.gz & out_ML.edges.gz** The internal structure of the inferred tree/graph is in these files

The file `out_ML.treeout.gz` has the inferred tree from the data and the first line is a Newick format ML tree and the remaining lines contain the migration edges. For the migration edges, the first line is the weight of the edge, followed (optionally) by the jackknife estimation of the weight, the jackknife estimates of the standard error, and the *p*-values. The subtree information then follows below the migration edge etc.

You can run OptM to estimate the optimal number of migration edges. See https://doi.org/10.1093/biomethods/bpab017 for details.

- Install R package {Sizer} from CRAN
- Install OptM package
- Load the package into R library(OptM)

To run OptM, you will need a folder of output files produced from TreeMix (like the one produced from running the bash script). The function optM will automatically search the folder for the `SAMIG.llik`, `SAMIG.modelcov.gz`, and `SAMIG.cov.gz` files; where `SAMIG` is that provided to the `-o` parameter of *treemix*.

Files should be in the format `SAMIG.i.M;` where `SAMIG` is any name you prefer,; *i* is the iteration number for that value of `M;` `M` is the number of migration edges used for the treemix run (`-m` parameter).

You need at least two iterations of each value of M (M>2) and you must have sequential integers (e.g., 1,2,3…).

Example code to run OptM in R from **https://cran.r-project.org/web/packages/OptM/readme/README.html**

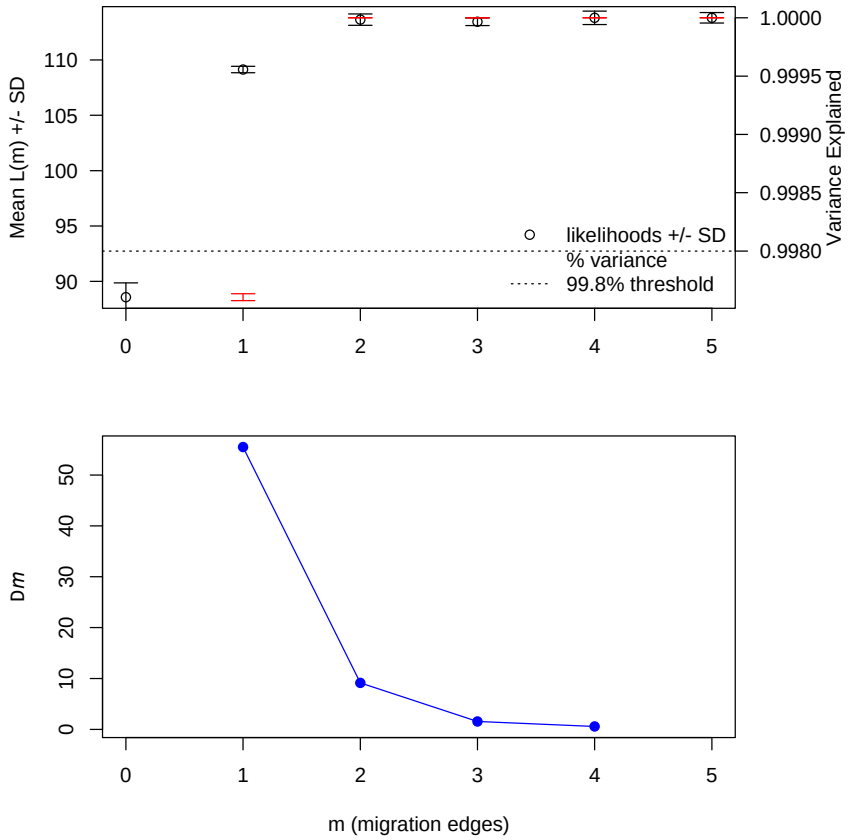
First load the data for the dataset above; 5 iterations for *M*={1-5} then Run *optM* using the default "Evanno"-like method:

```
test.optM = optM(folder="path_to_folder",method="Evanno")
```

The path to the folder of your results – generated earlier. Then, plot the results:

```
plot_optM(test.optM, method = "Evanno")
```

You should see an output like the below:



Alternatively, you can run linear modeling estimates rather than the *ad hoc* statistic:

```
test.linear = optM(folder="path_to_folder", method = "linear")
plot_optM(test.linear, method = "linear")
```

OR using *SiZer*:

```
test.sizer = optM(folder-"path_to_folder", method = "SiZer")
plot_optM(test.sizer, method = "SiZer")
```

Once you have calculated your best model – best log likelihood or change in delta you can visualise your tree!

