Coalescent: an open-science framework for importance sampling in coalescent theory

Susanta Tewari and John L. Spouge

National Center of Biotechnology Information, Bethesda, MD, United States

Submitted 25 May 2014
Accepted 30 July 2015
Published 18 August 2015

Corresponding author
Susanta Tewari,
statsusant@gmail.com

Academic editor
Claus Wilke

Additional Information and Declarations can be found on page 19

DOI 10.7717/peerj.1203

ABSTRACT

Background. In coalescent theory, computer programs often use importance sampling to calculate likelihoods and other statistical quantities. An importance sampling scheme can exploit human intuition to improve statistical efficiency of computations, but unfortunately, in the absence of general computer frameworks on importance sampling, researchers often struggle to translate new sampling schemes computationally or benchmark against different schemes, in a manner that is reliable and maintainable. Moreover, most studies use computer programs lacking a convenient user interface or the flexibility to meet the current demands of open science. In particular, current computer frameworks can only evaluate the efficiency of a single importance sampling scheme or compare the efficiencies of different schemes in an ad hoc manner.
Results. We have designed a general framework (http://coalescent.sourceforge.net; language: Java; License: GPLv3) for importance sampling that computes likelihoods under the standard neutral coalescent model of a single, well-mixed population of constant size over time following infinite sites model of mutation. The framework models the necessary core concepts, comes integrated with several data sets of varying size, implements the standard competing proposals, and integrates tightly with our previous framework for calculating exact probabilities. For a given dataset, it computes the likelihood and provides the maximum likelihood estimate of the mutation parameter. Well-known benchmarks in the coalescent literature validate the accuracy of the framework. The framework provides an intuitive user interface with minimal clutter. For performance, the framework switches automatically to modern multicore hardware, if available. It runs on three major platforms (Windows, Mac and Linux). Extensive tests and coverage make the framework reliable and maintainable.
Conclusions. In coalescent theory, many studies of computational efficiency consider only effective sample size. Here, we evaluate proposals in the coalescent literature, to discover that the order of efficiency among the three importance sampling schemes changes when one considers running time as well as effective sample size. We also describe a computational technique called "just-in-time delegation" available to improve the trade-off between running time and precision by constructing improved importance sampling schemes from existing ones. Thus, our systems approach is a potential solution to the "2^8 programs problem" highlighted by Felsenstein, because it provides the flexibility to include or exclude various features of similar coalescent models or importance sampling schemes.

Subjects Computational Biology, Genetics, Mathematical Biology, Science Policy, Statistics
Keywords Framework, Population genetics, Open-science, Importance sampling, Likelihood, Infinite sites model, Coalescent theory

INTRODUCTION

Felsenstein et al. (*1999*; Section 14) describes "the 2^8 programs problem" obstructing computational inference in population genetics, namely, that each variation in a statistical model or computational method requires a new computer program, even if underlying concepts remain similar. For example, importance sampling for population genetic models is an active area of research, but the 2^8 programs problem obstructs the comparision of novel and existing ideas, because no available computational framework can readily compare different importance sampling proposals. A computer program directly modelling the underlying concepts would provide a systematic solution, but for various reasons, existing implementations do not typically reflect the systematic approach (*Stodden, 2013a*) characterizing the development of theory. A systematic approach has the potential to improve the reliability of the implementation of base concepts and dramatically reduce the programming effort required to benchmark new ideas.

We therefore took a systems approach to population genetic models and developed a computational framework for importance sampling. Currently, the framework implements a standard neutral coalescent model of a single, well-mixed population of constant size over time under the infinite sites model of mutation. By design, however, it circumvents the 2^8 programs problem, so that the programming effort to augment a model with a new feature is linear in time (by definitions given by *Felsenstein et al. (1999)* and its discussion of the 2^8 programs problem). The framework can compare any subset of proposals programmed into it. In particular, we implemented the three standard proposals (Ethier–Griffiths–Tavaré, Stephens–Donnelly, and Hobolth–Uyenoyama–Wiuf) to compare them within a single framework. Because the framework already implements concepts like "likelihood", "genealogy", etc., adding a new proposal only requires specifying the corresponding algorithm.

The systems approach here mirrors our previous approach to computing exact coalescent probabilities (*Tewari & Spouge, 2012*). Previously, we implemented the computation of the exact likelihood for a general class of coalescent models based on recursions. The general implemention of recursions permits many useful computations (such as counting the total number of ancestral configurations, total number of genealogies etc.) with programming effort only linear in time, as explained above. The implementation permitted us to program and study several competing algorithms for computing exact probabilities (e.g., the forward algorithm of *Wu, 2009*), and we exemplified its use with both the infinite alleles and infinite sites model of mutation. Although exact probabilities help to benchmark computations and approximations in small datasets, they also aid intuition, with the potential to improve proposals in importance sampling. Thus, our two frameworks have complementary purposes.

Table 1 **Data set for Model K69, similar to Fig. 8.6 in *Wakeley (2009)*.** The characters are encoded as 0 and 1. Mutations are encoded as numbers from 1 to 4. The blue cells are the haplotype matrix, whose rows give the characters in each haplotype. The column vector to the right counts each haplotype in the sample data set, so the total number of genetic samples is 5.

Allele	**Segregating sites**				**Count**
	1	***2***	***3***	***4***	
a1	1	0	0	0	2
a2	1	1	1	1	1
a3	1	1	0	0	1
a4	0	0	0	0	1

BACKGROUND

Infinite-sites model (K69)

Excellent overviews of various coalescent models are available (e.g., *Hein, Schierup & Wiuf, 2005*; *Wakeley, 2009*). Here for the sake of completeness, we briefly describe the infinite-sites model (denoted "*K69*", after *Kimura, 1969*; also see *Watterson, 1975*). Most of our notation follows *Wakeley (2009)*.

Consider an aligned sample of DNA sequences, noting that alignment columns can contain gaps. An alignment column lacking gaps is called a "site", and Model K69 considers only sites. Under Model K69, the sample evolves from most recent common ancestor (MRCA) by reproduction, with occasional mutation at the sites. Model K69 is most suitable for long DNA sequences with low mutation rates, because it permits at most one mutation at each site during the evolution of the sampled sequences. The state of each site in a sampled sequence (its "character") can therefore be summarized by a binary digit: 0, if the corresponding DNA letter agrees with the MRCA; and 1, otherwise. A site is segregating if some sequences in the relevant sample contain the character 1. Thus, the segregating sites comprise the essential data in the sample. The sample data can be represented as $D = [X, \nu]$, where X is a binary matrix (i.e., $X_{i,j} \in \{0,1\}$) with distinct rows X_i ("haplotypes") and ν is an column vector such that ν_i counts the multiplicity of the haplotype X_i among the sampled sequences. Let n denote the total number of alleles i.e., $n = \sum_{(i)} \nu_i$. Thus, the character of haplotype X_i at site j is $X_{i,j} \in \{0,1\}$. See Table 1 for a sample dataset $D = [X, \nu]$ similar to Figure 8.6 in *Wakeley (2009)*.

As an aid to visualization, data conforming to Model K69 always have a unique *gene tree*. Figure 1, e.g., shows the gene tree corresponding to Table 1. Within a gene tree, the order of mutations on any edge is arbitrary, and permutation of the column order in the haplotype matrix X does not affect the gene tree for $D = (X, \nu)$. *Gusfield (1991)* gives an efficient algorithm for constructing gene trees.

