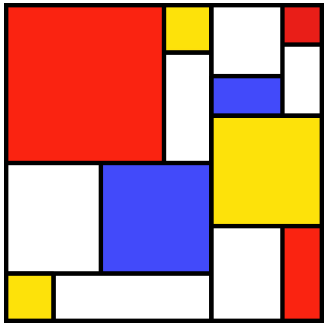
SLiM

Workshop Series



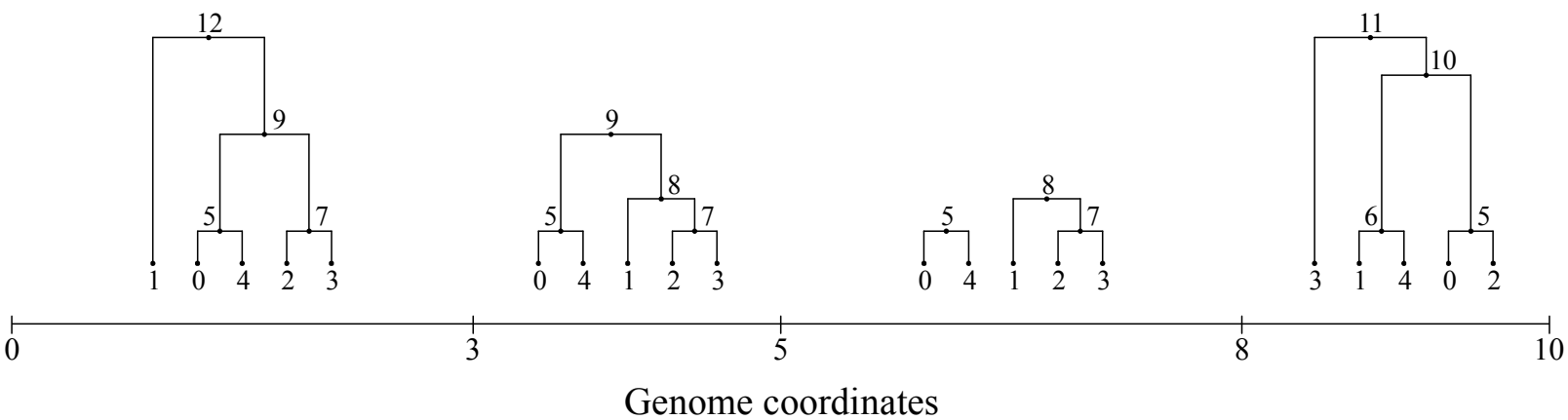
#20: Tree-sequence Recording

Benjamin C. Haller

Messer Lab, Cornell University

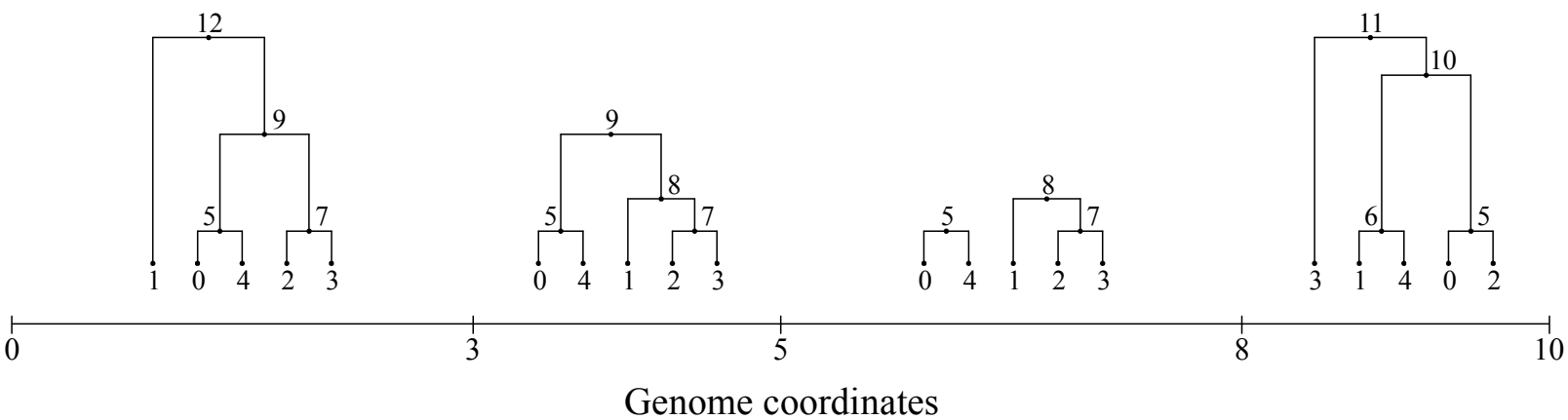
Tree sequences

- A record of the ancestry at every position
  - Originally from coalescent modeling (`msprime`)
  - Extremely compact due to correlations
  - Very fast to traverse and calculate statistics



