`ClonalFrame` User Guide

Version 1.1

Xavier Didelot and Daniel Falush
Peter Medawar Building for Pathogen Research
Department of Statistics
University of Oxford
Oxford OX1 3SY, UK
`{didelot,falush}@stats.ox.ac.uk`

Software from `http://bacteria.stats.ox.ac.uk/`

Contents

1 Introduction

`ClonalFrame` is a computer package for the inference of bacterial microevolution using multilocus sequence data, available freely from `http://bacteria.stats.ox.ac.uk/`. The statistical methodology underpinning `ClonalFrame` is formally described in the article "Inference of Bacterial Microevolution Using Multilocus Sequence Data" by X. Didelot and D. Falush, Genetics, Vol. 175, 1251-1266, March 2007. This is the appropriate citation for this program.

The `ClonalFrame` method does for recombining bacteria what phylogenetic methods do for non-recombining organisms, while also providing information on recombination. Specifically, it estimates the clonal relationships between the members of a sample, while also estimating the chromosomal position of homologous recombination events that have disrupted the clonal inheritance.

The method should be useful for most recombining bacteria. It could also work for other genetic systems that evolve by copying and gene conversion, such as gene families. For viruses, recombination takes place by crossing over during genome duplication and can in principle result in half of the genome coming from one source and the other half coming from another, so the genome does not have a single parent that accounts for most of its ancestry, i.e. there is no well defined clonal frame. Nevertheless, the method might in practice do a reasonable job of detecting which parts of the tree show strict clonal inheritance and also account usefully for many of the recombination events. Try it and see!

The method is less likely to be useful, and might even be positively misleading, for eukaryotic organisms that undergo meiosis. It is also unlikely to be ideal for establishing relationships at the genus level or above, or for populations of organisms that recombine so frequently that little clonal structure is evident, such as *Helicobacter pylori* in humans. It is also best to avoid regions of the genome which evolve so rapidly that alignment is difficult.

The more DNA sequence fragments that are inputted into `ClonalFrame` and the longer each one is, the more powerful and accurate inference will be. It is not appropriate to run the model using only a a single contiguous stretch of DNA unless it is significantly larger than the largest possible genetic import. Short stretches may be completely disrupted by a single genetic import and are therefore not appropriate on their own for estimating the clonal frame. In principle, the model could be adapted to use different datatypes such as Single Nucleotide Polymorphisms (SNPs) or microsatellites (otherwise known as VNTRS). However, alternative data types are not available in the current version.

While the computational methods implemented here are fairly powerful, some care is needed in running the program in order to ensure sensible answers. For example, it is not possible to determine suitable run lengths theoretically and this requires some experimentation on the part of the user. In addition to reading these guidelines, the user should consult the accompanying paper Didelot and Falush (2007) for a more detailed description of the uses and limitations of the method.

2 Version history

2.1 Version 1.0

Version 1.0 of `ClonalFrame` was the first one to be released to the public, in October 2006.

2.2 Version 1.1

Version 1.1 of `ClonalFrame` was released in June 2007. It contains a number of corrections for minor bugs both in the `ClonalFrame` program and in the Graphical User Interface. Both the main program and the GUI othose of f `ClonalFrame` Version 1.1 are completely compatible with those of Version 1.0.

Version 1.1 also contains two new features in the main program:

- the user now has the possibility to assume exponential growth of the population rather than constant size using the `-E` option (cf. Section 5.3);
- the program can now automatically count the number of mutation and recombination events happening in the evolution of the sample at each iteration (cf. Section 5.5).

The graphical user interface of Version 1.1 contains four new features:

- the program can now automatically compare the clonal genealogies inferred by different runs of the program (cf. Section 6.3.4);
- the levels of confidence in the nodes of a consensus tree can now be read and/or displayed (cf. Section 6.5.6);
- the program can test the external to internal branch ratio for deviations from neutrality. This can be useful to see if population dynamics and/or selection had an effect on the data (cf. Section 6.5.2);
- the user can now zoom on particular parts of the genome when using the genomic representation of events (cf. Section 6.2.3).

3 Installation

In order to install `ClonalFrame`, you need to download from `http://bacteria.stats.ox.ac.uk/` the version of the package that is appropriate for your computing environment (DOS/Windows, Linux or Mac).

It is recommended to complete the registration form on the `ClonalFrame` homepage before starting to use the program. This should help us to assess the level of interest in `ClonalFrame`, and where to focus our efforts for future releases. Registered users will also be notified when a new version of `ClonalFrame` becomes available. If you have downloaded the program without first registering, it is not too late to register now. Any feedback about `ClonalFrame` sent to `didelot@stats.ox.ac.uk` is also greatly appreciated.

In Windows, an executable installer will install the `ClonalFrame` package to the location specified by the user and create some links in the “Start menu” to the `ClonalFrame` executable, the Graphical User Interface and the user guide. In Linux, the user needs to unzip the package using the `unzip` command.

Each package contains:

- The `ClonalFrame` executable (`ClonalFrame.exe` in the Windows version)
- The graphical user interface `cfgui` (`cfgui.bat` in the Windows version) as well as all libraries required for the GUI to work

- An example input file `bac.dat` containing the *Bacillus* MLST dataset as used in the paper
- An example output file `bac.out` for the *Bacillus* dataset contained in `bac.dat`
- This user guide in PDF format (`UserGuide.pdf`)

The source code for the `ClonalFrame` executable and the graphical user interface are available on request. The `ClonalFrame` executable was written in C++ and the GUI in Matlab 7.2.

4 Format for the input file

`ClonalFrame` can be applied to any kind of sequence data, from a single fragment of DNA to whole genomes. It is well suited for the analysis of MLST data (cf. for example `http://pubmlst.org`), where 7 gene fragments have been sequenced but becomes progressively more powerful as the sequenced regions increase in length and number up to whole genomes (cf. for example the Genomes OnLine Database at `http://www.genomesonline.org/` for a list of genome sequencing projects). However, it requires the sequences to be aligned. If you have genomic data that is not aligned, we recommend using `Mauve` (available at `http://gel.ahabs.wisc.edu/mauve/`) which produces alignment of whole bacterial genomes in exactly the format required for analysis with `ClonalFrame`.

`ClonalFrame` uses extended multi-FASTA file type (XMFA), which is the output filetype of many alignment programs such as Mauve or Shuffle-Lagan. An XMFA file is made of blocks of aligned sequences. Each block is a multi-FASTA file followed by the `=` symbol. Moreover, each FASTA header line starts with:

`> seq_id:start-end [+/-]`

where `seq_id` is a unique identifier for the isolate, `start` and `end` indicate where the block is located on the genome of the isolate and `[+/-]` indicates on which DNA strand the sequence is found.

Here is a simple example of an XMFA file with 3 isolates and 2 blocks:

```
> ST1:11-20 +
ACGTACGTAC
> ST2:11-20 +
ACGTAAGTAC
> ST3:535-544 +
ACGTAC-TAA
=
> ST1:21-25 +
ACGTA
> ST2:21-25 +
ACGTT
> ST3:5733-5737 -
ACGTA
=
```

Any line starting with the `#` symbol is ignored. Any block that does not contain all the strains is ignored. If a whole fragment is missing for one of the strains and the user still wants to use this fragment in the analysis, that fragment must be input as a long gap using the `-` symbol on all positions of the missing fragment as was done for the gene *mutX* of the strain ST12 in the example file below. The order of the strains need not be the same from one fragment to the next.

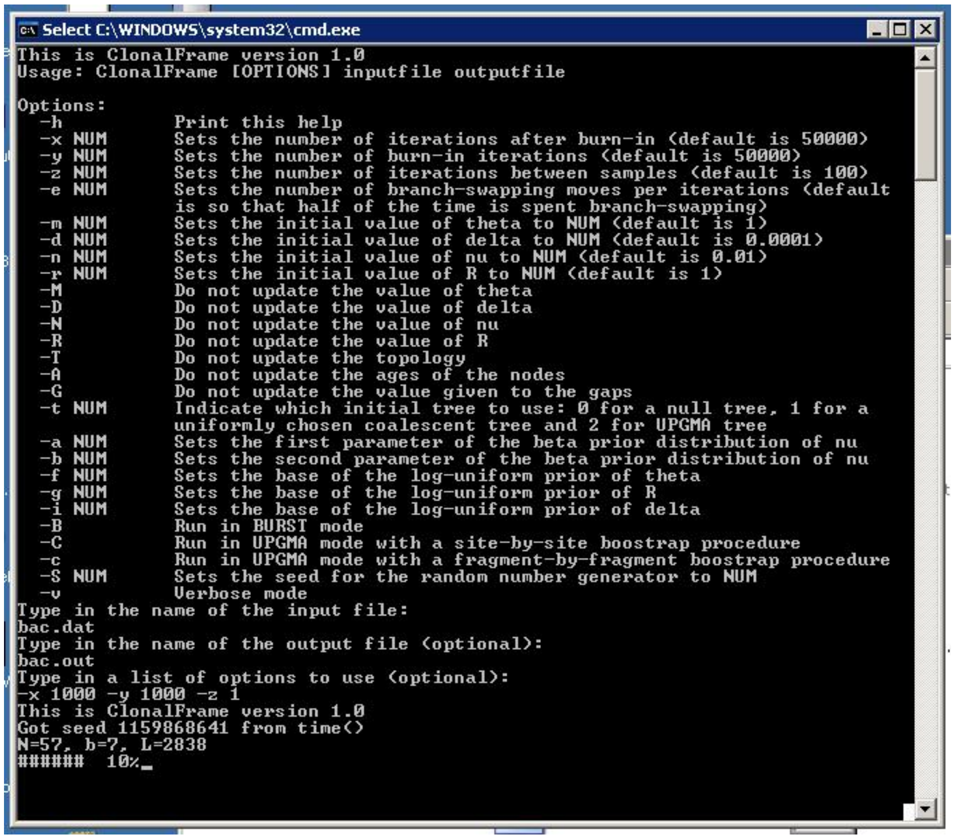
The current version of `ClonalFrame` does not actually use the position of the blocks in computation. Instead, each block is assumed to be distant from the previous one so that each recombination event only affects a single block. The GUI does use the start location of the first isolate in order to decide the spacing of the blocks relative to each other in plots. The genomic location and direction of blocks can be omitted in an input file. This is useful for example for MLST data where this information is unavailable. In this case, the GUI places the fragments next to each oter as in the paper. The simplest input format is therefore a FASTA file, or several FASTA files concatenated and delimited by the `=` symbol on an otherwise empty line as in the following example with three strains and three fragments:

```
# First gene: actA
> ST12
CGAGGAGTCCAGAAGCCTTAGCAA
> ST25
CGAGGAGTCCAGTAGCCTTAGCAA
> ST53
CGAGGAGTCCAGAAGCCTT-GCAA
=
# Second gene: recB
> ST25
CGCAGGAGTCCCAAAGAAGCCACTTAGAAACAA
> ST12
CGCAGGAGTCCCATAGAAGCCACTTAGAAACAA
> ST53
CGCAGGAGTCCC---GAAGCCACTTAGAAACAA
=
# Third gene fragment: mutX
> ST53
TGCATGCATGTGCATGCCCGAGAGCGCGATC
> ST25
-------------------------------
> ST12
TGCATGCATGTGAATGCCCGAGAGCGCGATC
=
```

