

1 **Title:**

2 Tree-sequence recording in SLiM opens new horizons for forward-time simulation of whole
3 genomes

4 **Running Title:** Tree-sequence recording in SLiM

5 **Authors:**

6 Benjamin C. Haller ‡
7 Dept. of Biological Statistics and Computational Biology
8 Cornell University, Ithaca, NY 14853, USA

9

10 Jared Galloway
11 Institute of Ecology and Evolution
12 University of Oregon, Eugene, OR 97403, USA

13

14 Jerome Kelleher
15 Big Data Institute, Li Ka Shing Centre for Health Information and Discovery
16 University of Oxford
17 Oxford, OX3 7FZ, UK

18

19 Philipp W. Messer *
20 Dept. of Biological Statistics and Computational Biology
21 Cornell University, Ithaca, NY 14853, USA

22

23 Peter L. Ralph ‡ *
24 Institute of Ecology and Evolution
25 University of Oregon, Eugene, OR 97403, USA

26

27 ‡ Corresponding authors

28 * Joint senior authors

29 **Corresponding Authors:**

30 Benjamin C. Haller, bhaller@benhaller.com

31 Peter Ralph, plr@uoregon.edu

32 **Abstract**

33 There is an increasing demand for evolutionary models to incorporate relatively realistic
34 dynamics, ranging from selection at many genomic sites to complex demography, population
35 structure, and ecological interactions. Such models can generally be implemented as individual-
36 based forward simulations, but the large computational overhead of these models often makes
37 simulation of whole chromosome sequences in large populations infeasible. This situation
38 presents an important obstacle to the field that requires conceptual advances to overcome. The
39 recently developed tree-sequence recording method (Kelleher et al., 2018), which stores the
40 genealogical history of all genomes in the simulated population, could provide such an advance.
41 This method has several benefits: (1) it allows neutral mutations to be omitted entirely from
42 forward-time simulations and added later, thereby dramatically improving computational
43 efficiency; (2) it allows neutral burn-in to be constructed extremely efficiently after the fact, using
44 “recapitation”; (3) it allows direct examination and analysis of the genealogical trees along the
45 genome; and (4) it provides a compact representation of a population’s genealogy that can be
46 analyzed in Python using the msprime package. We have implemented the tree-sequence
47 recording method in SLiM 3 (a free, open-source evolutionary simulation software package) and
48 extended it to allow the recording of non-neutral mutations, greatly broadening the utility of this
49 method. To demonstrate the versatility and performance of this approach, we showcase several
50 practical applications that would have been beyond the reach of previously existing methods,
51 opening up new horizons for the modeling and exploration of evolutionary processes.

52 **Keywords**

53 pedigree recording, coalescent, background selection, genealogical history, selective sweeps, tree
54 sequences

55 **Introduction**

56 Forward simulations are increasingly important in population genetics and evolutionary biology.
57 For example, they can be useful for modeling the expected evolutionary dynamics of real-world
58 systems (Fournier-Level et al., 2016; Cotto et al., 2017; Matz et al., 2018; Ryan et al., 2018), for
59 discovering the ecological and evolutionary mechanisms that led to present-day genomic patterns
60 in a species (Enard et al., 2014; Nowak et al., 2014; Arunkumar et al., 2015; Patel et al., 2018),
61 for testing or validating empirical and statistical methods (Haller and Hendry, 2013; Caballero et
62 al, 2015; Ewing et al., 2016; Haller and Messer, 2017a), and for exploring theoretical ideas about
63 evolution (Haller et al., 2013; Assaf et al., 2015; Mafessoni and Lachmann, 2015; Champer et. al,
64 2018), among other purposes. Because of this broad utility, there is a growing desire to run
65 simulations with increased realism in a variety of areas: longer genomic regions up to the scale of
66 full genome sequences, large populations, selection at multiple loci with linkage effects, complex
67 demography, ecological interactions with other organisms and the environment, explicit space
68 with continuous landscapes, spatial variation in environmental variables, spatial interactions such
69 as competition and mate choice between organisms, and so forth.

70 However, this type of realism comes at a price, in both processing time and memory usage.
71 Since computational resources are finite, this can often make it difficult or, in practical terms,
72 impossible to run some models. Advances in computing power have gradually extended the
73 boundaries of what is possible, as have performance improvements due to improved forward
74 simulation software (Messer, 2013; Thornton, 2014; Haller and Messer, 2017b), but
75 computational overhead continues to hold back progress in the field by limiting the level of
76 realism that can be attained in models.

77 From this perspective, the recently developed pedigree recording or “tree-sequence recording”
78 method (Kelleher et al., 2018) is potentially transformative. Kelleher et al. (2018) have shown
79 that, perhaps counterintuitively, the recording of all ancestry information for the entire population
80 can actually improve the runtime by orders of magnitude. These gains in efficiency are made
81 possible by the succinct tree sequence data structure (or “tree sequence”, for brevity) that lies at
82 the heart of the msprime coalescent simulator (Kelleher et al., 2016), subsequently refined in
83 Kelleher et al. (2018). The tree sequence data structure is a concise encoding of the correlated
84 genealogies along a chromosome resulting from evolution in sexually reproducing populations
85 (Figure 1). The sequence of trees along a genome has been studied for some time (Hudson,
86 1983), and is closely linked to the concept of an “Ancestral Recombination Graph” or ARG
87 (Griffiths, 1991; Griffiths & Marjoram, 1997). The use of the term “ARG” has historically been
88 ambiguous, however, sometimes referring to the stochastic process generating these trees, rather
89 than the resulting tree sequence itself, so we use the term “tree sequence” here to refer to this
90 sequence of trees in the particular representation described by Kelleher et al. (2016, 2018).
91 Precisely the same tree sequence data structure can be used to record each generation’s parent–
92 child relationships. This data structure will then record who each individual inherited each
93 section of chromosome from, for every individual that ever lived. However, there is a massive
94 amount of redundancy in this information, since many of the individuals simulated in the past
95 will leave no descendants in the extant population. The key insight of Kelleher et al. (2018) was
96 to provide an efficient algorithm to remove this redundancy by periodically “simplifying” the tree
97 sequence. This combination – the tree sequence data structure and an efficient algorithm for
98 simplifying it – allows complete genealogies for all extant individuals to be recorded efficiently
99 in forward simulations for the first time.

100 The most immediate advantage of recording a tree sequence during forward simulation is that it
101 allows neutral mutations to be omitted entirely; neutral mutations can simply be overlaid onto the
102 tree sequence after forward simulation has completed, because by definition they do not affect the
103 genealogies. This provides an immense efficiency benefit, since neutral mutations then only need
104 to be added along those branches of the tree from which the individuals of interest at the end of
105 the simulation have inherited; all other ancestral branches, which typically comprise the vast
106 majority of the full tree, can be ignored since they do not contribute to those individuals. Given
107 that many forward simulations spend the large majority of their time managing neutral mutations,
108 with considerable bookkeeping overhead in each generation, neutral mutation overlay following
109 forward simulation has been shown to improve performance by an order of magnitude or more
110 while producing provably statistically identical results (Kelleher et al., 2018).

111 A second advantage of recording genealogies is that the recorded tree sequence from a forward
112 simulation can be used as the basis for the construction of a neutral “burn-in” history for the
113 simulated population after forward simulation is complete, using (usually much faster) coalescent
114 simulation. The burn-in period of a simulation can be immensely time-consuming, often taking
115 much longer than the simulation of the evolutionary dynamics that are actually of interest; the
116 overhead of burn-in can therefore present a large obstacle for many models. With a method that
117 we call “recapitation”, we can leverage the information in the tree sequence to prepend a
118 coalescent simulation of the burn-in period, speeding up the burn-in process by many orders of
119 magnitude.

120 A third important advantage is that the pattern of ancestry and inheritance is in itself very
121 useful. For many statistics of interest, and in particular for inferring specific events that occurred

122 in the past, sequence-based data from mutations is essentially an extra layer of noise over the
123 signal of interest contained in the genealogies. Direct access to the precise genealogical history
124 of the simulated population allows the signal to be analyzed without the noise, gaining significant
125 statistical power. An expanding set of open-source tools makes it possible to load, analyze, and
126 even manipulate a recorded tree sequence using simple Python code, allowing open-ended
127 flexibility in analysis.

128 A fourth compelling advantage is that the recorded tree sequence files are very small and
129 enable very efficient calculation of population-genetic statistics (Kelleher et al. 2016, 2018). The
130 files output from even the largest simulations are rarely bigger than a few hundred megabytes,
131 and may be tens of thousands of times smaller than alternatives such as VCF and Newick.
132 Despite this high level of compression, tree sequences can be processed very efficiently; statistics
133 of interest such as allele frequencies within cohorts can often be computed incrementally, leading
134 to very efficient algorithms (Kelleher et al. 2016). Calculation of statistics of this sort from
135 simulated data can be very time-consuming, especially when long genomes are involved and
136 many replicate simulation runs have been performed, so the ability to speed up such calculations
137 is quite important.

138 Given these advantages, we have worked to integrate tree-sequence recording into SLiM 3, a
139 new major release of the free, open-source SLiM simulation software package
140 (<http://messerlab.org/slim/>). It is now possible to enable tree-sequence recording in any SLiM
141 model with a simple flag set in the model's script, and then to output the recorded tree sequence
142 at any point in the simulation. In addition, we have extended the original tree-sequence recording
143 method (Kelleher et al. 2018) to allow for the recording of mutations during forward simulation.