Visualisation

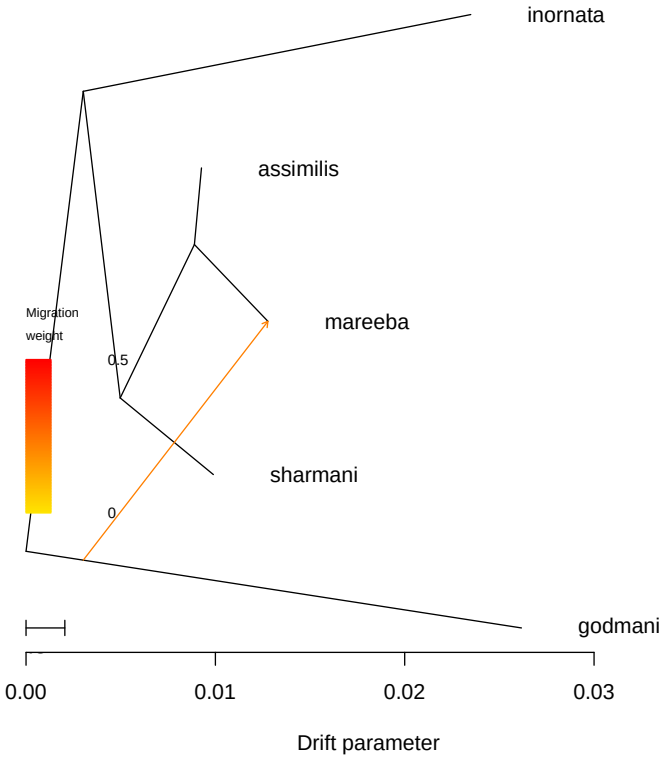
1. Graph visualisation

To visualise the graph, use the R script plotting_funcs.R (in src folder in TreeMix download). Execute this script within R

```
source("src/plotting_funcs.R")
plot_tree("path_to_file_with_correct_m_model")
```

for example `plot_tree("SAMIG_k10.1.1")`

Make sure are in your directory of outputs. This will plot the ML tree you highlight with the correct value of migration based on OptM results. It will plot the tree along with any migration edges with the weight of the migration associated with the colour of the lines. See example below.

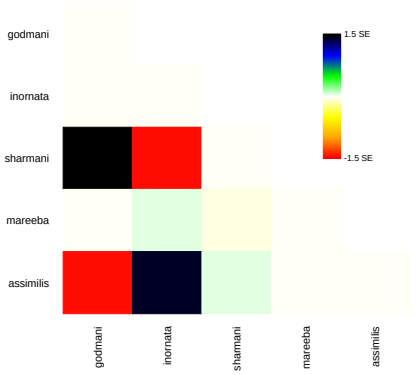


Residual visualisation

You can visualise the residuals from the fit of the model to the data. Evaluating the fit of the model can help identify populations that are not well-modeled (e.g., due to additional migration). This is also run in R

```
plot_residuals("SAMIG_k10.1.1", "poporder")
```

The file `poporder` is simply a list of the names of the populations in the order you would like them to be plotted. This can be a text file.



Where have we come?

The above Session was designed to give you an overview of the TreeMix analyses.

Having completed this Session, you should now be familiar with the following concepts.

- Familiarity with TreeMix analyses.
- Estimating a maximum likelihood tree from SNP data.
- Building a ML tree with migration.
- Estimating bootstrap values on the ML tree.
- Running a bash script to run multiple replicates per model of migration.
- Using R software packages to visually interpret the results and determine the optimal number of migration events.

