ARBORES – INSTRUCTIONS

KARI HEINE, MARIA DE IORIO, ALEX BESKOS, AJAY JASRA, DAVID BALDING

1. Overview

Arbores[1] is a Markov chain Monte Carlo (MCMC) algorithm for simulating ancestral recombination graphs (ARG's) from a Bayesian posterior distribution for given DNA polymorphism data.

Usage:

Mac: `./Arbores data n_iter mu rho seed output_folder initialisation`
Win: `ArboresWin64 data n_iter mu rho seed output_folder initialisation`

Description:

`data`	Name of the data file (see details in Section 2)
`n_iter`	Number of MCMC iterations. This is the number of bridge segment or time jittering iterations. The number of iterations where all bridge segments and times are sampled once is smaller by amount that depends on the number of observed sequences and the length of the observed sequences.
`mu`	Mutation rate parameter (mutations per site per effective population size generations)
`rho`	Recombination rate parameter (recombinations per site per effective population size generations)
`seed`	Integer seed for the random generator.
`output_folder`	Name of the folder where results are stored. A subfolder with given name is created (unless it already exists) in the current folder, i.e. the folder with the executable.
`initialisation`	OPTIONAL argument: the name of the file that provides initialisation for the MCMC chain (see details in section Initialisation)

The binaries are compiled and tested on macOS Sierra 10.12.1 and Windows 7 Enterprise 64-bit.

2. Data format

Data must be provided in a text file containing a space (' ') separated array, each row having an equal number of entries and each row ending with a new line after the last numeric entry (i.e. space cannot be the final character).

The first row contains integer site indices. The subsequent rows contain the DNA polymorphism data represented as zeros and ones separated by exactly one space character. An example of a valid data file is provided with the program (`testset_9_10_a.txt`).

The data must, in addition, satisfy the following conditions:

(1) The first data row (i.e. the 2nd row of the file) must contain only zeros.
(2) Site indices must be in increasing order.
(3) The distance between sites must be at least 2 sites.
(4) Indexing must start from 1.
(5) Each column must have at least two ones.

3. Output

The program generates the following files in the subfolder `output_folder`:

```
chain.txt
diagnostics.txt
map.tex
mrca.txt
```

`chain.txt`: Each row represents the state of the MCMC chain after the completion of all time jittering and bridge segment proposal steps. Because `n_iter` is the number of chain iterations with time jittering and bridge segment proposals considered as individual iterations, the number of rows in `chain.txt` is less than `n_iter`, by amount that depends on the length of the observed sequences, number of observed sequences and randomness in the algorithm, so an explicit expression for the number of rows cannot be provided.

Let N_l be the number of observed sequences. Each row then contains the following data as a space (' ') separated list:

Column index	Content
1	N_l
$2 \dots 2(N_l-1)+1$	Integers that define a coalescent tree structure.
$2(N_l-1)+2 \dots 3(N_l-1)+1$	Times of the coalescences in a coalescent tree.
$3(N_l-1)+2$	Site of the coalescent tree.
$3(N_l-1)+3 \dots 3(N_l-1)+4$	Parameters of the recombination as a subtree-prune-and-regraft operation (index of the pruning and regrafting node).
$3(N_l-1)+5$	Time of the recombination.

The above pattern of $3(N_l - 1) + 5$ numeric entries is repeated on the same row as many times as there are recombinations in the current state of the chain. For example, if there is only one recombination, then the above pattern appears twice. If there are 7 recombinations the patter appears 8 times. Each occurrence of this pattern represents a coalescent tree and all entries, except for the 1st (number of leaves) entry may chance from one occurrence to another.

To study the `chain.txt` file, the executable `ChainReader` (Mac) or `ChainReaderWin64` (Win64) is provided. This command line tool takes two arguments `filename` and `row`. `filename` is the name of a file with identical format as `chain.txt`. `ChainReader` reads the line specified by `row` (row indexing starting from zero) and generates a standalone tex-file `state.tex`. For more details about the usage of this tex-file, see below the description of `map.tex` file. An example usage of `ChainReader`:

Mac: `./ChainReader chain.txt 256`
Win: `ChainReaderWin64 chain.txt 256`

`diagnostics.txt`: The diagnostics file contains diagnostics data for each iteration. Time jittering and bridge segment iterations are considered individually and thus the number of rows in `diagnostics.txt` is equal to the `n_iter` provided as an input argument.

Each row contains the following data as a space (' ') separated list:

Row	Description
1	acceptance probability
2	current free time density
3	proposed free time density
4	current log likelihood
5	proposed log likelihood
6	current log prior density
7	proposed log prior density
8	proposed number of free times
9	current number of free times
10	proposed number of recombinations
11	current number of recombinations
12	acceptance indicator (0 = reject, 1 = accept)
13	jittering step indicator (0 = bridge segment step, 1 = time jittering step)
14	log likelihood after accept/reject decision
15	log prior density after accept/reject decision
16	log posterior (up to proportionality) after accept/reject decision
17	sampling set cardinality ratio
18	proposed recombination time density
19	current recombination time density

`map.tex`: The program writes a tex-file that represents the maximum a posteriori (MAP) ARG structure with coalescence and recombination times averaged over all iterations with the corresponding MAP structure.

The tex-file is ready to compile with the following preamble that has to supported by the LaTeX compiler:

```
\documentclass{standalone}
\usepackage{tikz}
\usetikzlibrary{shapes,arrows,positioning,shapes.geometric}
\usepackage{ifthen}
```

Also, to compile the file, the tex-file `phytree.tex` is needed. This file is provided with the software. Note also that `map.tex` has similar structure as the `state.tex` generated by `ChainReader`. Therefore same conditions for compiling the file apply to `state.tex`.

After compilation, the resulting picture represents the MAP ARG as a decomposition of coalescent trees. The first genomic site index where the tree appears, is given below each tree (i.e. if site 1234 is written below the tree, then a recombination occurs between the sites 1233 and 1234). Recombinations are illustrated by red arrows. The starting point of the arrow is the recombination time and the next tree after the recombination contains a new node at the time where the red arrow ends. Times are measured in the number of effective population size generations.

The resulting trees also contain two indices for each non-leaf node. The indexing below the node is the time ordered indexing within a tree, and the index above the node on the right hand side is a global indexing in the tree sequence. Each non-leaf node has a unique global index.

`mrca.txt`: The file contains times to the most recent common ancestor (MRCA) for each pair of observed sequences. Time to the MRCA is defined as the minimum over the times to the MRCA for given pair of leaf nodes in each coalescent tree. Times are given for each complete iteration (after iterating all time jittering and bridge segment steps once). Each row corresponds to one iteration and times are space (' ') separated.

Each column in `mrca.txt` corresponds to a specific sequence pair which can be determined as follows: the time to MRCA of the observed sequences (i, j), where $i = 0, ..., N_l - 1$ and $j = i + 1, ..., N_l - 1$ is reported on column $(\sum_{k=1}^{i} N_l - k) + j - i - 1$, with the column indexing starting from 0.

4. Initialisation

Initialisation argument (`initialisation`) is optional. If one wants to continue an previously executed run from the state where the previous run was terminated, this is possible by providing the optional `initialisation` argument. Initialisation is provided as a name of the file which is present in the same folder with the executable. File must consist of one row of data which is of the same format as the `chain.txt` file. Thus, the easiest way to generate a valid initialisation file is to copy the last row (or any other row where the

simulation is to be resumed at) of previously generated `chain.txt` file and save the file in the same folder with the executable.