144 This allows the tree-sequence output format, a `.trees` file, to be used in SLiM as a way of saving
145 and then restoring the state of a simulation while preserving information about ancestry, and
146 allows the mutations that occurred during forward simulation to be accessed later in Python-based
147 analyses.

148 To illustrate the large advantages provided by tree-sequence recording, and to show how to
149 take advantage of those benefits when using SLiM for forward simulation, we will present four
150 practical examples of the method. In the first example, we will show the impressive performance
151 benefits that can be achieved with tree-sequence recording compared to a classical forward
152 simulation. The second example will use tree-sequence recording to efficiently simulate
153 background selection near genes undergoing deleterious mutations, quantifying the expected
154 effect of background selection on levels of neutral diversity by measuring the heights of trees in
155 the recorded tree sequence. Our third example will be a model of admixture between two
156 subpopulations, showing how to use the recorded tree sequence in calculating the mean true local
157 ancestry at every position along a chromosome. Finally, the fourth example will illustrate how
158 the “recapitation” method allows msprime to be used to extremely efficiently add a “neutral burn-
159 in” history to a completed SLiM simulation of a selective sweep, by coalescing the simulation’s
160 initial population backward in time.

161 **Examples**

162 Examples were executed on a MacBook Pro (2.9 GHz Intel Core i7, 16 GB RAM) running
163 macOS 10.13.5, using Python 3.4.8, R 3.5.0, SLiM 3.1, msprime 0.6.1, and pyslim 0.1. Reported
164 times were measured with the Python `timeit` package. Peak memory usage for SLiM runs was
165 assessed with SLiM’s `-m` command-line option. The timing comparison (Figure 2) was executed

166 on the same hardware, with macOS 10.13.4, R 3.4.3, SLiM 3.0, and msprime 0.6.0, using the
167 Unix tool `/usr/bin/time` for timing (summing the reported user time and system time); we
168 believe the times measured would not change significantly with the newer software versions. The
169 full source code for the examples and timing tests, including timing and plotting code that is
170 omitted here, may be found at <https://github.com/bhaller/SLiMTreeSeqPub>. These examples use
171 the `matplotlib` (Hunter, 2007) and `numpy` (Oliphant, 2006) packages for Python.

172 *Example I: A simple neutral model*

173 Our first example is a model of a neutrally evolving chromosome of length $L = 10^8$ base
174 positions, with uniform mutation rate $\mu = 10^{-7}$ and recombination rate $r = 10^{-8}$ (both expressed as
175 the event probability per base per generation), in a panmictic diploid population of size $N = 500$,
176 running for a duration of $10N = 5000$ non-overlapping generations. The SLiM configuration
177 script for this basic model is very simple:

```
178     initialize() {  
179         initializeMutationRate(1e-7);  
180         initializeMutationType("m1", 0.5, "f", 0.0);  
181         initializeGenomicElementType("g1", m1, 1.0);  
182         initializeGenomicElement(g1, 0, 1e8-1);  
183         initializeRecombinationRate(1e-8);  
184     }  
185     1 {  
186         sim.addSubpop("p1", 500);  
187     }  
188     5000 late() {  
189         sim.outputFull("ex1_noTS.slimbinary", binary=T);  
190     }
```

191 This sets up a single “genomic element” spanning the full length of the chromosome, with
192 neutral mutations of type `m1` generated at the desired rate, and with the desired recombination
193 rate. In generation 1 a new subpopulation of the desired size is created, and the model runs to
194 generation 5000, after which it outputs the full simulation state. The SLiM manual provides

195 additional explanation of these concepts (Haller and Messer, 2016). This model took 211.9
196 seconds to run, and reached a peak memory usage of 443.8 MB.

197 Tree-sequence recording can easily be enabled for this model with a call to
198 `initializeTreeSeq()`:

```
199     initialize() {  
200         initializeTreeSeq();  
201         initializeMutationRate(0);  
202         initializeMutationType("m1", 0.5, "f", 0.0);  
203         initializeGenomicElementType("g1", m1, 1.0);  
204         initializeGenomicElement(g1, 0, 1e8-1);  
205         initializeRecombinationRate(1e-8);  
206     }  
207     1 {  
208         sim.addSubpop("p1", 500);  
209     }  
210     5000 late() {  
211         sim.treeSeqOutput("ex1_TS.trees");  
212     }
```

213 Note that we have now also set the mutation rate to zero; SLiM no longer needs to model
214 neutral mutations because they can be overlaid in a later step more efficiently. A `.trees` file is
215 output at the end of the run, instead of calling SLiM's `outputFull()` method, so that the recorded
216 tree sequence is preserved. In all other respects these models are identical. This is typical of
217 adapting a SLiM model to use tree-sequence recording: in general, the aim is to remove the
218 modeling of neutral mutations while preserving other aspects of the model verbatim.

219 After simulation has completed, neutral mutations are overlaid upon the saved tree sequence.
220 The full model – running the SLiM model and then doing the final mutation overlay step – can be
221 executed with a simple Python script:

```
222     import subprocess, msprime, pyslim  
223  
224     # Run the SLiM model  
225     subprocess.check_output(["slim", "-m", "-s", "0", "ex1_TS.slim"])  
226  
227     # Overlay neutral mutations  
228     ts = pyslim.load("ex1_TS.trees")  
229     mutated = msprime.mutate(ts, rate=1e-7, random_seed=1, keep=True)  
230     mutated.dump("ex1_TS_overlaid.trees")
```

231 This script uses the msprime Python package to overlay neutral mutations upon the recorded
232 tree sequence. The result is precisely the same, statistically, as if the neutral mutations were
233 included in the forward simulation, except that the vast majority of the bookkeeping work in each
234 generation is avoided because mutations only need to be overlaid upon the ancestral genomic
235 regions that persisted to the end of the simulation.

236 Note that pyslim is used to load the `.trees` file; this package provides a bridge between SLiM
237 and msprime, and should generally be used to load and save `.trees` files in Python if the files are
238 coming from or going to SLiM. The pyslim package extends the msprime tree sequence class by
239 adding support for SLiM's metadata annotations to the tree sequence, providing an interface for
240 reading or modifying that metadata as well as for generating SLiM-compliant `.trees` files that
241 contain the required metadata. The `.trees` files output by SLiM can be read directly by msprime,
242 but the returned object will have reduced functionality compared to those returned by pyslim.

243 The total time to execute this Python code is 4.37 seconds, almost 50 times faster than the
244 model without tree-sequence recording. Most of the runtime (4.09 seconds) is spent running the
245 SLiM model; the final mutation overlay by msprime is extremely fast. The peak memory usage
246 during the SLiM run is 145.8 MB, less than one-third of the memory usage of the model without
247 tree-sequence recording. Tree-sequence recording can often reduce memory usage, since the tree
248 sequence data structure is quite compact compared to SLiM's in-memory representation of the
249 neutral mutations that would be segregating in such a model. Tree sequences are also very
250 compact on disk; the final `.trees` file here, with mutations overlaid, takes about 8.9 MB, as
251 compared to 84.2 MB for the `ex1_noTS.slimbinary` file from the SLiM model without tree-
252 sequence recording, 559 MB for a Newick file, and 366 MB for a VCF file – even though the

253 .trees file contains ancestry information not included by the SLiM and VCF formats. A VCF
254 file containing the sequences of the final generation can be produced from a .trees file with
255 msprime's `write_vcf()` method, but the ancestry information is lost.

256 The speedup produced by this tree-sequence recording method can vary dramatically
257 depending upon the details of the simulation; all of the work to track neutral mutations is
258 eliminated, but new work is added involving the recording of all the recombination events that go
259 into producing the tree sequence. In general, the largest speedup will be observed with very long
260 chromosomes with many neutral mutations when the recombination rate is not too high; indeed,
261 when modeling a very short chromosome the overhead of tree-sequence recording can outweigh
262 the savings from omitting neutral mutations (see Discussion).

263 To further illustrate the performance benefits of tree-sequence recording, we conducted a set of
264 timing comparisons between SLiM without tree-sequence recording, SLiM with tree-sequence
265 recording, and msprime's coalescent simulation method. These comparisons involved essentially
266 the same model as shown above: a neutral panmictic model of diploids with non-overlapping
267 generations, with a population size $N = 500$, recombination rate $r = 10^{-8}$ per base position per
268 generation, and mutation rate $\mu = 10^{-7}$ per base position per generation. The chromosome length
269 L was varied over $\{10^5, 10^6, 10^7, 10^8, 10^9, 10^{10}\}$, with ten runs of each model at each value of L
270 using different random seeds. The number of generations varied with L (details below). The
271 msprime coalescent was run both with a final haploid sample size n equal to the full population
272 size ($n = 2N$), and with a much smaller sample size ($n = 2N/100$); in both cases, $N_e = N$ was used.
273 To verify that tree-sequence recording produced results equivalent to the coalescent, we checked

274 that the mean TMRCA for the $L = 10^{10}$ runs for the two methods did not differ significantly
275 ($p = 0.7791$).