Session 5: Snapper

Introduction

SNAPP is a software package designed for inferring species trees from genetic sequence data. It operates within a Bayesian statistical framework and is part of the BEAST2 platform, a suite of software for evolutionary analysis. SNAPP takes an approach that allows direct estimation of species trees without first inferring gene trees, thus avoiding the gene-tree/species-tree reconciliation required by other methods. SNAPP is however computationally expensive which, in the context of modern SNP datasets, demands heavy subsampling of individuals or loci.

Snapper is a phylogenetic approach implemented in Beauti/BEAST that can handle larger SNP datasets than can SNAPP. Computation time of Snapper is still impacted by the number of populations and the numbers of loci but is not particularly sensitive to the number of individuals in each terminal taxon (= population).

For this reason, particular attention needs to be directed at amalgamating populations where their joint taxonomic identity is without question and reducing the number of SNP loci by prudent filtering. Removing monomorphic loci, increasing the reliability of loci (read depth, reproducibility), minimizing missing data (call rate), removing multiple SNPs in a single sequence tag (secondaries) are all good options. You may wish also to remove SNP loci that are polymorphic within only a single population (autapomorphies).

If computational time is still an issue, run the analysis on a series of subsamples of loci (say `nloc=300`) to see if a consistent topology is obtained, then adopt that topology as the final result.

Note that there is a downside to manipulating your data to achieve reasonable computation times. Omission of sequence tags that are invariant during the SNP calling process, removal of monomorphic loci generated during taxon selection, and removing autapomorphic loci will all affect branch lengths, perhaps differentially, and so compromise branch lengths and estimates of divergence times. Fortunately, the topology should be little affected.

Finally, the analysis relies on certain assumptions. First is that the structure is one of a bifurcating tree and not a network. One needs to assign individuals to populations in advance of the analysis, confident that they are discrete entities and free of horizontal transfer. A second assumption is that the loci scored for SNPs are assorting independently. This is probably a reasonable assumption for SNPs derived from sparse representational sampling (e.g. DArT), but if dense SNP arrays are being used, then some form of thinning will be required. Of course, multiple SNPs on a single sequence tag will be linked, so filtering all but one SNP per sequence tag is required.

Workflow

The workflow is somewhat complicated, as with all BEAST analyses.

- Pre-work the dataset to minimize the number of populations (OTUs) and the number of loci, in the interests of managing computation time.
- Execute `gl2snapper()` to generate the input file for Beauti.

- Install BEAST2 and its associated components.
- Run beauti.exe and set the template to snapper [File | Template | Snapper].
- Load the nexus file produced by `gl2snapper()` which by default is snapper.nex.
- Select and set the parameters you consider appropriate.
- Save the xml file [File | Save As].
- Load the xml file into BEAST and execute.
- When beast is finished, examine the diagnostics with Tracer.
- Visualize the resultant trees using DensiTree and FigTree.
- If using the command line to run beast, the command is `beast -threads myxmlfile`. Progress can be monitored with awk (`awk '{print $1, $2}' snapper.log |tail`). When beast is finished, transfer the log and tree files to a windows platform and use Tracer, DensiTree and FigTree as above.

`gl2snapper` does not work with SilicoDArT data.

Worked Example: Snapper

This workflow is probably best illustrated by example.

Pre-work the dataset

The example dataset `gl` has already been minimized in terms of populations and loci.

Generate the input file for Beauti

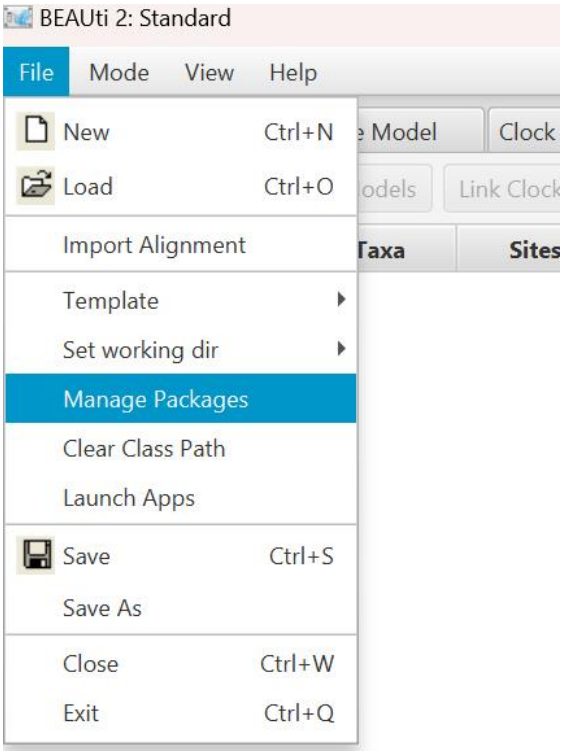
Run with default parameters

```
gl2snapper(gl)
```

