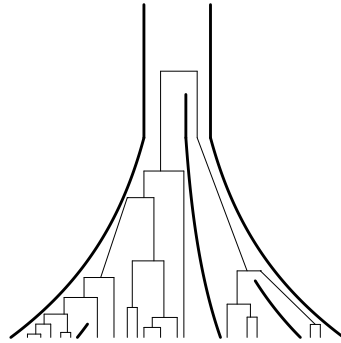


Batwing User Guide



Bayesian Analysis of Trees With Internal Node Generation.

Ian Wilson, David Balding and Mike Weale

Correspondence address:

Ian Wilson,
Department of Mathematical Sciences
University of Aberdeen
King's College,
Aberdeen, AB24 3UE, UK

Email: I.J.Wilson@ncl.ac.uk

BATWING Home Page: <http://www.mas.ncl.ac.uk/~nijw>

This guide gives more details of the program described in **Wilson, Weale & Balding 2003**. Inferences from DNA data: population histories, evolutionary processes and forensic match probabilities. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, **166**: 155-188.

COPYRIGHT NOTICE

©2002 by Ian Wilson, David Balding and Mike Weale. Permission is granted to copy this document provided that no fee is charged for it and that this copyright notice is not removed. Guide v1.03, Last Edited: 26 June 2003

Contents

1	Introduction	3
1.1	General description	3
1.2	Modelling assumptions	4
1.3	The Bayesian paradigm	4
2	The Batwing software	5
2.1	Downloading and Installation	5
2.1.1	Unpacking on Unix	5
2.1.2	Unpacking on Windows, and Macintosh	5
2.2	The Distribution	5
2.3	BATWING command line	6
2.4	Data settings	6
2.5	Population model settings	7
2.5.1	Population size	7
2.5.2	Population structure	7
2.6	Mutation model settings	7
2.6.1	STR (microsatellite) loci	7
2.6.2	UEP sites	8
2.7	Time scaling and parameterisation options	9
2.8	Prior distributions	10
2.9	MCMC control settings	12
2.10	Output	14
2.10.1	The newick file format	16
2.10.2	Post-processing	16
3	Data Simulation	17
4	Example session	17
5	Alphabetical listing of infile options	22

BATWING	Bayesian Analysis of Trees With Internal Node Generation
MCMC	Markov chain Monte Carlo
MRCA	Most recent common ancestor
SMM	Stepwise mutation model
SNP	Single nucleotide polymorphism
STR	Short tandem repeat (= microsatellite)
TMRCa	Time since MRCA
UEP	Unique event polymorphism

Table 1: Acronyms used in the text.

1 Introduction

1.1 General description

BATWING is a program written in C for the analysis of population genetic data, written by Ian Wilson, now at the University of Aberdeen. Its initial development was in collaboration with David Balding, now at Imperial College, London, and formed part of a research project funded by the UK EPSRC under the Stochastic Modelling in Science and Technology programme. BATWING is described in Wilson, Weale & Balding (*JRSS A* **166**: pp 155-188). Please send comments on this User Guide to I.J.Wilson@ncl.ac.uk.

BATWING reads in multi-locus haplotype data, and model and prior distribution specifications, and uses a Markov chain Monte Carlo (MCMC) method based on coalescent theory to generate approximate random samples from the posterior distributions of parameters such as mutation rates, effective population sizes and growth rates, and times of population splitting events. It also generates approximate posterior samples of the entire genealogical tree underlying the sample, including the tree height, which corresponds to the Time since the Most Recent Common Ancestor (TMRCa). BATWING does not model the effects of either recombination or selection, and hence implicitly assumes that these effects are negligible. Note also that BATWING is intended for within-species data, and not between-species data for which many phylogenetic software packages are available.

BATWING is a direct descendant of MICSAT, described in Wilson & Balding (*Genetics* **150**, pp 499-510, 1998). The principal differences are: (1) in addition to the stepwise mutation model (SMM) for microsatellite loci, there are mutation models for unique event polymorphisms (UEP); and (2) the population demography is extended from constant size, random mating, populations (the standard coalescent model), to include population growth and subdivision.

A manuscript giving detailed illustrations of BATWING-based analyses, “Inferences from DNA data: population histories, evolutionary processes, and forensic match probabilities”, by Wilson, Weale, and Balding, is currently under review for publication; a preprint is available from any of the authors.

1.2 Modelling assumptions

Underlying BATWING are various modelling assumptions, many of which are also assumptions underpinning coalescent theory (see e.g. Nordborg 2001). In the case that the data are drawn from a single population, two key assumptions are that the data form a random sample from the population, and that the population is random mating. For *structured* data, drawn from k subpopulations, BATWING assumes a common, random-mating ancestral population which split into two isolated random-mating subpopulations. If $k > 2$ then one or both of these subpopulations split again, and so on until k random-mating subpopulations existed at the time of sampling.

Three models are available for the population size: (0) constant, (1) pure exponential growth, and (2) exponential growth from a constant ancestral population size. For structured data, the start-of-growth time in model (2) is the same across all subpopulations, and there is also a common growth rate under models (1) and (2).

Natural populations are unlikely to satisfy BATWING’s modelling assumptions. In particular, BATWING assumes that population splitting events are instantaneous, with no subsequent migration, whereas in reality splits may be gradual and followed by migration. However, more realistic models will almost certainly have too many parameters for useful inferences to be drawn. We have tried to find a satisfactory compromise between reality and tractability, so that the most important features of the data are exploited to quantify the major underlying demographic events. It must be recognised that some questions of interest about historic demography cannot be answered from present-day genetic data alone.

1.3 The Bayesian paradigm

BATWING implements the Bayesian paradigm for statistical inference. Most scientists have been trained in the classical paradigm, which dominated 20th century science. The Bayesian approach is older, having been the predominant mode of inference in the 19th century, but has returned to prominence in recent years, due in large part to the availability of fast computing, and algorithms such as MCMC.

A key feature of the Bayesian paradigm is that a probability distribution is associated with all the unknowns. The investigator specifies both a *prior* distribution, representing knowledge about the unknowns before the present data are taken into account, and a *likelihood* function which assigns probabilities to the data given values for the unknowns. Bayes Theorem can then be employed to convert the prior distribution and likelihood into a *posterior* distribution.