276 The average runtimes obtained are shown in Figure 2. As L increased, the benefit of tree-
277 sequence recording compared to SLiM without tree-sequence recording became increasingly
278 large, topping out at a performance improvement of more than two orders of magnitude for
279 $L = 10^9$ and $L = 10^{10}$. Coalescent simulations with msprime were much faster than the tree-
280 sequence recording method, as expected, except at $L = 10^{10}$, where msprime's speed was
281 comparable to that of SLiM with tree-sequence recording. It appears that SLiM with tree-
282 sequence recording would be faster for L larger than 10^{10} . The number of events the coalescent
283 must simulate is quadratic in L , empirically, but with a small leading coefficient such that
284 msprime is quite fast even for reasonably large chromosome sizes (Kelleher et al. 2016). With
285 very large values of L , however, this $O(L^2)$ term begins to dominate and SLiM with tree-sequence
286 recording becomes faster. This may be chiefly of theoretical interest, since $L = 10^{10}$ is already a
287 very long chromosome (approximately three times the length of the full human genome). It is
288 also noteworthy that the msprime coalescent is only marginally faster for a sample of $n = 2N/100$
289 than for a full population sample of $n = 2N$; as more samples are added to a gene tree, the new
290 samples tend to attach to already existing branches quite quickly (Kingman, 1982).

291 Although the coalescent remains an order of magnitude faster for most practical purposes, it
292 can only be used in a few simple scenarios such as this; for models that require forward
293 simulation, tree-sequence recording offers large performance benefits over more traditional
294 forward simulation techniques. It is also worth noting that the coalescent is only an
295 approximation of the Wright–Fisher model, and will diverge from it under certain conditions

296 (Wakeley et al., 2012; Bhaskar et al., 2014) – one such condition being a sample size that is no
297 longer small compared to the population size, as is the case for our $n = 2N$ msprime runs here.
298 Forward simulation may therefore be preferable in order to obtain exact results under such
299 conditions.

300 **How long do we run it?** In general, it is desirable to run forward-time simulations “until
301 convergence” – until the effects of the starting configuration are forgotten. This occurs (in most
302 situations) when all genealogical trees have coalesced, meaning that at every position in the
303 genome a common ancestor to the entire final generation has appeared. In practice, models are
304 often run for $10N$ generations, a rule of thumb that is thought to suffice in most cases. However,
305 this is a thorny problem: longer chromosomes tend to require longer for coalescence, simply
306 because with more sites it is more likely that coalescence takes exceptionally long at some site.
307 In the simulations of Figure 2, we ran each simulation for the expected number of generations
308 required for coalescence at that value of L , which increased linearly with $\log(L)$, from about $3N$
309 for $L = 1e5$ to $15N$ for $L = 1e10$. This sufficed to make the comparison between SLiM and
310 msprime “fair”, but a better practical solution, recapitation, will be shown in Example 4. We
311 determined the expected number of generations empirically by running the same model 500 times
312 at each value of L with “coalescence detection” enabled (by passing `checkCoalescence=T` to
313 `initializeTreeSeq()`). The mean and other summary statistics for each value of L (Table S1)
314 agree with expectations from extreme value theory (Berman, 1964), with the expected time until
315 coalescence growing roughly as $1000 \log(L) - 10000$.

316 *Example II: Background selection*

317 Our second example is a model of background selection, a term which describes the effect that
318 purifying selection against deleterious mutations imposes on genetic variation at linked sites.
319 Such purifying selection should be particularly common in genic regions, where many genomic
320 positions should be subject to selective constraints. This background selection, like many types
321 of linked selection more generally, is expected to produce a “dip in diversity” in the surrounding
322 non-coding regions, with a signature of decreasing genetic diversity with decreasing distance to
323 the nearest gene (Charlesworth et al. 1993; Hudson 1994; Sattath et al., 2011; Elyashiv et al.,
324 2016). Here is a SLiM model that uses tree-sequence recording to model this scenario:

```
325     initialize() {  
326         defineConstant("N", 10000); // pop size  
327         defineConstant("L", 1e8); // total chromosome length  
328         defineConstant("L0", 200e3); // between genes  
329         defineConstant("L1", 1e3); // gene length  
330         initializeTreeSeq();  
331         initializeMutationRate(1e-7);  
332         initializeRecombinationRate(1e-8, L-1);  
333         initializeMutationType("m2", 0.5, "g", -(5/N), 1.0);  
334         initializeGenomicElementType("g2", m2, 1.0);  
335  
336         for (start in seq(from=L0, to=L-(L0+L1), by=(L0+L1)))  
337             initializeGenomicElement(g2, start, (start+L1)-1);  
338     }  
339     1 {  
340         sim.addSubpop("p1", N);  
341         sim.rescheduleScriptBlock(s1, 10*N, 10*N);  
342     }  
343     s1 10 late() {  
344         sim.treeSeqOutput("ex2_TS.trees");  
345     }
```

346 This model sets up a chromosome that consists of genes of length $L_1 = 1$ kb, separated by non-
347 coding regions of length $L_0 = 200$ kb. The total chromosome length is $L = 10^8$ bases, and 496
348 genes fit within it. The model uses a mutation rate of $\mu = 10^{-7}$ for deleterious mutations that can
349 arise within the genes; no other mutations are modeled. The deleterious mutations are given
350 selection coefficients drawn from a Gamma distribution with mean $-5/N$ and shape parameter
351 $\alpha = 1$ (modeling a scenario of moderately deleterious mutations with $2Ns = -10$ on average). We

352 assume co-dominance with $h = 0.5$. A population of size $N = 10000$ is started in generation 1,
353 and the model runs until generation $G = 10N$ (the output event, $s1$, is rescheduled dynamically to
354 that generation).

355 We can run this model and then conduct post-run analysis with a Python script:

```
356 import os, subprocess, msprime, statistics, pyslim
357 import matplotlib.pyplot as plt
358 import numpy as np
359
360 # Run the SLiM model and load the resulting .trees file
361 subprocess.check_output(["slim", "-m", "-s", "0", "ex2_TS.slim"])
362 ts = pyslim.load("ex2_TS.trees").simplify()
363
364 # Measure the tree height at each base position
365 height_for_pos = np.zeros(int(ts.sequence_length))
366 for tree in ts.trees():
367     mean_height = statistics.mean([tree.time(root) for root in tree.roots])
368     left, right = map(int, tree.interval)
369     height_for_pos[left: right] = mean_height
370
371 # Convert heights along the chromosome into heights at distances from a gene
372 height_for_pos = height_for_pos - np.min(height_for_pos)
373 L, L0, L1 = int(1e8), int(200e3), int(1e3)
374 gene_starts = np.arange(L0, L - (L0 + L1) + 1, L0 + L1)
375 gene_ends = gene_starts + L1 - 1
376 max_distance = L0 // 4
377 height_for_left_distance = np.zeros(max_distance)
378 height_for_right_distance = np.zeros(max_distance)
379 for d in range(max_distance):
380     height_for_left_distance[d] = np.mean(height_for_pos[gene_starts - d - 1])
381     height_for_right_distance[d] = np.mean(height_for_pos[gene_ends + d + 1])
382 height_for_distance = np.hstack([height_for_left_distance[::-1],
383     height_for_right_distance])
384 distances = np.hstack([np.arange(-max_distance, 0), np.arange(1, max_distance + 1)])
385
386 # Make a simple plot
387 plt.plot(distances, height_for_distance)
388 plt.show()
```

389 The first line after the import statement runs the SLiM model; this took 15643 seconds (4.35
390 hours) to execute. This is not short – it is still a fairly complex model! – but it is far shorter than
391 the alternative, a SLiM model without tree-sequence recording and including neutral mutations in
392 the non-coding regions. That alternative model would take ~83 hours, by extrapolation –
393 probably a conservative estimate, since the model had not yet reached mutation–selection balance
394 and was still slowing down when its timing was measured. The use of tree-sequence recording

395 here results, then, in a relatively modest speedup of 19 times. This makes sense, since the model
396 with tree-sequence recording still must keep track of a very large number of segregating
397 deleterious mutations. However, it is worth noting that the final result from this alternative
398 model would provide far less statistical power, since inference from it would be based only upon
399 the observed pattern of neutral mutations in one run, rather than the actual pattern of ancestry at
400 each chromosome position; to provide the same power, this alternative model would likely have
401 to be run many times or use a much higher mutation rate. If more performance gains were
402 needed, the model could perhaps be rescaled as well (see Discussion).

403 The rest of the code conducts post-run analyses. First, the `.trees` file from the SLiM run is
404 read in with `pyslim.load()` as in the previous example; here, however, we call `simplify()`
405 (Kelleher et al. 2018) upon the loaded tree sequence, which requires some explanation. SLiM
406 automatically retains, in the tree sequence, nodes corresponding to the original ancestors of each
407 subpopulation that was created with `addSubpop()`. This is done for various reasons, including
408 allowing ancestry to be more easily traced and enabling recapitation (see Example 4). When
409 SLiM saves a `.trees` file, these ancestors are present in the tree sequence but are not marked as
410 “samples”, and will therefore disappear after a `simplify()` operation. In many cases these
411 ancestors are harmless, as in Example 1; in fact, in Example 1, calling `simplify()` to remove
412 them would mean that mutations would be overlaid only back to the point of coalescence, rather
413 than to the beginning of forward simulation. Here, however, since we want to measure the
414 heights of trees in the tree sequence, these ancestors would complicate things for us; all trees
415 would be rooted in those ancestors, at the beginning of forward simulation. We therefore call

416 `simplify()` to remove them (when the model has coalesced below them; they are retained when
417 still in use by the tree sequence). Example 4 will delve into this matter further.

418 Next, a vector containing the mean tree height at each base position (`height_for_pos`) is
419 constructed by walking through the tree sequence to find the set of trees representing the ancestry
420 of every individual in the final generation at a given position. The mean tree height is a metric of
421 the time to the most recent common ancestor at a given base position, and thus of diversity at that
422 base position; background selection will tend to reduce the mean tree height, thereby lowering the
423 expected levels of diversity at a locus.

