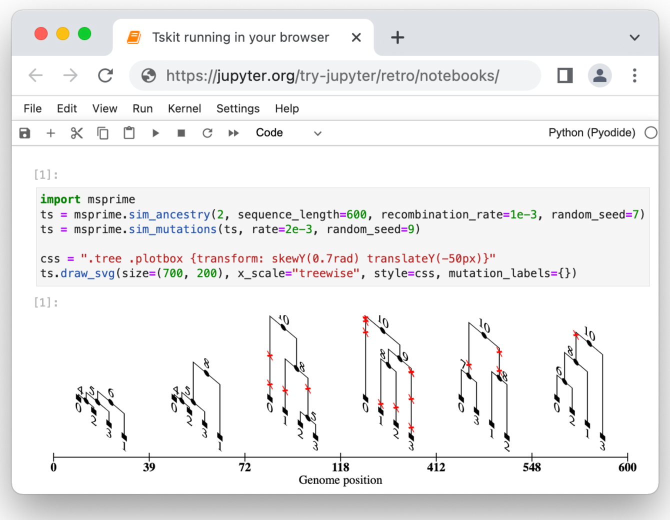
tskit: POPULATION-SCALE GENOMICS AND PHYLOGENETICS

Ben Jeffery[‡], Jerome Kelleher[‡], Nathaniel Pope[†], Peter Ralph[†],
Georgia Tsambos,[†§], Yan Wong[‡], and the rest of the tskit-dev group.
Code: github.com/petrelharp/progen-2023.
† University of Oregon ‡ University of Oxford § University of Washington

Interfaces and interoperability

We aim to provide *stable*, *well-tested* and *well-documented* software so others can reliably build with it – including a *backwards compatibility* guarantee. `tskit` is already being used in a number of inference and simulation packages. The core functionality is implemented via a C API, and the primary interface is via a Python library, but others are available:

- Well-tested API used by many software packages: SLiM, fwdpy11, msprime, tsinfer/tsdate, Relate, slendr, etc.
- Available in multiple programming languages:
- Runs in-browser (no install required!) for quick demos / teaching (see screenshot).
- Can represent full Ancestral Recombination Graphs; includes ARG likelihood calculations.
- Interoperable with other packages (e.g., VCF output for sequence data, newick/nexus output to Dendropy, numpy arrays to scikit-allel)



Try it!

Metadata

`tskit` now has integrated metadata for all objects (genomes, mutations, sites, etc). For instance *(spoiler alert)*, the complete ARG for 1.26 million SARS-Cov-2 genomes (until mid-2021): fits in 57MB, and loads in under 1 second! It is 819 MB once decompressed, and has metadata attached to three tables.

```
-rw-rw-r-- 1 jk jk 57M Mar  2 13:32 SARS-Cov-2-ARG.ts.tsz

ts = tszip.decompress("SARS-Cov-2-ARG.ts.tsz")

CPU times: user 775 ms, sys: 533 ms, total: 1.31 s
Wall time: 842 ms
```

Tree Sequence	
Trees	1496
Sequence Length	29904.0
Time Units	days
Sample Nodes	1265685
Total Size	819.3 MiB
Metadata	No Metadata

Table	Rows	Size	Has Metadata
Edges	1458146	44.5 MiB	
Individuals	0	24 Bytes	
Migrations	0	8 Bytes	
Mutations	1213193	45.8 MiB	✅
Nodes	1453347	716.5 MiB	✅
Populations	0	8 Bytes	
Provenances	1	874 Bytes	
Sites	29422	1.4 MiB	✅

The integrated data model links nodes, edges, sites and mutations, and now allows annotation of all objects with arbitrary external metadata. For instance, here's the first five sites in the SARS-Cov-2 ARG, and metadata for a sample (available as a dictionary!):

```
ts.tables.sites[:5]
```

id	position	ancestral_state	metadata
0	56	G	{'masked_samples': 727232}
1	57	A	{'masked_samples': 726137}
2	58	T	{'masked_samples': 725063}
3	59	C	{'masked_samples': 724533}
4	60	T	{'masked_samples': 721663}

```
dataclasses.asdict(ts.node(1026732))

{'id': 1026732, 'flags': 1, 'time': 60.0,
 'metadata': {'Imputed_lineage': 'B.1.1.7',
  'Nextclade_pango': 'B.1.1.7',
  'clade': '20I (Alpha, V1)',
  'country': 'Germany', 'date': '2021-05-01',
  'date_submitted': '2021-05-17',
  'gisaid_epi_isl': 'EPI_ISL_2122637',
  'sc2ts_qc': {'num_masked_sites': 150,
   'original_base_composition': {'-': 103,
    'A': 8902, 'C': 5473, 'G': 5847, 'T': 9578}},
  'strain': 'Germany/un-RKI-I-137988/2021',
  'totalSubstitutions': 36.0}}
```

Overview

`tskit` is the C and python library providing tools for working with *succinct tree sequences*. We provide solid, stable, well-tested software for you to use and build on. Why might you want to use tree sequences?