An example input file (`bac.dat`) is included in the `ClonalFrame` installation package.

5 Running `ClonalFrame`

The first stage in using `ClonalFrame` is to run the MCMC algorithm on the data. This first part is time consuming and the algorithm provides only minimal feedback during the run. Once the output is produced at the end of the run, the GUI can be used to visualise multiple summaries of the analysis as described in Section 6.



```
Select C:\WINDOWS\system32\cmd.exe
This is ClonalFrame version 1.0
Usage: ClonalFrame [OPTIONS] inputfile outputfile

Options:
  -h          Print this help
  -x NUM      Sets the number of iterations after burn-in (default is 50000)
  -y NUM      Sets the number of burn-in iterations (default is 50000)
  -z NUM      Sets the number of iterations between samples (default is 100)
  -e NUM      Sets the number of branch-swapping moves per iterations (default
              is so that half of the time is spent branch-swapping)
  -m NUM      Sets the initial value of theta to NUM (default is 1)
  -d NUM      Sets the initial value of delta to NUM (default is 0.0001)
  -n NUM      Sets the initial value of nu to NUM (default is 0.01)
  -r NUM      Sets the initial value of R to NUM (default is 1)
  -M          Do not update the value of theta
  -D          Do not update the value of delta
  -N          Do not update the value of nu
  -R          Do not update the value of R
  -T          Do not update the topology
  -A          Do not update the ages of the nodes
  -G          Do not update the value given to the gaps
  -t NUM      Indicate which initial tree to use: 0 for a null tree, 1 for a
              uniformly chosen coalescent tree and 2 for UPGMA tree
  -a NUM      Sets the first parameter of the beta prior distribution of nu
  -b NUM      Sets the second parameter of the beta prior distribution of nu
  -f NUM      Sets the base of the log-uniform prior of theta
  -g NUM      Sets the base of the log-uniform prior of R
  -i NUM      Sets the base of the log-uniform prior of delta
  -B          Run in BURST mode
  -C          Run in UPGMA mode with a site-by-site boostrap procedure
  -c          Run in UPGMA mode with a fragment-by-fragment boostrap procedure
  -S NUM      Sets the seed for the random number generator to NUM
  -v          Verbose mode
Type in the name of the input file:
bac.dat
Type in the name of the output file (optional):
bac.out
Type in a list of options to use (optional):
-x 1000 -y 1000 -z 1
This is ClonalFrame version 1.0
Got seed 1159868641 from time()
N=57, b=7, L=2838
######  10%_
```

Figure 1: ClonalFrame in action on a Windows system

5.1 General usage

5.1.1 Windows Users

The easiest way to use `ClonalFrame` in Windows is to click on "Start", "All programs", "Clonal-Frame" and "Clonal Frame". A window similar to the one shown in Figure 1 will then pop up. At the top of the window is indicated the version number of `ClonalFrame` in use, as well as a list of options (cf. below for an explanation of the role of all these options).

The user is invited to type in the name of an input file, which must be in the format described in Section 4. The user can indicate either the name of a file if this file is in the directory where `ClonalFrame` was installed (for example `bac.dat` to use the example input file provided with the program); or use a path relative to the directory where `ClonalFrame` was installed (for example `Work\data.dat` if there is a file called `data.dat` in a subdirectory called `Work` of the installation directory; or use an absolute path (for example `D:\Work\data.dat` if the input file `data.dat` is in the directory `Work` on the `D:` drive.

After entering the name of an input file, the user is asked for the name of an output file (which can be indicated either relatively to the installation directory or absolutely as described above). If none is entered, the program will automatically use the name `XXX.out` where `XXX` is the location and name of the input file.

Finally, the user can add some options. In the example on Figure 1, the user indicated the options `-x 1000 -y 1000 -z 1` which are explained below.

The algorithm then starts running and an indicator of progress is shown (in the example of Figure 1, the chain has run for approximately 10% of the iterations).

5.1.2 Other systems

The general usage of `ClonalFrame` is as follows:

```
ClonalFrame [OPTIONS] inputfile outputfile
```

For example, in order to reproduce the Windows session shown in Figure 1, the user would type in:

```
ClonalFrame -x 1000 -y 1000 -z 1 bac.dat bac.out
```

Note that on many systems, for security reasons, the current directory is not included in the path where the program looks for executables. The user would then have to type in `./ClonalFrame` instead of `ClonalFrame`.

5.2 Basic options

One very important decision for the user is how long to run the program for. The program is started from a random configuration, and from there takes a series of steps through the parameter space, each of which depends (only) on the parameter values at the previous step. This procedure induces correlations between the state of the Markov chain at different points during the run. The hope is that by running the simulation for long enough, the correlations will be negligible. There are two issues to worry about: (1) burnin length: how long to run the simulation before collecting data to minimize the effect of the starting configuration, and (2) how long to run the simulation after the burnin to get accurate parameter estimates. To choose an appropriate burnin length, it is really helpful to look at the values of summary statistics that are printed out by the program (i.e. the values of θ, R, ν, δ and the likelihood) to see whether they appear to have converged. It is also important to compare the results of independent simulations, started at different initial conditions, to check that they have arrived at the same posterior distribution of the summary statistics and the topology. If not, it implies that the program has not been run long enough for convergence to take place.

The number of possible topologies of the genealogy increases rapidly with the number of strains in the sample. Consequently, mixing takes more iterations as the number of strains increases. Conversely, increasing the length or number of sequences increases the information on the genealogy and might facilitate slightly faster convergence.

The basic options relative to the run of the MCMC are as follows:

- `-x NUM` sets the number of MCMC iterations after the burn-in period to NUM (default is 50000).
- `-y NUM` sets the number of burn-in iterations to NUM (default is 50000).
- `-z NUM` sets the number of iterations that are performed between recording the parameter values in the posterior sample (also called the thinning interval) to NUM (default is 100).

- `-e NUM` sets the number of branch-swapping attempts per iteration. If this option is not set, the number of branch-swapping attempts per iteration is automatically chosen so that at least half of the time of each iteration is spent on branch-swapping. Note that in the paper Didelot and Falush (2007) only one branch-swapping attempt was performed per iteration with the consequence that the algorithm did not mix as well as it does now.

5.3 Advanced options

Initial values are a random tree (which will be different on each run) and particular values for the parameters θ, R, ν and δ. It can be useful to change the initial values of these parameters to check that the algorithm has been run sufficiently long for the chain to have reached convergence. Further, the user may also wish to check the prior if either (1) the default values are not appropriate to the particular dataset (2) there is good information about values from other source (3) the user wishes to check that the priors do not have an undue influence on the posterior.

These are the options:

- `-m NUM` sets the initial value of θ to NUM (default is 1). θ is the mutational rate and is assumed to be constant on the branches of the topology. This option can be used with the `-M` option to force the chain to use a constant predetermined value of θ.
- `-d NUM` sets the initial value of δ^{-1} to NUM (default is 0.0001). δ is the average tract length of a recombination event. This option can be used with the `-N` option to force the chain to use a constant predetermined value of ν.
- `-n NUM` sets the initial value of ν to NUM (default is 0.015). ν is the rate of new polymorphism introduced by recombination. This option can be used with the `-N` option to force the chain to use a constant predetermined value of ν.
- `-r NUM` sets the initial value of R to NUM (default is 1). R is the recombination rate and is assumed to be constant of the branches of the topology. This option can be used with the `-R` option to force the chain to use a constant predetermined value of R.
- `-M` prevents the chain from updating the value of θ.
- `-D` prevents the chain from updating the value of δ.
- `-N` prevents the chain from updating the value of ν.
- `-R` prevents the chain from updating the value of R.
- `-t NUM` indicates which initial tree to use. NUM can either be 0 for the tree ((((1,2),3),4),...), 1 for a uniformly chosen coalescent tree or 2 for a scaled UPGMA tree. By default, a uniformly chosen coalescent tree is used. Because the UPGMA tree represents a good guess as to the likely shape of the topology, it may aid convergence, particularly if there are many strains in the dataset, however it is dangerous choice to make since the algorithm could easily get stuck in a local optimum close to the tree.
- `-a NUM` sets the first parameter of the beta prior distribution of ν (default is 1).
- `-b NUM` sets the second parameter of the beta prior distribution of ν (default is 1).