Information used in formulating the prior may come from previous studies of related systems, theory, and informal judgement. Sometimes scientists choose to downweight or ignore available prior information, and formulate a diffuse prior, giving support to a wide range of values for the unknowns. Although there is not usually a unique prior distribution, and hence not a unique posterior, computational power is now available to explore a range of priors: often the

posterior is dominated by the likelihood, so that the posterior distribution is essentially invariant over a wide range of priors. If not, the investigator learns the crucial information that his/her data are only weakly informative about some parameters of interest, and hence background information must be chosen with care and accurately reported in any publication.

BATWING offers a range of probability distributions from which investigators can choose priors for mutation rates, effective population sizes and growth rates, and population splitting times (see Section 2.8). For unknowns which are properties of the genealogical tree, such as the TMRCA, BATWING offers several models from coalescent theory, outlined above in Section 1.2 and in more detail below in Section 2.5. Note that we speak of coalescent *models*, even though they specify *priors* for the unknown genealogical tree. Although model and prior are usually thought of as distinct, the distinction is not fundamental and is a matter of custom and convenience.

2 The Batwing software

2.1 Downloading and Installation

The main distribution, and the only available for unix systems, is in the file `batwing.tar.gz`, a zipped tar file. The first step is to download this file which is available at <http://www.mas.ncl.ac.uk/~nijw>. If you work under Windows or Macintosh (MacOS 8.0 or above) systems, executable files `batwing.exe` and `MacBatwing.hqx` are also available at the same site (but `batwing.tar.gz` is still needed for documentation and other files).

2.1.1 Unpacking on Unix

The downloaded file, `batwing.tar.gz`, is unpacked using these command

```
gunzip -c batwing.tar.gz | tar xf -
```

on a command line. To produce the executable file type `make` (or alternatively on some systems `gmake`). This should make the `batwing`, `sample` and `prior` programs and put them in the `bin` subdirectory.

2.1.2 Unpacking on Windows, and Macintosh

Under these systems any decompression software, such as `winzip` should unpack `batwing.tar.gz`.

2.2 The Distribution

The distribution when unpacked will extract the files into a directory `batwing` with subdirectories `bin` (executable files), `data` (data files), `examples` (example files), `doc` (documents, including this Guide), `sample` (files for programs to sample from the model) and `src` and `R` (source files). To run the programs from a command line you will need to add the `bin` directory to your path, or copy the program into the directory with your input files.

In this document we use **this font** to indicate BATWING parameters and filenames, and *this font* for variable names. File listings use a fixed-width font. The first line of a file listing gives the path to the file in the distribution, to enable readers to find the file in the BATWING distribution. The `#` symbol is used to denote a comment, which is ignored by BATWING.

2.3 Batwing command line

The unix and windows versions of BATWING are command line driven. (although BATWING can be run by double-clicking under a Windows environment). Syntax:

```
batwing infile outfile <seed>
```

The `<>` notation indicates that `seed` is optional; `infile` and `outfile` are required and will be prompted for if not supplied.

infile The input filename; the default `infile` is assumed if not specified, e.g. if BATWING is run by double-clicking `batwing.exe` in a Windows environment. Can contain settings for: (1) data file path; (2) models and prior distributions and (3) MCMC control parameters, such as the number of outputs and the number of update steps between outputs. All settings are made via the syntax `name: value` on a separate line. Default values exist for many settings, see Sections 2.4 to 2.9 and Section 5.

outfile The output filename stem; default `out`. Various output files are created with this stem and different extensions: `outfile.par` contains a copy of the `infile` information and information about MCMC acceptance rates; `outfile.iit`, `outfile.end`, and `outfile.x` (where *x* is an integer) give states of the genealogical tree and other parameters visited by the MCMC algorithm, while `outfile` (without an extension) gives the parameter values output by the algorithm. See Section 2.10

seed Seed for the random number generator, default value 1. Can also be specified in the `infile` but a command line seed takes priority.

2.4 Data settings

The location of the observed data is specified by a line of the form

```
datafile: <<path/>filename>
```

in the `infile`. If `path/` is omitted the current directory is assumed, otherwise the path should be specified relative to the current directory; default setting is `datafile`.

Data files have one line per haplotype, with one or more spaces separating the alleles at distinct loci. Following `#`, the remainder of the line is ignored (the whole line is ignored if `#` is the first non-space character).

First come the UEP alleles which may be coded by any two single alphanumeric characters (e.g. “0” and “1”, or “A” and “T”). Next come the microsatellite, or STR (= short tandem repeat), alleles, coded by an integer value giving

the number of tandem repeats at that locus (a constant added to each allele length at a locus has no effect on inferences; also, arbitrary numeric allele labels are allowed under the k -alleles model, Section 2.6.1). Missing STR data can be specified using `-1`.

If the data are drawn from several distinct populations (i.e. `migmodel=1`, see Section 2.5.2) the positive integer label specifying the source subpopulation is stored in a file `locationfile`, in the same way as for `datafile`. Rows of the `locationfile` should correspond to the rows of the `datafile`. Subpopulation codes may be any positive integers. Missing location information can be specified using `-1`; if this is used the total number of subpopulations must be assigned to `Npopulations` in the `infile`.

If you would like different labels for the Newick trees that are output from BATWING, then one can set the option `labelfile`. The contents of the `labelfile` should be the names for each of the samples, and these are then output as part of the trees.

The option `popnames`, allows you to set the names of the different populations to get more friendly output. *Not done yet.*

2.5 Population model settings

2.5.1 Population size

The population growth model is specified via `sizemodel` which can be assigned one of three values (default 0):

- 0 Constant (effective) population size N (chromosomes); in this case BATWING implements the standard coalescent model.
- 1 Pure exponential growth at rate α to a current population size N .
- 2 Exponential growth at rate α from a constant-size ancestral population of size N , with growth starting at (scaled) time β before present.

2.5.2 Population structure

Two options are available for population structure. Setting `migmodel=0` specifies that the data are drawn from a single population (the default). If `migmodel=1` then the subpopulation from which a haplotype in `datafile` is drawn must be specified by a positive integer on the corresponding row of `locationfile`, with `-1` indicating missing data. If there are missing data then `Npopulations` must be assigned the number of subpopulations. If no data are missing, `Npopulations` is set automatically to the number of distinct codes used in `locationfile`.