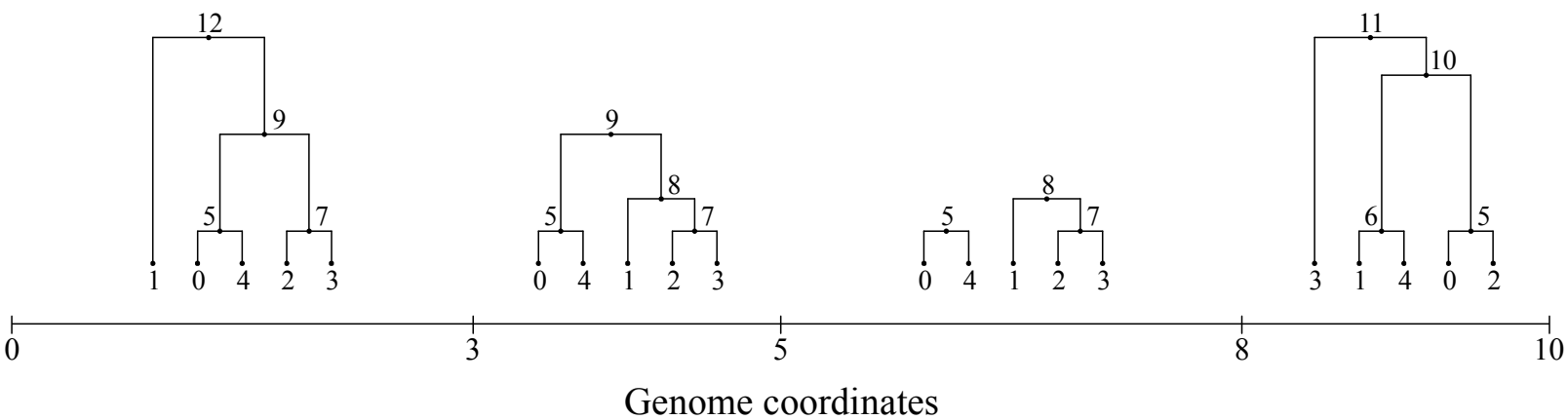
Tree sequences

- Tree sequence structure
  - Leaves are "samples" – often extant individuals
  - Internal nodes are ancestors
  - In SLiM, roots are the first generation



Tree-sequence recording

- Tracks the ancestry tree at every position
  - Neutral mutations can be overlaid after the fact
  - Neutral burn-in can be done with the coalescent
  - Recapitation can construct a coalescent history



Tree-sequence recording

- Records every new genome as a *node*
- Records every crossover as an *edge*
- Records every mutation

Tree-sequence recording

- Records every new genome as a *node*
- Records every crossover as an *edge*
- Records every mutation
- This produces a huge memory footprint!
  - **Simplification** needs to be done periodically
  - Discards branches that are extinct
  - Discards intermediate nodes along branches
  - SLiM automatically simplifies periodically
  - Explicitly simplifying can improve performance

Tree-sequence recording

- Enable tree-sequence recording
  - `initializeTreeSeq()`
- Control simplification if desired
  - `simplificationRatio, simplificationInterval`
  - `treeSeqSimplify()`
- Remember particular individuals if desired
  - `treeSeqRememberIndividuals()`
- Output a `.trees` file at completion
  - `treeSeqOutput()`

A complete tree-seq model

```
initialize() {
    initializeTreeSeq();
    initializeMutationRate(0);
    initializeMutationType("m1", 0.5, "f", 0.0);
    initializeGenomicElementType("g1", m1, 1.0);
    initializeGenomicElement(g1, 0, 1e8-1);
    initializeRecombinationRate(1e-8);
}
1 {
    sim.addSubpop("p1", 500);
}
5000 late() {
    sim.treeSeqOutput("final.trees");
}
```

- Calls `initializeTreeSeq()` and `treeSeqOutput()`
- Uses a (neutral) mutation rate of zero