- `-f NUM` sets the base of the log-uniform prior of θ (default is $e = 2.718...$).
- `-g NUM` sets the base of the log-uniform prior of R (default is $e = 2.718...$).
- `-i NUM` sets the base of the log-uniform prior of δ (default is $e = 2.718...$).
- `-E NUM` sets the model of population dynamics to exponential growth with rate NUM. The default value is 0 which is equivalent to a constant population size model.

5.4 Miscelleneous options

It is also possible to use `ClonalFrame` to run other algorithms, which may be useful for comparative purposes, as illustrated for example in Didelot and Falush (2007).

- `-v` set the program to run in verbose mode. All the information about the moves proposed as the MCMC runs will then be displayed. In the default non-verbose mode, a simple indicator of progress is shown.
- `-B` set the program to run in BURST mode.
- `-C` set the program to run in UPGMA mode with a site-by-site bootstrap procedure.
- `-c` set the program to run in UPGMA mode with a fragment-by-fragment bootstrap procedure.
- `-S NUM` sets the seed for the random number generator to NUM. If this option is not used, the random number generator is seeded using `/dev/random` if available (i.e. on a Unix or Linux system) or a system call to `time()` otherwise. This option is especially useful to repeat the run of a chain exactly as it happened, for example in conjunction with the `-v` verbose mode option in order to see more details about the chain moves.

5.5 Count of the number of evolutionary events

When run in the verbose mode (using the `-v` option), `ClonalFrame` outputs three lines at each recorded iteration of the chain in the following format:

```
mut=X1,rec=X2,mutrec=X3,cf=X4
from isolates to MRCA: mut=X5,rec=X6,mutrec=X7,cf=X8
mutin=X9,mutout=X10,recin=X11,recout=X12
```

The meaning of the four values on the first line are as follows:

- `X1` is the total number of mutation events happening on all of the branches of the topology;
- `X2` is the total number of recombination events;
- `X3` is the total number of substitutions introduced by recombination;
- `X4` is the size of the clonal frame, ie. the number of sites in the dataset for which no recombination is found anywhere in the genealogy.

The four quantities on the second line are the same as ones on the first line, except that each one is now calculated not for the whole topology, but is an average for the path between each leaf and the topmost node of the genealogy (ie. the path between each isolate and the most recent common ancestor of the sample).

Finally the four figures displayed on the third line are as follows:

- `X9` is the total number of mutations into a nucleotide found somewhere else on the tree;
- `X10` is the total number of mutations into a nucleotide not found anywhere else on the tree;
- `X11` is the total number of substitutions caused by recombination that change the site into a type found somewhere else on the tree;
- `X12` is the total number of substitutions caused by recombination that change the site into a type not found anywhere else on the tree.

Note that `X9+X10=X1` and `X11+X12=X3`.

If the ratio `X9` over `X1` is high, it might indicate that many substitutions identified by `ClonalFrame` as mutations are in fact caused by intra-population recombination.

The ratio of `X11` over `X3` is a measure of the proportion of intra-population recombination compared to inter-population recombination in the fragments that `ClonalFrame` has identified as being imported. It is therefore a crude measure of the genetic isolation of the population under study.

6 Using the Graphical User Interface

The Graphical User Interface (GUI) of `ClonalFrame` is written in Matlab 7.2. However, it does not require the user to have Matlab installed on the computer.

The GUI is contained in an executable called `cfgui` (or `cfgui.bat` for Windows users) which can take up to two arguments: the location of a `ClonalFrame` output file, and an optional label file. If no argument is provided (for example by double clicking on the program in Windows), the GUI lets the user select a `ClonalFrame` output file through a file selection dialog window.

A label file is a file containing as many lines as there are strains in the dataset, and the *i*-th line specifies an alternate name for the *i*-th strain. The order has to be the same as for the first fragment of the input file. If no label file is specified, the GUI simply uses the labels of the strains in the input file. Using a label file is therefore completely optional but can be useful for example to see how different isolates with the same sequence type are located.

Once the file is loaded, a window appears containing a 50% consensus tree based on the posterior sample of topologies inferred by `ClonalFrame` (as in Figure 2). This window is the main window of the GUI, from which all analyses are performed.

6.1 Definition of a consensus tree

Majority-rule consensus trees is the method used by the `ClonalFrame` GUI to summarise the information contained on all the trees in the posterior sample in a single tree. Another, potentially much more powerful method is to use a split graph as implemented in the program `SplitsTree`. See Section 6.3.7 to see how `SplitsTree` can be used on the output of `ClonalFrame`.

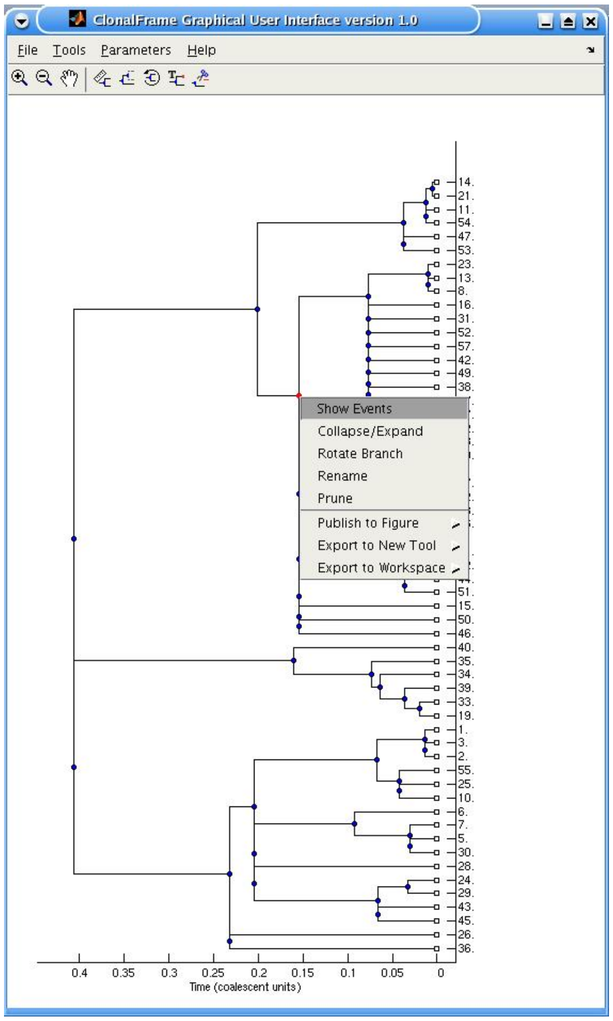


Figure 2: Main window of the Graphical User Interface

A majority-rule consensus tree is defined by its threshold x. The consensus trees produced by default by `ClonalFrame` are 50% consensus trees which means that $x = 0.5$, but the GUI allows the user to change the value of the threshold (cf. Section 6.5). Any tree can be seen as a set of splits, where each internal node of the tree defines a split, which is equal to the set of taxa than cluster below that node. For example the tree shown in Figure 3A contains the splits: (C,D) and (B,C,D) whereas the tree shown in Figure 3B contains the splits: (A,B) and (C,D).

A majority-rule consensus tree with threshold x contains exactly all the splits found in at least a proportion x of the trees sampled. For example, imagine a sample where the tree from Figure 3B occurs three times and the tree from Figure 3A occurs one time. The total list of splits found in the set of all trees is: (C,D) found 4 times, (B,C,D) found 1 times and (A,B) found 3 times. This means that if $x > 0.75$, only the split (C,D) will be in the consensus tree as shown in Figure

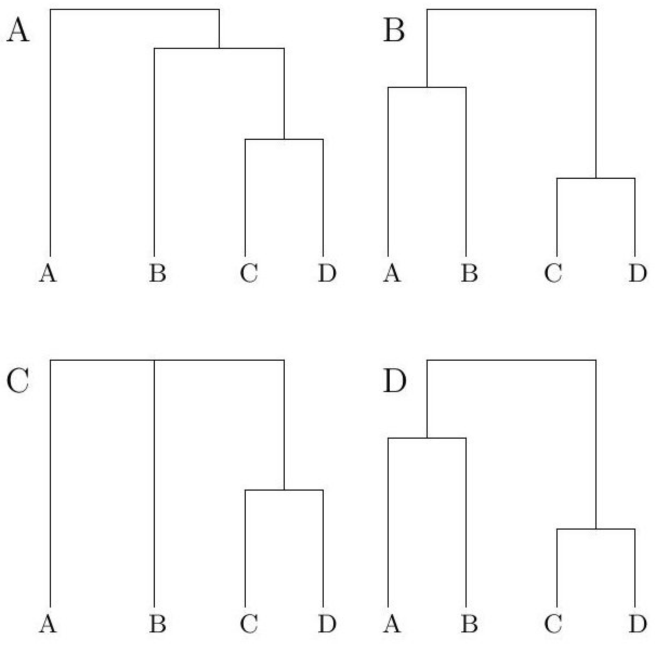


Figure 3: Illustration of how consensus trees are built

3C. However, if $x < 0.75$, both the split (A,B) and the split (C,D) will be in the consensus tree as shown in Figure 3D.