2.6 Mutation model settings

2.6.1 STR (microsatellite) loci

The default mutation model is the SMM, under which the repeat number changes by one at each mutation event, with decreases and increases being

equally likely. The default prior distribution on the ancestral allele size is uniform on the positive integers. This is an improper prior, but always leads to a proper posterior distribution. Although negative allele lengths are not excluded under our model, positive observed states and a positive ancestral state means that intermediate negative states are extremely unlikely.

An alternative is the k -allele mutation model, under which a mutant allele is equally likely to take any one of the k possible states, irrespective of the previous state. To specify this model, **kalleles** should be assigned a list (separated by spaces) of the numbers of alleles possible at each locus, the loci having the same order as in **datafile**.

Under either SMM or k -alleles model, by default all loci have the same mutation rate; this default can be explicitly set by assigning **locustypes** the integer 1. If **locustypes** is assigned an integer equal to the number of STR loci, there is a distinct mutation rate for each locus. The only other setting permitted for **locustypes** is a list of positive integers whose sum is the number of STR loci. If for example there are 11 STR loci and the list is “3 6 2”, then there are three distinct mutation rates, one common to the first three loci, one shared by the next six loci, and one shared by the final two loci. Clearly, the loci have to be ordered so that loci with the same mutation rate are neighbouring.

2.6.2 UEP sites

Unique Event Polymorphism (UEP) sites are polymorphic sites at which only two alleles are observed and the investigator assumes that a single historical mutation event was responsible for the observed polymorphism. These may include insertion/deletion polymorphisms, such as an *Alu* insertion, or single-nucleotide polymorphisms (SNP). The number of UEP sites should be assigned to **infsites** (default: 0).

There is an ascertainment problem common to many genetic data types but which is often particularly acute for UEP data. Such sites may have been included in a genetic survey because they were known in advance to be polymorphic; possibly they were known to be highly polymorphic in several populations. Inferences which may validly be drawn from a site ascertained in this way may differ substantially from valid inferences had exactly the same data been observed at a site which was chosen “at random”.

To allow some investigation of ascertainment effects BATWING allows three ways to incorporate UEP sites into the analysis, selected by setting **infotype** (default is 0):

- 0 Conditions on there being a single mutation at each UEP site. Hence UEP positions on the tree contribute to the tree likelihood, and the posterior density, but no inference is drawn about the mutation rate.
- 1 Assumes the same UEP mutation rate for all UEP loci, with a uniform prior. UEP positions within the tree contribute to the tree likelihood and posterior density.
- 2 Only trees consistent with the UEP data are permitted, but UEP data are not used in any other way.

For insertion/deletion polymorphisms, it may be reasonable to assume that the ancestral state is known. For SNP sites modelled as a UEP, the ancestral state is typically unknown. If the parameter `ancestralinf` is assigned a UEP haplotype, this is taken to be the haplotype at the root node (i.e. the haplotype of the MRCA at the UEP loci). To use this feature, UEP alleles must be specified in the same way both in the `datafile` and for `ancestralinf`. Currently, an ancestral state can be assigned either to all or none of the UEP loci.

2.7 Time scaling and parameterisation options

BATWING follows the coalescent theory convention of working with times (such as the TMRCA, or a population split time) expressed in units of N generations. Time 0 corresponds to the present, and large positive times correspond to far back in the past. The value of N used by BATWING for the time scaling is the *current* effective population size (N_c) under model 1, but the *ancestral* effective population size (N_a) under model 2. The reason for this choice is that N_a is not defined under model 1, whereas under model 2 it is N_a that has the most important effect on the data; N_c is realised only instantaneously.

BATWING users can choose to work either with the unscaled growth rate *alpha* and mutation rate *mu*, or the scaled growth rate *omega* = $N \times \textit{alpha}$ and scaled mutation rate *theta* = $2N \times \textit{mu}$ (the “2” appears in the definition of *theta* for historical reasons). A feature of working with scaled rates and times is that the likelihood under BATWING’s modelling assumptions does not depend on N , which may thus be eliminated from any analyses. This reduction in the number of unknowns may lead to more precise inferences about the remaining unknowns. However, if N is unknown then interpretation of the results is severely limited because times and rates cannot be expressed in terms of generations or years.

For this reason it is often preferable to work with unscaled rates, together with explicit assignment of prior and posterior distributions to N . A scaled time output by BATWING can then be converted into an unscaled time (in generations) via multiplication by the prevailing value of N (a further multiplication by generation time leads to a time in years). Note, however, that inferences about unscaled parameters usually depend more sensitively on the prior specification than do inferences for the scaled parameters. For example, when the sample size is very large the data may convey substantial information about $N \times \textit{mu}$ so that the posterior for *theta* may be effectively independent of the prior, but how this information is allocated to N and *mu* separately may depend sensitively on their joint prior distribution.

A further parameterisation option is offered by BATWING under growth model 2, where users can choose to work either with a (scaled or unscaled) growth rate or with *kappa* (the natural logarithm of current to ancestral population sizes). Since

$$N_c = N_a e^{N_a \times \textit{alpha} \times \textit{beta}},$$

it follows that

$$\textit{kappa} = N_a \times \textit{alpha} \times \textit{beta} = \textit{omega} \times \textit{beta}.$$

Population size model	code	Parameter						
		N	μ	θ	α	ω	κ	β
0	U	*	*					
	S			*				
1	U	*	*		*			
	S			*		*		
2	U	*	*		*			*
	U'	*	*				*	*
	S			*		*		*
	S'			*			*	*

Table 2: Parameterisations available within BATWING under the different models for population size (see Table 3 for brief definitions of the parameters). Users must choose one of the parameterisation options and assign a prior distribution to each of the parameters indicated by a * in the corresponding row. The code U denotes that N is included as a parameter and hence inferences may be obtained for unscaled rates and times; S denotes that only inferences about scaled rates and times are possible. Under model 2, the ' denotes that κ is included instead of a growth rate parameter.

The parameterisation options available within BATWING under the three population size models are summarised in Table 2. An option is selected by assigning a prior distribution to the corresponding parameters (see Section 2.8 below). A brief summary of all BATWING parameters that may need to be assigned a prior distribution is given in Table 3.