424 An aside: there can be a set of trees for a given position, rather than just a single tree, if the
425 forward simulation was not run sufficiently long for coalescence to have occurred at every
426 position in the genome. In `msprime` this is modelled by allowing trees to have multiple roots.
427 Each root represents the most recent common ancestor of some subset of the extant population at
428 that location in the genome; if coalescence has not occurred, then the final population should still
429 contain genetic variation that was segregating in the initial population, since different individuals
430 inherit from different roots of the ancestry tree. Since the model here ran for $10N$ generations, we
431 can hope that it has coalesced at most or all positions; but unless a model is explicitly run out to
432 coalescence (or recapitated), it is always possible that multiple roots will exist, and so robust code
433 ought to handle that case by looping over the roots for each tree as we do here.

434 These mean tree heights along the chromosome are then converted to mean tree heights at
435 distances from the nearest gene (`height_for_distance`), taking into account the somewhat
436 complex genetic structure of the model. Finally, the relationship between distance to the nearest
437 gene and tree height is plotted. These analyses took 12.39 seconds to complete. Note that neutral

438 mutations were never simulated at all; the analysis is based upon the tree sequence itself, not
439 upon the distribution of neutral mutations.

440 A plot of the results can be seen in Figure 3, showing the well-known “dip in diversity”
441 realized here through simulation. As the distance to the nearest gene decreases, diversity dips due
442 to the background selection exerted by selection against deleterious mutations within the gene.

443 *Example III: True local ancestry mapping*

444 Our third example involves mapping the true local ancestry at every position along a
445 chromosome in a two-subpopulation admixture model with adaptive introgression at two partially
446 linked loci. This is an important dynamic in all sorts of biological systems, from human–
447 Neanderthal admixture to hybrid zones between divergent bird populations; one often wishes to
448 be able to find which ancestral population each chromosomal region traces back to. The SLiM
449 model looks like this:

```
450 initialize() {  
451     defineConstant("L", 1e8);  
452     initializeTreeSeq();  
453     initializeMutationRate(0);  
454     initializeMutationType("m1", 0.5, "f", 0.1);  
455     initializeGenomicElementType("g1", m1, 1.0);  
456     initializeGenomicElement(g1, 0, L-1);  
457     initializeRecombinationRate(1e-8);  
458 }  
459 1 late() {  
460     sim.addSubpop("p1", 500);  
461     sim.addSubpop("p2", 500);  
462     sim.treeSeqRememberIndividuals(sim.subpopulations.individuals);  
463  
464     p1.genomes.addNewDrawnMutation(m1, asInteger(L * 0.2));  
465     p2.genomes.addNewDrawnMutation(m1, asInteger(L * 0.8));  
466  
467     sim.addSubpop("p3", 1000);  
468     p3.setMigrationRates(c(p1, p2), c(0.5, 0.5));  
469 }  
470 2 late() {  
471     p3.setMigrationRates(c(p1, p2), c(0.0, 0.0));  
472     p1.setSubpopulationSize(0);  
473     p2.setSubpopulationSize(0);  
474 }  
475 2: late() {  
476     if (sim.mutationsOfType(m1).size() == 0)
```

```
477     {
478         sim.treeSeqOutput("ex3_TS.trees");
479         sim.simulationFinished();
480     }
481 }
482 10000 late() {
483     stop("Did not reach fixation of beneficial alleles.");
484 }
```

485 The `initialize()` callback sets up tree-sequence recording with a mutation rate of $\mu = 0$ and a
486 recombination rate of $r = 10^{-8}$ along a chromosome of length $L = 10^8$. Although the mutation rate
487 is zero, a mutation type `m1` is defined representing beneficial mutations with a selection
488 coefficient of $s = 0.1$; mutations of this type will be added in generation 1.

489 In generation 1 we create two subpopulations, `p1` and `p2`, of 500 individuals each; these are the
490 original subpopulations that will admix. We tell SLiM to remember these individuals forever as
491 ancestors in the tree sequence, with `treeSeqRememberIndividuals()`, because we want them to
492 act as the roots of all recorded trees so that we can establish local ancestry using them. Note that
493 this is not strictly necessary, since (as discussed in Example 2) SLiM automatically retains the
494 root ancestors for each population; we could rely upon that, and we would be fine as long as we
495 did not `simplify()` after loading the tree sequence in Python. The use of
496 `treeSeqRememberIndividuals()` has been shown here for purposes of illustration, however,
497 since some models may wish to remember non-root individuals for analysis. Next, we add a
498 beneficial mutation at $0.2L$ in `p1`, and another at $0.8L$ in `p2`; the expectation is that by the end of
499 the run all individuals will be recombinants that carry both of these mutations. Finally, we create
500 subpopulation `p3` and tell SLiM that it will be composed entirely of migrants from `p1` and `p2` in
501 equal measure.

502 By the end of generation 2, subpopulation `p3` has received its offspring generation from `p1` and
503 `p2` as intended, so we can now remove `p1` and `p2` from the model and allow `p3` to evolve. At this

504 stage, all individuals in p3 are still unmixed, having been generated from parents in either p1 or
505 p2, but beginning in generation 3 they will start to mix.

506 Finally, we have some output and termination code. If both m1 mutations fix, they are
507 converted to Substitution objects by SLiM, and when that is detected the model writes out a
508 final .trees file and terminates. If we reach generation 10000 without that happening, the
509 admixture failed, and we stop with an error. This model is conceptually similar to recipe 13.9 in
510 the SLiM manual (Haller and Messer, 2016), but has been converted to use tree-sequence
511 recording, so you can refer to the manual's recipe for additional commentary.

512 We can run this model from a Python script and do post-run analysis, as we did in Example 2:

```
513 import os, subprocess, msprime, pyslim
514 import matplotlib.pyplot as plt
515 import numpy as np
516
517 # Run the SLiM model and load the resulting .trees file
518 subprocess.check_output(["slim", "-m", "-s", "0", "ex3_TS.slim"])
519 ts = pyslim.load("ex3_TS.trees").simplify()
520
521 # Assess the true local ancestry at each base position
522 breaks = np.zeros(ts.num_trees + 1)
523 ancestry = np.zeros(ts.num_trees + 1)
524 for tree in ts.trees(sample_counts=True):
525     subpop_sum, subpop_weights = 0, 0
526     for root in tree.roots:
527         leaves_count = tree.num_samples(root) - 1 # subtract one for the root
528         subpop_sum += tree.population(root) * leaves_count
529         subpop_weights += leaves_count
530     breaks[tree.index] = tree.interval[0]
531     ancestry[tree.index] = subpop_sum / subpop_weights
532 breaks[-1] = ts.sequence_length
533 ancestry[-1] = ancestry[-2]
534
535 # Make a simple plot
536 plt.plot(breaks, ancestry)
537 plt.show()
```

538 The first line after the import statements runs the SLiM model, which completes in just 0.416
539 seconds, with peak memory usage of 55.6 MB; since it tracks only two mutations, and typically
540 terminates by generation 150 or so, it is very quick.

541 The equivalent SLiM model to achieve true local ancestry mapping without tree-sequence
542 recording has to model a mutation at each base position, as can be seen in recipe 13.9 in the SLiM
543 manual (Haller and Messer, 2016). A direct comparison is not possible, because recipe 13.9
544 scaled up to a chromosome length of $L = 10^8$ would take an estimated 7.2 days to run, and worse,
545 would require 8.1 TB (terabytes) of memory. Those estimates are derived from the pattern of
546 performance observed for recipe 13.9 with $L = 5 \times 10^5$, $L = 10^6$, and $L = 2 \times 10^6$ (the upper limit on
547 our test machine due to memory usage), extrapolated out to $L = 10^8$. Implementing this model
548 with tree-sequence recording therefore reduces the runtime by a factor of more than 1.35 million,
549 and reduces the memory usage by a factor of more than 160,000.

550 Similar to Example 2, the post-run analysis walks through the tree sequence, but in this case,
551 computes the mean true local ancestry (the fractional ancestry from subpopulation p1 versus p2)
552 for each tree. This is done by finding the roots for the tree, assessing the subpopulations of origin
553 of those root individuals, and averaging those together weighted by the number of descendants
554 from each root. A simple plot is then produced. In this example, the analysis took 62.2 seconds;
555 the analysis runtime is relatively long because the trees here typically have many roots, so the
556 inner loop is executed a great many times.

557 The final plot of true local ancestry by chromosome position is shown in Figure 4. The mean
558 true local ancestry at the points where the beneficial mutations were introduced into p1 and p2 has
559 to be 100% p1 and 100% p2, respectively, since both beneficial mutations fixed by the end of the
560 run. At other points along the genome there is more variation, but with a general pattern of being
561 more completely admixed at the chromosome ends and middle, with gradations toward the
562 absolute p1 and p2 points. Since this is a single run of the model, the pattern is quite stochastic;

563 an average across many runs of this model could produce a smooth plot if desired, and since it
564 takes only a couple of minutes to execute the pipeline here, that would be very quick to do. This
565 method of calculating true local ancestry could be used by any SLiM model with tree-sequence
566 recording, so models with more complex demography, under any scenario of selection and
567 mating, with any recombination map, etc., could just as easily be explored.