Under Model K69, the MRCA (represented by a matrix with a single row of 0s) evolves into the sample data $D = [X, \nu]$ by passing stepwise through a sequence of ancestral configurations, which have a form $C = [X, \nu]$ similar to the sample data. In the following, a "singleton row i" is a row with count $\nu_i = 1$. Starting from sample back in time to the MRCA, each step corresponds to one of three possible evolutionary operations on the

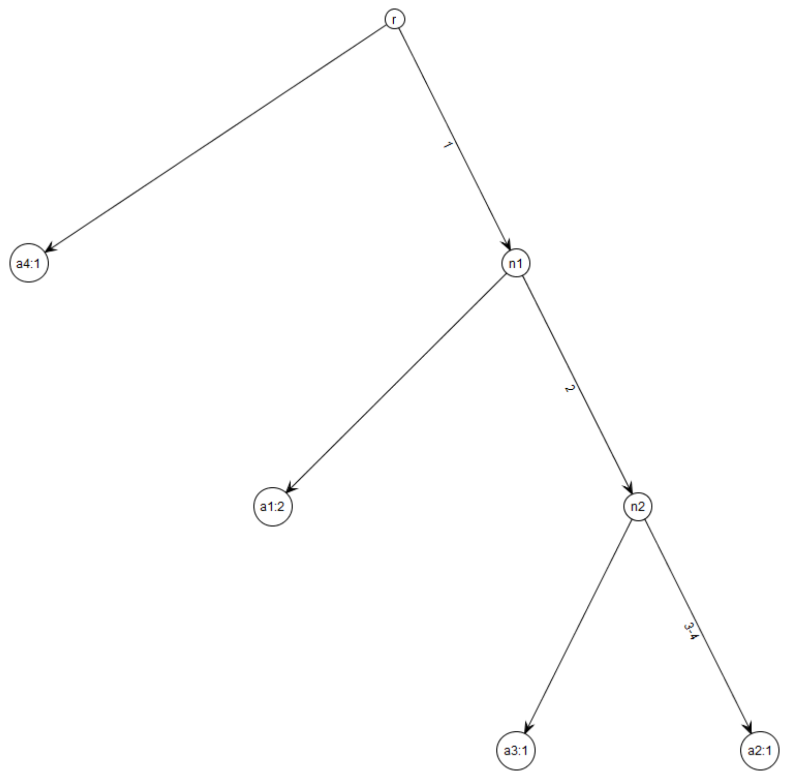


Figure 1 Gene tree for data in Table 1. Note that gene trees are not coalescent trees where time information is present. The leaf nodes correspond to the alleles and their count, e.g., node a_1: *2* indicates allele a_1 with count *2*. Mutations are labelled on the edges and all the children node inherit those mutations, e.g., Mutation *1* is inherited by all the leaf nodes except a_4:*1*. This figure was drawn using the framework in *Tewari & Spouge (2012)*.

current ancestral configuration: (1) coalescence (deleting one from a set of identical rows); (2) removing a "type I" mutation, which leaves the sequence still unique (changing the only 1 in some column j into 0 and leaves the corresponding singleton row i unique in the ancestral configuration); and (3) removing a "type II" mutation, which makes the sequence identical to one or more others (changing the only 1 in some column j into 0, to make the corresponding singleton row i the same as some other row(s) in the ancestral configuration). Removal in both mutational types I and II is restricted to "the only 1 in some column" and "a singleton row", because Model K69 permits at most one mutation at each site. Once a mutation is removed, the corresponding site is no longer a segregating site (i.e., the corresponding column in the new binary matrix has only 0s). Figure 2 illustrates these operations. Thus, a computer can efficiently represent the removal of a mutation simply by removing the corresponding column from X, a representation we now use. Under the representation, the MRCA becomes an empty matrix with count 1.

To represent the three evolutionary operations mathematically, consider an ancestral configuration $C = [X, \nu]$, let e_i denote a column vector with 1 in the i-th position, and 0 elsewhere. Given C, let A denote the set of singleton rows i, and $A(i)$ denote column j with the smallest index in row i, so that row i and column $A(i)$ satisfy the restrictions

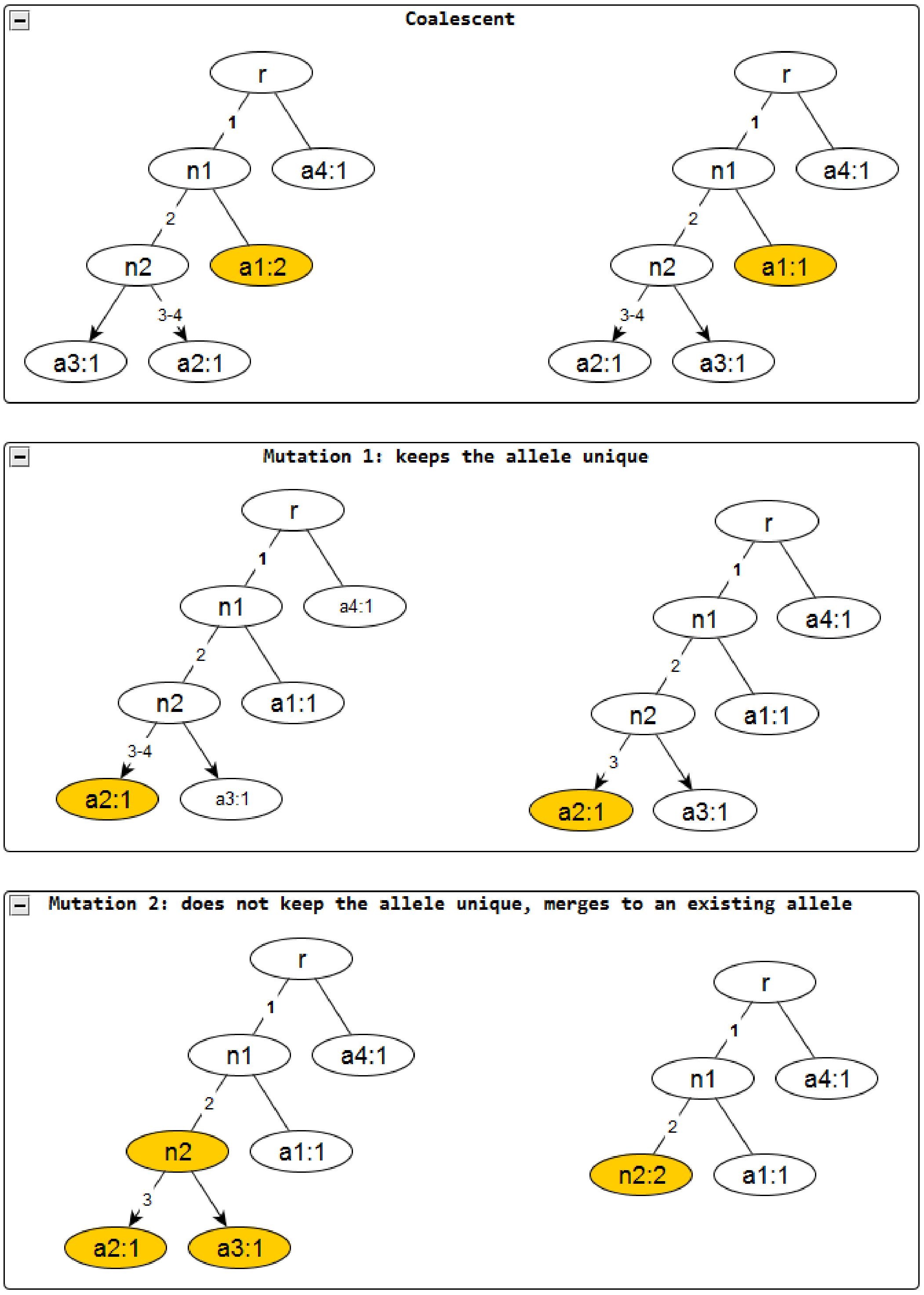


Figure 2 Evolutionary operations in the infinite sites model. The before and after configurations are drawn for each evolutionary operation. The affected nodes are represented in color. Coalescent events reduce the node frequency by *1* but does not affect the edge labels. Mutation events remove the edge labels one at a time. The first kind keeps the affected allele unique whereas the second one does not and merges with an existing allele, incresing the frequnecy of the merged allele by one.

on mutations of type I (In the following, the arbitrary choice of the smallest column index $A(i)$ is feasible, because column order is irrelevant to the gene tree.). Let δ_i be the corresponding evolutionary operator that deletes column $A(i)$ from X, creating the new ancestral configuration $[\delta_i X, \nu]$. Similarly, let B denote the set of singleton rows i, each with a single column j satisfying the restrictions on mutations of type II. For each i, let $B(i)$ be the row index of the "merge haplotype", the haplotype that row i becomes when the 1 in column j is changed to 0. Let Δ_i be the corresponding evolutionary operator, which deletes row i and column j from X, creating the haplotype matrix $\Delta_i X$, and which also deletes the i-th row of the column vector ν, so the new ancestral configuration is $\left[\Delta_i X, \Delta_i\left(\nu + e_{B(i)}\right)\right]$.