2.8 Prior distributions

The distributions supported by BATWING for univariate parameters are:

`uniform< (v1,v2) >`; uniform on the interval $(v1, v2)$; default interval is $(0, \infty)$, which specifies an improper prior.

`constant(v1)`, or just $v1$.

`normal(v1,v2)`; mean = $v1$; SD = $v2$.

`lognormal(v1,v2)`; if X has this distribution then $\log(X)$ has the `normal(v1,v2)` distribution.

`gamma(v1,v2)`; mode = $(v1-1)/v2$, mean = $v1/v2$.

All the parameters within one of the parameterisation options shown in Table 2 must be assigned a prior distribution from the above list. In addition, in models with population structure, *split* and *prop* must also be assigned a prior. The latter is a multivariate parameter, and the only prior distribution available is the `dirichlet(v1,v2,...,vn)`. The case $v1 = v2 = \dots = vn = 1$ gives the (multivariate) uniform distribution. The case $v1 = v2 = \dots = vn = 2$ is the default prior. When $n = 2$ the `dirichlet` is also known as the beta distribution.

Parameter	Description and comments
N	Effective population size (haploid); current (N_c) under model 1, ancestral (N_a) under model 2
mu	Mutation rate; a list of prior distributions is required if <code>locustypes</code> $\neq 1$
$theta$	$theta = 2N \times mu$
$alpha$	Unscaled growth rate (per generation); only positive values allowed; if <code>migmodel</code> =1, all subpopulations have the same growth rate
$omega$	$omega = N \times alpha$
$kappa$	$kappa = \log_e(N_c/N_a) = N_a \times alpha \times beta$
$beta$	$beta$, the time at which population growth starts
$split$	Time of the first population split
$prop$	Proportion of the total population size taken up by each subpopulation; <code>dirichlet</code> is only permitted prior family and <code>dirichlet(2, 2, \dots, 2)</code> is the default

Table 3: Summary of the BATWING parameters that may need assignment of a prior distribution, according to the parameterisation option selected from those given in Table 2. The final two parameters are used whenever `migmodel`=1.

Prior distributions for scaled genealogical times are assigned implicitly via choice of the demographic model; for example, under model 0 (the standard coalescent) the prior TMRCA when the sample size is two has the exponential distribution with expectation and SD both equal to 1 coalescent unit; as the sample size gets large, the prior expectation and SD of the TMRCA approach 2 and 1.08 units, respectively.

The prior for unscaled genealogical times is assigned implicitly by choice of demographic model and the choice of prior for N ; the prior expectation of the unscaled TMRCA (in generations) of a large sample is approximately twice the prior expectation of N .

To assign a prior distribution, append “**prior**” to the parameter name shown in Table 3, followed by a colon, space, and the distribution, all on one line of the `infile`. For example,

```
thetaprior: gamma(20,2)
```

assigns a gamma prior to $theta$. The prior mean is 10 and the prior mode is 9.5.

If the ancestral states at the UEP loci are known, then these values can be fixed in the BATWING analyses via `ancestralinf`. For example,

```
ancestralinf: 00000
```

assigns 0 to the ancestral state of the five UEP loci. If `ancestralinf` is not set, the two states at each locus are *a priori* equally likely.

2.9 MCMC control settings

The starting state of the genealogical tree and other parameters may be chosen automatically by BATWING, or a starting tree may be specified in Newick format (see Section 2.10.1) using an `initialfile` assignment in the `infile`. The latter option is useful for resuming a BATWING run which has been interrupted either by the user or by a computer failure.

`initialfile` contains complete information on an instance of a tree from a previous MCMC run – such as those output as `outfile.iit`, `outfile.end`, or as `outfile.x` (where `x` is an integer) – these files consist of information about the tree in Newick format and information about the other parameters and the population tree (if appropriate). New MCMC settings can be specified via the `infile` as usual. If `seed` is specified in the `infile`, it takes precedence over the `initialfile` settings (and a command line `seed` takes precedence over an `infile` setting). All other settings in the `infile` (e.g. `priors`) are over-ruled by the settings specified in the `initialfile`.

If the initial tree is not specified, it is generated by BATWING, with the `badness` parameter controlling the suitability of this tree for the data: 1 specifies a random tree, chosen independently of the data, while 0 specifies a tree obtained via a parsimony heuristic, with each coalescence occurring between the two nodes with the most similar haplotypes. A value between 0 and 1 allows a mixture between these two extremes. See Wilson & Balding (1998) for more details.

BATWING allows `warmup` to be set in the `infile`. However, this merely specifies an additional number of outputs to be generated. No portion of a BATWING run is automatically discarded: the user must explicitly choose how many of the initial outputs to discard as burn-in. Therefore, use of `warmup` is optional, and if used a warning message is generated.

One important diagnostic tool for an MCMC run are the plots of the autocorrelation function (ACF) of the output values, both for the parameters of interest and for the log-likelihood. Ideally, the ACFs should be as small as possible. However, it is usually not worthwhile investing in reducing the ACF provided that the values decrease approximately monotonically to zero, and become negligible at lag much less than the square root of the number of outputs. The two BATWING parameters controlling the “thinning” of the BATWING output, and hence the ACF, are `Nbetsamp`, which gives the number of changes that are made to model parameters between sampling occasions, and `treebetN`, the number of changes to the tree attempted between changes in the model parameters. We make more changes to the tree than to the model parameters because tree changes are computationally cheap.