6.2 Exploring the consensus tree

By right-clicking on a node of the consensus tree and selecting "View Events" (as illustrated in Figure 2), it is possible to see the events inferred by `ClonalFrame` on the branch directly above that node. A label is automatically given to the selected node ('A' for the first node selected, 'B' for the second, etc.) and this label is indicated at the top of a newly created window that shows the events, so that the user can keep track of which node the window correspond to. An example is shown in Figure 4.

Two different types of representations are available (the choice of the representation is done in the "Parameters" menu, by clicking on "Representation mode" and then either "Genetic" or "Genomic").

6.2.1 Genetic representation of the events

The genetic representation mode (illustrated in Figure 5) shows all the blocks on one line, with grey vertical lines between the blocks. Each inferred substitution is indicated by a cross, the intensity of which indicates the posterior probability for that substitution. The red line indicates at each locus

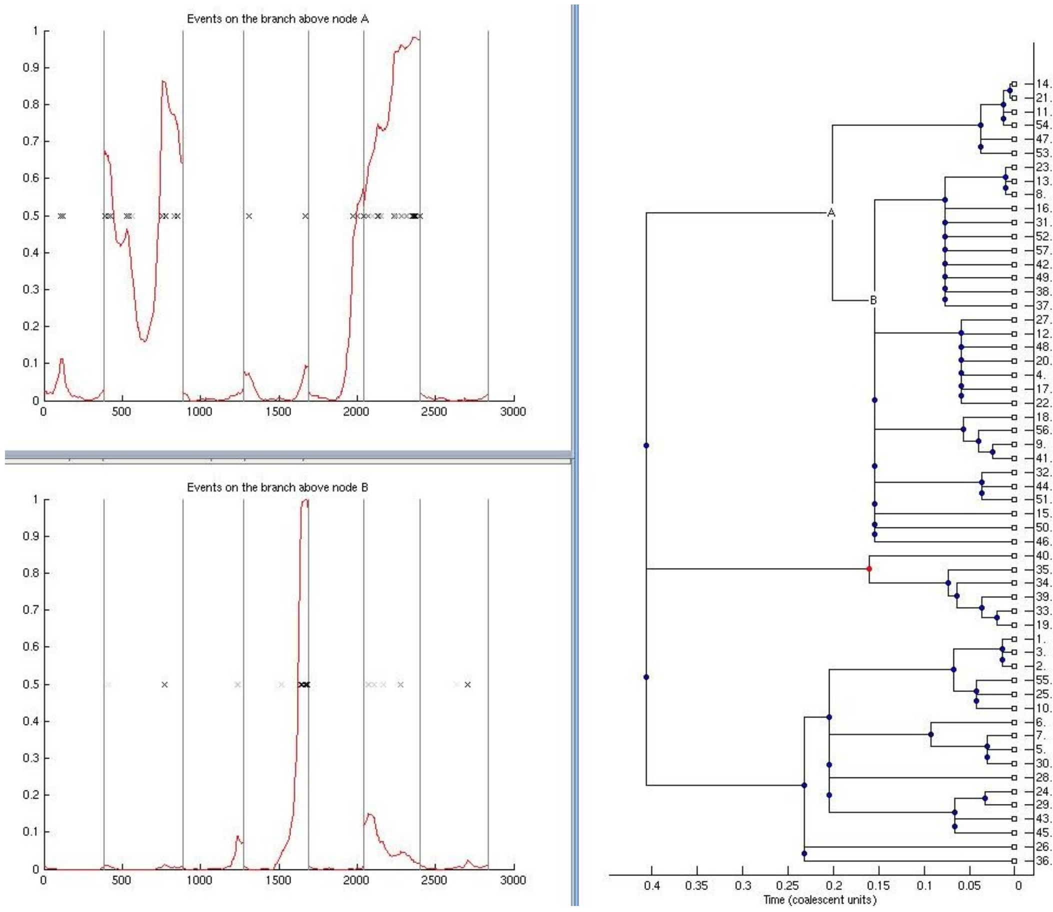

Events on the branch above node A
Events on the branch above node B
A
B
Time (coalescent units)

Figure 4: Example of how the events on the branches of the consensus tree can be looked at

the probability for an import on a scale from 0 (bottom of the y axis) to 1 (top of the Y axis). This representation mode is well adapted when the size L of the input alignment is relatively small, for example with MLST data.

6.2.2 Genomic representation of the events

The genomic representation mode (illustrated in Figure 6) shows the alignment on as many lines of X bp as required by the size of the input alignment L, where X is an option that can be set when switching the genomic representation on. Substitutions happening more than 50% of the time in the posterior are indicated by green triangles. The color of each position indicates the posterior probability for an import at that position with black for a probability of 0 and red for probability of 1. This representation mode is well adapted for the exploration of genomic data, but can also be useful as a compact way to represent the events for MLST data (e.g. Figure 7). When setting the representation mode to genomic through the "Parameters" menu (cf. Section 6.5), the GUI allows

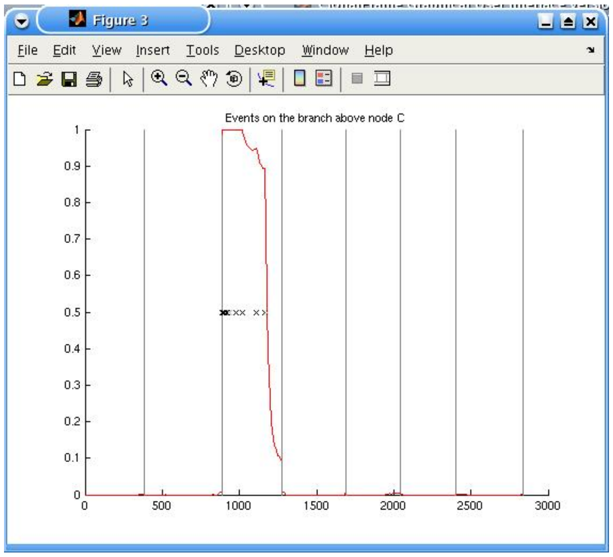


Figure 5: Example of genetic representation of the mutation and recombination events

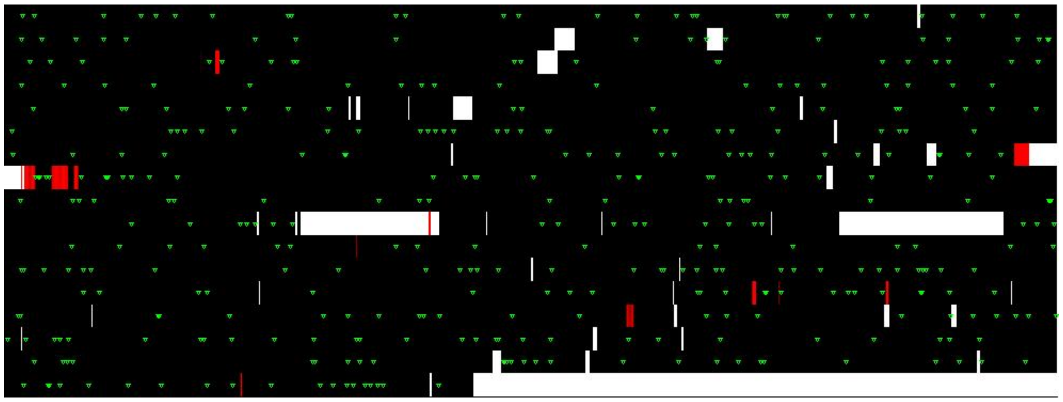


Figure 6: Example of genomic representation of the mutation and recombination events for a whole genome alignment

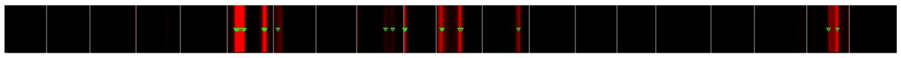


Figure 7: Example of genomic representation of the mutation and recombination events for a dataset comprising 20 gene fragments

the user to customize the representation, by showing the substitutions or not, showing the limits between blocks of alignment or not and setting the maximum number of base pairs per line.

6.2.3 Genetic zoom in the genomic mode of representation

When looking at the events on a branch in genomic representation mode, it is possible to zoom in onto a small region of the alignment to better see the events in that region. To do so, the user needs to click on "File", and then "Genetic zoom". The user must then click on the middle of the region of interest, which will then be displayed in a separate window and using the genetic mode of representation. The length of the fragments shown when using the genetic zoom can be specified when turning the genomic mode of representation on.

6.3 The "File" menu

Here we describe the important options available in the "File" menu.

6.3.1 Open a ClonalFrame output file

The "Open a ClonalFrame output file..." option lets the user select a `ClonalFrame` output file on which to work. This is equivalent to closing down the graphical user interface and restarting it for another output file.

6.3.2 Open a label file

The "Open a label file..." option allows the user to add new labels to the strains in the dataset. The user is asked to select a file which must contain as many lines as there are strains in the sample, and the name on the i-th line is then used as an alternate name for the i-th strain in the sample.

6.3.3 Convergence assessment

The "Compare for convergence with..." option allows the user to assess the convergence of the MCMC by comparing it with the output from other chains run on the same dataset and with the same options. This follows the method of Gelman and Rubin (1992). The user has to select one or several `ClonalFrame` output files that he wants to compare the currently opened file to. For the test to be valid, the different chains should have been run using the same data and parameters, but with distinct starting points. The Gelman and Rubin test is then performed on the values of θ, R, ν and δ as well as the TMRCA of the genealogy. A Gelman and Rubin statistic above 1.2 indicates poor convergence. Note however that this optic does not test the mixing of the tree topology which may be the slowest element of the parameters to converge.