568 *Example IV: Neutral burn-in for a non-neutral model*

569 Our final example illustrates a solution to the problem of neutral burn-in. In many applications
570 one wishes to execute a non-neutral forward simulation beginning with an equilibrium amount of
571 extant neutral genetic diversity, and the simulation needed to generate that pre-existing diversity,
572 typically called the model “burn-in”, can take quite a long time – often much longer than it takes
573 to execute the non-neutral portion of the simulation. For a model with a long chromosome or
574 large population size, this burn-in can be so long as to limit the practical scale of the simulations
575 that can be conducted. One solution to this is a “hybrid” approach, in which a forward simulation
576 is initialized with the result of a (much faster) coalescent simulation (similar to Bhaskar 2014).
577 This is now possible using tree sequences in SLiM, but we go a step further: even a great deal of
578 the work done in a coalescent simulation of this burn-in period is unnecessary. All of the
579 genealogical branches that go extinct are irrelevant; all that matters are those segments of
580 ancestral genomes from which the final generation inherits. With tree-sequence recording, one
581 can simulate only the histories of those segments, saving an immense amount of computation
582 relative to a forward-time burn-in simulation.

583 Here we will look at a fairly large model ($N = 10^5$; $L = 10^6$) that evolves under neutral
584 dynamics until coalescence (the neutral burn-in), after which follows some relatively brief non-

585 neutral dynamics (a selective sweep). Running the burn-in period for this model in SLiM would
586 take an exceedingly long time, given the scale of the model, as we will see below. A better idea
587 is to use what we call “recapitation”: we can run the SLiM model forward from an initial state
588 that conceptually follows burn-in, and then use msprime to generate *after the fact* the coalescent
589 history for the initial individuals of the forward simulation. This can be done without simulating
590 neutral mutations, but if neutral mutations are desired as an end product of the simulation, they
591 can be overlaid at the end as in Example 1.

592 We begin with the SLiM model, which simulates the introduction and sweep to fixation of a
593 beneficial mutation. For simplicity, we will select a run of the model that happens to result in
594 fixation, rather than using a recipe that is conditional upon fixation; the random number seed
595 specified in the Python script below should produce that outcome. The SLiM model:

```
596 initialize() {  
597     initializeTreeSeq(simplificationRatio=INF);  
598     initializeMutationRate(0);  
599     initializeMutationType("m2", 0.5, "f", 1.0);  
600     m2.convertToSubstitution = F;  
601     initializeGenomicElementType("g1", m2, 1);  
602     initializeGenomicElement(g1, 0, 1e6 - 1);  
603     initializeRecombinationRate(3e-10);  
604 }  
605 1 late() {  
606     sim.addSubpop("p1", 100000);  
607 }  
608 100 late() {  
609     sample(p1.genomes, 1).addNewDrawnMutation(m2, 5e5);  
610 }  
611 100:10000 late() {  
612     mut = sim.mutationsOfType(m2);  
613     if (mut.size() != 1)  
614         stop(sim.generation + ": LOST");  
615     else if (sum(sim.mutationFrequencies(NULL, mut)) == 1.0)  
616     {  
617         sim.treeSeqOutput("ex4_TS_decap.trees");  
618         sim.simulationFinished();  
619     }  
620 }
```

621 This specifies a simple model with population size $N = 10^5$ diploid individuals, chromosome
622 length $L = 10^6$ base positions, and a recombination rate of $r = 3 \times 10^{-10}$ per base position per

623 generation, without mutation. It runs to generation 100 and then introduces the sweep mutation
624 (the delay before introduction is just to provide separation between the simulation start and the
625 start of the sweep in the plot produced below). When the sweep mutation is found to have fixed,
626 it then outputs a `.trees` file and stops. It specifies an infinite “simplification ratio” in the call to
627 `initializeTreeSeq()` so that simplification happens only once, at the point when the `.trees` file
628 is written out at the end; with this large of a model simplification takes a significant amount of
629 time, so this optional setting speeds the model up somewhat at the price of a higher peak memory
630 footprint.

631 As in previous examples, we will run this from a Python script that does post-run analysis:

```
632 import os, subprocess, msprime, pyslim
633 import numpy as np
634 import matplotlib.pyplot as plt
635
636 # Run the SLiM model and load the resulting .trees file
637 subprocess.check_output(["slim", "-m", "-s", "2", "ex4_TS.slim"])
638 ts = pyslim.load("ex4_TS_decap.trees") # no simplify!
639
640 # Calculate tree heights
641 def tree_heights(ts):
642     heights = np.zeros(ts.num_trees + 1)
643     for tree in ts.trees():
644         if tree.num_roots > 1: # not fully coalesced
645             heights[tree.index] = ts.slim_generation
646         else:
647             root_children = tree.children(tree.root)
648             real_root = tree.root if len(root_children) > 1 else root_children[0]
649             heights[tree.index] = tree.time(real_root)
650     heights[-1] = heights[-2] # repeat the last entry for plotting with step
651     return heights
652
653 # Plot tree heights before recapitation
654 breakpoints = list(ts.breakpoints())
655 heights = tree_heights(ts)
656 plt.step(breakpoints, heights, where='post')
657 plt.show()
658
659 # Recapitate
660 recap = ts.recapitate(recombination_rate=3e-10, Ne=1e5, random_seed=1)
661 recap.dump("ex4_TS_recap.trees")
662
663 # Plot tree heights after recapitation
664 breakpoints = list(recap.breakpoints())
665 heights = tree_heights(recap)
666 plt.step(breakpoints, heights, where='post')
667 plt.show()
```


668 After the import, we run the SLiM model (which takes 46.05 seconds) and load the `.trees` file
669 it saves out. We then immediately make a plot of mean tree heights along the chromosome. This
670 is similar to what we did in Example 2, but here it requires some extra finesse because we did not
671 simplify the tree sequence after loading it as we did then. To perform recapitation, we cannot
672 first simplify – we need the ancestral individuals that started the SLiM simulation to remain in the
673 tree sequence, so that recapitation can build upon them correctly. For this reason, every root in
674 the loaded tree sequence has the same time, corresponding to the beginning of the forward
675 simulation. The code in the `tree_heights()` function corrects for that, getting the height of the
676 child of the root if the forward simulation has coalesced below the original ancestor. This
677 provides the red line in Figure 5, showing that the area immediately around the introduced
678 mutation has coalesced at the time of the introduction (due to hitchhiking), but that the remainder
679 of the chromosome has not yet coalesced and thus has a tree height corresponding to the start of
680 forward simulation. These uncoalesced plateaus are what we will fix with recapitation.

681 The next step, then, is to perform the recapitation. This process works backwards from the tree
682 sequence information recorded by SLiM, constructing a full coalescent history for all of the
683 individuals alive at the end of the run. Since the non-neutral dynamics eliminated much of the
684 genetic diversity from the population as it existed at the beginning of forward simulation, this
685 coalescence requires very little work – much less than even a normal coalescent simulation for
686 this population size would require. In the example run discussed here, the process took 0.41
687 seconds. If neutral mutations are desired, they can then be overlaid on the recapitated tree
688 sequence following the method of Example 1; that code is not shown again here, but that
689 operation took another 0.58 seconds (with $\mu = 10^{-7}$).

690 Finally, we plot the mean tree heights for the recapitated tree sequence; this produces the black
691 line in Figure 5. The uncoalesced plateaus have now coalesced to times as far as a million
692 generations in the past. This plot nicely illustrates the classical sweep pattern in which regions
693 closer to the position of the sweep tend to coalesce more recently, due to hitchhiking, than
694 regions farther away (Maynard-Smith and Haigh, 1974).

695 Simulating the neutral burn-in period in SLiM instead, with neutral mutations occurring at a
696 rate of $\mu = 10^{-7}$, would take an estimated 114.7 hours (from extrapolation; this is a very
697 conservative estimate since the model was nowhere near mutation–drift balance when times were
698 measured). Recapitation and neutral mutation overlay, with a total time of 0.99 seconds,
699 therefore sped up the burn-in process in this example by more than 400,000 times.

700 Recapitation is clearly much faster than conducting burn-in with forward simulation, then; it
701 should be faster than a rescaled forward simulation model too (since rescaling can generally not
702 be taken that far without introducing problematic artifacts; see Discussion), and faster even than
703 constructing the burn-in state with the coalescent (since recapitation is based upon the coalescent
704 but handles far fewer events). Recapitation provides other benefits as well, since it means that
705 neutral burn-in can be deferred until after forward simulation is complete, and can even be
706 conducted as an afterthought on existing model output. It also allows the non-neutral forward
707 simulation to run without a burn-in history needing to be loaded (likely making it faster and
708 leaner), and allows one to avoid the question of how many generations must be simulated for
709 complete burn-in. It is worth noting that the coalescent (and thus recapitation) does not produce
710 identical results to forward simulation of a Wright–Fisher model, but the differences are small
711 and are mostly in the pattern of the most recent branches (Wakeley et al., 2012; Bhaskar et al.,

712 2014); using recapitation as an approximation for neutral forward simulation should therefore
713 produce practically identical results as long as the forward portion of the simulation runs for at
714 least a few generations. Similarly, although spatial models differ substantially from the standard
715 coalescent, this difference is mostly seen in the more recent portion of the trees; lineages that
716 have “mixed” across the species range without coalescing behave statistically like lineages in a
717 randomly mating population (Wilkins, 2004; Matsen and Wakeley, 2006). Recapitation with an
718 unstructured coalescent should therefore be a good approximation to pre-existing diversity in a
719 spatial simulation as well.

