September 20, 2018

KimTree version 2.0.1

User Manual

Contents

1 Overview

The software package KimTree implements a hierarchical Bayesian model to estimate divergence times in a population tree, from allele count data at many single-nucleotide polymorphisms (SNPs) (Gautier and Vitalis, 2013). In KimTree, the allele frequencies are modelled along each branch of a specified tree, using Kimura's time-dependent diffusion approximation for genetic drift (Kimura, 1964). The joint analysis of autosomal and X-linked polymorphisms allows KimTree to infer the effective sex ratios or ESR (defined as the female proportion of the effective population), along each branch (Clemente *et al.*, 2018). KimTree is written in C programming language. The source code is available under the GNU General Public License (see http://www.gnu.org/licenses/gpl-3.0.en.html). Once compiled, the KimTree binary reads data files supplied by the user, and a number of options can be passed through the command line. This document provides information on how to format the data files, how to specify the user-defined parameters, and how to interpret the results.

2 Before you start

2.1 How to get KimTree?

Download the archive from http://www1.montpellier.inra.fr/CBGP/software/kimtree/, and extract it from a terminal:

```
tar -xzvf kimtree_2.0.1.tar.gz
```

Binaries for OS X, Windows, Linux are not available. Therefore, you need to recompile KimTree from the source files provided (see the next subsection).

2.2 How to compile KimTree?

The source files are available in the `src` subdirectory. KimTree is coded using C programming language and can therefore be compiled for any system supported by gcc. To do so, Windows users may need to get a gcc compiler, e.g. by installing MinGW, mingw-64, or Cygwin. To compile the code and get the `kimtree` binary, use the Makefile provided:

```
make clean all
```

KimTree uses OpenMP to implement multithreading, which allows parallel calculation on computer systems that have multiple CPUs or CPUs with multiple cores. The gcc version included with OS X may generate executable code

that results in runtime error ("Abort trap: 6") when more than one thread is used. In that case, you first need to install a recent version of gcc, following the instructions in http://hpc.sourceforge.net/. Then, you can recompile KimTree using (assuming `gcc` has been installed in `/usr/local/`):

```
make clean all CC=/usr/local/bin/gcc
```

3 Underlying principles of KimTree

KimTree is a method designed to estimate divergence times on a diffusion time scale from large single-nucleotide polymorphism (SNP) data sets, conditionally on a population history represented as a multifurcating tree.

3.1 The data

By default, the data consist of allele counts at many SNPs, collected from individuals sampled in a set of populations. KimTree can also handle pooled-population genotyping data, using the `-pool` option. In that case, the data consist of read counts, collected from population pools. Given the allele frequencies in each pool, the conditional distribution of read counts is then assumed to be binomial (see Günther and Coop, 2013; Gautier, 2015).

When X-linked polymorphisms are available, KimTree provides estimate of effective sex ratios or ESR (defined as the female proportion of the effective population) for each branch of the rooted tree topology that summarizes the history of the populations of interest. To do so, the X-linked data are provided in an additional file, using the `-Xfile` option.

3.2 The population genetics model

The method is based on a diffusion approximation for the distribution of allele frequencies in multiple populations, whose demographic history is represented as a multifurcating tree. We recommend you read carefully the details of the models in Gautier and Vitalis (2013) and Clemente *et al.* (2018). Consider a sample of populations sharing a common history, represented as a tree. At each locus, assuming Hardy-Weinberg Equilibrium (HWE) in each sampled population, the conditional distribution of the observed count of the reference allele (which is arbitrarily defined) is binomial, given the sample size and the (unknown) allele frequency. In the absence of mutation, assuming that branch i with effective size $N_{e,i}$ diverged from its ancestor for t_i discrete non-overlapping generations, the distribution of the allele frequency in the ith branch of the tree, conditional upon the allele frequency in the parental

population and the branch length $\tau_i \equiv t_i / (2N_{e,i})$, is given by Kimura's time-dependent diffusion approximation (see Eqs 4.9 and 4.16 in Kimura (1964)). The integration over the allele frequencies along all the branches of the tree is then achieved by means of a hierarchical Bayesian model, which is detailed in Gautier and Vitalis (2013) and Clemente *et al.* (2018).