6.3.4 The tree comparison tool

The "Compare consensus tree with..." option is a very general tool that allows the user to select another `ClonalFrame` output file, and shows the level of confidence in each node of the current consensus tree according to this second output file. The second output file must be based on an alignment comprising the same isolates as the current output file. This tree comparison tool can be useful in three different contexts:

- As a way to assess the convergence and mixing properties of the MCMC, when the second output file is based on exactly the same data and parameter priors as the first one;

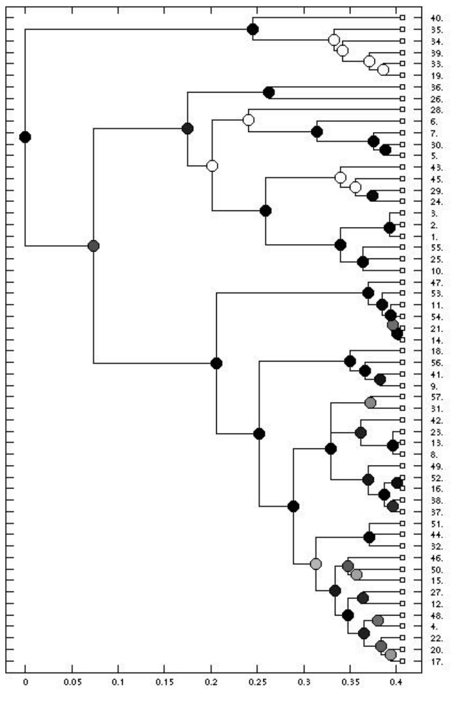
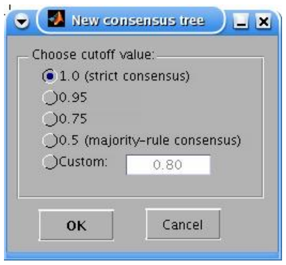


Figure 8: Example of the tree comparison tool

- As a way to measure the effect of the prior on each parameters, when the second output file is based on exactly the same data but different values of the prior as specified by the options `-a`, `-b`, `-f`, `-g` and `-i`;

- As a way to compare the results based on different datasets, for example to compare the results based on the sequence of two different sets of genes.

An example of the output of the tree comparison tool is shown in Figure 8. The shading of the nodes is proportional to how well supported they are according to the second analysis. For example, the nodes shown in white are the ones that are found in the first analysis but not in the second, and the ones shown in black are the ones found in both. For the comparison of the two output files to be complete, it is necessary to perform another comparison with the second file loaded first and then compared to the first one. This would allow the user to see in white the nodes found by the second analysis but not by the first one.



Choose cutoff value:

- (● 1.0 (strict consensus))
- 0.95
- 0.75
- 0.5 (majority-rule consensus)
- Custom: 0.80

Figure 9: Window for the selection of a cut-off value when creating a new consensus tree

6.3.5 Consensus tree from several files

The "Build consensus tree with..." option allows the user to build a consensus tree from the output of several `ClonalFrame` output files. The resulting consensus tree is based on the union of the posterior samples contained in these output files. The user can first select one or several files on which the consensus tree is to be based, along with the currently opened file. Then a window such as the one on Figure 9 will pop-up, allowing the user needs to specify a cut-off value for the new consensus tree.

6.3.6 Export the consensus tree as a Newick file

The "Export as Newick file" option allows the user to export the consensus tree as a Newick file, which can then be used with different programs, for example MEGA (freely available from `http://www.megasoftware.net`) which has more extended drawing capabilities than the `ClonalFrame` GUI. For example, this is how the Figures from Didelot and Falush (2007) were produced.

6.3.7 Export posterior sample as Newick file

The "Export posterior sample as Newick file" option allows the user to export all the trees sampled after the burn-in period into a Newick file, which can then be used with a different program, such as `SplitsTree4` (freely available from `http://www.splitstree.org/`) which contains a very extensive set of methods to summarize a set of trees into a split network or a reticulate network. This is useful because the consensus trees used by the graphical user interface of `ClonalFrame` are a simplistic way to summarize the information contained in several trees. Figure 10 shows an example of a split network drawn by `SplitTree4` based on a sample of trees exported from `ClonalFrame` using this option.

6.3.8 Export as a distance matrix

This option allows the user to create a file containing a distance matrix in the Phylip format as described here:

`http://evolution.genetics.washington.edu/phylip/doc/distance.html`

The file contains a distance matrix where the distance between any pair of strains is equal to the average of their distance during the run of the chain. Note that this is not based on the consensus tree and therefore provides a different way to summarize the topologies explored during the run of

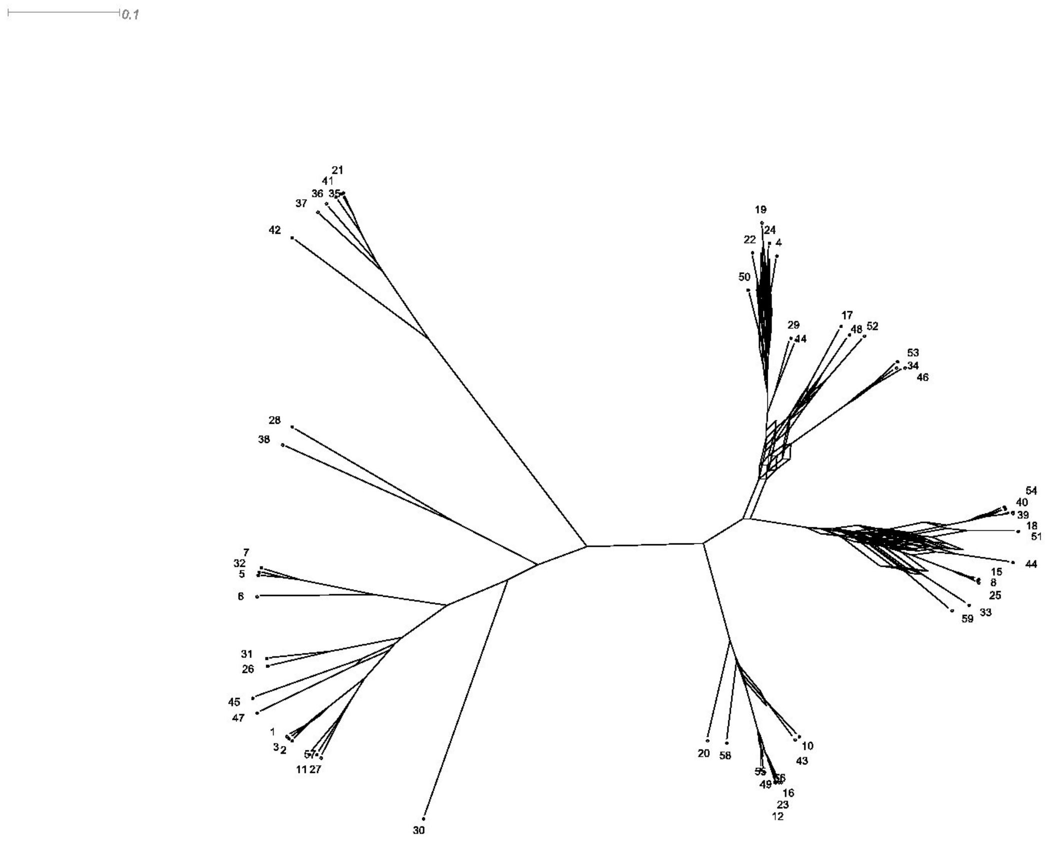


Figure 10: Example of split network produced by SplitTree4 based on a sample of trees exported using the "Export posterior sample as Newick file" option

the chain. This file can then be used in other software packages such as `emboss`, freely available from `http://emboss.sourceforge.net`, or `SplitsTree`, available from `http://www.splitstree.org/`.

6.3.9 Export as a picture file

This option allows the user to export the current view of the consensus tree to a wide variety of graphics file format including JPEG, PostScript and PDF.

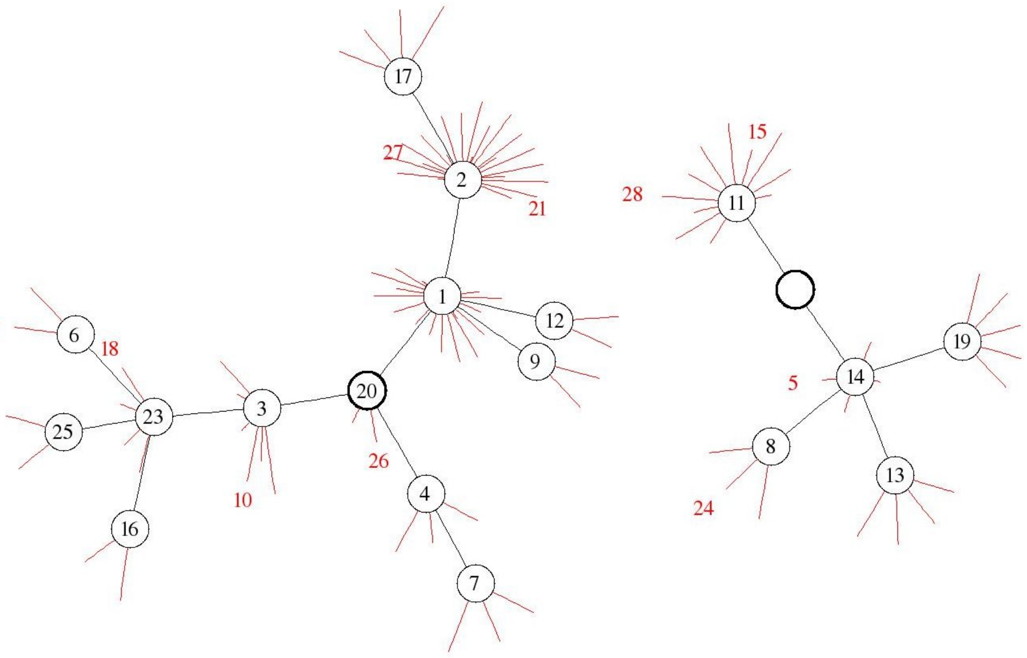
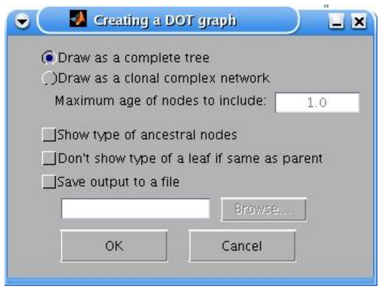


