tsdate

Release 0.1.6.dev0+g7c4afeb.d20220609

Jun 09, 2022

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.abi826421

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 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 but has issues with numerical stability, while the maximization approach is slightly less accurate empirically, but is 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
   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

`tsdate.`**`date`**(*tree_sequence*, *mutation_rate*, *Ne=None*, *recombination_rate=None*, *time_units=None*, *priors=None*, *, *return_posteriors=None*, *progress=False*, **kwargs*)

Take a tree sequence (which could have `uncalibrated` node times) and assign new times to non-sample nodes using the *tsdate* algorithm. If a mutation_rate is given, the mutation clock is used. The recombination clock is unsupported at this time. If neither a mutation_rate nor a recombination_rate is given, a topology-only clock is used. Times associated with mutations and non-sample nodes in the input tree sequence are not used in inference and will be removed.

Parameters

- **`tree_sequence`** (*`TreeSequence`*) – The input `tskit.TreeSequence`, treated as one whose non-sample nodes are undated.
- **`Ne`** (*`float`*) – The estimated (diploid) effective population size used to construct the (default) conditional coalescent prior. This is what is used when `priors` is `None`: a positive `Ne` value is therefore required in this case. Conversely, if `priors` is not `None`, no `Ne` value should be given.
- **`mutation_rate`** (*`float`*) – The estimated mutation rate per unit of genome per unit time. If provided, the dating algorithm will use a mutation rate clock to help estimate node dates. Default: `None`
- **`recombination_rate`** (*`float`*) – The estimated recombination rate per unit of genome per unit time. If provided, the dating algorithm will use a recombination rate clock to help estimate node dates. Default: `None`
- **`time_units`** (*`str`*) – The time units used by the `mutation_rate` and `recombination_rate` values, and stored in the `time_units` attribute of the output tree sequence. If the conditional coalescent prior is used, then this is also applies to the value of `Ne`, which in standard coalescent theory is measured in generations. Therefore if

you wish to use mutation and recombination rates measured in (say) years, and are using the conditional coalescent prior, the `Ne` value which you provide must be scaled by multiplying by the number of years per generation. If `None` (default), assume `"generations"`.

- **priors** (*NodeGridValues*) – NodeGridValue object containing the prior probabilities for each node at a set of discrete time points. If `None` (default), use the conditional coalescent prior with a standard set of time points as given by *build_prior_grid()*.
- **return_posteriors** (*bool*) – If `True`, instead of returning just a dated tree sequence, return a tuple of `(dated_ts, posteriors)`. Note that the dictionary returned in `posteriors` (described below) is suitable for reading as a pandas `DataFrame` object, using `pd.DataFrame(posteriors)`.
- **eps** (*float*) – Specify minimum distance separating time points. Also specifies the error factor in time difference calculations. Default: 1e-6
- **num_threads** (*int*) – The number of threads to use. A simpler unthreaded algorithm is used unless this is >= 1. Default: None
- **method** (*string*) – What estimation method to use: can be "inside_outside" (empirically better, theoretically problematic) or "maximization" (worse empirically, especially with gamma approximated priors, but theoretically robust). If `None` (default) use "inside_outside"
- **probability_space** (*string*) – Should the internal algorithm save probabilities in "logarithmic" (slower, less liable to to overflow) or "linear" space (fast, may overflow). Default: "logarithmic"
- **ignore_oldest_root** (*bool*) – Should the oldest root in the tree sequence be ignored in the outside algorithm (if "inside_outside" is used as the method). Ignoring outside root provides greater stability when dating tree sequences inferred from real data. Default: False
- **progress** (*bool*) – Whether to display a progress bar. Default: False

Returns A copy of the input tree sequence but with altered node times, or (if `return_posteriors` is True) a tuple of that tree sequence plus a dictionary of posterior probabilities from the "inside_outside" estimation `method`. Each node whose time was inferred corresponds to an item in this dictionary, with the key being the node ID and the value a 1D array of probabilities of the node being in a given time slice (or `None` if the "inside_outside" method was not used). The start and end times of each time slice are given as 1D arrays in the dictionary, under keys named `"start_time"` and `end_time"`.