In Gautier and Vitalis (2013), the prior distribution of the frequency π_j of the reference allele in the root population followed a beta distribution Beta$(1.0, 1.0)$. In Clemente *et al.* (2018), the model has been improved in several directions. First, KimTree has been extended to estimate the hyper-parameters of the Beta(α, β) prior for allele frequencies in the root population. Estimating the hyper-parameters of the beta distribution allows for a more flexible allele frequency distribution at the root, potentially shifting the total age of the tree. This option is set by default. However, the user may fix the values of the parameters α and β, as in Gautier and Vitalis (2013), using the option `-fixed_beta`, in which case their values is set using the options `-beta_a` and `-beta_b`, respectively (by default, $\alpha = \beta = 0.7$). Second, the model has been extended to account for the fact that the dataset consists, by construction, of polymorphic sites only. In SNP datasets, indeed, sites that are fixed across the entire sample have been filtered out. This is a non-trivial issue, since the fraction of sites that are monomorphic in the sample, but were polymorphic in the root population, contains information on the branch lengths. Ignoring this information may therefore result in biased estimates of the branch lengths. This is set by default, but may be unset using the option `-unascertained`. Last, the model was extended to jointly analyze allele frequencies from both autosomal and X-linked markers and provide estimates of the ESR for each branch of the tree. In that case, X-linked data must be provided using the option `-Xfile` to specify the filename.

3.3 The framework for statistical inference

The framework for statistical inference from this model consists in a hierarchical Bayesian model (see Gelman *et al.*, 2004), for which the directed acyclic graph (DAG) is shown in Figure 1. KimTree is based on a componentwise Markov chain Monte Carlo (MCMC) algorithm to sample from the joint posterior distribution of the model parameters. Some parameters of the MCMC algorithm can be adjusted by the user. In particular, proposal distributions are adjusted during short pilot runs, in order to get acceptance rates between 0.25 and 0.40 (see, e.g., Gilks *et al.*, 1996). After the pilot runs, a burn-in period may be defined, before samples are drawn from the Markov chain. Then, samples are taken from the chain, with the number

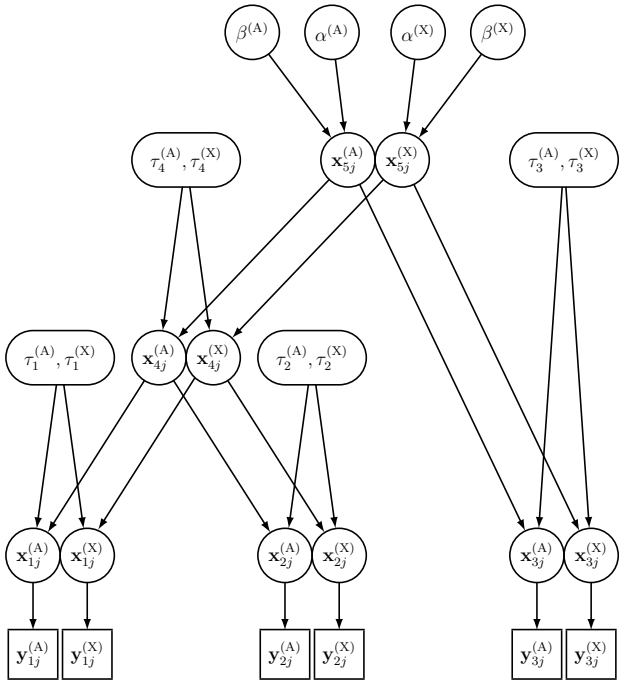


Figure 1: **Directed acyclic graph (DAG) of the hierarchical Bayesian model for a three-population example tree.** This graph represents the most complete model with both autosomal (A) and X-linked (X) data. Because all parameters and data are specific to one or the other genetic system, we use the index Ω ($\Omega \in \{\mathrm{A}, \mathrm{X}\}$). With autosomal data only, the DAG is simplified by removing all parameters and data with index (X). The square nodes characterize the data, i.e. $y_{ij}^{(\Omega)}(\Omega \in \{\mathrm{A}, \mathrm{X}\})$ represents the observed allele counts from autosomal and X-linked data in population i at SNP j. The circles and rounded rectangles represent the parameters to be estimated: $x_{ij}^{(\Omega)}$ is the (unknown) allele frequency in population i; $\tau_i^{(\Omega)}$ is the length (in a diffusion time scale) of the branch leading to population i; $\alpha^{(\Omega)}$ and $\beta^{(\Omega)}$ are the shape and scale parameters of the beta distribution, which describes the allele frequency distribution π_j in the root population. Unidirectional edges (arrows) represent direct stochastic relationships within the model. They indicate the conditional dependency between connected nodes. If the `-pool` option is set, then the data consist in read counts, that depend upon the (unobserved) allele counts. If the option `-fixed_beta` is set, then the parameters α and β of the (beta) prior distribution of π_j are not estimated.