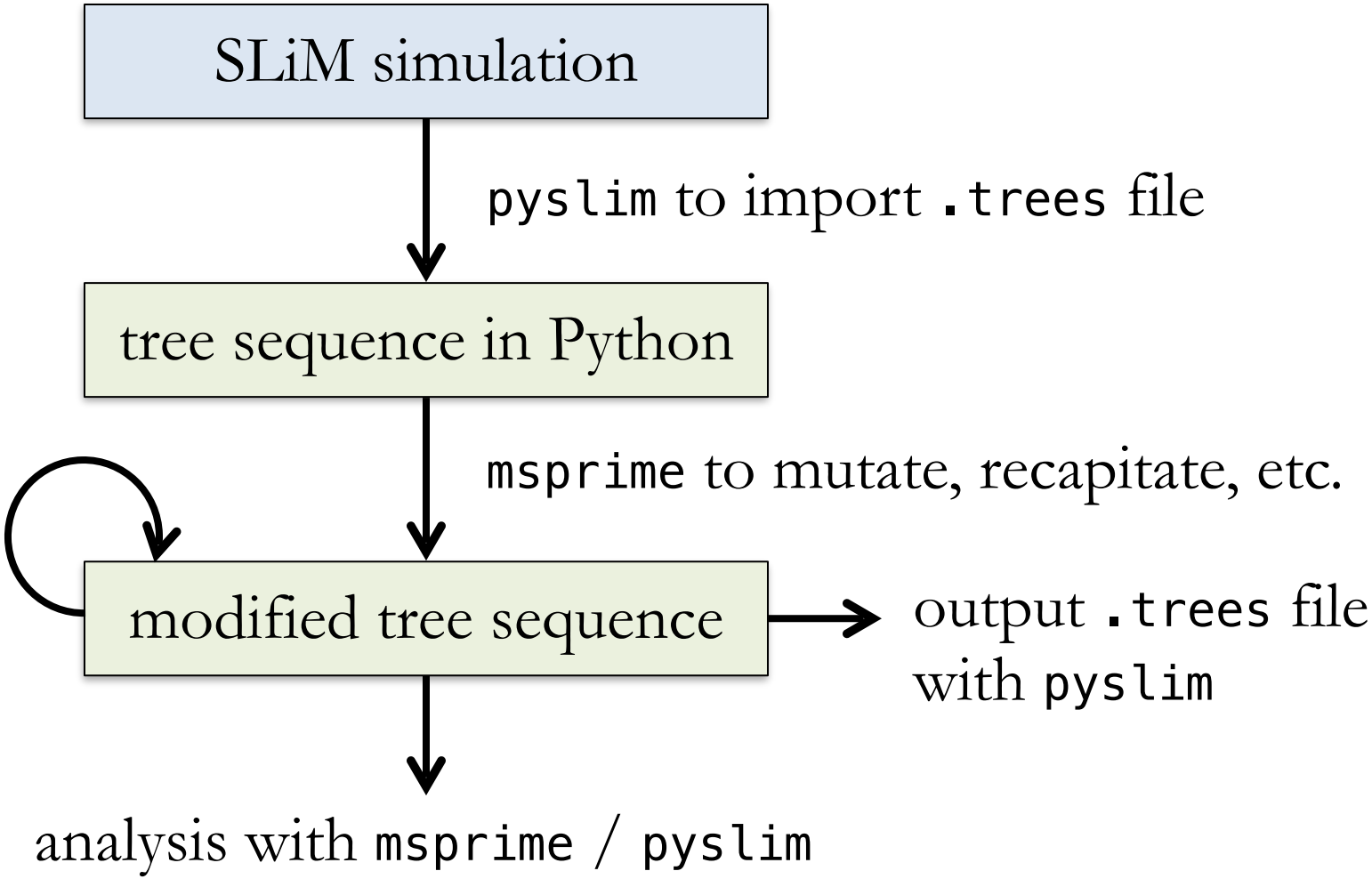
Tree-sequence analysis in Python

- SLiM:
  - runs forward genetic simulations
- `tskit`:
  - provides a foundation for tree sequences
- `msprime`:
  - performs mutation and coalescence
- `pyslim`:
  - reads and writes SLiM `.trees` files

Tree-sequence analysis in Python

- Typical workflow:
  - run a simulation in SLiM and save a `.trees` file
  - read the `.trees` file from SLiM with `pyslim`
  - mutate it, recapitate it, etc. with `msprime`
  - perform analyses upon it with Python
  - write out a modified `.trees` file with `pyslim`