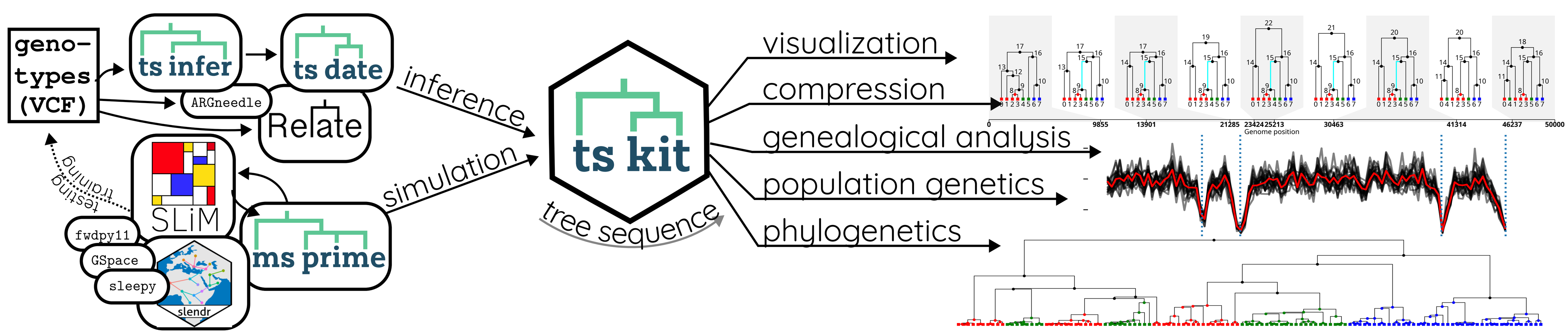
- For large samples, stores genotypes losslessly using (estimated) underlying genealogical relationships in orders of magnitude less space,
- ...and allows fast processing and exploration, in seconds, not hours.
- Genealogical relationships – "the trees" – are often closer to things we want to learn about
- ...and explicitly include a *time dimension*.
- History of a process can be recorded in a simulation, not just the genotypic outcome,
- ...and simulations can be much faster/more efficient.

In summary: by representing genomes using the genealogical process that generated the data, we get both a huge advantage storing and manipulating genomic data, as well as a more direct look at the processes that generated the data.

Here's some **silly slogans**, care to suggest any more?

- "`tskit`: launching your genomes into the time dimension!"
- "`tskit`: tree thinking, for popgen"
- "`tskit`: stable software for genome-scale trees"
- "`tskit`: all your insights, much faster!"

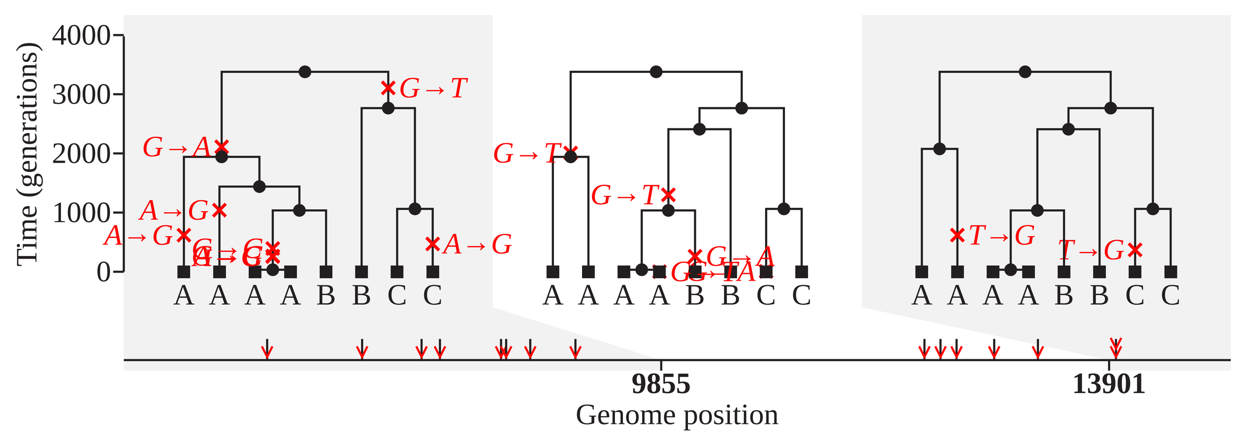
Documentation and examples: `https://tskit.dev/`