of iterations between any two samples set by the thinning interval. This is aimed at reducing autocorrelation between successive values of the parameters along the Markov chain. Typically (as set by default) 100,000 updating steps are completed after 25 short pilot runs of 1,000 iterations each and a burn-in of 25,000 steps. All the model parameters are sampled every 25 steps (thinning), yielding 5,000 observations.

4 Using KimTree

4.1 Input files format

4.1.1 Input tree

In KimTree, the topology needs to be specified a priori. The topology is encoded an oriented tree (see, e.g., Kelleher *et al.*, 2016). An oriented tree is a sequence of integers $\pi_1, \pi_2, \ldots, \pi_u$, such that π_u is the parent node of u and u is the root if $\pi_u = 0$. In KimTree, we further assume that the sampled populations (leaf nodes) are mapped to the integers $1, \ldots, n$. For every internal node u, we have $n < u < 2n$. By convention, the last integer in the sequence is the root node of the tree. Therefore, for a strictly bifurcating tree, the $2n-2$ non-zero entries occur at $u = 1, \ldots, 2n-2$, and the root is $2n-1$ (i.e., $\pi_{2n-1} = 0$). For example, a tree formatted as $((1,2),3)$ in Newick format would read:

```
--- file begins here ---
4  4  5  5  0
--- file ends here ---
```

A star-shaped tree formatted as $(1,2,3,4,5)$ in Newick format would read:

```
--- file begins here ---
6  6  6  6  6  0
--- file ends here ---
```

A more complex tree formatted as $(((1,2),3),(4,5,6))$ in Newick format would read:

```
--- file begins here ---
7  7  8  9  9  9  8 10 10 0
--- file ends here ---
```

These example oriented trees are represented in Figure 2. These figures can easily be obtained using the R function `draw.tree()` from the `KimTree.R`

file in the `R` subdirectory of the archive. The `draw.tree()` function can therefore be used beforehand, in order to check any tree topology considered for KimTree analyses.

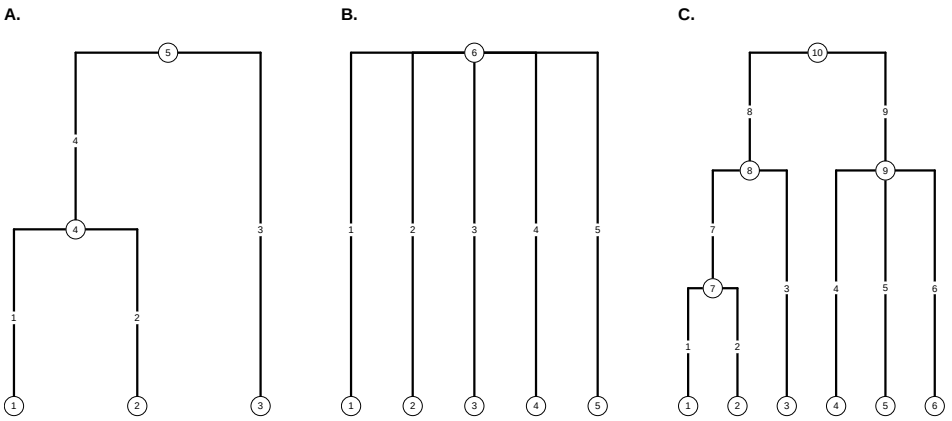


Figure 2: **Example oriented trees.** From left-to-right, these trees are defined by the sequences $\langle 4, 4, 5, 5, 0 \rangle$ (A), $\langle 6, 6, 6, 6, 6, 0 \rangle$ (B) and $\langle 7, 7, 8, 9, 9, 9, 8, 10, 10, 0 \rangle$ (C). The node numbers are indicated within circles, and the branch numbers (that correspond to the numbers given in KimTree output files) are indicated at the midpoint of each branch.

