Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Application Notes

OXFORD

Subject Section

ftprime: Recording the pedigree: efficient simulation of whole genomes

Corresponding Author [1,*], Co-Author [2] and Co-Author [2,*]

[1]Department, Institution, City, Post Code, Country and
[2]Department, Institution, City, Post Code, Country.

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

To use genomic data for inference and prediction it is often necessary to obtain whole-genome information from individual-based simulations, but the computational burden of endowing each simulated individual with an entire genome can be substantial. In this note we describe how to both (a) dramatically reduce this burden and (b) efficiently record the entire history of the population. We do this by simulating only those loci that may affect reproduction (those having non-neutral variants), and recording the entire history of genetic inheritance in an efficient data structure, on which neutral mutations can be quickly placed afterwards. *make more clear data structure was already developed? refer to 'tree sequence' by name?* The algorithm is implemented in python, and is designed to be easily used by any forwards-time simulation software. **Availability and implementation:**
Contact:
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

In recent years the increase in computing power – gradual only in comparison to the increase in our ability to sequence genomes – has made it possible to simulate the evolution of whole genomes of realistically-sized populations. This major milestone promises to remove the field's reliance on approximations of unknown applicability. Thus far, much of our understanding of how genomes evolve comes from models of one or a few linked loci (**?**), and/or results derived under the assumption of neutrality. These are certainly important, but researchers today have widely differing views on the practical importance of widespread selection (**???**). Thus far what little analytical progress has been made on models of ubiquitous selection are hard to check due to a lack of realistic simulation.

Simulations of organisms endowed with functional, varying genomes interacting with each other and spatially and temporally varying environments will not only allow us to develop and test our understanding of evolution, but also to produce quantitative predictions of population dynamics – for instance, the spread of insecticide resistance and behavior modification of mosquitos.

The most commonly used simulation methods to date rely on neutrality, as this assumption coupled with random mating allows the use of coalescent methods (**?**). These drastically reduce the amount of required computation, because they exploit a time-reversal duality of such models to *only* simulate the portion of history that determines the genomes in the modern population. How does this work? Suppose that to the *population pedigree* – the entire history of parent-offspring relationships of an entire population going back to a remote time – we add information encoding the genetic outcomes of each ancestral meiosis – who inherited which parts of which parental chromosomes. This embellished graph is known as the *ancestral recombination graph*, or ARG (**?**). Combined with ancestral genotypes and the origins of new mutations, it completely specifies the genomic sequence of any individual in the population at any time. However, much less than the entire ARG is needed to specify relationships between any given set of samples – only those portions of it from which those samples have actually inherited, back to their most recent common ancestors. For instance, *add diagram of simple example?* The assumptions of coalescent theory – random mating and neutrality – imply that the stochastic law of this random set forms a Markov process looked at *backwards* in time, and can hence be simulated without reference to the unnecessary remainder. Although this point was in principle known since **?**, only recent algorithmic advances made it possible to actually simulate the process correctly, without approximation, across whole genomes (**?**).

These efficient methods are not available if either key assumption of coalescent theory – neutrality or random mating – are broken. (Extensions

can be made for only a few loci under selection (**??**).) Forwards-time, individual-based simulation, however, is burdened by recording and passing on entire genomes, which typically have tens to hundreds of millions of varying sites. Such simulations must explicitly record any non-neutral genetic variants, because these affect the demographic process. However, neutral variants, by definition, do not, and so if the demographic history is recorded they can be simulated afterwards.

Wrap-up paragraph. This is what we do, which lets us both record history and be more efficient.

2 Methods and Implementation

Since msprime provides a number of tools for manipulating tree sequences, adding mutations, and producing output (e.g., VCF files) we need only determine a method for properly recording history and passing it to msprime.

2.1 msprime: Tree sequences and coalescence records

First we recall the data storage methods used by msprime (see **?**, for more details).

To record the history and identity of a set of sampled genomes, it is only necessary to record commmon ancestry (or, *coalescence*) events – most recent common ancestors of distinct segments of sampled genomes – and the identity and location of any mutations that these most recent common ancestors carry that are not shared by any other recorded ancestor. *(not quite right)*

The topology of the ARG is recorded by msprime as a list of Coalescence Records, which are tuples of the form `(left,right,parent,children)`, which indicates that the individuals listed in `children` have inherited the segment of genome between `left` and `right` from the ancestor `parent`. Individuals are identified by integers, and there is also a table recording the *birth time* of each individual.

For a set of coalescence records to be consistent, it is required that:

1. The birth time of any `children` must be strictly after the birth time of the `parent`.
2. The set of intervals on which individual `a` is a child must be disjoint, for every `a`.
3. The set of intervals on which individual `a` is a parent must be disjoint, for every `a`.

These ensure that the coalescence records unambiguously specify a subset of the genealogy at any location on the genome, but do not ensure that the entire history of the samples is specified. Furthermore, to allow for efficient algorithms, msprime also requires that

4. The samples must be numbered `0,...,n-1`, and the smallest non-sampled label must be an internal node. *perhaps remove this one*
5. The list of `children` in a coalescence record must be sorted.
6. Records must be sorted in nondecreasing order by birth time of the `parent`.
7. Birth times must be strictly greater than zero.

2.2 Recording history

Simulation software often gives a unique identifier to each *diploid* individual, but the natural exchangeable unit of consideration for the ARG (and msprime) is gamete – so, each diploid individual is the union of two haploid individuals: one consisting of the set of chromosomes inherited maternally, the other paternally. To translate between the two, we adopt the convention that the maternally inherited set of chromosomes of diploid individual k is given haploid ID $2k$, and the paternally inherited set $2k+1$. ID numbers referred to below are haploid IDs.

Since we do not know from which ancestors the eventual genetic samples will inherit, to record the history of the samples we must record the results of all reproduction events – the entire ARG. To do this, and conform with the format above, we maintain an ordered dictionary `arg`, with one entry for each parent, whose entries are of the form `id:(time,records)`, where `id` is the integer ID of the parent, `time` is the birth time of the parent, and `records` is a list of nonoverlapping coalescence records, sorted by left endpoint. We update this structure each time a new individual is born. Suppose that individual I is born at time t to parents p_0 and p_1, with recombination breakpoints $0 = x_0 < x_1 < x_2 < \cdots < x_n \leq L$, so that I inherits the genomic segment $[0, x_1)$ from p_0, the segment $[x_1, x_2)$ from p_1, and so forth (here $n \geq 1$), and where L is the total length of the chromosome. When this occurs, we:

1. append a new entry `I:(t,[])` to the list `arg`; and
2. for each $1 \leq k \leq n$, `merge` the new record `(x[k-1],x[k],p[k mod 2],(I,))` into the existing records in 'arg[p[k mod 2]]'.

The second step requires we merge the new record with existing ones so as to retain the criteria described above. To `merge` a record saying that individual `I` also inherits from this `parent` on interval `[a,b)` into an existing set of records of the form `(left,right,parent,children)`, we need to

1. split any record for which `left<a<right` or `left<b<right` into two at the corresponding endpoint,
2. and then add `I` to `children` for any intervals overlapping `[a,b)`.

This is done easily, maintaining the fact that records are not overlapping and are maintained in sorted order. *could alternatively write out algorithm but it is a bit nitpicky*

3 Results and Discussion

4 Conclusion

Acknowledgements

Funding