Visuzalization – see more at `https://tskit.dev/tutorials/viz.html`

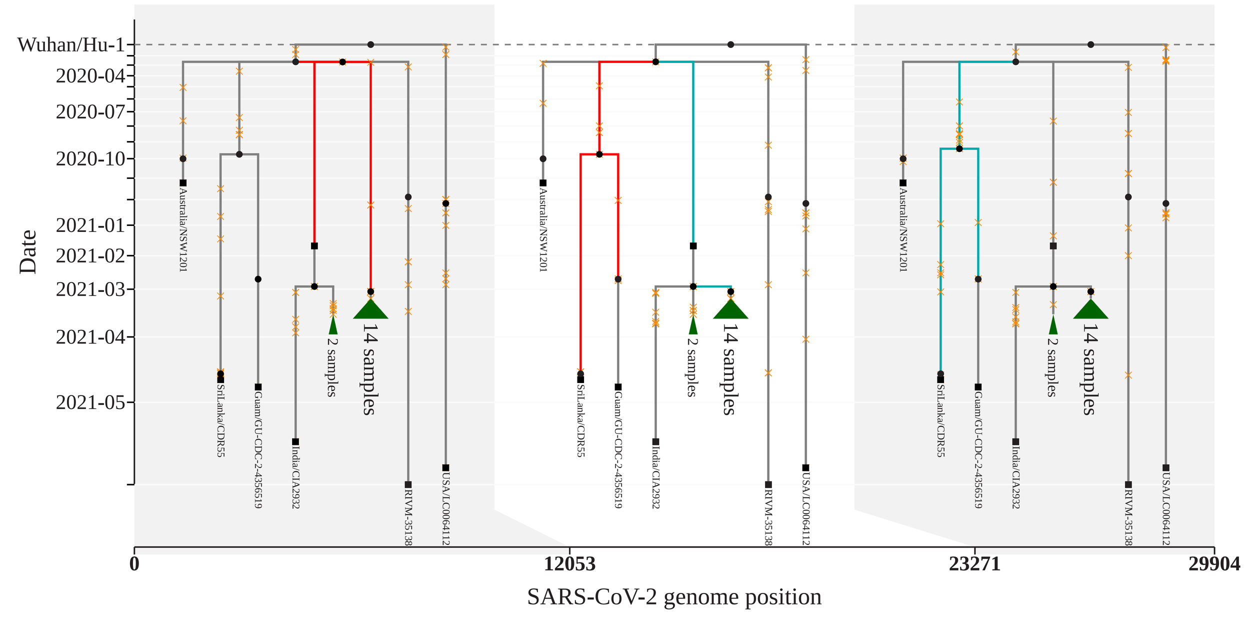
SVG-based visualization allows flexible styling of local trees.

- set labels for nodes, mutations, and tickmarks: e.g., using metadata
- color elements: e.g., to highlight branches that change between trees, mutations by type, or samples by location
- transform elements: e.g., rotate labels, alter node symbols, even 3D effects!
- timescale titles show time units by default (scaling can be linear, log, or rank)
- interaction possible via mouseover events and javascript animation
- text-based plots also available for simple debugging



Example: SARS-CoV-2 samples affected by recombination, subset from a 1.2M sample Covid tree sequence

```
simp_ts.draw_svg(
    size=(800, 400), canvas_size=(850, 405),
    style=style + "".join(node_styles),
    y_axis=True, time_scale="log_time",
    symbol_size=4.5, y_label = "Date",
    x_label = "SARS-CoV-2 genome position",
    y_ticks = y_ticks, mutation_labels={},
    y_gridlines=True, node_labels=node_labels,
    root_svg_attributes={"id": "ns_rec"},
)
```



Contributors

`tskit` is developed by an open and inclusive community. Want to get involved? All skill levels welcome – email us at `admin@tskit.dev`.