4.1.2 Allele count data (by default)

The data file reads as follows:

```
--- file begins here ---
6
100
81 19 86 14 2 98 8 92 32 68 23 77
89 11 81 19 9 91 1 99 27 73 27 73
89 11 91 9 11 89 15 85 77 23 80 20

[...97 more lines...]

--- file ends here ---
```

In this example, there are 6 populations (the first number in the file), and 100 SNPs (the second number in the file). Each line that follows corresponds

to one SNP. The number of columns is twice the number of populations. Each pair of numbers corresponds to the allele counts in one population. For example, at the first SNP, in the first population, there are 81 copies of the first allele, and 19 copies of the second allele. In the second population, there are 86 copies of the first allele, and 14 copies of the second allele, etc.

4.1.3 Read count data (using the `-pool` option)

The data file reads as follows:

```
--- file begins here ---
6
100
50 50 50 50 50 50
71 8 115 0 61 36 51 39 10 91 69 58
82 0 91 0 84 14 24 57 28 80 18 80
93 28 112 30 90 48 0 113 33 68 0 106

[...97 more lines...]

--- file ends here ---
```

In this example, there are 6 populations (the first number in the file), and 100 SNPs (the second number in the file). The size of each pool (expressed as a number of genes, i.e. twice the number of diploid individuals) is indicated in line 3. In the above example, each pool is made of 50 gene copies (25 diploid individuals). Each line that follows corresponds to one SNP. The number of columns is twice the number of populations. Each pair of numbers corresponds to the allele counts in one population. For example, at the first SNP, in the first population, there are 71 reads of the first allele, and 8 reads of the second allele. In the second population, there are 115 reads of the first allele, and 0 read of the second allele, etc.

4.2 Running KimTree

KimTree is a command-line executable. The ASCII hyphen-minus ("-") is used to specify options. As specified below, some options take integer or float values and some options do not. Here is an example call of the program:

```
./kimtree -threads 8 -file infile.dat -tree tree.dat
          -outputs example -thin 20 -npilot 5 -burnin 1000
          -length 10000
```

In this example run, some (autosomal) data would be read from the file `infile.dat`, the topology would be read from the file `tree.dat` and the outputs would be printed out in the `example/` subdirectory. 10,000 updating steps would be completed after 5 short pilot runs of 1,000 iterations each and a burn-in of 1,000 steps. The calculations would use 8 threads. Samples would be collected for all the model parameters every 20 steps (thinning), yielding 500 observations. All the options are detailed below, in § 4.6, and the list of output files is provided in § 4.7.

4.3 Sanity checks

4.3.1 Assessing convergence

We advise to assess convergence, e.g., by computing the multivariate extension of Gelman–Rubin's diagnostic (Brooks and Gelman, 1998) on independent Markov chains. The Gelman–Rubin's diagnostic is based on the computation of the ratio of the pooled-chains variance over the within-chain variance. The Gelman–Rubin's diagnostic can be calculated using the coda package (Plummer *et al.*, 2006), as implemented for R (R Core Team, 2017), using the traces of the τ_i and the (α,β) parameters that are printed out in the `trace_tau.out` file and the `trace_beta.out` file, respectively.

4.3.2 Checking mixing properties

Also, we strongly recommend assessing the mixing properties of the MCMC by inspecting the trace of the parameters in the `trace_xxx.out` files. The trace shall show relatively good mixing (reasonably low autocorrelation, AND random variation around a stationary value). The autocorrelation can be measured using the `coda` package (Plummer *et al.*, 2006), as implemented for R (R Core Team, 2017). Otherwise, you may want to increase the length of the burn-in period and/or the total length of the Markov chain. KimTree also reports the effective sample size (ESS) for various parameters in the `logfile.log` output file. The ESS is a measure of how well a Markov chain is mixing. The ESS represents the number of effectively independent draws from the posterior distribution that the Markov chain is equivalent to (then the ESS must be compared to the chain length). Low ESS (due to strong autocorrelation) indicates poor mixing of the Markov chain.

4.4 Interpreting the results

Bayesian inference is based on evaluation of the posterior distribution. The posterior distributions of the model parameters can be plotted using the `trace_xxx.out` output files. Also, the mean and standard deviation of the model parameters are saved in the `summary_xxx.out` output files.

Because the tree topology is generally unknown, we implemented a model choice procedure to characterize, for any given dataset, the strength of evidence for alternative population histories. Following Gautier and Vitalis (2013), we used the deviance information criterion (DIC), which is a standard criterion for model selection Spiegelhalter *et al.* (2002). The `DIC.out` output file contains the value of the DIC. Models with smaller DIC should be preferred to models with larger DIC.