Figure 11: Example of DOT graph representation

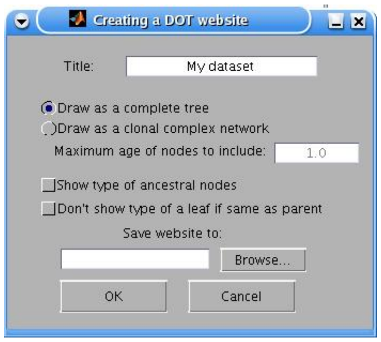


[✓ Draw as a complete tree]
Draw as a clonal complex network
Maximum age of nodes to include: 1.0
Show type of ancestral nodes
Don't show type of a leaf if same as parent
Save output to a file

Figure 12: Window presenting the options when representing the consensus tree with a DOT graph

6.3.10 Create a DOT graph

The "Show a DOT graph" option can be used to see the results as a Graphviz network (this is done using the program Graphviz `neato`, which is fully documented on the GraphViz site `http://www.graphviz.org/`). This option is particularly useful for the exploration and display of large datasets (over a hundred isolates). The user can choose four options in a window such as the one represented in Figure 12. The first one is the maximum age of nodes to include in the graph. The second one is whether or not the type of ancestral nodes should be shown. The third one is whether the type of isolates which are identical to their parent node should be shown or not. The last one is the location of the output file if the user wants to save the DOT graph. This can


Creating a DOT website

Title: My dataset

(● Draw as a complete tree) ○ Draw as a clonal complex network

Maximum age of nodes to include: 1.0

☐ Show type of ancestral nodes

☐ Don't show type of a leaf if same as parent

Save website to:

Figure 13: Window presenting the options when creating a DOT website

be saved in DOT, JPG or PostScript format.

The resulting network shows ancestral nodes in black and the location of isolates in red, with each red line indicating a single isolate (Figure 11 shows an example). If the type of an ancestral node is not found amongst the isolates, it is shown as an empty circle even if the option for ancestral nodes label was selected. The ancestral node of each network component is indicated by a thicker circle.

6.3.11 Create a DOT website

In the "File" menu, the "Create a DOT website" option allows the user to quickly set up a website on which to present the results of an analysis. The user needs to select a location where the website will be saved, a title for the analysis as well as the first three options described in the above section "Viewing as a DOT graph" since the website uses a DOT graph. These choices are made thanks to a window similar to the one illustrated in Figure 13.

The "Create a DOT website" option creates an "index.htm" file in the selected directory as well as several png images. The created webpage contains a link showing each of the parameters and statistics available in the "Parameters" menu (cf. Section 6.5), as well as a clickable DOT network representing the inferred consensus tree. Clicking on a node of this network shows the events that were inferred on the branch directly above that node as described in the Section 6.2.1.

6.4 The "Tools" menu

The "Tools" menu can be used to select a "Tool", ie. an action associated to a mouse left click on the consensus tree. The different tools are also available in the icons bar at the top of the window.

The available options include:

- "Zoom in", "Zoom out", "Reset view", "Fit window" and "Pan" which can be used to control which proportion of the tree is shown
- "Inspect" shows some information about the nodes such as their distances to one another
- "Collapse/Expand" can be used to hide and show whole branches of the tree
- "Rotate" can be used to rotate a branch

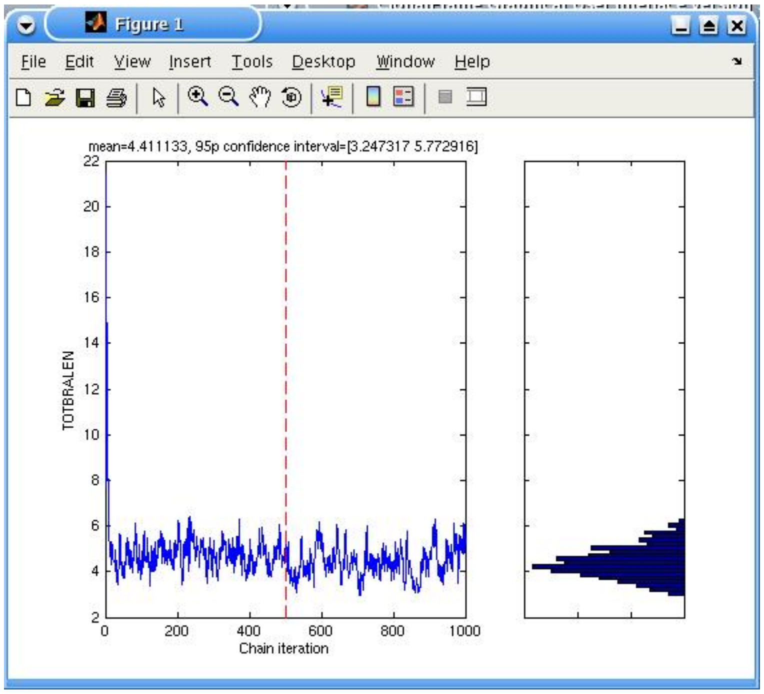


Figure 1
File Edit View Insert Tools Desktop Window Help
mean=4.411133, 95p confidence interval=[3.247317 5.772916]
TOTBRALEN
Chain iteration

Figure 14: Example of parameter representation

- "Rename" is used to rename the leaves
- "Prune" is used to prune whole branches of the tree
- "Threshold collapse" can be used to hide nodes depending on their age
- "Expand all" cancels all the collapses performed
- "Find Leaf/Branch" can be used to find a strain in the tree

6.5 The "Parameters" menu

6.5.1 Visualisation of parameters and statistics

The "Parameters" menu can be used to see the evolution of different parameters and statistics during the MCMC run. All of the options below produce a graph (cf. Figure 14) where the X-axis represent the iterations of the MCMC, and the Y-axis represent the different values taken by a variable for the different iterations of the chain. In all the graphs, a vertical red dotted line represents the end of the burn-in period and the beginning of the sampling. The histogram on the left represents the posterior distribution inferred by `ClonalFrame`. The different options are:

- "View theta", "View nu", "View R" and "View delta" can be used to follow the evolution of the components of $\mathcal{M}$ in the MCMC (this can also be used to assess convergence and mixing

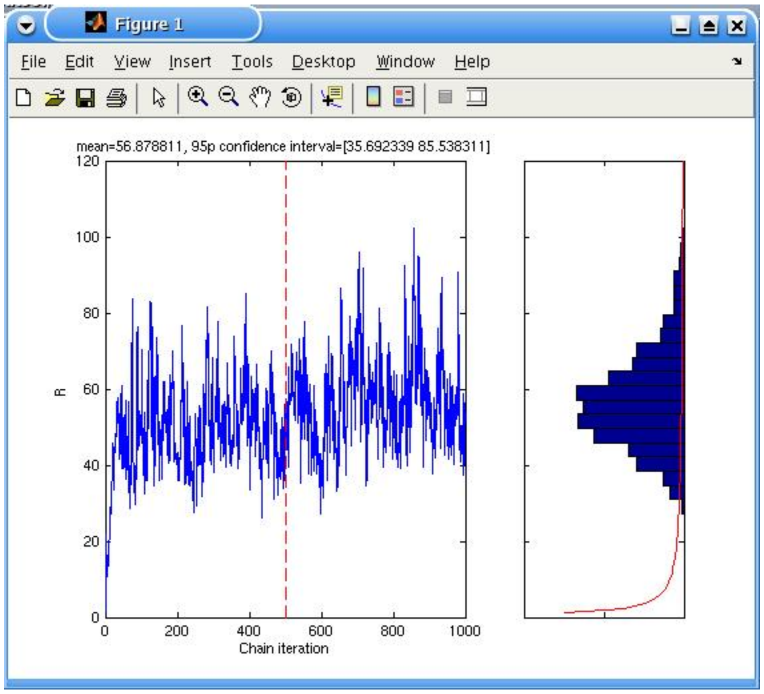


Figure 1
File Edit View Insert Tools Desktop Window Help
mean=56.878811, 95p confidence interval=[35.692339 85.538311]
R
Chain iteration

Figure 15: Another example of parameter representation

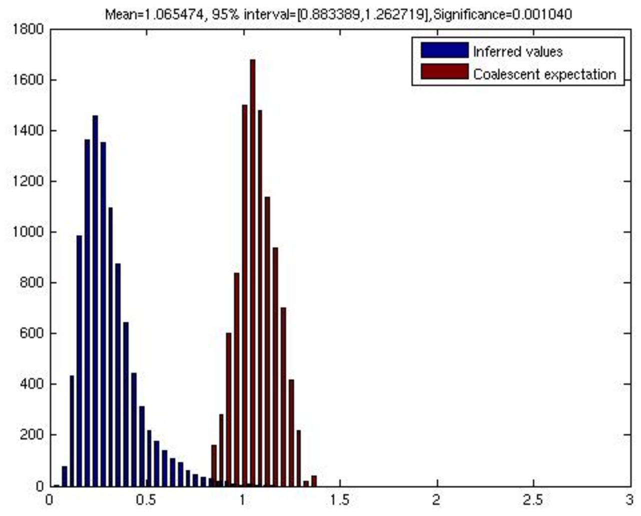


Figure 16: Illustration of the test of external to internal branch lengths ratio

properties of the chain for example). When visualising the θ, ν, R or δ, the prior distribution of these parameters is indicated on the right panel by a red line as in Figure 15.