to generate an input file for BEAUti. Type `?gl2snapper` for help. You can examine the contents of this input file using your favourite editor. The file is called `snapper.nex`.

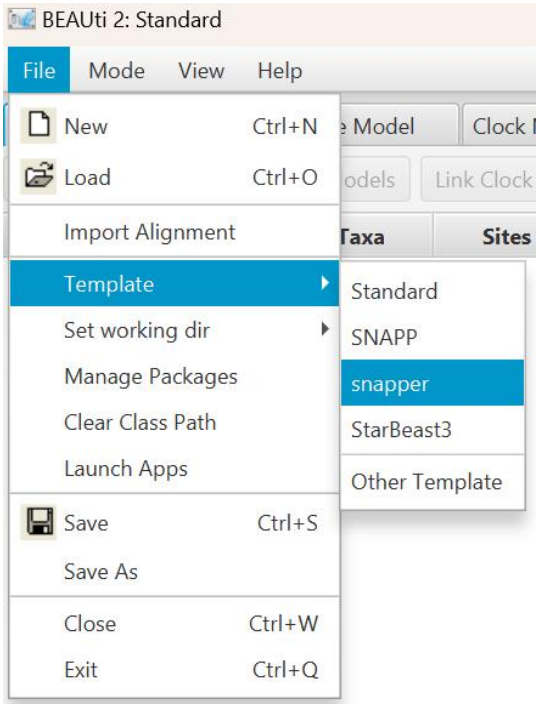
Run Beauti

We need first to install the snapper package via the package manager.



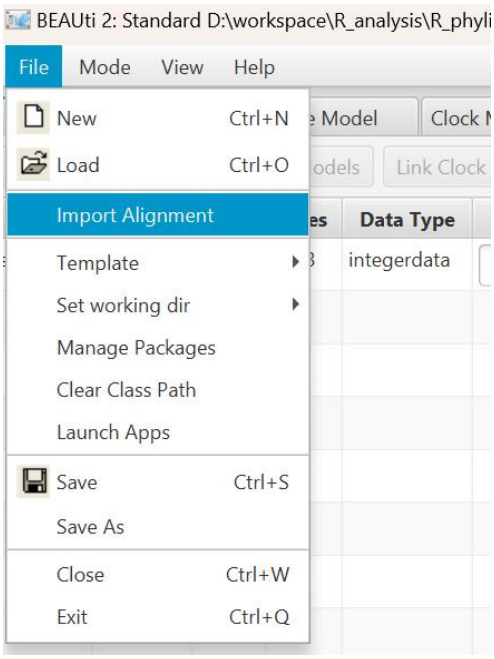
Select and install Snapper.

Now select the Snapper template.



Load the Nexus file

Import the nexus file produced by `gl2snapper()`.



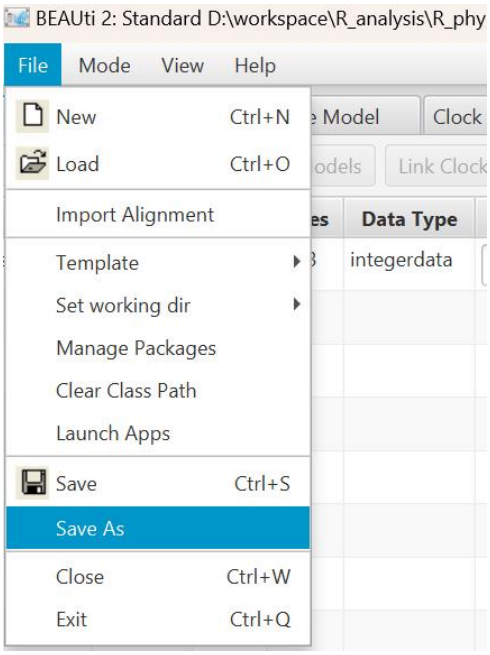
Check the taxon abbreviations, and rectify if necessary.