4.5 Worked example

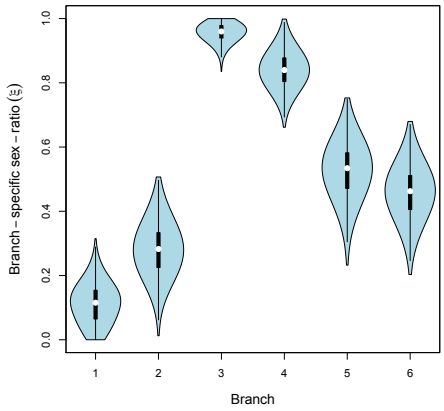
In the following, it is assumed that the current (working) directory is at the root of the `kimtree_2.0.1.tar.gz` archive, that contains several files and subdirectories (`data/`, `man/`, `R/` `src/`). From a terminal, execute the following command line:

```
./src/kimtree -npilot 20 -lpilot 500 -burnin 10000
              -length 20000 -thin 20 -file /data/auto.dat
              -Xfile /data/chrx.dat -tree /data/test.tre
              -threads 8 -outputs run-example/
```

In this example run, the autosomal data are read from the `data/auto.dat` file, the X-linked data from the `data/chrx.dat` file, and the tree topology from the `data/test.tre` file. The outputs are printed out in the `run-example/` subdirectory. The data consist in a simulation of a four-population tree with topology ((1,2),(3,4)). The root population was made of 50,000 males and 50,000 females, and the internal branches correspond to populations made of 5,000 males and 5,000 females. Branch 1 was made of 1,000 females and 9,000 males ($\xi_1 = 0.1$); branch 2 was made of 2,000 females and 8,000 males ($\xi_2 = 0.2$); branch 3 was made of 9,000 females and 1,000 males ($\xi_3 = 0.9$); branch 4 was made of 8,000 females and 2,000 males ($\xi_4 = 0.8$). The two successive splits occurred 1,000 and 3,000 generations before present time. The mutation rate was fixed at $\mu = 1.5 \times 10^{-7}$. 50 females were sampled per population, and genotyped at 5,000 autosomal SNPs and 5,000 X-linked SNPs.

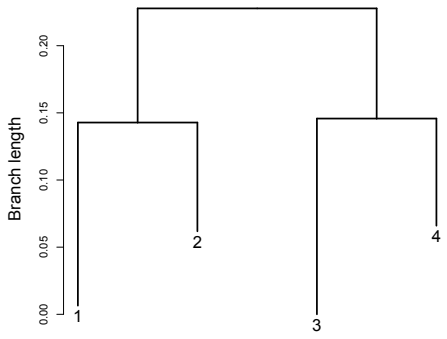
Once KimTree has been executed, you may analyze the results, using R. To do so, launch R, and set the working directory to the root of the archive, using the `setwd()` function. Then, you may represent the posterior distributions of the ESR for each branch using, e.g.:

```
> library(vioplot)
> xi <- read.table("trace_xi.out",header = TRUE)
> plot(1,xlim = c(0.75,6.25),ylim = c(0,1),
  ylab = expression(Branch-specific~sex-ratio~(xi)),
  xlab = "Branch",type = "n",cex.lab = 1.25)
> vioplot(xi$deme_1,xi$deme_2,xi$deme_3,xi$deme_4,
  xi$deme_5,xi$deme_6,col = "lightblue",add = TRUE)
```



Alternatively, summaries (mean and standard deviation) of the model parameters are saved in the `summary_xxx.out` output files. Also, one can use the R function `draw.tree()` from the `KimTree.R` file in the `R` subdirectory of the archive, to plot the tree topology using the estimated values of the branch lengths:

```
> source(R/KimTree.R)
> draw.tree(tree_file = "data/test.tre",n = 4,
  summary_tau_file = "outputs/summary_tau_auto.out",
  leafs = TRUE,nodes = FALSE,edges = FALSE)
```



Last, the `DIC.out` file contains the value of the deviance information

criterion (DIC) Spiegelhalter *et al.* (2002), which may be used to choose between alternative histories.