Having defined the sample space of ancestral configurations $[X, \nu]$ and the steps that Model K69 permits, we now determine the corresponding probability measure $p[X, \nu]$, which implicitly depends on a population mutation parameter θ. The MRCA probability is $p([\,],\ [1]) = 1.0$, and the probabilities $p[X, \nu]$ satisfy the recursion

$$\begin{aligned} n(n-1+\theta)p[X,\nu] &= \sum_{i:\nu_i \geq 2} \nu_i(\nu_i - 1)p[X, \nu - e_i] + \theta \sum_{i:i \in A} p[\delta_i X, \nu] \\ &\quad + \theta \sum_{i:i\in B} \left(\nu_{B(i)} + 1\right) p\left[\Delta_i X, \Delta_i\left(\nu + e_{B(i)}\right)\right], \end{aligned} \tag{1}$$

For introductory examples see *Hein, Schierup & Wiuf, 2005*; *Wakeley, 2009*.

Importance sampling for computing likelihood

Wakeley (2009) gives an overview of importance sampling for computing likelihood in coalescent theory. Briefly, here are the key concepts. To make the dependence of the probability on the mutational parameter θ explicit, let $p(D;\theta) = p[X, \nu]$ for $D = (X, \nu)$. The probability of the data in (1) can also be written as the following:

$$p(D;\theta) = \sum_G p(D \mid G;\theta)p(G) \tag{2}$$

where the sum is over all genealogies G consistent with the data D. Let $q(.)$ be any probability measure, and $E_q[.]$ be an expectation under $q(.)$, and define the ratio $w(G) = p(G)/q(G)$. If $p(.)$ is absolutely continuous with respect to $q(.)$, i.e., if $q(G) > 0$ wherever $p(G) > 0$, then

$$\begin{aligned} p(D;\theta) &= \sum_G p(D \mid G;\theta)\frac{p(G)}{q(G)}q(G) \\ &= E_q\left[p(D \mid G;\theta)\frac{p(G)}{q(G)}\right] \\ &= E_q\left[p(D \mid G;\theta)w(G)\right]. \end{aligned} \tag{3}$$

Usually, in the context of importance sampling, $p(.)$ is called the target distribution; $q(.)$, the trial distribution; and $w(.)$, the importance sampling weight (*Hammersley & Handscomb, 1964*; *Liu, 2002*). Given R realizations G_r $(r = 1, \ldots, R)$ of the genealogy G *independently* sampled from the trial distribution $q(.)$, then the strong law of large numbers implies that with probability 1,

$$p(D;\theta) = \lim_{R\to\infty} R^{-1}\sum_{r=1}^{R} p(D \mid G_r;\theta)\, w(G_r). \tag{4}$$

Thus, importance sampling provides an estimator for the likelihood

$$\hat{p}_{IS}(D;\theta) \approx R^{-1}\sum_{r=1}^{R} p(D \mid G_r;\theta)\, w(G_r). \tag{5}$$

Equation (1) provides a sequence of steps from a population sample to its most recent common ancestor (MRCA), each step corresponding to a single ancestral coalescence or the loss of a single mutation. A Monte Carlo simulation can therefore assign trial probabilities $q(.)$ to these time-steps, to create a sequential importance-sampling scheme (*Liu, 2002*). Many coalescent processes are Markovian, so sequential importance sampling (SIS) is a natural choice to simulate them, because events occurring at different time-steps are independent. The coalescence literature often uses the terminology "proposals" for the sampling choices, although "proposals" does not follow the standard usage in the Monte Carlo literature (The non-standard usage might be derived from the Metropolis method, which accepts or rejects "proposals"). In any case, this paper adheres to standard Monte Carlo terminology (*Liu, 2002*; *Robert & Casella, 2004*) whenever there is no conflict with terminology in the coalescent literature.

Standard sequential samplers

Sequential samplers choose among the evolutionary operations corresponding to the different terms in Eq. (1). Because each operation is determined once the corresponding haplotype X_i in the ancestral configuration $C = (X, \nu)$ is known, we let $q(i \mid C)$ with various subscripts denote corresponding trial probability.

The Ethier–Griffiths–Tavare (EGT) sequential sampler

The EGT recursion in Eq. (1) directly suggests a sequential sampling scheme (*Griffiths & Tavare, 1994*):

$$q_{GT}(i \mid C) \propto \left\{ \begin{array}{cc} \nu_i - 1 & \nu_i \geq 2 \\ \frac{\theta}{n} & i \in A \\ \frac{\theta\left(\nu_{B(i)}+1\right)}{n} & i \in B \\ 0 & \text{otherwise} \end{array} \right\}. \tag{6}$$

The Stephens–Donnelly (SD) sequential sampler

Stephens & Donnelly (2000) developed a sampling scheme by characterizing the target distribution and then approximating it with

$$q_{SD}(i \mid C) \propto \left\{ \begin{array}{cc} \nu_i & \nu_i \geq 2 \text{ or } i \in A \text{ or } i \in B \\ 0 & \text{otherwise} \end{array} \right\}. \tag{7}$$

The Hobolth–Uyenoyama–Wiuf (HUW) sequential sampler

Hobolth, Uyenoyama & Wiuf (2008) approximated the effects of all mutations on the probabilities for the next step from the sample to the MRCA, to derive

$$q_{HUW}(i \mid C) \propto \left\{ \begin{array}{cc} \sum_m u_{i,m}(\theta_0) & \nu_i \geq 2 \text{ or } i \in A \text{ or } i \in B \\ 0 & \text{otherwise} \end{array} \right\}, \tag{8}$$

where θ_0 is a fixed value of θ,

$$u_{i,m}(\theta) = \left\{ \begin{array}{cc} p_\theta(d_m)\dfrac{\nu_i}{d_m} & X_{i,m} = 1 \\ \left[1 - p_\theta(d_m)\right]\dfrac{\nu_i}{(n - d_m)} & X_{i,m} = 0 \end{array} \right\}$$

$$d_m = \sum_m X_{i,m}\nu_i$$

and

$$p_\theta(d_m) = \frac{\sum_{k=2}^{n-d_m+1} \dfrac{d_m - 1}{n-k}\dfrac{1}{k-1+\theta}\binom{n-d_m-1}{k-2}\binom{n-1}{k-1}^{-1}}{\sum_{k_0=2}^{n-d_m+1} \dfrac{1}{k_0-1+\theta}\binom{n-d_m-1}{k_0-2}\binom{n-1}{k_0-1}}$$

$$p_\theta(1) = \frac{\frac{1}{n-1+\theta}}{\sum_{k_0=2}^{n} \dfrac{1}{k_0-1+\theta}\dfrac{k_0-1}{n-1}},$$

where $d_m = \sum_i X_{im}\nu_i$ is the total number of alleles containing mutation m and $p_\theta(d_m)$ is the probability that the next evolutionary operation (coalescence or mutation) involves a row X_i where $X_{i,m} = 1$ (i.e., haplotype i bears mutation m) and $u_{i,m}$ denotes the probability of involving row X_i in the next mutation event m. The proposal probability in Eq. (8) sums $u_{i,m}$ over all mutations m for row X_i.

IMPLEMENTATION

We now describe the architecture of the framework, diagramming the key classes and interfaces with the unified modelling language (UML), while displaying the various connections and assumptions. The diagrams use standard UML notations (explained in the *UML-Wiki*). The framework consists of several packages, which progressively narrow the most general concepts down to the specifics of *K69*, the infinite-sites model of mutation.

Core framework

The core framework models the concepts for any domain of sampling. It corresponds to the package *commons.is*. Figure 3 displays the key classes: *Sampler*, *Proposal*, and *Factor*.