The appropriate balance between `Nbetsamp` and `treebetN` requires experimentation, and depends on the number of loci and the sample size. The effects of varying them are illustrated by the three `infile` shown at the top of Figure 1. Resulting autocorrelation plots for *theta* and *L*, up to lag 30, are shown at the bottom of the figure. Increasing `treebetN` at the cost of a proportional reduction in `Nbetsamp` has a slightly deleterious effect on the autocorrelations, while decreasing computation times by about 25%. All the ACF shown in the

```
#../examples/thinning/thin1.in
datafile: ../../data/examples/ex1.data
treebetN: 2
Nbetsamp: 50
```

```
#../examples/thinning/thin2.in
datafile: ../../data/examples/ex1.data
treebetN: 5
Nbetsamp: 20
```

```
#../examples/thinning/thin3.in
datafile: ../../data/examples/ex1.data
treebetN: 10
Nbetsamp: 10
```

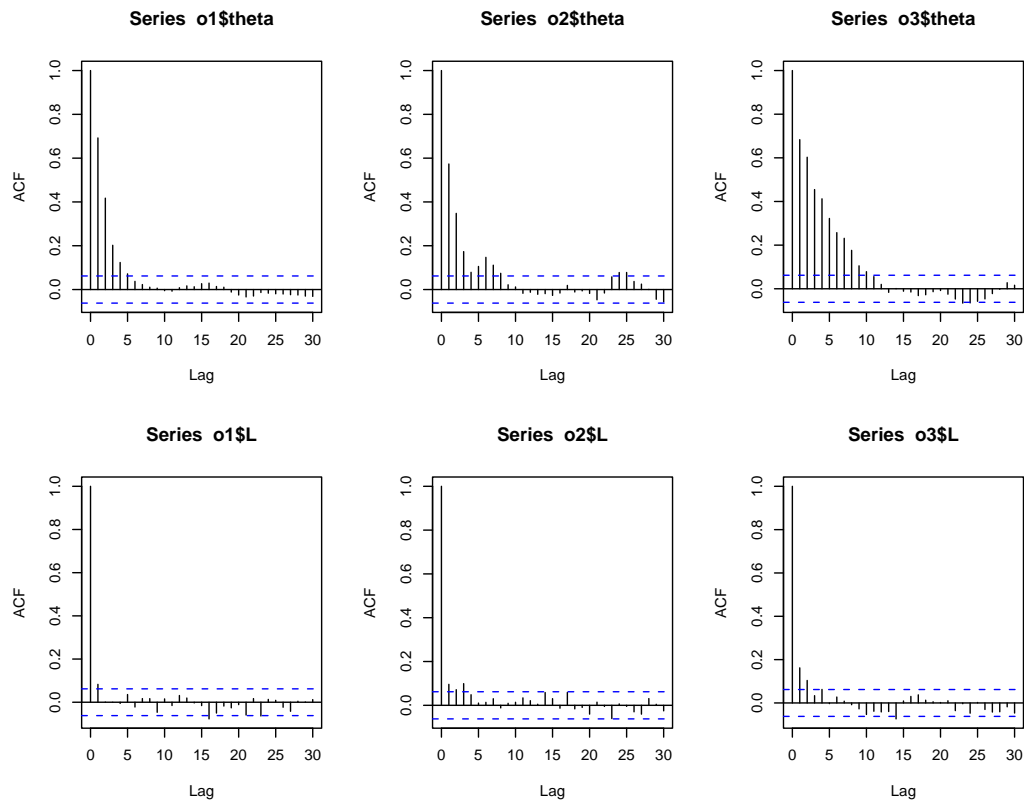


figure would usually be regarded as acceptable (the number of outputs is the default 1000).

2.10 Output

BATWING generates a text file **outfile**, of which each line after the first contains a space-separated list of floating point or integer numbers giving the values of the unknowns at one instant of the BATWING run. The first line of **outfile** gives names of the output variables, which we now briefly describe:

1. **ll_times**, the log-likelihood of the coalescent times of all interior nodes in the genealogical tree.
2. (if there are STR loci) **ll_mut**, the log-likelihood of all the STR mutation events that occur between all nodes in the genealogical tree.
3. (if **infsites** > 0) **ll_inf**, the log-likelihood of all the UEP mutation events that occur between all nodes in the genealogical tree. If **inftype** is set to 2 in **infile**, the value 0 is always output (see 2.6.2)
4. **ll_allpriors**, the log-prior probability of the current values of all the model hyperparameters. This column is also a catch-all for all things not covered in the first 3 columns. For example, if population splitting is specified in the **infile**, then the log-likelihood of all population splitting times is also included here. The sum of the first 4 columns gives the log-posterior probability (up to a constant).
5. the STR mutation rate, either unscaled (*mu*) or scaled (*theta*). If separate rates have been specified for different STR loci using **locustypes**, then they all appear here in separate columns.
6. (if a prior has been assigned) *N*, the effective population size (N_c if **sizemodel**=1, N_a if **sizemodel**=2).
7. (if **inftype**=1) the UEP mutation rate.
8. *T*, the TMRCA, or time since the most recent common ancestor of the sample, in coalescent units (multiply by *N* to obtain TMRCA in generations).
9. *L*, the total branch length of the tree, in coalescent units.
10. (if **sizemodel** ≠ 0) the population growth rate, either unscaled *alpha* or scaled *omega*.
11. (if **sizemodel**=2) *beta*, the time at which growth starts.
12. (if **sizemodel**=2) $kappa = N_a \times alpha \times beta = omega \times beta$.
13. (if **infsites** > 0) the next **infsites** columns give the binary states of the UEP loci at the root node of the tree (will always equal **ancestralinf** if set).

14. (if `migmodel=1`) the next k columns, $k = \#$ subpopulations, give the relative subpopulation sizes.
15. (if `migmodel=1`) the next $2(k-2)$ columns are pairs of numbers for each population merging event, starting with the most recent one and ending with the penultimate one. The first number indicates which clades merge at this point. To interpret this, first convert the number into binary form with k digits in the order corresponding to the `locationfile` codes (with Pop 1 as the least significant digit and Pop k as the most significant digit); 1s indicate which populations are beneath this node. The second number of the pair gives the time of the merging event. After these $k-2$ pairs of columns, a final column gives the time of the final coalescence event.
16. (if `infsites > 0` and `UEPtimes=1`) the next `infsites` \times 2 columns are pairs of numbers for each UEP locus, giving the descendent and ancestral node times of the branch on which the UEP mutation occurred.
17. (if `countcoals=1`) the number of coalescences occurring more recently than the start of growth.

When BATWING has finished running, it creates an `outfile.par` text file with details of model settings used in the analysis and acceptance rates for the following Metropolis-Hastings proposals:

`cutjoin` : moving a node to somewhere else in the tree.

`times` : changing the time of a node.

`haplotype` : changing the STR haplotype of a node.

`splitprop` (if `migmodel = 1`): changing the subpopulation size proportions.

`splittime` (if `migmodel = 1`): changing one of the subpopulation split times.

`mu` : changing the value of the STR mutation rate.

`N` : changing the value of N (if used).

`alpha` (if `sizemodel \neq 0`): changing the value of *alpha* or *omega*.