4.6 Details of KimTree options

`-help`

This option prints out the full list of options accepted by KimTree, i.e.:

```
usage: ./kimtree [ options ]
valid options are :
-help               print this message
-version            print version
-file               name of the input data file (default: data.dat)
-Xfile              name of the input file for X-linked data
-tree               name of the input tree file (default: data.tre)
-outputs            directory where the outputs will be produced (default: current directory)
-seed               initial seed for the random number generator (default: computed from current time)
-threads            number of threads to be used (default: number of cpu available)
-length             run length of the Markov chain (default: 125000)
-thin               thinning interval size (default: 25)
-burnin             length of the burn-in period (default: 25000)
-npilot             number of pilot runs (default: 25)
-lpilot             length of each pilot run (default: 1000)
-pool               option to analyse data from pooled DNA samples (default: unset)
-unascertained      option to compute the likelihood for unascertained data (default: unset)
-fixed_beta         option to fix the shape parameters of the beta prior distribution of pi (default: unset)
-beta_a             shape parameter (a) of the prior for allele frequencies alpha_ir's in the root node (default: 0.70)
-beta_b             shape parameter (b) of the prior for allele frequencies alpha_ir's in the root node (default: 0.70)
-dlt_cnt            half window width from which updates of allele counts are randomly drawn (default: 5)
-dlt_frq            half window width from which updates of alpha are randomly drawn (default: 0.25)
-dlt_tau            half window width from which updates of tau are randomly drawn (default: 0.025)
-dlt_beta_mu        half window width from which updates of the beta mu parameters are drawn (default: 0.025)
-dlt_beta_nu        standard deviation of the lognormal distribution from which updates of the beta nu parameters are
                    drawn (default: 1.00)
-verbose            option to print the traces of all parameters (generates big output files!)
```

`-version`

This option prints out the KimTree version.

`-file`

This option gives the name of the input file. If the option is not specified, the input file name is "data.dat".

`-Xfile`

This option gives the name of the input file for X-linked data.

`-tree`

This option gives the name of the input file for the tree topology.

`-outputs`

This option gives the directory where all the outputs will be saved. If the option is not specified, then all the output files will be saved in the current directory (where `kimtree` is executed).

`-seed`

This option gives the initial seed (integer) for the random number generator. If the option is not specified, then the initial seed is computed from the current computer time. Note that because KimTree code is parallelized, two different runs with the same initial seed may provide different sequences of random numbers, hence different outputs.

`-threads`

This option gives the number of threads to be used. If the option is not specified, then all available cpu are used.

`-length`

This option gives the total length of the MCMC (i.e., the number of iterations run after the burn-in period). By default, `length` = 50000 (i.e., `-length 50000`).

`-thin`

This option gives the size of the thinning (i.e., the number of iterations between any two records from the MCMC). By default, `thin` = 25 (i.e., `-thin 25`).

`-burnin`

This option gives the length of the burn-in period (i.e., the number of iterations before the first record from the MCMC). By default, `-burnin` = 10000 (i.e., `-burnin 10000`).

`-npilot`

This option gives the number of pilot runs (i.e., the number of runs used to adjust the parameters of the MCMC proposal functions, to get acceptance rates between 0.25 and 0.40). By default, `-npilot` = 25 (i.e., `-npilot 25`).

`-lpilot`

This option gives the length of each pilot run (i.e., the number of iterations for each run). By default, `-lpilot` = 1000 (i.e., `-lpilot 1000`).

`-pool`

This option enables the analysis of pooled-population genotyping data (see § 3.1). By default, this option is not set.

`-unascertained`

This option cancels the computation of the conditional likelihood model. Using this option (which is unset by default), the model does not account for the exclusive presence of polymorphic markers in SNP datasets.

`-fixed_beta`

This option is used to fix the shape parameters (α and β) of the (beta) prior distribution of π_j. By default, this option is not set.

`-beta_a`

If the `-fixed_beta` option is set, then this option is used to set the shape parameter α of the (beta) prior distribution of π_j. By default, α = 0.7 (i.e., `-beta_a 0.7`).

`-beta_b`

If the `-fixed_beta` option is set, then this option is used to set the shape parameter β of the (beta) prior distribution of π_j. By default, β = 0.7 (i.e., `-beta_b 0.7`).