- "View r over m" and "View rho over theta" show the evolution of r/m and ρ/θ as the chain ran. This two statistics can be used to assess the relative contribution of recombination and mutation in the creation of the sample from a common ancestor. ρ/θ is the ratio of rates at which recombination and mutation occur. It is therefore a measure of how often recombination events happen relative to mutations. r/m is the ratio of probabilities that a given site is altered through recombination and mutation. It is therefore a measure of how important the effect of recombination was in the diversification of the sample relative to mutation.
- "View likelihood" can be used to see the evolution of the log-likelihood of the MCMC
- "View TMRCA" and "View total branch length" can be used to see the evolution of the time to the most recent common ancestor and total branch length of the clonal genealogy $\mathcal{T}$ during the run of the MCMC

6.5.2 External to internal branch length ratio test

The "View external/internal branch length ratio" option shows in red the distribution of the sum of the lengths of the external branches (ie. the ones that connect a leaf of the tree) divided by the sum of the lengths of the internal branches (ie. the ones that connect two internal nodes of the tree). It also computes and shows in blue the expected distribution of the external to internal branch length ratio under the coalescent model. Finally, it calculates the statistical significance of the deviation between the observed and expected ratios. An example of this test of external to internal branch lengths ratio is shown in Figure 16 (note that the legend is wrongly inverted in some versions of ClonalFrame as in Figure 16).

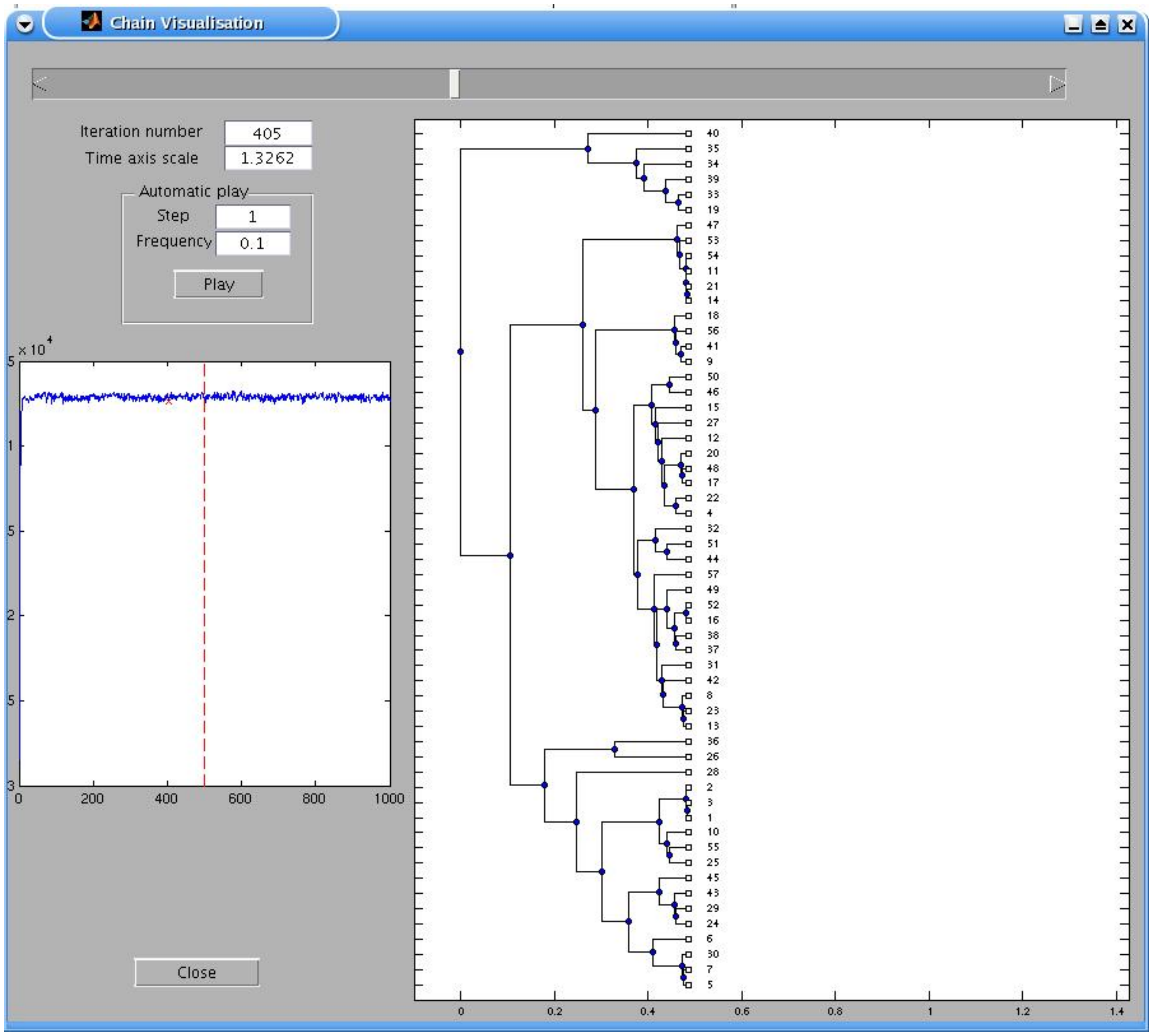
If the external to internal branch length ratio is significantly higher than expected (as in Figure 16), it means that the inferred genealogy is unexpectedly "star-like", which is consistent with either an expansion of the population size or the acquisition of a fitness advantage early in the history of the sample.

6.5.3 Visualisation of the sampled trees

The "View chain" option in the "Parameters" menu can be used to see the succession of states taken by the topology $\mathcal{T}$ during the run of the chain. A special window illustrated in Figure 17 is used to visualise the different topologies. A slider at the top allows the user to move along the chain run. It is also possible to produce a slide show of the different genealogies by setting the "Step" (number of genealogies between two successive views) and the "Speed" (time spent on each genealogy) and clicking on the "Play" button. The genealogies are represented on the panel on the right whereas the panel on the left shows the values taken by either one of the four parameters θ, ν, R and δ or the the log-likelihood.

6.5.4 Representation modes

The "Representation mode" option of the "Parameters" menu, allows the user to select the representation mode to be used for events on each branch (either genetic or genomic, as described in Sections 6.2.1 and 6.2.2). If the genomic representation mode is chosen, 4 more options can be set:



Iteration number: 405
Time axis scale: 1.3262
Step: 1
Frequency: 0.1

Figure 17: Visualisation of the sampled genealogies during the run of the chain

- If the "Show substitutions" option is activated, the substitutions inferred by the program on each branch with posterior probability > 0.5 are shown as green triangles;
- If the "Show beginning and end of blocks" option is activated, the location of the beginning and end of the blocks in the alignment are shown as grey vertical segments;
- The user can specify the "Maximum length of lines" which is the number of base pairs that should be shown on each line of the graphical representation;
- The user can also specify the "Length of zoom" which is the number of base pairs that should

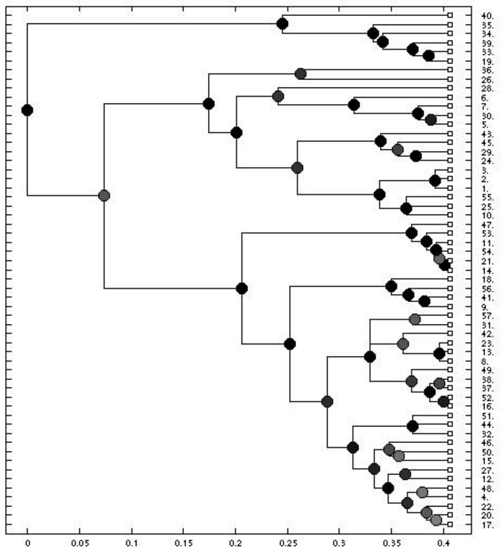


Figure 18: Example of the visualisation of the confidence in the nodes of the consensus tree

be shown when using the "Genetic zoom" option of the genomic representation mode.

6.5.5 Recomputation of the consensus tree

The "Recompute consensus tree" option of the "Parameters" menu allows the user to compute a new consensus tree. The user must first choose the cut-off value used by the majority-rule consensus tree. The default value of 0.5 is a very relaxed cut-off which might result in some nodes of the consensus tree not being very well supported. This however allows to show a maximum of nodes, which is useful for example in conjunction when the confidence in the nodes is shown (cf. below). A value of 0.95 is recommended if the user wants to make sure that his consensus tree only include well supported nodes. A value of 1.0 corresponds to a strict consensus tree, which is not recommended because its output will be heavily dependent on the length for which the chain is run.