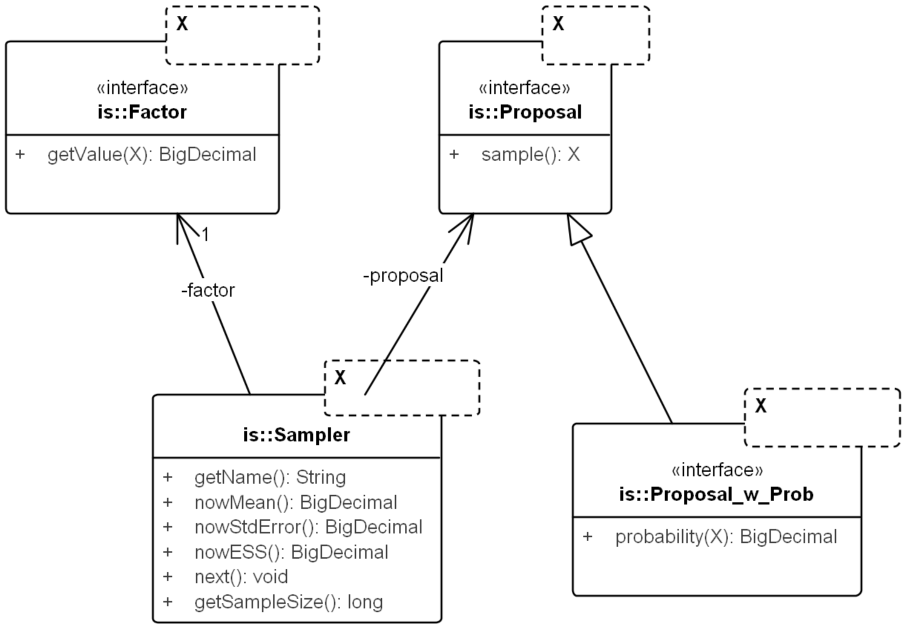


Figure 3 UML diagram of the core importance sampling (IS) framework. The diagram shows key classes corresponding to the concepts of sampler, proposal and importance weight (*Factor*). They are parametric over the domain of sampling, denoted by *X*. For coalescent models, the domain of sampling is *Genealogy*. *Proposal* specifies the sampling algorithm and its extension, *Proposal_w_Prob* specifies its probability distribution. The sampler performs the sampling by repeatedly calling *next()*. The various *nowX(.)* methods give estimates based on the sampled values so far. In text, a sampler is referred by its proposal name when the context is clear.

Sampler

Sampler generalizes Eq. (5) as

$$\hat{E}_q[Y] = \sum_{r=1}^{R} h(Y_r)w(Y_r), \tag{9}$$

where $h(Y)$ is called a "mean function" i.e., a function whose mean (or expectation) is to be computed, and $w(Y_r) = p(Y_r)/q(Y_r)$, $p(.)$ and $q(.)$ being the target and trial distributions, respectively. For coalescent models, Y corresponds to the genealogy G, and $h(Y)$ denotes the conditional probability of observing data D given the genealogy Y. Note that for coalescent models, Y represents the relevant events in the entire genealogical history of the sample, including coalescent events. To estimate the probabilities in the standard sequential sampling schemes above, take $h(Y) = 1$ identically for all Y.

Proposal & factor

Proposal draws an independent sample Y each time *sample()* is called and *Factor* computes $w(Y)$ in Eq. (9). *Factor* can be created by computing the weight $w(Y)$ directly (by implementing sub-interface *Proposal_w_Prob*) or by implementing an analytical expression for the ratio, if available (e.g., the so-called "functional path" F_j in Eq. 12 of *Griffiths & Tavare, 1994*).

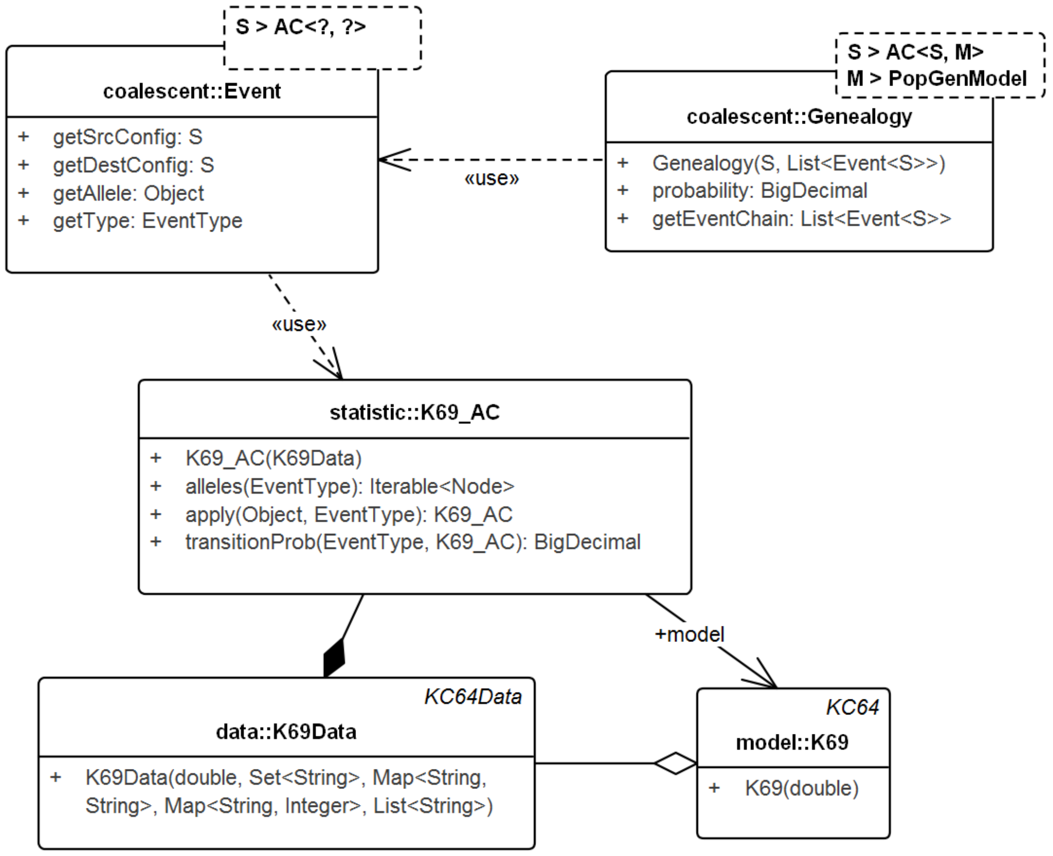

S > AC<?, ?>
coalescent::Event
+ getSrcConfig: S
+ getDestConfig: S
+ getAllele: Object
+ getType: EventType
«use»
S > AC<S, M>
M > PopGenModel
coalescent::Genealogy
+ Genealogy(S, List<Event<S>>)
+ probability: BigDecimal
+ getEventChain: List<Event<S>>
«use»
statistic::K69_AC
+ K69_AC(K69Data)
+ alleles(EventType): Iterable<Node>
+ apply(Object, EventType): K69_AC
+ transitionProb(EventType, K69_AC): BigDecimal
+model
KC64Data
data::K69Data
+ K69Data(double, Set<String>, Map<String, String>, Map<String, Integer>, List<String>)
KC64
model::K69
+ K69(double)

Figure 4 UML diagram of the key classes in *Tewari & Spouge (2012)* used for sampling genealogies. The diagram shows the connection between *K69_AC* (an *AC* in model *K69*) and *Genealogy*. *AC*s generate Events via method *apply()* recursively and the chain of events in turn create the *Genealogy*. Events do not carry the time information. Its allele property specifies the allele of the source configuration that generated the destination configration. Note that, *Genealogy* is connected to the underlying data and model and carries enough information to compute its own probability. *Event* and *Genealogy* are both generic and work across a range of types. *Genealogy*, for example, would work with any type *S* that inherits from *AC* (such as *K69_AC*) and type *M* that inherits from *PopGenModel* (such as *K69*).

Coalescent models

The following subsection describes SIS schemes for coalescent genealogies. Our framework for exact probabilities in coalescent models (*Tewari & Spouge, 2012*) already contains the general concepts for implementing sequential schemes for specific models. Figure 4 sketches the interactions among these general concepts.