Set parameters

We will run with default parameters for the sake of this exercise.

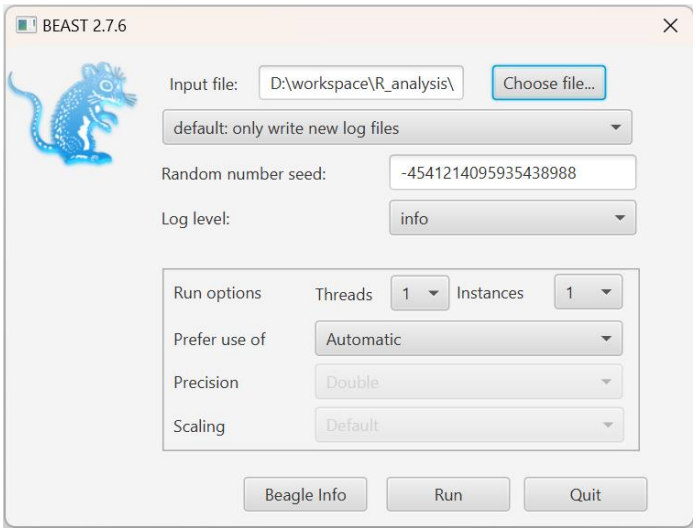
Save the xml file

Save the xml file (which serves as the input file for BEAST) as `snapper.xlm`.



Execute the xml file in BEAST

Specify the input file as snapper.xml and click Run.



Herein lies the rub. Computational time. Advice on how to manage this is given in the documentation associated with `gl2snapper()`. Type `?gl2snapper` for help. On a basic PC using a single thread, the script using the full example dataset took 10 days to run. It is a good thing we decided to subset the loci down to 500 for the sake of the exercise.

Session 5: SilicoDart analysis

IN PREPARATION

Exercises

IN PREPARATION

References

Exercises

	**Exercise 1**

	**Exercise 2**

	**Exercise 3**

Where have we come?

In this Session, we have covered a range of topics on data entry, the storage of data and some preliminary approaches to examining those data. Having completed the Session, you should understand

- What is a sensible pipeline for preliminary handling of your SNP data.
- How a genlight object is organised in terms of the SNP scores (which are different from the scores used by DArT PL) and how locus and sample metadata are associated with the genotypes.
- The different types of locus metadata generated by DArT PL, and how to look up what each metric means.
- How to read data from DArT Pty Ltd into a genlight object.
- How to interrogate the locus and individual (specimen/sample) metadata.

References

Jombart T. and Caitlin Collins, C. (2015). Analysing genome-wide SNP data using adegenet 2.0.0. http://adegenet.r-forge.r-project.org/files/tutorial-genomics.pdf

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

Jombart, T., Kamvar, Z.N., Collins, C., Lustrik, R., Beugin, M.P., Knaus, B.J., Solymos, P., Mikryukov, V., Schliep, K., Maié, T., Morkovsky, L., Ahmed, I., Cori, A., Calboli, F. and Ewing, R.J. (2018). Package 'adegenet'. Version 2.1.1. Exploratory Analysis of Genetic and Genomic Data. https://github.com/thibautjombart/adegenet

Posada, D and Crandall, KA (2001). Selecting the Best-Fit Model of Nucleotide Substitution, *Systematic Biology* 50:580–601. https://doi.org/10.1080/10635150118469.

Kimura, M. (1980) A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16:111–120.

Saitou, N. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees". *Molecular Biology and Evolution. 4:406–425.* doi:10.1093/oxfordjournals.molbev.a040454

Ende