6.5.6 Confidence in the nodes of the consensus tree

The last option of the "Parameters" menu is called "Show confidence in nodes". It creates a new figure in which the confidence in each node of the current consensus tree is shown as a colour, with black indicating total confidence and white indicating no confidence at all. The exact value for the confidence in each node can also be read simply by placing the mouse on top of a node of the consensus tree. This, as well as the "Show confidence in nodes" option, is not available after an input file has been loaded into the GUI but requires the user to first recompute a consensus tree using the option "Recompute consensus tree" in the "Parameters" menu. An example of the output of this option is shown in Figure 18

7 Output file

The output of `ClonalFrame` is a single file, containing 13 different parts. Each of the 13 parts starts with a line starting with a `#` symbol followed by the name of that part. The file ends with the line `#end`. Here we describe each of these parts in the order that they appear in the output file. However, when using the graphical user interface described in Section 6, one does not need to look at the output file directly. This section is therefore written for advanced users who want to make the most of `ClonalFrame`, for example by writing there own scripts to interpret the output file. The `ClonalFrame` output format was designed to be highly computer-readable, so that the design of such script should be easy.

An example is given for each part below to illustrate what the output file looks like. This example correspond to a very short run of `ClonalFrame` with parameters `-x=100 -y=50 -z=10` on a very small input file of two blocks and three strains called carriage1, carriage2 and case1.

```
#gene mutA
> carriage1
ACGTACGTAC
> carriage2
ACGTAAGTAC
> case1
ACGTAC-TAA
=

#gene gapR
> case1
ACGTA
> carriage1
ACGTT
> carriage2
ACGTA
=
```

Obviously such a short run of the program on such a small amount of data is only interesting to describe the different parts of the output file and does not have any inference value.

7.1 Consensus tree part

This part starts with the line `#constree`. It is made of a single line containing the 50% majority-rule consensus tree built from the genealogies sampled by the MCMC. The consensus tree is in Newick file format which is described here:
`http://evolution.genetics.washington.edu/phylip/newicktree.html`.

In our example, this part looks as follows:

```
#constree
((1:0.013890,3:0.013890)4:0.063136,2:0.077025)5:0.000000
```

This means that the program has inferred that 1 and 3 are more closely related to each other than to 2. Here the reference number 1, 2 and 3 correspond to strains carriage1, carriage2 and case1 respectively (ie. they are in the order that they are found in the first fragment of the input file); reference number 4 corresponds to the most recent common ancestor of 1 and 3 and reference number 5 correspond to the root of the tree.

7.2 Strain names

This part starts with the line `#names`. It is a simple list of the names used in the input multi-FASTA file for the different strains, in the order that they are for the first block of alignment.

In our example, this part looks as follows:

```
#names
carriage1
carriage2
case1
```

7.3 Consensus events part

This part starts with the line `#consevents`. It contains the events found on the consensus tree of Section 7.1. It is made of a $N \times (2M)$ matrix where the number of lines N is the number of nodes in the consensus tree, and the number of columns is two times M where M is the number of reference sites (cf. Section 7.7). The first number of each pair for each reference site indicates the posterior probability of recombination at that site and the second one the posterior probability of substitution (either through recombination or through mutation).

In our example, this part looks as follows (5 decimal places have been removed for simplicity):

```
#consevents
6.0e-01 0.0e+00 6.0e-01 2.0e-01 6.0e-01 0.0e+00 6.0e-01 0.0e+00 1.0e+00 0.0e+00 1.0e+00 8.0e-01
8.0e-01 0.0e+00 8.0e-01 1.0e-00 8.0e-01 0.0e+00 8.0e-01 0.0e+00 6.0e-01 0.0e+00 6.0e-01 0.0e+00
1.0e+00 0.0e+00 1.0e+00 2.0e-01 1.0e+00 0.0e+00 1.0e+00 8.0e-01 8.0e-01 0.0e+00 8.0e-01 0.0e+00
3.3e-01 0.0e+00 3.3e-01 0.0e+00 3.3e-01 0.0e+00 3.3e-01 0.0e+00 6.6e-01 0.0e+00 6.6e-01 0.0e+00
0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00
```

Let's look at a few examples:

- On the branch above node 1 (ie. the leaf of the tree representing the first strain carriage1), at the first reference site, there is 60% of chance that a recombination happened, but 0% of chance that this resulted in a substitution.
- On the same branch but on the last reference site, there is 100% chance that a recombination happened, and 80% of chance that is resulted in a substitution.
- At the second reference site and on the branch above node 2 (ie. the left of the tree representing the second strain carriage2), there is 80% of chance that a recombination happened and 100% of chance that a substitution happened, which means that there is 20% of chance that a mutation happened.

7.4 Consensus information part

This part starts with the line `#consinfo`. It contains additional information about the consensus tree. For each node in the consensus tree, the first column indicates its reference number as used in the consensus tree of Section 7.1, the second one indicates the name of the isolates it correspond to in the input alignment (or 0 for an internal node), the third column indicates its average age and the last column indicates the average distance to its father.

In our example, this part looks as follows:

```
#consinfo
1 1 0.000000 0.037435
2 2 0.000000 0.055999
3 3 0.000000 0.041030
4 0 0.013890 0.059356
5 0 0.077025 0.000000
```

Here we see that the node with reference number one correspond in fact to the first strain of the first block (ie. carriage1), that its average age in the chain was 0.0 and the average age of its father was 0.037435.

7.5 MCMC part

This part starts with the line `#mcmc`. It contains only three numbers, which are the length of the MCMC, the length of the burn-in period and the thining interval between two measures. These numbers correspond to the parameters used to run the chain and provided by the option `-x`, `-y` and `-z` respectively.

In our example, this part looks as follows:

```
#mcmc
100
50
10
1.000000
1.000000
2.718282
2.718282
2.718282
```

This indicates that the chain was run for 100 iterations plus 50 burn-in iterations, with a thining interval of 10 (ie. the program was run with the options `-x 100 -y 50 -z 10`). The prior on ν was Beta(1.0,1.0), which is default but could be specified using the `-f` and `-g` options. Finally, the logarithms functions for the log-uniform priors of θ, R and δ all used the default base of $e = 2.718282$, which is default but could be specified using the `-f`, `-g` and `-i` options.

7.6 Topologies part

This part starts with the line `#phy`. It contains the list of topologies taken by the MCMC. There is a line for each recorded iteration of the MCMC. Each line contains a tree in Newick format. The numbers used for the leaves of these trees are the same as for the consensus tree.

In our example, this part looks as follows:

```
#phy
((1:0.000000,3:0.000000):0.179537,2:0.000000):2.523944
((1:0.000000,2:0.000000):0.011917,3:0.000000):0.201797
((1:0.000000,2:0.000000):0.245128,3:0.000000):0.403253
(1:0.000000,(2:0.000000,3:0.000000):0.112972):0.154433
(1:0.000000,(2:0.000000,3:0.000000):0.024046):0.040212
((1:0.000000,3:0.000000):0.018968,2:0.000000):0.022107
(1:0.000000,(2:0.000000,3:0.000000):0.013318):0.056893
((1:0.000000,3:0.000000):0.007379,2:0.000000):0.030962
((1:0.000000,2:0.000000):0.088611,3:0.000000):0.150165
((1:0.000000,3:0.000000):0.015322,2:0.000000):0.125000
```

We can follow the evolution of the chain using this part:

- The algorithm started with 1 and 3 clustered together (line 1)
- Then branches were swapped so that 1 and 2 clustered together (lines 3 and 4)
- Then 2 and 3 clustered together (lines 4 and 5)
- Then 1 and 3 clustered together (line 6)
- Then 1 and 2 clustered together (line 7)
- Then 1 and 3 clustered together (line 8)
- Then 1 and 2 clustered together (line 9)
- Finally the branches were swapped back to get 1 and 3 clustered together (line 10)

The fact that in the second half (after the burn-in, ie. lines 6 to 10) 3 out of 5 genealogies (on lines 6, 8 and 10) have 1 and 3 clustered together explains why they are clustered together in the consensus tree of Section 7.1.

7.7 Reference sites part

This part starts with the line `#poly`. It contains a vector indicating the location of the reference sites on the first (reference) isolate. The reference sites are the sites at the beginning or end of each block, the polymorphic sites and all sites that are at a distance from the start of the block they belong to which is a multiple of 50.

In our example, this part looks as follows:

```
#poly
1
6
7
10
12
16
```

This means that the reference sites are the 1st, 6th, 7th, 10th, 12th and 16th sites on the genome of the first strain. Since we did not provide any information about the location of the fragments in the input file, the program placed them next to one another at the beginning of the first genome.

7.8 Likelihood part

This part starts with the line `#ll`. It contains the list of values taken by the log-likelihood during the run of the MCMC.

In our example, this part looks as follows:

```
#ll
-2.208693e+01
-1.652939e+01
-1.626068e+01
-2.477261e+01
-2.218586e+01
-1.830525e+01
-2.225068e+01
-1.925356e+01
-2.228668e+01
-1.768745e+01
```

7.9 Blocks part

This part starts with the line `#blocks`. It contains a list of numbers indicating the beginning and end of blocks of alignment, in number of reference sites. For example the vector 2,3,2 indicates that the first two reference belong to a block, the next three to another block and the last two to a third block.

In our example, this part looks as follows:

```
#blocks
4
2
```

This means that the first 4 reference sites (1, 6, 7 and 10) are on the first block whereas the last 2 (12 and 16) are on the second one.

7.10 theta, nu, delta and R parts

These parts start with the lines `#theta`, `#nu`, `#delta` and `#R` respectively. All these parts contain a vector which indicates the values taken by the parameters θ, ν, δ and R during the run of the MCMC.

In our example, these parts look as follows (8 values were replaced in four places by the symbol `[...]` for clarity):

```
#theta
1.024949e+00
8.357529e-01
```

```
[...]
#nu
8.623121e-02
6.464725e-02
[...]
#delta
9.555236e-05
9.953685e-05
[...]
#R
9.683404e-01
9.080195e-01
[...]
```