Genealogy & AC

Genealogy defines the chain of events from the sample to the MRCA. *AC* denotes the ancestral configuration in a generic coalescent model and given the allele and event types, specifies the recursion. Ancestral configuration (K69_AC—for model K69) provides alleles on which events (coalescent, mutation) occur and also generates the resulting ancestral configurations. The recursive events create genealogies that start from the sample and end at the MRCA. The Event object captures the source and destination

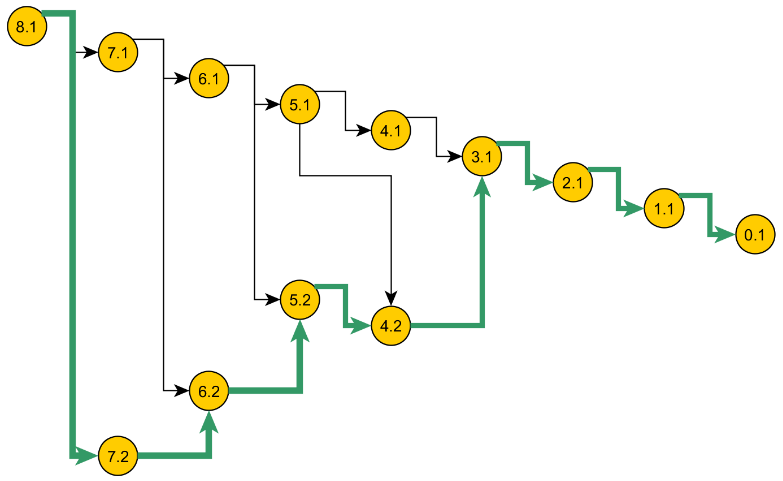


Figure 5 Space of genealogies as domain of sampling. The paths in the diagram constitute the space of genealogies for data in Table 1. Total number of genealogies grows exponentially with the sum of the number of alleles and the mutation count present in data. A sample path from this space is marked. Nodes represent the ancestral configurations (*AC*) and their labels have format: [*eventstoMRCA*] :[*serial-no.*]. The *AC*s are grouped by their *eventstoMRCA* which acts as a sequential step in the sampling. A particular branch is selected with chance proportional to the weight specified in the proposal.

configurations, the event type, the allele on the source configuration that created the destination configuration, but not the time information.

GProposal

GProposal implements SIS via *Proposal_ w_Prob*. It recursively builds the sample and its probability, at each step using the alternatives in Eq. (1) (The previous version of our framework already calculated the exact probabilities, reducing programming effort). Figure 5 illustrates SIS in a coalescent process. *GProposal* specifies the SIS of all coalescent models based on Eq. (1). Specific proposals (e.g., Eqs. (6), (7) and (8)) need only implement the abstract method *proposalWeight*. The program design suits Monte Carlo well, because it localizes errors and limits the scope of problems associated with a specific proposal. It also stabilizes results when comparing sequential sampling schemes: general optimizations might improve the efficiency of several schemes, but the implementation of each scheme shares the gain, thereby maintaining their relative efficiencies.

The infinite-sites model K69

For the infinite-sites model of mutation (*K69*), we implemented three standard proposals (Eqs. (6), (7) and (8)) with the abstract method *proposalWeight*, as described above. If the root (i.e., the state of the MRCA) is unknown, the user has two choices: (1) repeat the computation by hand for all possible roots (future versions will automate the repetition); or (2) follow the accepted expedient of substituting the consensus sequence for the root *McMorris, 1977*. The proposals are collected in *GProposals_K69* (see Fig. 6), which follows the factory design pattern (*Gamma et al., 1995*). The framework for exact probabilities already specifies the interface *K69_AC* of the ancestral configuration under infinite-sites model of mutation, so we used it to specify the three proposals. Figure 7 illustrates

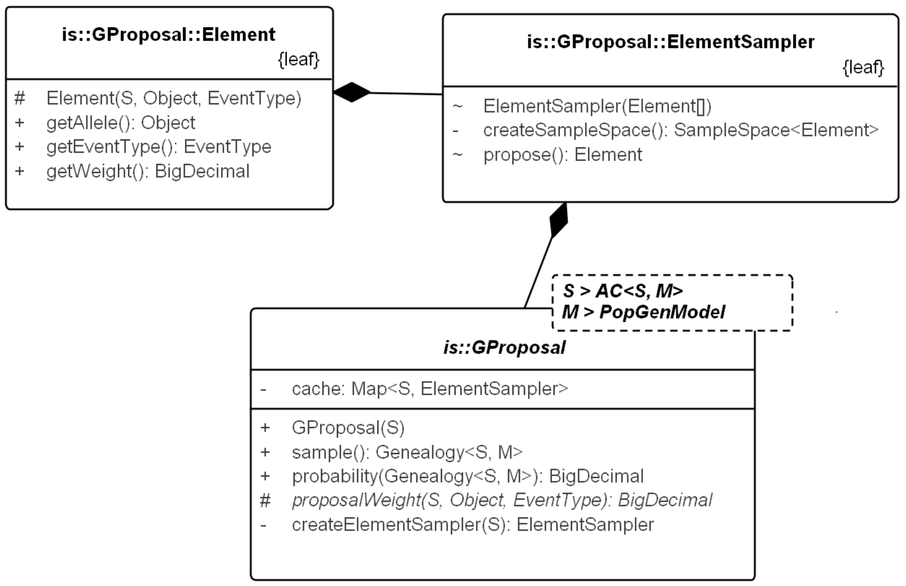
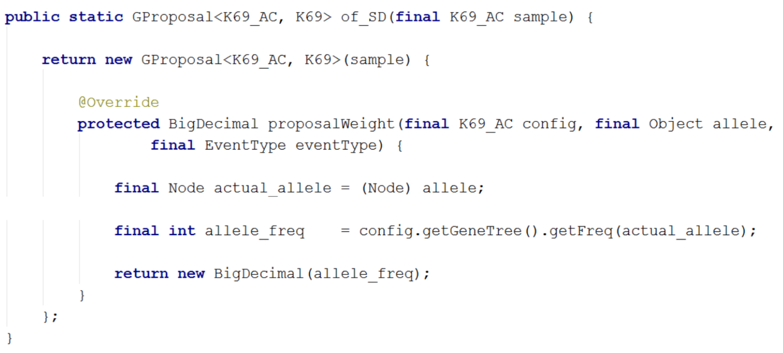

is::GProposal::Element
{leaf}
# Element(S, Object, EventType)
+ getAllele(): Object
+ getEventType(): EventType
+ getWeight(): BigDecimal
is::GProposal::ElementSampler
{leaf}
~ ElementSampler(Element[])
- createSampleSpace(): SampleSpace<Element>
~ propose(): Element
S > AC<S, M>
M > PopGenModel
is::GProposal
- cache: Map<S, ElementSampler>
+ GProposal(S)
+ sample(): Genealogy<S, M>
+ probability(Genealogy<S, M>): BigDecimal
# proposalWeight(S, Object, EventType): BigDecimal
- createElementSampler(S): ElementSampler

Figure 6 UML diagram explaining the internals of GProposal. GProposal implements *Proposal_w_Prob* with *Genealogy* for type parameter *X*. It has two central methods: (1) *sample()* and (2) *probability(genealogy)* whose implementations are correlated (second method reuses parts of the first) to be efficient. Branch weights are encapsulated in structure *Element* and one *Element* is proposed by the *ElementSampler* at each sequential step of the sampling. As such, new proposals need only proivde the specification of the proposal equation via *proposalWeight(.)* method. All the standard proposals have been implemented in this way. The implementation of SD proposal is shown in Fig. 7.


```
public static GProposal<K69_AC, K69> of_SD(final K69_AC sample) {

    return new GProposal<K69_AC, K69>(sample) {

        @Override
        protected BigDecimal proposalWeight(final K69_AC config, final Object allele,
                final EventType eventType) {

            final Node actual_allele = (Node) allele;

            final int allele_freq    = config.getGeneTree().getFreq(actual_allele);

            return new BigDecimal(allele_freq);
        }
    };
}
```

$$q_{SD}\left(i \mid C\right) \propto \left\{ \begin{matrix} v_i & v_i \geq 2 \text{ or } i \in A \text{ or } i \in B \\ 0 & \text{otherwise} \end{matrix} \right\}$$

Figure 7 Writing a new proposal. Demonstrates writing of a new proposal using the SD proposal. Note that the implementation is a close translation of the corresponding equation.

the implementation of SD Proposal, which demonstrates that proposals can be written compactly from the corresponding equations, leaving the framework to encapsulate the details.

Multiple parameters

Most population genetic models with multiple parameters (including model K69) lack analytical expressions for the corresponding likelihoods, so computers are essential for

Table 2 **Tests and coverage for the framework.** The table shows the number of tests and the percentage of coverage for various packages in the framework. Typically, 70% coverage is considered stable.