`growth` (if `sizemodel = 2`): changing the time of start of growth.

`infroot` (if `infsites > 0`): changing the MRCA UEP haplotype.

In addition to the `outfile.par` file, BATWING also generates a number of text files providing a detailed description the current state of the genealogical tree: `outfile.1` ... `outfile.x` describe the tree at regular intervals, where $x = \text{samples/picgap}$; `outfile.iit` and `outfile.end` describe the tree at the start and at the end of the process. Each of these files starts with a description of the current instance of the tree in newick format, suitable for viewing by “Treeview” and other packages. The information for each node includes its ID number, the UEP and STR haplotypes, and the time in coalescence units. Following the

newick tree description, there is information on the current state of the random number generator and values of the model parameters. This information allows a further BATWING run to continue from exactly the same place using the `initialfile` option in the `infile`. Following the hyperparameter information, a final line provides details of the current tree in exactly the same format as it would appear in a line of the `outfile`.

If `outroot` is set in the `infile`, by including a line

`outroot:`

in the `infile`, then the MRCA haplotype will be output. If `UEPtimes` is set then the minimum and maximum time at which each UEP mutation could have occurred are output. If `countcoals` is set the number of coalescences more recent than the start of growth is output.

2.10.1 The newick file format

The newick file format is a way of describing trees in computer files. A number of documents about it are available at:

http://evolution.genetics.washington.edu/phylip/newick_doc.html

<http://evolution.genetics.washington.edu/phylip/newicktree.html>

2.10.2 Post-processing

The use of Splus (commercial) or R (freely available at <http://cran.r-project.org>) is recommended for post processing of the output, however most simple analyses can be performed using spreadsheet software such as excel.

A set of S/R functions is available for reading and writing output from BATWING, and is included in the distribution as `R/batwing.R`. Appendix 1 of this document describes their use and appendix 2 gives shows their use for our example analysis.

It is recommended that analysis of output from BATWING is done using S-plus (commercial) or R (freely available from

<http://cran.r-project.org>

The file `batwing.r` in the distribution has functions for input and analysis of output from batwing – and organises the output into a more usable format – as well as producing tables for input into latex documents and word processors.

In addition the batwing R library available from <http://www.mas.ncl.ac.uk/~nijw> gives a set of functions for the post-processing of BATWINGoutput.

BATWING allows the output of trees from the posterior distribution, and in order to look at these a tree drawing package is needed. A number of programs are available for tree drawing, some are listed at:

<http://evolution.genetics.washington.edu/phylip/software.html>

These programs were designed for phylogenetic (species) trees, but can also be used for genealogical trees.

3 Data Simulation

The two programs SAMPLE and PRIOR, distributed with BATWING in the directory sample, can be used to simulate data for testing. They use an **infile** similar to that for BATWING, but without a **datafile** assignment. Instead they require **samplesize** to be assigned a list of sample sizes for the simulated data (if the list is of length one, an unstructured sample is simulated).

For SAMPLE we also need to assign a value to **nSTR**, the number of STR loci.

An optional parameter is **height**, which if assigned a positive real number *hval* results in a tree with approximate height *hval*, obtained by sampling trees until one is within 0.1% of *hval*.

The output files for SAMPLE are:

outfile a text file with data in a format for input into batwing.

<out>.data a text file includes the tree in newick format and includes more information on other parameters.

PRIOR gives output in the same form as BATWING, but is samples from the prior distribution only, ignoring any observed data.

4 Example session

This session is an example of a test session using BATWING, **sample**, and R and TreeView for postprocessing.

SAMPLE was used to simulate data using syntax:

```
sample samplein sampout
```

The input file **samplein** specifies 10 STR loci, no UEP loci, a sample size of 10 haplotypes (Figure 2, top).

Two output files are produced: **sampout**, which lists the simulated data (Figure 2, middle), and **sampout.data** which is a newick format file containing information about the total length of the tree and the height in the last line (Figure 2, bottom). The tree can be displayed using the program **treetool** (solaris) or **treeview** (windows) or other programs (Figure 3).

We analyse the data in **sampout** using BATWING with syntax:

```
batwing infile out
```

The **infile** (Figure 3, bottom) calls for 11 000 BATWINGparameter outputs, and 220 (= 11000/500) tree outputs. The file **out.par** repeats information from **infile**, together with MCMC acceptance rates (Figure 4). The file **out** consists of 11 000 rows each giving values for the six parameters: **lltimes**, **llmut**, **llprior**, **theta**, **T**, and **L** (Figure 4, bottom). The 220 tree outputs are given in files **out.x**, for $x = 1, 2, \dots, 220$.