Return type tskit.TreeSequence or (tskit.TreeSequence, dict)

4.1.1 Specifying Prior and Time Discretisation Options

tsdate.**build_prior_grid**(*tree_sequence*, *Ne*, *timepoints=20*, ***, *approximate_priors=False*, *approx_prior_size=None*, *prior_distribution='lognorm'*, *eps=1e-06*, *progress=False*)

Using the conditional coalescent, calculate the prior distribution for the age of each node, given the number of contemporaneous samples below it, and the discretised time slices at which to evaluate node age.

Parameters

- **tree_sequence** (*TreeSequence*) – The input `tskit.TreeSequence`, treated as undated.
- **Ne** (*float*) – The estimated (diploid) effective population size: must be specified. Using standard (unscaled) values for `Ne` results in a prior where times are measures in generations.

- **timepoints** (*int_or_array_like*) – The number of quantiles used to create the time slices, or manually-specified time slices as a numpy array. Default: 20
- **approximate_priors** (*bool*) – Whether to use a precalculated approximate prior or exactly calculate prior. If approximate prior has not been precalculated, tsdate will do so and cache the result. Default: False
- **approx_prior_size** (*int*) – Number of samples from which to precalculate prior. Should only enter value if approximate_priors=True. If approximate_priors=True and no value specified, defaults to 1000. Default: None
- **prior_distr** (*string*) – What distribution to use to approximate the conditional coalescent prior. Can be "lognorm" for the lognormal distribution (generally a better fit, but slightly slower to calculate) or "gamma" for the gamma distribution (slightly faster, but a poorer fit for recent nodes). Default: "lognorm"
- **eps** (*float*) – Specify minimum distance separating points in the time grid. Also specifies the error factor in time difference calculations. Default: 1e-6

Returns A prior object to pass to tsdate.date() containing prior values for inference and a discretised time grid

Return type base.NodeGridValues Object

4.2 Preprocessing Tree Sequences

tsdate.**preprocess_ts**(*tree_sequence*, *, *minimum_gap=1000000*, *remove_telomeres=True*, ***kwargs*)

Function to prepare tree sequences for dating by removing gaps without sites and simplifying the tree sequence. Large regions without data can cause overflow/underflow errors in the inside-outside algorithm and poor performance more generally. Removed regions are recorded in the provenance of the resulting tree sequence.

Parameters

- **tree_sequence** (*TreeSequence*) – The input :class'tskit.TreeSequence' to be preprocessed.
- **minimum_gap** (*float*) – The minimum gap between sites to remove from the tree sequence. Default: "1000000"
- **remove_telomeres** (*bool*) – Should all material before the first site and after the last site be removed, regardless of the length. Default: "True"
- ****kwargs** – All further keyword arguments are passed to the `tskit.simplify` command.

Returns A tree sequence with gaps removed.

Return type tskit.TreeSequence

4.3 Functions for Inferring Tree Sequences with Historical Samples

tsdate.**sites_time_from_ts**(*tree_sequence*, *, *unconstrained=True*, *node_selection='child'*, *min_time=1*)

Returns an estimated "time" for each site. This is the estimated age of the oldest MRCA which possesses a derived variant at that site, and is useful for performing (re)inference of a tree sequence. It is calculated from the ages of nodes, with the appropriate nodes identified by the position of mutations in the trees.

If node times in the tree sequence have been estimated by `tsdate` using the inside-outside algorithm, then as well as a time in the tree sequence, nodes will store additional time estimates that have not been explictly constrained by the tree topology. By default, this function tries to use these "unconstrained" times, although this is likely to fail (with a warning) on tree sequences that have not been processed by `tsdate`: in this case the standard node times can be used by setting `unconstrained=False`.

The concept of a site time is meaningless for non-variable sites, and so the returned time for these sites is `np.nan` (note that this is not exactly the same as tskit.UNKNOWN_TIME, which marks sites that could have a meaningful time but whose time estimate is unknown).