Test area	Number of tests	Coverage (line)
Common	47	70%
Model	10	75%
Data	31	92%
Phylogeny	11	86%
Recursion	23	62%
Statistic	27	84%
Providers	30	61%
Importance sampling	*62*	76%
	241 (total)	*75.75%* (avg.)

calculating maximum likelihood estimates (MLEs). Separate likelihood computations for each set of parameter values is often inefficient, because some models (e.g., K69) permit computations for each set to reuse every realization, vastly improving performance. Different proposals exploit this opportunity differently, however. SD, e.g., does not use the parameter value corresponding to mutation, EGT uses scaling (Eq. 12 in *Griffiths & Tavare, 1994*), and HUW indirectly avoids the parameter value by using asymptotics from Eqs. 12 and 13 of *Hobolth, Uyenoyama & Wiuf (2008)*. Computations for different proposals reuse realizations differently, so a poor design might easily require a programmer to specify a proposal twice. In contrast, our framework avoids duplication of effort and allows a single proposal specification with a single set of parameter values to sweep through several sets of parameter values by specifying a separate implementation for reusing realizations.

Parallel proposals

The framework design is object-oriented, thereby promoting the natural use of multiple computer cores. Generally, it emphasizes modularity over optimization of running times, enhancing platform-independence and the possibility of running the framework on common multi-core machines. Multiple schemes can run on separate *threads*, reducing computing time and providing direct visual feedback on their running times. Some proposals both increase statistical power and reduce computing time relative to other proposals (e.g., SD over EGT) but more typically, a nuanced trade-off occurs.

Tests and coverage

The expansion of the framework over time must not break existing features and design contracts (*Freeman & Pryce, 2009*). Thus, extensibility puts constraints on, e.g., the graphical user interface. *Automated unit tests* can permit debugging to remain manageable and coverage of *checked exceptions* (known causes of disruption) to expand. The *coverage* (the proportion of code lines that the unit tests execute) is an accepted figure of merit for unit tests. Table 2 provides the number of tests and coverage for the packages in the framework. Typically, more than 70% coverage should inspire confidence (*Limjap, 2008*).

Features

Some features of the framework follow, along with some notes on each feature. Readers can consult the project website for more information. Framework version numbers are indicated in square brackets, and Version *1.4.2* retains all features in version *1.3.0*. Version *1.4.2* corresponds to this article; Version *1.3.0*, to the article *Tewari & Spouge (2012)*.

1. [*1.3.0*] *Support for two models: Infinite Alleles Model (KC64) and Infinite Sites Model (K69)*. The models include mutation under the standard neutral coalescent model of a single, well-mixed population of constant size over time. Data for *K69* are read from an *xml* file; data for *KC64*, from a string literal within the framework.
2. [*1.3.0*] *Checks Phylogeny of binary data using Gusfieldś algorithm or the Four Gamete‘s algorithm. The user chooses the algorithm.*
3. [*1.3.0*] *Draws Phylogeny of K69 data.* The framework uses Gusfield‘s algorithm to build the gene tree.
4. [*1.3.0*] *Exact computation of various model statistics.* For *K69* and *KC64* on small datasets, the framework can: (1) compute the exact likelihood and the probabilities of ancestral configurations; (2) count and build ancestral configurations and genealogies; (3) profile the recursion cache, which can be useful in improving exact algorithms.
5. [*1.4.2*] *Maximum Likelihood Estimation (MLE) using exact computation.* The MLE can be computed for *K69* and *KC64* on small datasets. The user specifies the minimum, maximum, and the increment for values of the population mutation rate.
6. [*1.4.2*] *Smart Data Integration.* Given a job, the *Data Panel* automatically lists all relevant datasets in (*xml*) files or string literals. The user can easily open and edit *xml* files within the application, which updates jobs as it auto-detects files added or deleted from the underlying system. The framework can also interpret data intended for the Infinite Sites Model *K69* for use with the Infinite Alleles Model *KC64*.
7. [*1.4.2*] *Maximum Likelihood Estimation (MLE) using Importance Sampling(IS).* Equation (10) below defines the effective sample size. For *K69* on datasets of moderate size, the framework can compute likelihoods for any combination of available proposals, including EGT, SD, and HUW. In addition, a programmer can simply specify a new proposal to add it to the framework. A user can run multiple proposals simultaneously to measure their relative efficiencies and limit the sampling by realizations (to estimate effective sample sizes) or by time (to estimate effective sample sizes per running time), with results available in either textual or graphical formats. To benchmark proposals, the framework can plot its likelihood estimates next to exact likelihoods (either provided or computed); during execution, it can also track and report various measures (e.g., standard errors, effective sample sizes, etc.).
8. [*1.4.2*] *Benchmarking Performance of Competing Proposals.* The framework simulate datasets to compare measures associated with different proposals over a grid of parameters like mutation rates and alleles in the gene tree. As above, the user can choose to present key metrics as graphs or texts.

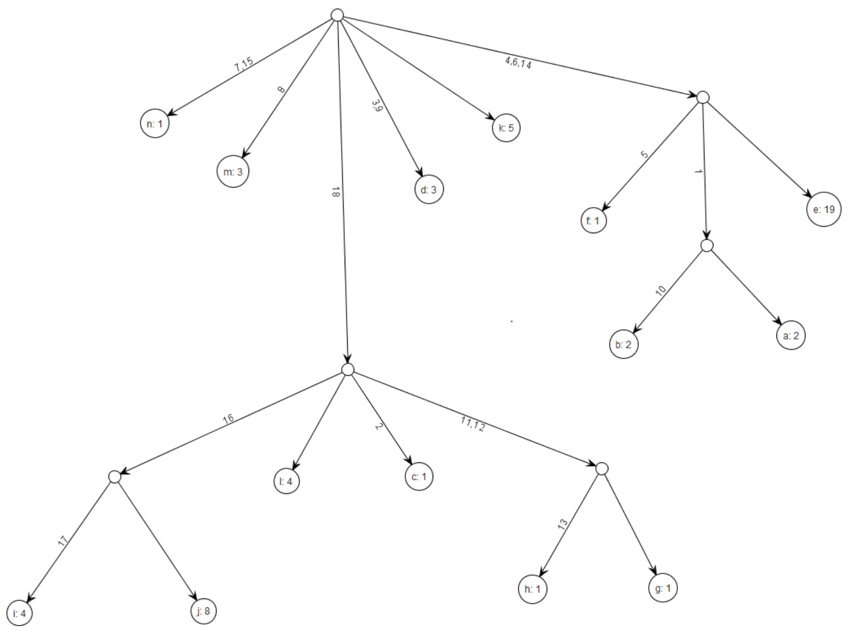


Figure 8 Gene tree for the benchmark data set in *Griffiths & Tavare (1994)*. There are 18 segregating sites and 14 distinct alleles and the total sample size is 55. The figure was drawn using the framework.

RESULTS & DISCUSSION

Our figures and results generally follow the presentation of *Hobolth, Uyenoyama & Wiuf (2008)*. Figure 8 displays the gene tree for the dataset of *Griffiths & Tavare (1994)*, a standard benchmark for comparing proposals. To compare the EGT, SD, and HUW proposals, we used effective sample size (ESS),

$$ESS = \frac{R}{1 + \mathrm{var}_q(w(Y))}, \tag{10}$$

where R is the number of realizations (samples), and w is the corresponding importance weight (*Liu, 2002*, p.35). Loosely, ESS quantifies the similarity of the target distribution to the trial distribution, so SIS improves as the ESS increases.

Validating MLE and the maximum likelihood

The framework computed the likelihood curve (Table 3). Table 4 for the MLEs resulting from each proposal is visually consistent with figures in *Wu (2009)* and verifies results for both standard error and ESS in *Hobolth, Uyenoyama & Wiuf (2008)*, namely, that the SIS performance order is EGT $<$ SD $<$ HUW. The running times for SD and EGT were nearly equal, however, and about half of the running time for HUW. Within the framework, all proposals share the same runtime infrastructure, so the accuracy in HUW comes at the price of approximately doubling the running time per realization.

Confirming proposal order

Hobolth, Uyenoyama & Wiuf (2008) investigated the performance of the three proposals by comparing ESSs as mutation rates and data sample size varied, their Fig. 6 showing

Table 3 Validating MLE of the mutation rate using multiple proposals. MLE is computed in the range [1.0, 10.0] with an increment of 0.1 using multiple proposals; corresponding published values are included from *Wu (2009)*.

