tsdate

Release 0.1.dev158+gc96e07d.d20230826

Aug 26, 2023

Contents:

This is the documentation for `tsdate`, a method for efficiently inferring the age of ancestors in a tree sequence.

CHAPTER 1

Introduction

`tsdate` is a scalable method for estimating the age of ancestral nodes in a tree sequence. The method uses a coalescent prior and updates node times on the basis of the number of mutations along each edge of the tree sequence (i.e. using the "molecular clock").

The method is designed to operate on the output of tsinfer, which efficiently infers tree sequence *topologies* from large genetic datasets. `tsdate` and `tsinfer` are scalable to the largest genomic datasets currently available.

The algorithm is described in this Science paper (preprint here). We also provide evaluations of the accuracy and computational requirements of the method using both simulated and real data; the code to reproduce these results can be found in another repository.

Please cite this paper if you use `tsdate` in published work:

> Anthony Wilder Wohns, Yan Wong, Ben Jeffery, Ali Akbari, Swapan Mallick, Ron Pinhasi, Nick Patterson, David Reich, Jerome Kelleher, and Gil McVean (2022) *A unified genealogy of modern and ancient genomes*. Science **375**: eabi8264; doi: https://doi.org/10.1126/science.abi8264

CHAPTER 2

Installation

To install `tsdate` simply run:

```
$ python3 -m pip install tsdate --user
```

Python 3.5, or a more recent version, is required. The software has been tested on MacOSX and Linux.

Once installed, `tsdate`'s *Command Line Interface (CLI)* can be accessed via:

```
$ python3 -m tsdate
```

or

```
$ tsdate
```

Alternatively, the *Python API* allows more fine-grained control of the inference process.

CHAPTER 3

Tutorial

3.1 Dating Tree Sequences

To illustrate the typical use case of `tsdate`'s Python API, we will first create sample data with msprime and infer a tree sequence from this data using tsinfer. We will then run `tsdate` on the inferred tree sequence.

Let's start by creating some sample data with `msprime` using human-like parameters.

```
import msprime

sample_ts = msprime.simulate(sample_size=10, Ne=10000,
                                          length=1e4,
                                          mutation_rate=1e-8,
                                          recombination_rate=1e-8,
                                          random_seed=2)
print(sample_ts.num_trees,
      sample_ts.num_nodes)
```

The output of this code is:

```
12 29
```

We take this simulated tree sequence and turn it into a *tsinfer* SampleData object as documented here, and then infer a tree sequence from the data

```
import tsinfer

sample_data = tsinfer.SampleData.from_tree_sequence(sample_ts)
inferred_ts = tsinfer.infer(sample_data)
```

Note: `tsdate` works best with simplified tree sequences (`tsinfer`'s documentation provides) details on how to simplify an inferred tree sequence. This should not be an issue when working with tree sequences simulated using `msprime`.

Next, we run *tsdate* to estimate the ages of nodes and mutations in the inferred tree sequence:

```
import tsdate
dated_ts = tsdate.date(inferred_ts, Ne=10000, mutation_rate=1e-8)
```

All we need to run `tsdate` (with its default parameters) is the inferred tree sequence object, the *estimated* effective population size, and *estimated* mutation rate. Here we have provided a human mutation rate per base pair per generation, so the nodes dates in the resulting tree sequence should be interpreted as generations.

3.1.1 Specifying a Prior

The above example shows the basic use of `tsdate`, using default parameters. The software has parameters the user can access through the `tsdate.build_prior_grid()` function which may affect the runtime and accuracy of the algorithm.

3.1.2 Returned dates

Several different node time estimates are returned by `tsdate.date()`. In particular, to conform to the requirements of a valid tree sequence, child nodes must always be strictly younger than their parents. In the unusual case that a child has an estimated time which is older than its parent, `tsdate` increases the node time of the parent in the tree sequence so that it is constrained to be fractionally older than its oldest child.

In the case of the inside-outside method (see below), the estimated time is based on the posterior mean node time. The true posterior mean time (unconstrained by the tree sequence requirements above) is stored in the node metadata. This can be consulted if you want to obtain the raw mean times from the posterior.

For even greater detail about the posterior time estimates for each node, you can use the `return_posteriors` option, which will return the full distribution of posterior probabilities in each time slice.

Inside Outside vs Maximization

One of the most important parameters to consider is whether `tsdate` should use the inside-outside or the maximization algorithms to perform inference. A detailed description of the algorithms will be presented in our preprint, but from the users perspective, the inside-outside approach performs better empirically, and returns a full posterior distribution, but occasionally has issues with numerical stability. In contrast, the maximization approach is slightly less accurate empirically, and will not return true posteriors, but has the advantage of always being numerically stable.