`-dlt_cnt`

This parameter gives the initial value of Δ_y, which is half the window width from which updates of allele counts y'_{ij} are drawn uniformly around the current value y_{ij}. The value of Δ_y is eventually adjusted, for each locus in each deme, during pilot runs to get acceptance rates between 0.25 and 0.40. By default, $\Delta_y = 5$ (i.e., `-dlt_cnt 5`).

`-dlt_frq`

This parameter gives the initial value of Δ_x, which is half the window width from which updates of allele frequency p'_{ij} are drawn uniformly

around the current value p_{ij}. The value of Δ_x is eventually adjusted, for each locus in each deme, during pilot runs to get acceptance rates between 0.25 and 0.40. By default, $\Delta_x = 0.25$ (i.e., `-dlt_frq 0.25`).

`-dlt_tau`

This option gives the initial value of Δ_τ, which is the standard deviation on the log scale of the lognormal distribution (with median equal to the current value M_i) from which updates of parameters Δ_τ are drawn. The value of Δ_τ is eventually adjusted, for each deme, during pilot runs to get acceptance rates between 0.25 and 0.40. By default, $\Delta_\tau = 0.025$ (i.e., `-dlt_tau 0.025`).

`-dlt_mu`

If the `-fixed_beta` option is not set, then the parameters of the (beta) prior distribution of π_j are updated. To that end, we follow Kruschke (2011) and parameterize the beta distribution using $\alpha = \mu\nu$ and $\beta = (1-\mu)\nu$. The `-dlt_mu` option gives the initial value of Δ_μ, which is half the window width from which updates of the mean allele frequency μ' are drawn uniformly around the current value μ. The value of Δ_μ is eventually adjusted during pilot runs to get acceptance rates between 0.25 and 0.40. By default, $\Delta_\mu = 0.025$ (i.e., `-dlt_mu 0.025`).

`-dlt_nu`

If the `-fixed_beta` option is not set, then the parameters of the (beta) prior distribution of π_j are updated. To that end, we follow Kruschke (2011) and parameterize the beta distribution using $\alpha = \mu\nu$ and $\beta = (1-\mu)\nu$. The `-dlt_nu` option gives the initial value of Δ_ν, which is the standard deviation on the log scale of the lognormal distribution (with median equal to the current value ν) from which updates of the parameter ν' are drawn. The value of Δ_ν is eventually adjusted during pilot runs to get acceptance rates between 0.25 and 0.40. By default, $\Delta_\nu = 0.5$ (i.e., `-dlt_nu 0.5`).

`-verbose`

This option is used to print the traces of all parameters in different output files. This may generate very large output files. By default, this option is not set, and only the traces of the τ_i, the ξ_i and the (α,β) parameters are printed out.

4.7 Format of the output files

KimTree produces several output files:

`logfile.log`

contains all the information that is printed on the console during execution

`diag_mcmc.log`

contains the log(likelihood) and the log(post. density) along the chain (second and third columns) and the acceptance rates for each category of parameters, i.e. (depending on the data and the options): the allele counts n_{ij}, the allele frequencies x_{ij}, the branch lengths τ_i, possibly the parameters μ and ν of the (beta) prior distribution of π_j.

`summary_beta_xxx.out`

contains the mean and standard deviation (std) of the shape parameters α and β of the (beta) prior distribution of π_{rj} in the root node. If X-linked data are provided, two files are produced: `summary_beta_auto.out` for autosomal data (A) and `summary_beta_chrx.out` for X-linked data (X). Otherwise, only the `summary_beta.out` file is written.

`summary_counts_xxx.out`

contains the mean and standard deviation (std) of the allele counts x_{ij}, if the `-pool` option is set, for each locus in each deme. If X-linked data are provided, two files are produced: `summary_counts_auto.out` for autosomal data (A) and `summary_counts_chrx.out` for X-linked data (X). Otherwise, only the `summary_counts.out` file is written.

`summary_freq_xxx.out`

contains the mean and standard deviation (std) of the allele frequencies x_{ij}, for each locus in each deme. If X-linked data are provided, two files are produced: `summary_freq_auto.out` for autosomal data (A) and `summary_freq_chrx.out` for X-linked data (X). Otherwise, only the `summary_freq.out` file is written.

`summary_tau_xxx.out`

contains the mean and standard deviation (std) of the τ_i parameter (lenght) for each branch of the tree. If X-linked data are provided,

two files are produced: `summary_tau_auto.out` for autosomal data (A) and `summary_tau_chrx.out` for X-linked data (X). Otherwise, only the `summary_tau.out` file is written.