Proposals	MLE		Sample size	
	Published	Framework	Published	Framework
EGT	4.8	4.8	200,000	100,000
SD	[4.5, 5.0]	4.9	100,000	100,000
HUW	[4.5, 5.0]	4.9	100,000	100,000

Table 4 Estimating likelihood at the MLE of mutation rate 4.8 by multiple proposals for fixed number of realizations (100,000). Published values are compared against the values computed using the framework. Exact likelihood is 8.71E–20 (*Wu, 2009*).

Proposals	Published	Framework			
	Likelihood	Likelihood	Std. error	ESS	Time(s)
EGT	7.76E–20	7.57E–20	8.31E–21	82	1347
SD	9.33E–20	9.14E–20	5.41E–21	283	1046
HUW	8.70E–20	9.01E–20	3.75E–21	572	2160

that the ESSs for three proposals have a stable relationship, EGT < SD < HUW. Using the same mutation rates and data sample sizes, our simulations independently confirmed their results as follows (Fig. 9). For each cell, the tool *ms* (*Hudson, 2002*) or *msms* acting as a cross-platform fallback (*Ewing & Hermisson, 2010*) simulated three independent sets of samples (denoted by different colours in the figure) for the corresponding mutation rate and data sample size. Within each cell, there are two plots. The left-hand panels show results if we limited all simulations to 108,000 realizations (approximating the 100,000 used in Fig. 6 of *Hobolth, Uyenoyama & Wiuf (2008)*); the right-hand panels, if we limited all samplers by time, terminating them when the slowest one completed 108,000 realizations. The left panels of each cell verify the results in Fig. 6 in *Hobolth, Uyenoyama & Wiuf (2008)*.

Effect of running time

Although the left panels of each cell verify the results in Fig. 6 in *Hobolth, Uyenoyama & Wiuf (2008)*, the right panels show that when the figure of merit is *ESS per running time*, SD improves relative to HUW, but EGT does not. Thus, although HUW requires more time than EGT to produce a realization, the improvement in the ESS more than compensates. HUW also requires more time than SD to produce a realization, but here the improvement in the ESS does not compensate so decisively. For the real dataset in Fig. 8, Table 5 tells the similar story. The message is clear: substantial improvements in the ESS for a fixed number of realizations do not always translate into a substantial practical reductions in the errors of estimates.

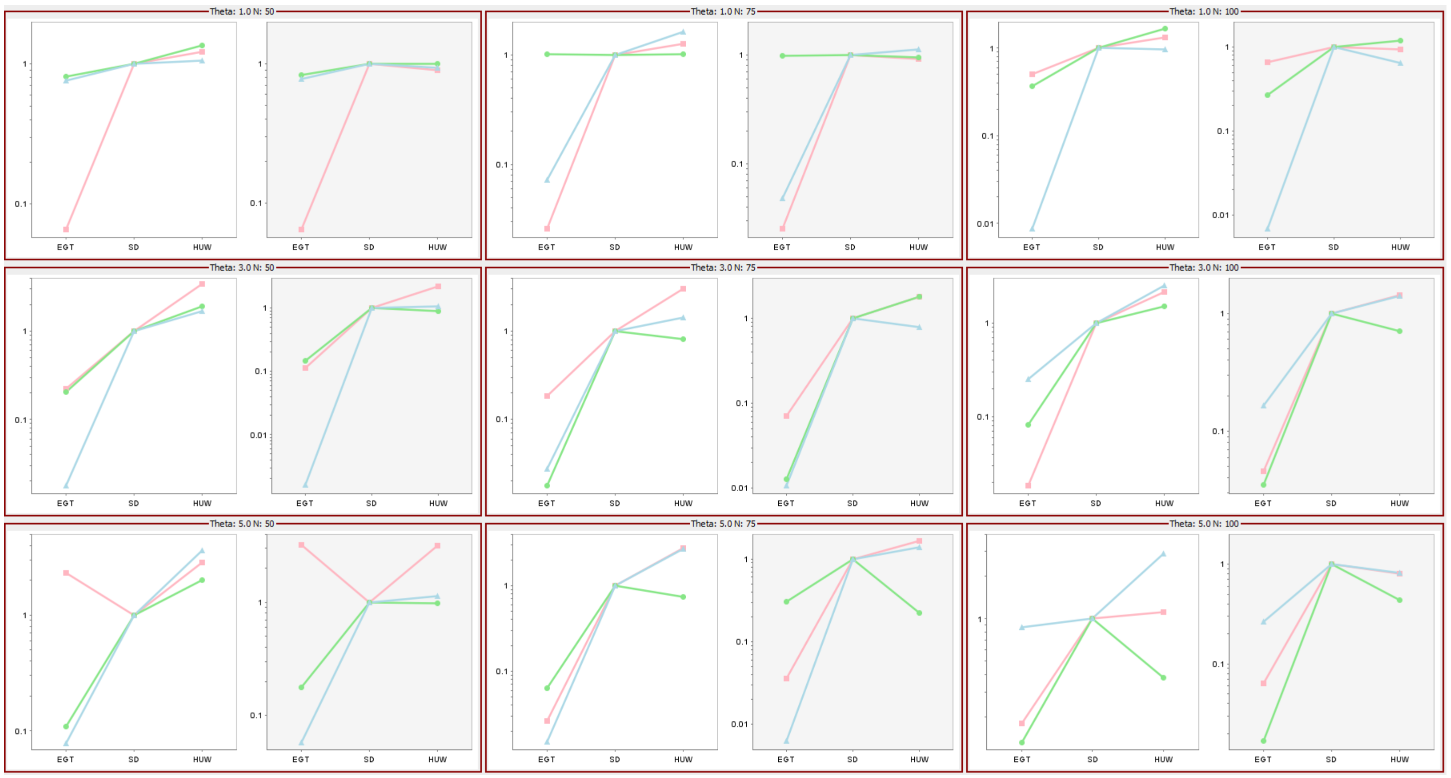


Figure 9 Significance of time in proposal efficiency. Three data sets are simulated under each combination of mutation rates (=1, 3, 5) and sample sizes ($n = 50, 75, 100$). Scaled ESS values (to the SD proposal) are estimated for each data set. All values are scaled to the ESS of the SD proposal, which corresponds uniformly to 1. Each cell has two plots: left panel verifies the proposal order for fixed number of observations but the right panel shows that when the running time is fixed, the increase in ESS may not be always decisive (e.g., HUW and SD).

Table 5 Estimating likelihood at the MLE of mutation rate 4.8 by multiple proposals under fixed time (3000 s). Proposals EGT and SD both sample more observations than HUW when run under the fixed time. The conclusions from Table 4 do not change. But, note that though EGT still ranks the same, the improvement of HUW over SD is less dramatic due to the difference in the number of realizations. Thus, running time should be included in comparing proposals.

Proposals	Likelihood	Std. Error	ESS	Realizations
EGT	8.43E–20	1.14E–20	54	212,277
SD	9.21E–20	5.66E–21	264	280,740
HUW	9.54E–20	5.16E–21	340	121,855

New mixed proposal via exact samplers

Wu (2009) motivated expanding the size limits on exact analysis of coalescent data by pointing out that exact analysis can validate Monte Carlo estimates. In the same spirit, we developed a framework for exact recursive computation of coalescent probabilities (*Tewari & Spouge, 2012*). Our present framework is a seamless extension of our recursive framework, and it permitted us to implement a new proposal, as follows.