720 Note that constructing a burn-in history with recapitation is only equivalent to a period of
721 forward simulation if the burn-in period is completely neutral. If a non-neutral burn-in to
722 equilibrium is needed, the best approach is probably to run the burn-in period in SLiM with tree-
723 sequence recording turned on and neutral mutations turned off (thus avoiding the cost of
724 simulating the neutral mutations during burn-in, as in Example 1). If a neutral burn-in is desired,
725 but the neutral mutations are then needed by the non-neutral portion of the simulation (perhaps
726 because some of the neutral mutations become non-neutral due to an environmental change), one
727 might simulate the burn-in period with the coalescent in msprime (including mutation), and then
728 save the result as a `.trees` file using `pyslim`; one could then read that `.trees` file into SLiM to
729 provide the initial state for further simulation. These techniques go beyond what we have space
730 to illustrate here, but the manual for SLiM 3 provides further recipes showing the use of tree-
731 sequence recording. Since it is possible to move simulation data with full ancestry records back
732 and forth between msprime and SLiM, one can imagine many ways to combine the two to
733 leverage their strengths while avoiding their weaknesses.

734 **Discussion**

735 We have integrated support for tree-sequence recording (Kelleher et al., 2018) into the popular
736 SLiM forward simulation software package. Tree-sequence recording can now be enabled in any
737 SLiM simulation, and the results output to a `.trees` file that can be loaded into Python for further
738 simulation or analysis using the `msprime` package (a part of the `tskit` framework). We have also
739 extended the tree-sequence recording method to allow the recording and output of mutations that
740 arise during forward simulation.

741 We provided four examples demonstrating the power of the tree-sequence recording method.
742 The first example, of a simple neutral model, showed how to enable tree-sequence recording with
743 a few trivial modifications to a SLiM model's script. The second example illustrated the use of
744 recorded tree sequences in post-simulation analysis in Python to estimate the characteristic
745 reduction in neutral diversity expected around functional regions due to background selection.
746 The third example mapped the mean true local ancestry along the chromosome in a model of the
747 admixture of two subpopulations, again using post-simulation Python analysis. Finally, our
748 fourth example illustrated the use of `msprime` to “recapitate” a SLiM run, using the coalescent to
749 construct a neutral burn-in period after the completion of forward simulation.

750 All of these examples illustrated the large performance benefits that can be achieved with tree-
751 sequence recording. Indeed, for very large neutral simulations our timing comparison indicated
752 that the speedup due to tree-sequence recording can exceed two orders of magnitude, and can put
753 the performance of forward simulation on par with an efficient coalescent-based simulation such
754 as `msprime` (Example 1). For a large simulation with many non-neutral mutations, we still
755 observed a speedup of more than an order of magnitude (Example 2); simulations with a lower
756 density of non-neutral mutations should benefit even more. Similarly, compared to standard

757 forward simulation methods, using recapitation to construct a neutral burn-in period provided a
758 speedup of more than five orders of magnitude (Example 4), and using the tree sequence to obtain
759 true local ancestry information provided a speedup of more than six orders of magnitude
760 (Example 3). Memory savings observed in these models were also quite substantial.

761 Although we have not made use of it in these examples, SLiM records substantial metadata in
762 the tree sequence it outputs about genomes, individuals, and mutations. This includes sex, age,
763 and spatial location of remembered individuals, and times of origination and selection
764 coefficients of mutations. This information, along with the tree sequence, could enable
765 substantially more detailed dissection of evolutionary trajectories. Access to this SLiM metadata
766 is mediated by the new `pyslim` package that bridges SLiM and `msprime`. Furthermore, the
767 `.trees` file contains all of the information necessary to reconstruct the internal state of the
768 simulation in SLiM, so it can be loaded back into SLiM, examined graphically using `SLiMgui`,
769 and even used as a starting point for further simulation (with some caveats; see the manual).

770 Tree-sequence recording is not a panacea. Models that do not involve neutral mutations will
771 not realize a speed benefit from tree-sequence recording's ability to defer neutral mutation
772 overlay; in fact, they will run more slowly, since the overhead of recording will not be
773 compensated by eliminating neutral mutation simulation. Models that involve a very high
774 recombination rate relative to the mutation rate may also not see a speed benefit from tree-
775 sequence recording, since tracking the recombination breakpoints can become very time-
776 consuming; informal tests indicate that this becomes important, for neutral simulations, when the
777 recombination rate is two or more orders of magnitude larger than the mutation rate, however, so
778 it may not be a practical concern for most models. Even if simulation performance is not

779 improved by tree-sequence recording, the ancestry information provided by the tree sequence
780 may still speed up analysis or provide additional statistical power, which can also be quite
781 important in reducing total runtimes. The benefit of tree-sequence recording also depends upon
782 factors such as the proportion of neutral to non-neutral mutations, the distribution of fitness
783 effects from which the non-neutral mutations are drawn, the genetic architecture, the frequency
784 with which tree-sequence simplification is performed, and many other factors. In practice, it may
785 be worthwhile to simply compare the performance of both methods for a particular model; it is
786 difficult to distill any reliable rule of thumb. These considerations were discussed further in
787 Kelleher et al. (2018).

788 A commonly used technique for speeding up large forward simulations is model rescaling.
789 This involves scaling down the population size (N) by some factor Q , while scaling up the
790 mutation rate (μ), the recombination rate (r), and selection coefficients (s) by the same factor; this
791 holds many common population-genetic parameters constant, since they involve products of these
792 variables (e.g., $N\mu$, Nr , and Ns). Since these factors (as well as genetic drift) are *rates*, one
793 generation in the rescaled model corresponds to Q generations in the original model. Therefore,
794 rescaling by a factor Q can provide a speedup of up to a factor of Q^2 due to the Q -times smaller
795 population size *and* the Q -times smaller number of generations that need to be simulated.
796 However, this technique has important limitations, because it can introduce artifacts due to the
797 discretization of mutation frequencies and of time. For example, if a model with an original
798 population size of $N = 10,000$ were rescaled to a model with $N = 100$, the smallest possible
799 mutation frequency will also have increased from 0.00005 to 0.005, which could severely affect
800 studies in which one is concerned about the behavior of low-frequency polymorphisms. There are

801 more serious issues related to the process of adaptation; since rescaled values of s are larger,
802 rescaling has the net effect of substituting many mutations of small effect with a single one of
803 large effect (with $Q=100$, replacing 100 mutations with $s=0.001$ by a single one of $s=0.1$). Thus,
804 rescaling must not be taken too far, and careful comparisons are needed between the unscaled and
805 the rescaled model to ensure that results are not altered by rescaling artifacts. The SLiM manual
806 (Haller and Messer, 2016) has an extended discussion of model rescaling and provides instructive
807 examples. Since tree-sequence recording does not introduce such artifacts, it probably ought to
808 be used to full advantage before any model rescaling is applied. If that does not bring the desired
809 simulation within practical computational bounds, rescaling may be used in conjunction with
810 tree-sequence recording, but with the same caveats mentioned above. Note, however, that the
811 effectiveness of combining both strategies is hard to predict, since the increased recombination
812 rate in the scaled model means that roughly the *same* number of recombination events must be
813 recorded.

814 Although tree-sequence recording is not appropriate in every model, the examples we have
815 presented demonstrate that the performance gains it provides can make simulations possible that
816 would previously have been beyond reach, opening up new horizons for exploration. The
817 software packages used here – SLiM, msprime, Python, R – are all free and open-source, and the
818 examples and analyses shown here are all available on GitHub. We hope that the practical
819 examples we have provided will raise the level of awareness among evolutionary biologists
820 regarding this exciting new method.

821 **Acknowledgements**

822 Thanks to Kevin Thornton and Dom Nelson for helpful discussions. This work was supported by
823 funding from the College of Agriculture and Life Sciences at Cornell University, Predator Free
824 2050 (grant SS/05/01), and the NIH (grants R21AI130635 and R01GM127418) to PWM; by
825 funding from the Sloan Foundation and the NSF (under DBI-1262645) to PLR; and by the
826 Wellcome Trust (grant 100956/Z/13/Z) to Gil McVean for JK.

827 **References**

- 828 Arunkumar, R., Ness, R.W., Wright, S.I., and Barrett, S.C. (2015). The evolution of selfing is
829 accompanied by reduced efficacy of selection and purging of deleterious mutations.
830 *Genetics* 199(3), 817-829.
- 831 Assaf, Z.J., Petrov, D.A., and Blundell, J.R. (2015). Obstruction of adaptation in diploids by
832 recessive, strongly deleterious alleles. *PNAS* 112(20), E2658-E2666.
- 833 Berman, S.M. (1964). Limit theorems for the maximum term in stationary sequences. *Ann. Math.*
834 *Statist.* 35(2), 502–516.
- 835 Bhaskar, A., Clark, A.G., and Song, Y.S. (2014). Distortion of genealogical properties when the
836 sample is very large. *PNAS* 111(6), 2385–2390.
- 837 Caballero, A., Tenesa, A., & Keightley, P.D. (2015). The nature of genetic variation for complex
838 traits revealed by GWAS and regional heritability mapping analyses. *Genetics* 201(4),
839 1601–1613.
- 840 Charlesworth, B., Morgan, M.T., and Charlesworth, D. (1993). The effect of deleterious
841 mutations on neutral molecular variation. *Genetics* 134(4), 1289–1303.
- 842 Champer, J., Liu, J., Oh, S.Y., Reeves, R., Luthra, A., Oakes, N., Clark, A.G., and Messer, P.W.
843 (2018). Reducing resistance allele formation in CRISPR gene drive. *PNAS (early access)*,
844 1–6. DOI: <https://doi.org/10.1073/pnas.1720354115>
- 845 Cotto, O., Wessely, J., Georges, D., Klonner, G., Schmid, M., Dullinger, S., Thuiller, W., and
846 Guillaume, F. (2017). A dynamic eco-evolutionary model predicts slow response of alpine
847 plants to climate warming. *Nature Communications* 8, 15399.
- 848 Elyashiv, E., Sattath, S., Hu, T. T., Strutsovsky, A., McVicker, G., Andolfatto, P., Coop, G. &
849 Sella, G. (2016). A genomic map of the effects of linked selection in *Drosophila*. *PLoS*
850 *Genetics* 12(8), e1006130.
- 851 Ewing, G.B., and Jensen, J.D. (2016). The consequences of not accounting for background
852 selection in demographic inference. *Molecular Ecology* 25(1), 135–141.
- 853 Enard, D., Messer, P.W., and Petrov, D.A. (2014). Genome-wide signals of positive selection in
854 human evolution. *Genome Research* 24(6), 885–895.