Jerome Kelleher	Yan Wong	Ben Jeffery	Murillo R.	Brian Zhang	Clemens Weiss
Peter Ralph	Kevin R. Thornton	Anthony Wilder Wohns	Jeremy Guez	Inés Rebollo	Savita Karthikeyan
Graham Gower	Georgia Tsambos	Daniel Goldstein	Saurabh Belsare	Duncan Palmer	marianne-aspbury
Andrew Kern	Shing Hei Zhan	nspope	Alexis Simon	Jeet Sukumaran	Jonathan Terhorst
Aaron Ragsdale	Hugo van Kemenade	Gertjan Bisschop	Ben Haller		

Statistics

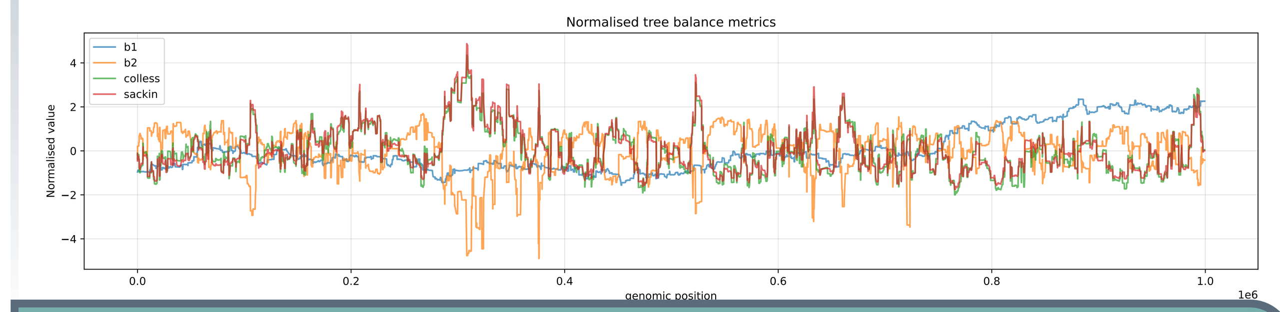
`tskit` lets you perform efficient calculations of statistics along the genome, often many times quicker than other software! You may be interested in calculating:

- the allele frequency spectrum or statistics derived from it, like nucleotide diversity, Tajima's D, f4, ...
- IBD-based quantities,
- summaries of tree topology, e.g., genealogical nearest neighbours and tree balance metrics,
- cross-coalescence rates (coming soon!)

Example: newly-implemented tree balance statistics. A balanced (binary) tree is perfectly symmetric in some way: each node's subtrees are of equal size, where 'size' is determined by some metric involving the tree's nodes and edges. `tskit` now implements several different metrics of balance:

```
imb = pd.DataFrame({
    "genomic position" : [t.interval[0] for t in ts.trees()],
    "b1" : [t.b1_index() for t in ts.trees()],
    "b2" : [t.b2_index() for t in ts.trees()],
    "colless" : [t.colless_index() for t in ts.trees()],
    "sackin" : [t.sackin_index() for t in ts.trees()]
}).set_index("genomic position")

imb = ((imb - imb.mean()) / imb.std())
imb.plot(figsize=(16, 4), alpha=0.7)
```



Notable new features

`tskit`'s contributors are actively working on new features, bug fixes, and improvements to the usability of existing features. Here's a shortlist of some recent additions:

Reference sequences By default, the sites in a tree sequence only give ancestral state at polymorphic sites. Remaining positions can now be specified using the `TreeSequence.reference_sequence`, and individual sample alignments can be obtained with the `TreeSequence.alignments()` iterator.

Structural operations We've expanded the set of utility functions for large edits on tree sequences. For instance, the `TreeSequence.decapitate` method removes all parts of a tree sequence that are older than some user-specified time, and `TreeSequence.union` joins together separate tree sequences, allowing parallel simulation across different branches of a phylogenetic tree.

Efficient array access The relationships between nodes in each tree can now be extracted as `numpy` arrays. When used with `numba`, Python-based calculations on the trees can be as fast as machine-level code. Here is numba + python computing total branch length (see example code) just as fast as the "built-in" method (implemented in C), and 10–100× faster than the un-numba'ed python code:

```
@numba.njit
def _total_branch_length(order, parent, tim
    tbl = 0
    for u in order:
      if parent[u] != -1:
        tbl += time[parent[u]] - time[u]
    return tbl

def get_total_branch_length(tree):
    return _total_branch_length(
      tree.preorder(), tree.parent_array,
      tree.tree_sequence.nodes_time)
```