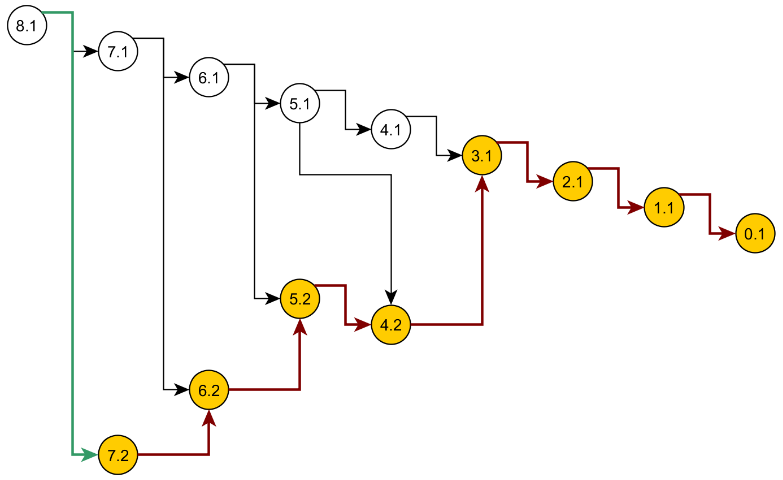


Figure 10 Genealogy of the exact singleton sampling for data in Table 1. Genealogy of the mixed-exact proposal contains a sub-graph (see text). Sub-graph nodes are highlighted, red arrows indicate exact sampling, and the green arrow indicates sampling via the delegate proposal.

Some nodes of the full ancestral graph permit exact sampling of transitions away from them. On the corresponding subgraphs, the framework can compute the most efficient proposal possible (the target distribution itself), while using a *delegate proposal* elsewhere. We call this idea *Exact Subgraph Sampling*. Consider, e.g., "singleton nodes", nodes of the ancestral subgraph whose sequences all have multiplicities $\nu_i = 1$, leading to an Exact Subgraph Sampling scheme, we call *Exact Singleton Sampling*. Figure 10 highlights a sampling path using exact singleton sampling for data in Table 1. For the real data in Fig. 8, the exact distribution for the singleton nodes can be computed under 10 s with average memory, whereas the full graph takes more than a day on a high-end PC (*Wu, 2009*). Because they are beyond the purview of this article, detailed results for Exact Singleton Sampling will be presented elsewhere, but preliminary results for the data in Fig. 8 suggest that Exact Singleton Sampling improves the ESS per computational time for HUW by a factor of about 4.

CONCLUSIONS

Running time is a significant practical consideration when comparing the computational efficiency of different importance sampling proposals. With ESS per running-time replacing ESS as a figure of merit used in *Hobolth, Uyenoyama & Wiuf (2008)*, the Hobolth-Uyenoyama-Wiuf proposal (HUW) retains only a small edge over the Stevens-Donnelly proposal (SD). In this case, therefore, the expected sample size (ESS) per running time and the ESS agree HUW as the optimal proposal, but disagree quantitively on the relative improvement. Running time is an important practical consideration, however, so when comparing proposals on a level playing field within a single computational framework, the ESS per running time should receive more weight than the ESS as a figure of merit. Thus, evaluations within computational frameworks like ours should be preferred to *ad hoc* evaluations of importance sampling schemes (ISSs). As an aside, our framework includes exact computations (*Tewari & Spouge, 2012*), so it easily implements the new proposal, Exact Singleton Sampling, described above.

Our project website (http://coalescent.sourceforge.net) contains open source code in the Java programming language under the GPLv3 license. Following the spirit of open science (*Stodden, 2013a*; *Stodden, 2013b*; *Stodden, 2013c*), our software reflects a systems approach. For instance, our claims can be verified by running our software, which comes with several test cases backed by a large test coverage. The software itself is scalable and its user interface is intuitive, requiring only a basic familiarity with the theory, and our Supplemental Information contain an illustrated stepwise instruction manual. The present framework augments our earlier framework for exact algorithms (*Tewari & Spouge, 2012*), also with a systems approach, adding another tool to the likelihood analysis for population genetics data under the infinite sites model of mutation.

ACKNOWLEDGEMENTS

It is our pleasure to acknowledge helpful conversations with Dr. Sergey Sheetlin and valuable feedback from Prof. Wu Yufeng.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This research was funded in part by National Institutes of Health (NIH); Intramural Research Program of the NIH, National Library of Medicine. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
National Institutes of Health.
Intramural Research Program of the NIH, National Library of Medicine.

Competing Interests

The authors declare there are no competing interests.

Author Contributions

- Susanta Tewari conceived and designed the experiments, performed the experiments, analyzed the data, contributed reagents/materials/analysis tools, wrote the paper, prepared figures and/or tables, reviewed drafts of the paper, development of the framework.
- John L. Spouge conceived and designed the experiments, analyzed the data, wrote the paper, reviewed drafts of the paper.

Data Availability

The following information was supplied regarding the availability of data:
http://coalescent.sourceforge.net/.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj.1203#supplemental-information.

REFERENCES

Ewing G, Hermisson J. 2010. MSMS: a coalescent simulation program including recombination, demographic structure and selection at a single locus. *Bioinformatics* **26**:2064–2065 DOI 10.1093/bioinformatics/btq322.

Felsenstein J, Kuhner MK, Yamato J, Beerli P. 1999. Likelihoods on coalescents: a monte carlo sampling approach to inferring parameters from population samples of molecular data. In: *Statistics in molecular biology and genetics, IMS lecture notes—monograph series,* vol. 33. 163–185.

Freeman S, Pryce N. 2009. *Growing object-oriented software guided by tests.* 1st edition. Addison-Wesley Professional.

Gamma E, Helm R, Johnson R, Vlissides J. 1995. *Design patterns elements of reusable object-oriented software.* 1st edition. Addison-Wesley.

Griffiths RC, Tavare S. 1994. Ancestral inference in population genetics. *Statistical Science* **9**:307–319 DOI 10.1214/ss/1177010378.

Gusfield D. 1991. Efficient algorithms for inferring evolutionary trees. *Networks* **21(1)**:19–28 DOI 10.1002/net.3230210104.

Hammersley JM, Handscomb DC. 1964. *Monte Carlo methods.* London: Methuen.

Hein J, Schierup MH, Wiuf C. 2005. *Gene genealogies, variation and evolution: a primer in coalescent theory.* 1st edition. Oxford University Press.

Hobolth A, Uyenoyama MK, Wiuf C. 2008. Importance sampling for the infinite sites model. *Statistical Applications in Genetics and Molecular Biology* **7(1)** DOI 10.2202/1544-6115.1400.

Hudson RR. 2002. Generating samples under a Wright–Fisher neutral model. *Bioinformatics* **18**:337–338 DOI 10.1093/bioinformatics/18.2.337.

Kimura M. 1969. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics* **61(4)**:893–903.

Limjap J. 2008. What is a reasonable code coverage % for unit tests (and why)? stackoverflow. *Available at http://stackoverflow.com/a/90021*.

Liu JS. 2002. *Monte Carlo strategies in scientific computing, Springer series in statistics.* New York: Springer.

McMorris F. 1977. On the compatibility of binary qualitative taxonomic characters. *Bulletin of Mathematical Biology* **39**:133–138 DOI 10.1007/BF02462853.

Robert CP, Casella G. 2004. *Monte Carlo statistical methods.* New York: Springer.

Stephens M, Donnelly P. 2000. Inference in molecular population genetics. *Journal of the Royal Statistical Society. Series* 605–635 DOI 10.1111/1467-9868.00254.

Stodden V. 2013a. Setting the default to reproducible. In: *Computational science research. Available at http://www.siam.org/news/news.php?id=2078*.

Stodden V. 2013b. Changes in the research process must come from the scientific community, not federal regulation. *Available at http://blog.stodden.net/2013/09/24/changes-in-the-research-process-must-come-from-the-scientific-community-not-federal-regulation/*.

Stodden V. 2013c. Toward reproducible computational research: an empirical analysis of data and code policy adoption by journals. *PLoS ONE* **8**(**6**):e67111 DOI 10.1371/journal.pone.0067111.

Tewari S, Spouge JL. 2012. Coalescent: an open-source and scalable framework for exact calculations in coalescent theory. *BMC Bioinformatics* **13**:257 DOI 10.1186/1471-2105-13-257.

UML-Wiki. Wikipedia article on unified modeling language. *Available at https://en.wikipedia.org/wiki/Unified_Modeling_Language*.

Wakeley J. 2009. *Coalescent theory. An introduction*. Greenwood Village: Roberts and Company Publishers.

Watterson GA. 1975. On the number of segregating sites in genetical models without recombination. *Theoretical Population Biology* **7**:256–276 DOI 10.1016/0040-5809(75)90020-9.

Wu Y. 2009. Exact computation of coalescent likelihood for panmictic and subdivided populations under the infinite sites model. *IEEE Transactions On Computational Biology And Bioinformatics* **7**:611–618.