- 855 Fournier-Level, A., Perry, E.O., Wang, J.A., Braun, P.T., Migneault, A., Cooper, M.D., Metcalf,
856 C.J.E., and Schmitt, J. (2016). Predicting the evolutionary dynamics of seasonal
857 adaptation to novel climates in *Arabidopsis thaliana*. *PNAS* 113(20), E2812–E2821.
- 858 Griffiths, R.C. The two-locus ancestral graph. In: Basawa, I.V., Taylor, R.L., eds. *Selected*
859 *Proceedings of the Sheffield Symposium on Applied Probability, 1989*. Hayward,
860 California: Institute of Mathematical Statistics, 1991: 100–117.
- 861 Griffiths, R.C., and Marjoram, P. (1997). An ancestral recombination graph. In: Donnelly P.,
862 Tavaré S., eds. *Progress in Population Genetics and Human Evolution*. Berlin, Germany:
863 Springer-Verlag, 1997: 257–270.
- 864 Haller, B.C., and Hendry, A.P. (2013). Solving the paradox of stasis: Squashed stabilizing
865 selection and the limits of detection. *Evolution* 68(2), 483–500.
- 866 Haller, B.C., R Mazzucco, R., and Dieckmann, U. (2013). Evolutionary branching in complex
867 landscapes. *American Naturalist* 182(4), E127-E141.
- 868 Haller, B.C., and Messer, P. W. (2016). SLiM: An Evolutionary Simulation Framework. URL:
869 http://benhaller.com/slim/SLiM_Manual.pdf
- 870 Haller, B.C., and Messer, P.W. (2017a). asymptoticMK: A web-based tool for the asymptotic
871 McDonald–Kreitman test. *G3: Genes, Genomes, Genetics* 7(5), 1569–1575.
- 872 Haller, B.C., and Messer, P.W. (2017b). SLiM 2: Flexible, interactive forward genetic
873 simulations. *Molecular Biology and Evolution* 34(1), 230–240. DOI:
874 <http://dx.doi.org/10.1093/molbev/msw211>
- 875 Hudson, R.R. (1983). Properties of a neutral allele model with intragenic recombination.
876 *Theoretical Population Biology* 23(2), 183–201.
- 877 Hudson, R.R. (1994). How can the low levels of DNA sequence variation in regions of the
878 *Drosophila* genome with low recombination rates be explained? *PNAS* 91(15), 6815–
879 6818.
- 880 Hunter, J.D. (2007). Matplotlib: A 2D graphics environment. *Computing In Science &*
881 *Engineering* 9(3), 90–95.
- 882 Kelleher, J, Etheridge, A.M., and McVean, G. (2016). Efficient coalescent simulation and
883 genealogical analysis for large sample sizes. *PLoS Computational Biology* 12(5):
884 e1004842. DOI: <https://doi.org/10.1371/journal.pcbi.1004842>
- 885 Kelleher, J., Thornton, K.R., Ashander, J., and Ralph, P.L. (2018). Efficient pedigree recording
886 for fast population genetics simulation. *PLoS Computational Biology* 14(11): e1006581.
887 DOI: <https://doi.org/10.1371/journal.pcbi.1006581>
- 888 Kingman, J.F.C. (1982). On the genealogy of large populations. *Journal of Applied Probability*
889 19, 27–43.
- 890 Mafessoni, F., and Lachmann, M. (2015). Selective strolls: fixation and extinction in diploids are
891 slower for weakly selected mutations than for neutral ones. *Genetics* 201(4), 1581–1589.
- 892 Matsen, F.A., and Wakeley, J. (2006). Convergence to the island-model coalescent process in
893 populations with restricted migration. *Genetics* 172(1), 701–708.

- 894 Matz, M.V., Treml, E.A., Aglyamova, G.V., and Bay, L.K. (2018). Potential and limits for rapid
895 genetic adaptation to warming in a Great Barrier Reef coral. *PLoS Genetics* 14(4),
896 e1007220.
- 897 Maynard-Smith, J., and Haigh, J. (1974). The hitch-hiking effect of a favourable gene. *Genetics*
898 *Research* 23(1), 23–35.
- 899 Messer, P.W. (2013). SLiM: Simulating evolution with selection and linkage. *Genetics* 194(4),
900 1037–1039.
- 901 Nowak, M.D., Haller, B.C., and Yoder, A.D. (2014). The founding of Mauritian endemic coffee
902 trees by a synchronous long-distance dispersal event. *Journal of Evolutionary Biology*
903 27(6), 1229–1239.
- 904 Oliphant, T.E. (2006). A guide to NumPy. U.S.A.: Trelgol Publishing.
- 905 Patel, R., Scheinfeldt, L.B., Sanderford, M.D., Lanham, T.R., Tamura, K., Platt, A., Glicksberg,
906 B.S., Xu, K., Dudley, J.T., and Kumar, S. (2018). Adaptive landscape of protein variation
907 in human exomes. *Molecular Biology and Evolution* 35(8): 2015–2025. DOI:
908 <https://doi.org/10.1093/molbev/msy107>
- 909 Ryan, S.F., Deines, J.M., Scriber, J.M., Pfrender, M.E., Jones, S.E., Emrich, S.J., and Hellmann,
910 J.J. (2018). Climate-mediated hybrid zone movement revealed with genomics, museum
911 collection, and simulation modeling. *PNAS* 115(10) E2284-E2291.
- 912 Sattath, S., Elyashiv, E., Kolodny, O., Rinott, Y., and Sella, G. (2011). Pervasive adaptive protein
913 evolution apparent in diversity patterns around amino acid substitutions in *Drosophila*
914 *simulans*. *PLoS Genetics* 7(2), e1001302.
- 915 Thornton, K.R. (2014). A C++ template library for efficient forward-time population genetic
916 simulation of large populations. *Genetics* 198(1), 157–166.
- 917 Wakeley, J., King, L., Low, B.S., and Ramachandran, S. (2012). Gene genealogies within a fixed
918 pedigree, and the robustness of Kingman’s coalescent. *Genetics* 190(4), 1433–1445.
- 919 Wilkins, J.F. (2004). A separation-of-timescales approach to the coalescent in a continuous
920 population. *Genetics* 168(4), 2227–2244.

921 **Data Accessibility**

922 SLiM 3 is available online at <https://messerlab.org/slim/>. It is open source, under the GPL 3.0

923 license, and its source code is on GitHub at <https://github.com/MesserLab/SLiM>.

924 msprime is available online at <https://pypi.org/project/msprime/>. It is open source, under the GPL

925 3.0 license, and its source code is on GitHub at <https://github.com/tskit-dev/msprime>.

926 pyslim is open source, under the MIT license, and is available on GitHub at

927 <https://github.com/tskit-dev/pyslim>.

928 The examples and performance assessments presented in this paper are available on GitHub at

929 <https://github.com/bhaller/SLiMTreeSeqPub>.

930 **Author Contributions**

931 *We have used the CRediT taxonomy for contributions (<https://casrai.org/credit/>).*