3.1.3 Command Line Interface Example

`tsdate` also offers a convenient *command line interface (CLI)* for accessing the core functionality of the algorithm.

For a simple example of CLI, we'll first save the inferred tree sequence we created in *the section above* as a file.

```
import tskit

inferred_ts.dump("inferred_ts.trees")
```

Now we use the CLI to again date the inferred tree sequence and output the resulting dated tree sequence to `dated_ts.trees` file:

```
$ tsdate date inferred_ts.trees dated_ts.trees 10000 1e-8 --progress
```

The first two arguments are the input and output tree sequence file names, the third is the estimated effective population size, and the fourth is the estimated mutation rate. We also add the `--progress` option to keep track of `tsdate`'s progress.

3.1.4 Troubleshooting tsdate

If numerical stability issues are encountered when attempting to date tree sequences using the Inside-Outside algorithm, it may be necessary to remove large sections of the tree which do not have any variable sites using `tsdate.preprocess_ts()` method.

3.2 Inferring and Dating Tree Sequences with Historical (Ancient) Samples

`tsdate` and `tsinfer` can be used together to infer tree sequences from both modern and historical samples. The following recipe shows how this is accomplished with a few lines of Python. The only requirement is a tsinfer.SampleData file with modern and historical samples (the latter are specified using the *individuals_time* array in a tsinfer.SampleData file).

```
import msprime
import tsdate
import tsinfer
import tskit

import numpy as np


def make_historical_samples():
    samples = [
        msprime.Sample(population=0, time=0),
        msprime.Sample(0, 0),
        msprime.Sample(0, 0),
        msprime.Sample(0, 0),
        msprime.Sample(0, 1.0),
        msprime.Sample(0, 1.0)
    ]
    sim = msprime.simulate(samples=samples, mutation_rate=1, length=100)
    # Get the SampleData file from the simulated tree sequence
    # Retain the individuals times and ignore the sites times.
    samples = tsinfer.SampleData.from_tree_sequence(
      sim, use_sites_time=False, use_individuals_time=True)
    return samples

def infer_historical_ts(samples, Ne=1, mutation_rate=1):
   """
   Input is tsinfer.SampleData file with modern and historical samples.
   """
   modern_samples = samples.subset(np.where(samples.individuals_time[:] == 0)[0])
   inferred_ts = tsinfer.infer(modern_samples) # Infer tree seq from modern samples
   # Removes unary nodes (currently required in tsdate), keeps historical-only sites
   inferred_ts = tsdate.preprocess_ts(inferred_ts, filter_sites=False)
   dated_ts = tsdate.date(inferred_ts, Ne=Ne, mutation_rate=mutation_rate) # Date␣
↪tree seq
   sites_time = tsdate.sites_time_from_ts(dated_ts)  # Get tsdate site age estimates
```

(continues on next page)

(continued from previous page)

```
    dated_samples = tsdate.add_sampledata_times(
        samples, sites_time) # Get SampleData file with time estimates assigned to sites
    ancestors = tsinfer.generate_ancestors(dated_samples)
    ancestors_w_proxy = ancestors.insert_proxy_samples(
        dated_samples, allow_mutation=True)
    ancestors_ts = tsinfer.match_ancestors(dated_samples, ancestors_w_proxy)
    return tsinfer.match_samples(
        dated_samples, ancestors_ts, force_sample_times=True)

samples = make_historical_samples()
inferred_ts = infer_historical_ts(samples)
```

We simulate a tree sequence with six sample chromosomes, four modern and two historical. We then infer and date a tree sequence using only the modern samples. Next, we find derived alleles which are carried by the historical samples and use the age of the historical samples to constrain the ages of these alleles. Finally, we reinfer the tree sequence, using the date estimates from tsdate and the historical constraints rather than the frequency of the alleles to order mutations in `tsinfer`. Historical samples are added to the ancestors tree sequence as proxy nodes, in addition to being used as samples.

CHAPTER 4

Python API

This page provides documentation for the `tsdate` Python API.

4.1 Running tsdate

4.1.1 Specifying Prior and Time Discretisation Options

4.2 Preprocessing Tree Sequences

4.3 Functions for Inferring Tree Sequences with Historical Samples

CHAPTER 5

Command line interface

`tsdate` provides a Command Line Interface to access the basic functionality of the *Python API*.

```
$ tsdate
```

or

```
$ python3 -m tsdate
```

The second command is useful when multiple versions of Python are installed or if the **`tsdate`** executable is not installed on your path.

5.1 Argument details

CHAPTER 6

Indices and tables

- genindex
- search