Parameters

- **tree_sequence** (*TreeSequence*) – The input :class'tskit.TreeSequence'.
- **unconstrained** (*bool*) – Use estimated node times which have not been constrained by tree topology. If `True` (default), this requires a tree sequence which has been dated using the `tsdate` inside-outside algorithm. If this is not the case, specify `False` to use the standard tree sequence node times.
- **node_selection** (*str*) – Defines how site times are calculated from the age of the upper and lower nodes that bound each mutation at the site. Options are "child", "parent", "arithmetic" or "geometric", with the following meanings
  - `'child'` (default): the site time is the age of the oldest node *below* each mutation at the site
  - `'parent'`: the site time is the age of the oldest node *above* each mutation at the site
  - `'arithmetic'`: the arithmetic mean of the ages of the node above and the node below each mutation is calculated; the site time is the oldest of these means.
  - `'geometric'`: the geometric mean of the ages of the node above and the node below each mutation is calculated; the site time is the oldest of these means
- **min_time** (*float*) – A site time of zero implies that no MRCA in the past possessed the derived variant, so the variant cannot be used for inferring relationships between the samples. To allow all variants to be potentially available for inference, if a site time would otherwise be calculated as zero (for example, where the `mutation_age` parameter is "child" or "geometric" and all mutations at a site are associated with leaf nodes), a minimum site greater than 0 is recommended. By default this is set to 1, which is generally reasonable for times measured in generations or years, although it is also fine to set this to a small epsilon value.

Returns Array of length tree_sequence.num_sites with estimated time of each site

Return type numpy.array

`tsdate.`**add_sampledata_times**(*samples*, *sites_time*)

Return a tsinfer.SampleData file with estimated times associated with sites. Ensures that each site's time is at least as old as the oldest historic sample carrying a derived allele at that site.

Parameters **samples** (*tsinfer.formats.SampleData*) – A tsinfer SampleData object to add site times to. Any historic individuals in this SampleData file are used to constrain site times.

Returns A tsinfer.SampleData file

Return type tsinfer.SampleData

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

This is the command line interface for tsdate, a tool to date tree sequences.

```
usage: tsdate [-h] [-V] {date,preprocess} ...
```

5.1.1 Positional Arguments

subcommand Possible choices: date, preprocess

5.1.2 Named Arguments

-V, --version show program's version number and exit

5.1.3 Sub-commands:

date

Takes an inferred tree sequence topology and returns a dated tree sequence.

```
tsdate date [-h] [-m MUTATION_RATE] [-r RECOMBINATION_RATE] [-e EPSILON]
            [-t NUM_THREADS] [--probability-space PROBABILITY_SPACE]
            [--method METHOD] [--ignore-oldest] [-p] [-v VERBOSITY]
            tree_sequence output Ne
```

Positional Arguments

tree_sequence	The path and name of the input tree sequence from which we estimate node ages.
output	The path and name of output file where the dated tree sequence will saved.
Ne	estimated effective (diploid) population size.

Named Arguments

-m, --mutation-rate	The estimated mutation rate per unit of genome per generation. If provided, the dating algorithm will use a mutation rate clock to help estimate node dates. Default: None
-r, --recombination-rate	The estimated recombination rate per unit of genome per generation. If provided, the dating algorithm will use a recombination rate clock to help estimate node dates. Default: None
-e, --epsilon	Specify minimum distance separating time points. Also specifies the error factor in time difference calculations. Default: 1e-6
-t, --num-threads	The number of threads to use. A simpler unthreaded algorithm is used unless this is >= 1. Default: None
--probability-space	Should the internal algorithm save probabilities in 'logarithmic' (slower, less liable to to overflow) or 'linear' space (faster, may overflow). Default: 'logarithmic'
--method	Specify which estimation method to use: can be 'inside_outside' (empirically better, theoretically problematic) or 'maximization' (worse empirically, especially with a gamma approximated prior, but theoretically robust). Default: 'inside_outside'
--ignore-oldest	Ignore the oldest node in the tree sequence, which is often of low quality when using empirical data.
-p, --progress	Show progress bar.
-v, --verbosity	How much verbosity to output.

preprocess

Remove regions without data from an input tree sequence.

```
tsdate preprocess [-h] [--minimum_gap MINIMUM_GAP]
                  [--trim-telomeres TRIM_TELOMERES] [-v VERBOSITY]
                  tree_sequence output
```

Positional Arguments

tree_sequence The tree sequence to preprocess.

output The path and name of output file where the preprocessed tree sequence will saved.

Named Arguments

--minimum_gap The minimum gap between sites to trim from the tree sequence. Default: '1000000'

--trim-telomeres Should all material before the first site and after the last site be trimmed, regardless of the length of these regions. Default: 'True'

-v, --verbosity How much verbosity to output.

CHAPTER 6

Indices and tables

- genindex
- search

Index