932 BCH: Conceptualization, Investigation, Methodology, Software, Validation, Visualization,

933 Writing – Original Draft Preparation, Writing – Review & Editing.

934 JG: Conceptualization, Methodology, Software, Writing – Review & Editing.

935 JK: Conceptualization, Methodology, Software, Validation, Visualization, Writing – Review &

936 Editing.

937 PWM: Conceptualization, Funding Acquisition, Supervision, Writing – Review & Editing.

938 PLR: Conceptualization, Funding Acquisition, Methodology, Software, Supervision, Validation,

939 Writing – Review & Editing.

940 Figures

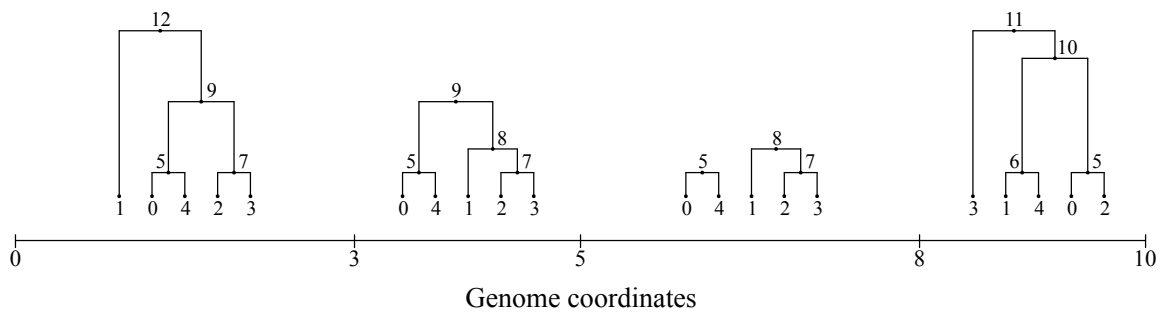
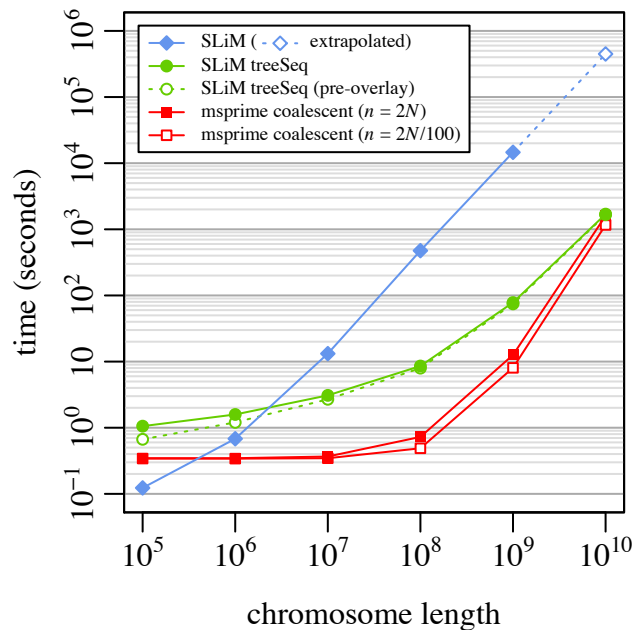
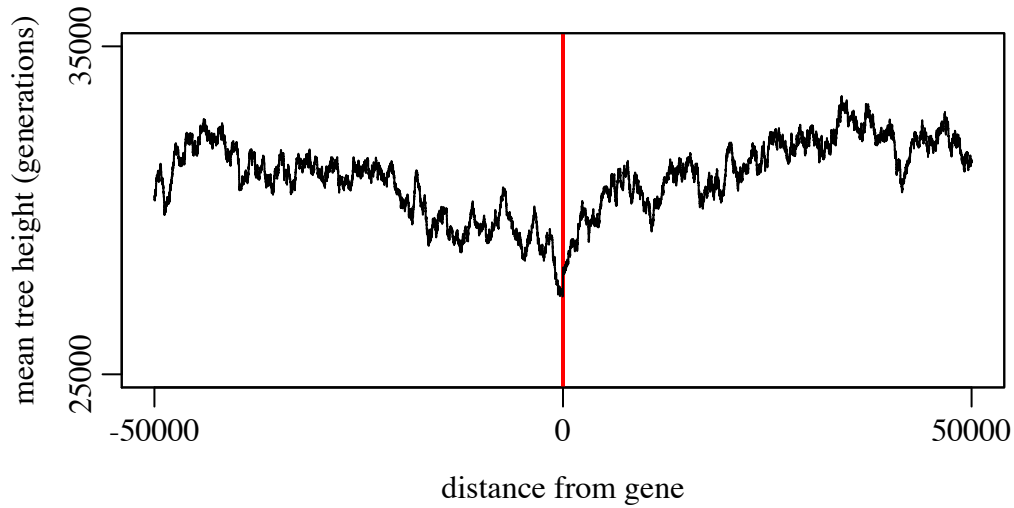


Figure 1. An example tree sequence for a model of five extant genomes, with a chromosome ten base positions long. Each interval between x axis ticks is a genomic interval with a distinct ancestry tree. The leaves of each tree [0–4] represent the extant genomes, whereas the internal nodes [5–12] represent ancestral genomes from which the extant genomes descend. The pattern of ancestry at adjacent sites is typically highly correlated, as seen here. Full coalescence has been achieved for the first, second, and fourth intervals, but the third interval has not yet fully coalesced; the tree for that interval therefore has multiple roots. See Kelleher et al. (2016, 2018) for further discussion of the tree sequence data structure.

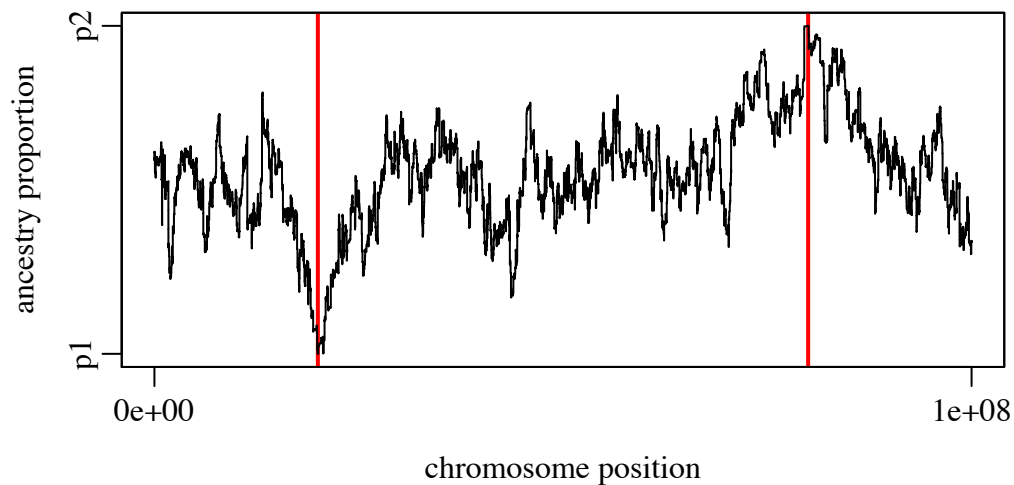


951

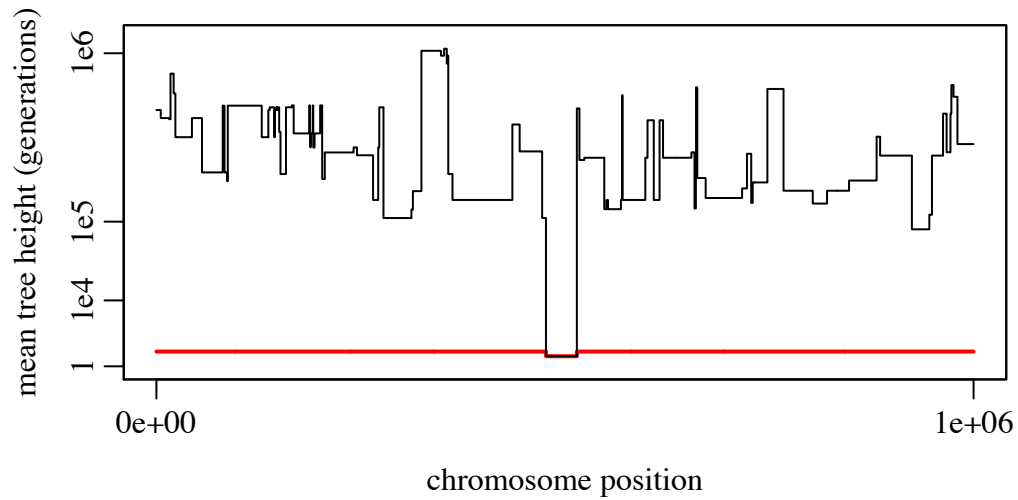
952 Figure 2. A speed comparison between SLiM without tree-sequence recording, SLiM with
953 tree-sequence recording and mutation overlay, and msprime's coalescent simulation for a
954 simple neutral model (Example 1; see text for model description). Each point represents
955 the mean runtime across 10 replicates using different random number seeds; bars showing
956 standard error of the mean would be smaller than the size of the plotted points in all cases.
957 Runs for SLiM without tree-sequence recording (filled blue diamonds) were not
958 conducted for $L = 10^{10}$ because the memory usage was prohibitive, so a linear
959 extrapolation is shown (hollow blue diamond). Runs for SLiM with tree-sequence
960 recording and mutation overlay (filled green circles) are subdivided here to show the
961 runtime for SLiM alone, prior to mutation overlay (hollow green circles), illustrating that
962 the time for mutation overlay is negligible. The runtimes for the msprime coalescent for a
963 full population sample of $n = 2N = 1000$ (filled red squares) and for a sample of size
964 $n = 2N/100 = 10$ (hollow red squares) are both shown. Note that the x and y axes are both
965 on a log scale.



966
967 Figure 3. Mean diversity (as measured by mean tree height) as a function of distance from the
968 nearest gene (Example 2). The center of the x-axis (red line) represents a distance of zero,
969 immediately adjacent to a gene; moving away from the x-axis center to the left/right
970 represents moving away from the nearest gene to the left/right respectively. The pattern
971 of decreased diversity near a gene is the “dip in diversity” due to background selection.



972
973 Figure 4. Mean true local ancestry at each position along the chromosome (Example 3). The
974 red vertical bars indicate the positions at which beneficial mutations were originally
975 introduced into p1 and p2. The beneficial mutations, which both fixed, are points where
976 the true local ancestry is 100% p1 or p2. True local ancestry regresses toward equal
977 admixture with increasing distance from those fixed points.



978
979 Figure 5. Mean tree height (on a cube-root-scaled y -axis) at each position along the
980 chromosome, before and after recapitation (Example 4). The red line shows mean tree
981 heights prior to recapitation; the region surrounding the introduced sweep mutation
982 coalesces at the start of the sweep, whereas the plateaus outside that region are
983 uncoalesced and have a height corresponding to the start of forward simulation (100
984 generations earlier). The black line shows heights after recapitation; the uncoalesced
985 plateaus have now been coalesced backward in time, producing tree heights as long as a
986 million generations in the past.