We post-process the parameter values in **out** using either R or Spls using the commands

```

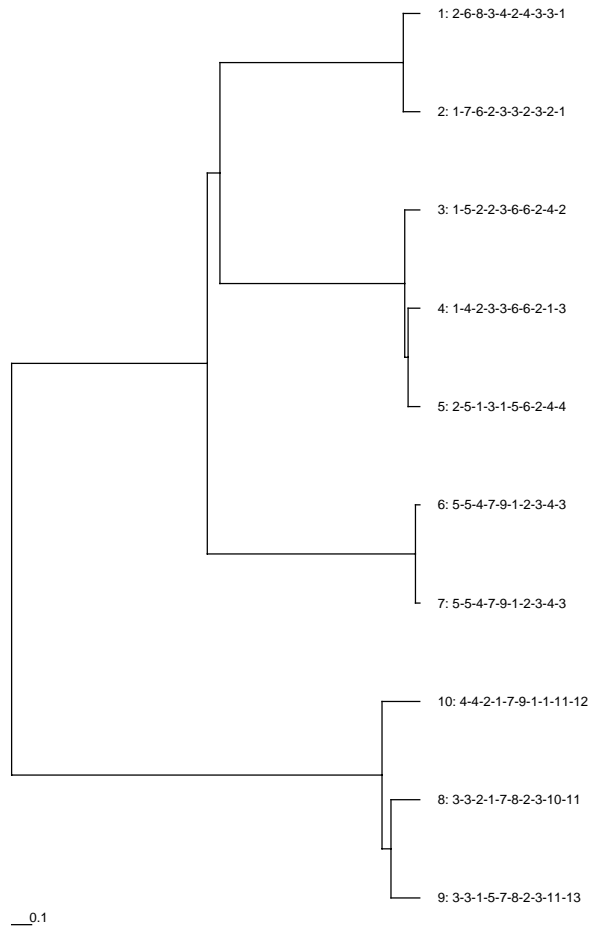
# examples\session1\samplein
# input file for example 1
samplesize: 10
theta: 10
nSTR: 10
seed: 16

# examples\session\sampout
# output file generated from above input
3 3 2 1 7 8 2 3 10 11
5 5 4 7 9 1 2 3 4 3
2 5 1 3 1 5 6 2 4 4
3 3 1 5 7 8 2 3 11 13
5 5 4 7 9 1 2 3 4 3
1 7 6 2 3 3 2 3 2 1
1 4 2 3 3 6 6 2 1 3
2 6 8 3 4 2 4 3 3 1
4 4 2 1 7 9 1 1 11 12
1 5 2 2 3 6 6 2 4 2

# examples\session\sampout.data
# output file generated from above input
((((('1: 2-6-8-3-4-2-4-3-3-1': 0.0820825,'2: 1-7-6-2-3-3-2-3-2-1': 0.0820825)'3:
2-6-7-3-4-2-3-3-3-1': 0.91005,('3: 1-5-2-2-3-6-6-2-4-2': 0.0751013,('4: 1-4-2-
3-3-6-6-2-1-3': 0.0587582,'5: 2-5-1-3-1-5-6-2-4-4': 0.0587582)'6: 2-5-2-3-3-6-6-
2-4-4': 0.0163431)'6: 2-5-2-3-3-6-6-2-4-4': 0.917031)'6: 4-5-3-2-5-3-3-2-6-5':
0.0627099,('6: 5-5-4-7-9-1-2-3-4-3': 0.0216645,'7: 5-5-4-7-9-1-2-3-4-3': 0.02166
45)'8: 5-5-4-7-9-1-2-3-4-3': 1.03318)'8: 5-5-3-3-4-3-3-1-6-5': 0.972001,('8:
3-3-2-1-7-8-2-3-10-11': 0.143024,'9: 3-3-1-5-7-8-2-3-11-13': 0.143024)'10: 3-
3-2-2-7-8-2-2-11-12': 0.0448991,'10: 4-4-2-1-7-9-1-1-11-12': 0.187923)'11: 3-3-
2-2-7-9-3-2-11-12': 1.83892)'11: 8-1-6-1-5-6-2-2-8-8';
summary
10 samples with 10 STR loci and 0 infinite sites.
theta 10
height 2.02684 length 6.66922

```

Figure 2: Listing of the `samplein` file used in the example (top), and the resulting `sampout` and `sampout.data` files (middle and bottom).



```
# \examples\session1\infile
# input file for analysis of sample output
datafile: sampout
tree_consensus: 1 #switch on consensus output
picgap: 500
samples: 10000
warmup: 1000
treebetN: 10
Nbetsamp: 20
thetaprior: uniform(0,100)
seed: 11
```

Figure 3: Tree output from treeview, using `sampout.data` file (top) and listing of `infile` for example (bottom).

```

# \examples\session1\out.par
# output file from example
datafile: sampout
outfile: out
infsites: 0

Analysis Details
=====
sizemodel: 0
migmodel: 0

Priors
-----
thetaprior: uniform(0,100)

Program Control Details
badness: 0.01
seed: 11
samples: 11000
Nbetsamp: 20
treebetN: 10
pop_consensus: 0
tree_consensus: 1

proportion accepted:
cutjoin 0.00898727
times 0.256097
haplotype 0.34313
theta: 0.0701045

# \examples\session1\out
# output file from example
lltimes llmut llprior theta T L
-5.85066 -258.997 -4.60517 27.7094 0.632908 2.82459
-4.40921 -266.008 -4.60517 60.3782 0.537683 2.36435
-5.01031 -255.684 -4.60517 47.6446 0.536507 2.51192
-3.6966 -242.024 -4.60517 38.9361 0.536507 2.157
...

```

Figure 4: Listing of output files `out.par` and `out` (first five lines) for example. See Figure 3 for `infile`.

```

> o <- read.table("out",header=T) # read in the data
> o <- o[-c(1:1000),] # remove the first 1000 rows as a burn in
> median(o[, "T"]) # calculate the median of T
> mean(o[, "theta"]) # calculate the mean of theta
> qres <- function(x)
> c(mean(x),median(x),quantile(x,probs=c(0.025,0.975)))
> # sets up a functions that calculates mean median and interval
> apply(o,2,qres)
> # applies function over all columns (see splus/r documentation).

```

note that the `#` symbol again indicates a comment.

obtain the means, medians and equal-tailed 95% posterior intervals shown in Table 4.

	mean	median	95% Interval
lltimes	-8.16	-7.86	(-14.45,3.702)
llmut	-246.0	-245.6	(-268.4,-225.6)
theta	16.83	15.31	(7.685,35.14)
T	1.172	1.566	(0.6643,3.65)
L	5.33	5.041	(2.34,10.11)

Table 4: Posterior 95% equal-tailed intervals obtained by post-processing the results in `out`.

5 Alphabetical listing of infile options

alphaprior *used:* see above *default:* none

The prior for alpha, the population growth rate, per generation. Currently, only positive values of alpha are allowed. If `migmodel = 1`, all subpopulations continue to grow at the same rate.

example– `alphaprior: gamma(1,100)`

ancestralinf *used:* when `infsites` $\neq 0$ *default:* NULL

If set, constrains the root node to have the UEP haplotype specified; value should be a list of space-separated 0/1 characters, with length = `infsites`.

example– `ancestralinf: 0 1 0 0 0 #5 UEP loci`

badness *used:* if `initialfile` not set *default:* 0.01

A real number used to obtain the initial genealogical tree of the MCMC chain. A value ≥ 1 means the initial tree joins the data sample nodes at random, without regard to their haplotype. A 0 value means that the first coalescence (going back in time) happens between the two most similar nodes, and so on. A value between 0 and 1 gives a compromise between these two extremes. See Wilson & Balding (1998) for more details.

example– `badness: 0.1`

betaprior *used:* see above *default:*

The prior for beta, the time at which population growth starts.