`summary_xi.out`

contains the mean and standard deviation (std) of the ξ_i parameter (effective sex-ratio) the for each branch of the tree.

`trace_beta_xxx.out`

contains the values of the shape parameters α and β of the (beta) prior distribution of π_{rj} along the Markov chain. If X-linked data are provided, two files are produced: `trace_beta_auto.out` for autosomal data (A) and `trace_beta_chrx.out` for X-linked data (X). Otherwise, only the `trace_beta.out` file is written. These files may be useful to check for convergence using, e.g., the CODA package in R.

`trace_counts_xxx.out`

contains the value of the allele counts y_{ij} along the Markov chain. This file is only printed out if the `-verbose` option is set. If X-linked data are provided, two files are produced: `trace_counts_auto.out` for autosomal data (A) and `trace_counts_chrx.out` for X-linked data (X). Otherwise, only the `trace_counts.out` file is written.

`trace_freq_xxx.out`

contains the value of the allele frequencies x_{ij} along the Markov chain. This file is only printed out if the `-verbose` option is set. If X-linked data are provided, two files are produced: `trace_freq_auto.out` for autosomal data (A) and `trace_freq_chrx.out` for X-linked data (X). Otherwise, only the `trace_freq.out` file is written.

`trace_tau_xxx.out`

contains the value of τ_i along the Markov chain. If X-linked data are provided, two files are produced: `trace_tau_auto.out` for autosomal data (A) and `trace_tau_chrx.out` for X-linked data (X). Otherwise, only the `trace_tau.out` file is written. These files may be useful to check for convergence using, e.g., the CODA package in R.

`trace_xi.out`

contains the value of the ξ_i parameter (effective sex-ratio) along the Markov chain. This file is only produced if X-linked data are provided.

`DIC.out`

contains the values of: the posterior mean deviance, $\bar{D}$, which can be interpreted as a Bayesian measure of fit); the Bayesian deviance evaluated at the posterior mean of the parameters, $D(\bar{\mathbf{\Theta}})$; the effective dimension of the hierarchical model, $p_D = \bar{D} - D(\bar{\mathbf{\Theta}})$; and the deviance information criterion (DIC), which is equal to $\big(2\bar{D} - D(\bar{\mathbf{\Theta}})\big)$.

5 Credits

KimTree uses Makoto Matsumoto and Takuji Nishimura's implementation of the Mersenne Twister random number generator, http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html.

6 Copyright

7 Contact

If you have any question, please feel free to contact me. However, I strongly recommend you read carefully this manual first.

Bibliography

Brooks, S., and A. Gelman, 1998 General methods for monitoring convergence of iterative simulations. Journal of Computational and Graphical Statistics 7: 434–455.

Clemente, F., M. Gautier, and R. Vitalis, 2018 Inferring sex-specific demographic history from SNP data. PLoS Genetics .

Gautier, M., 2015 Genome-wide scan for adaptive divergence and association with population-specific covariates. Genetics 201: 1555–1579.

Gautier, M., and R. Vitalis, 2013 Inferring population histories using genome-wide allele frequency data. Molecular Biology and Evolution 30: 654–668.

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin, 2004 *Bayesian Data Analysis*. Chapman & Hall, New York, 2nd edition.

Gilks, W. R., S. Richardson, and D. J. Spiegelhalter, 1996 *Markov Chain Monte Carlo in Practice*. Chapman & Hall, New York, 2nd edition.

Günther, T., and G. Coop, 2013 Robust identification of local adaptation from allele frequencies. Genetics 195: 205–220.

Kelleher, J., A. M. Etheridge, and G. McVean, 2016 Efficient coalescent simulation and genealogical analysis for large sample sizes. PLoS Comput Biol 12: e1004842.

Kimura, M., 1964 Diffusion models in population genetics. Journal of Applied Probability 1: 177–232.

Kruschke, J. K., 2011 *Doing Bayesian data analysis: A tutorial with R and BUGS*. Academic Press, Oxford.

Plummer, M., N. Best, K. Cowles, and K. Vines, 2006 Coda: output analysis and diagnostics for MCMC. R News 6: 7–11.

R Core Team, 2017 *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Spiegelhalter, D. J., N. G. Best, B. P. Carlin, and A. Van Der Linde, 2002 Bayesian measures of model complexity and fit. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 64: 583–639.