example– `betaprior: gamma(2,1)`

countcoals *used:* if `sizemodel=2` *default:* 0

This is a tool for investigating the convergence behaviour of the chain. If set to “1”, the number of coalescences more recent than the start of growth is included in each output.

example– `countcoals: 1`

infsites *used:* always *default:* 0

The number of UEP loci.

example– `infsites:`

inftype *used:* whenever we have UEP loci *default:* 0

Code for treatment of UEP loci. 0 = Use UEP data to condition permissible genealogical trees, but not to affect the tree likelihood or posterior density in any other way. 1 = assume the same UEP mutation rate for all UEP loci, with a uniform prior. UEP positions within the tree contribute to the tree likelihood and posterior density.

example– `inftype: 0`

initialfile *used:* optional *default:*
 The name of a `.iit` file containing complete information on an instance of a tree from a previous MCMC run. Used to set up all information on the data, model settings and prior setting as specified in the file, and it sets the random number seed to the same value that it was when the `.iit` file was saved. MCMC settings still can be specified in the `infile` (e.g. `samples`, `Nbetsamp`, `treebetN`, `warmup`, `seed`). If `seed` is specified in the `infile`, it takes precedence over the `initialfile` settings. If `seed` is specified in the command line, it takes precedence over the `infile` settings. All other settings in the `infile` (e.g. priors etc.) are over-ruled by the settings specified in the `initialfile`.
example– `initialfile:`

labelfile *used:* When we want to use alternative labelling of trees *default:* NULL
 Gives the filename of a file of labels for the leaves of the posterior coalescent trees. The file should contain the same number of rows as the `datafile`
example– `labelfile: e.label #` For a label file `e.label`

kalleles *used:* whenever we have STR loci *default:* NULL
 Determines whether a k-alleles model is used for mutations of the STR loci. Value is a list of the numbers of alleles permissible at each locus. Under the k-alleles model, a mutation causes a change to *any* of the k alleles (including the original allele) with equal probability. If not set, the SMM is used, in which a mutation causes the allele repeat number to increase or decrease by one unit with equal probability.
example– `kalleles: 4 4 4 4 #` for 4 STR loci

kappaprior *used:* see above *default:* none
 The prior for $\kappa = N_0 \times \alpha \times \beta$. *kappa* is the natural log of the ratio of current population size to initial population size.
example– `kappaprior: gamma(5,1)`

locustypes *used:* always *default:* 1
 If = 1, all STR loci have the same mutation rate. If = # STR loci, each STR locus has a distinct mutation rate. If = a list of integers whose sum = # STR loci, e.g. “3 6 2”, then the first 3 STR loci have the same mutation rate, then the next 6 have their own mutation rate, and then the final 2 have theirs.
example– `locustypes: 3 6 2`

meantime *used:* always *default:* 0
 If set to 1 the mean pairwise coalescence times between individuals in the sample is output.
example– `meantime: 1`

migmodel *used:* always *default:* 0
0 = No population substructure; 1 = samples drawn from subpopulations specified in **locationfile**

example- migmodel: .

muprior *used:* see above *default:* none
The prior distribution(s) for the STR mutation rate. Value is a list of distributions, of length according to value of **locustypes**.

example- muprior: gamma(1,1000)

Nbetsamp *used:* always *default:* 10
The number of times that changes to hyperparameters are attempted between outputs.

example- Nbetsamp: 10

npopulations *used:* if missing data in locationfile *default:* 0
Specifies the number of sub-populations. Normally, this is assigned automatically to the number of distinct values in **locationfile**.

example- npopulations: 3

Nprior *used:* see above *default:* none
The prior for N , the effective population size (N_c if **sizemodel**=1, N_a if **sizemodel**=2).

example- Nprior: lognormal(9,1)

omegaprior *used:* see above *default:* none
The prior for $N \times \alpha$.

example- omegaprior: uniform

outroot *used:* when **infsites** > 0 *default:* none
if this is set then the root haplotype is output.

example- outroot: 0

picgap *used:* always *default:* 100
The number of BATWING parameter outputs between each output of the tree to a **outfile.x** file.

example- picgap: 1 000

pop_consensus *used:* if there is a population tree *default:* 0
If this is set then an additional output file **outfile.pcsn** is output, giving a list of the population tree shapes in newick format. This can be used as input to a program like CONSENSE from Phylip to produce a consensus tree.

example- pop_consensus: 1

propprior *used:* if `migmodel = 1` *default:* Dirichlet(2, 2, ..., 2)

The prior for proportion of the total population size taken up by each geographic group. Must be Dirichlet.

example- `propprior: dirichlet(4,4,4)`

samples *used:* always *default:* 1 000

The number of outputs from a BATWING run (beyond any specified by `warmup`). Each output is written on a separate line in the `outfile`. N.B. `samples` has nothing to do with data sample size.

example- `samples: 2000`

seed *used:* always *default:* 1

A long unsigned integer specifying the random number generator seed. The order of precedence for seed assignment is: Command Line > `infile` > `initialfile` > default.

example- `seed: 123`

sizemodel *used:* always *default:* 0

Code for the population growth model: 0, constant population size; 1, exponential growth at rate *alpha* at all times; 2, exponential growth at rate *alpha* from a constant-size population, with growth starting at time *beta*.

example- `sizemodel: 2`

splitprior *used:* if `migmodel=1` *default:* none

The prior for the time of the first population split.

example- `splitprior: gamma(1,1)`

thetaprior *used:* see above *default:* none

The prior for $2N \times \mu$.

example- `thetaprior: uniform`

treebetN *used:* always *default:* 10

The number of times that changes to the genealogical tree are attempted before any changes to the hyperparameters are attempted. Thus BATWING outputs are separated by `treebetN` × `Nbetsamp` attempted tree updates.

example- `treebetN: 10`

tree_consensus *used:* if there is a population tree *default:* 0

If this is set then an additional output file `outfile.tcsn` is generated, listing the shapes of the subpopulation tree in newick format. This can be used as input to a program like CONSENSE from Phylip to produce a consensus tree.

example- `tree_consensus: 1`

warmup *used:* always *default:* 0

The number of additional outputs to be generated to allow for burn-in of the MCMC algorithm. However, the additional outputs are not automatically discarded: this must be done explicitly by the user (and the number eventually discarded need bear no relation to the value of `warmup`).

example- warmup: 200