Tree-sequence analysis in Python



A complete Python analysis script

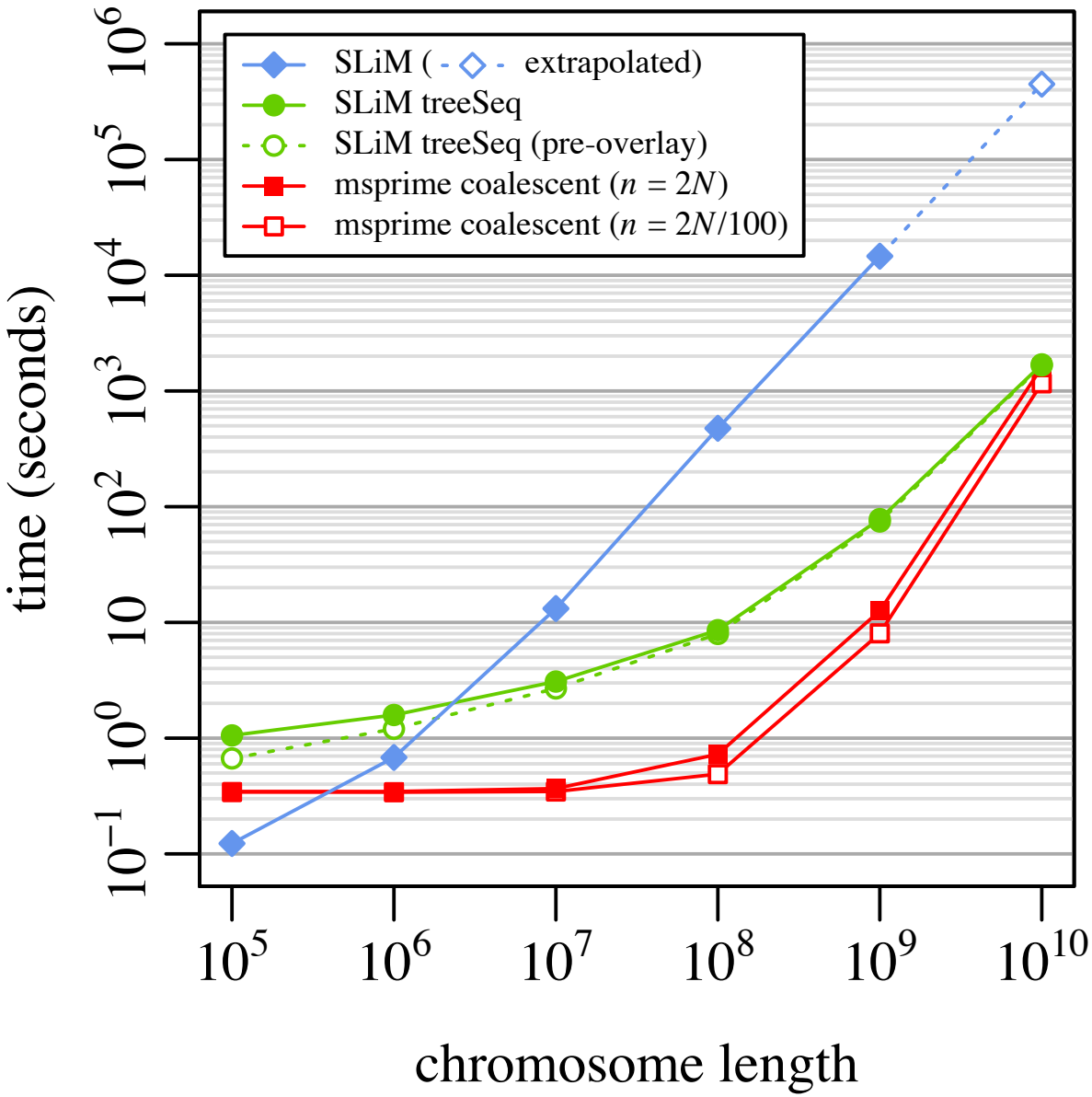
```
import msprime, pyslim

ts = pyslim.load("final.trees").simplify()
mutated = msprime.mutate(ts, rate=1e-7, random_seed=1, keep=True)
mutated.dump("final_overlaid.trees")
```

- Import `msprime` and `pyslim` packages
- Load the saved `.trees` file with `pyslim`
- Use `msprime` to overlay mutations
- Write out the new tree sequence

Tree-sequence recording

- What's the point again?
- Ancestry information is useful
- **Speed**
  - Without tree-seq, 211.9 seconds
  - With tree-seq, 4.37 seconds
  - Almost a **50×** speedup
  - Why? Neutral mutations are *overlaid*
  - Memory usage is also lower



Haller et al. (2019), Molecular Ecology Resources

Recapitation

- Forward simulation needs burn-in
  - provides an equilibrium initial state
- Burn-in can take a very long time!
- `msprime` can do a coalescent burn-in
- But recapitation is even better:
  - allows neutral burn-in to be skipped
  - a coalescent history is added *afterwards*
  - neutral mutations can then be overlaid
  - **even faster** than a coalescent burn-in

Resources

Received: 10 September 2018 | Revised: 6 November 2018 | Accepted: 9 November 2018
DOI: 10.1111/1755-0998.12968

RESOURCE ARTICLE

WILEY MOLECULAR ECOLOGY RESOURCES

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

Benjamin C. Haller[1] | Jared Galloway[2] | Jerome Kelleher[3] |
Philipp W. Messer[1,*] | Peter L. Ralph[2,*]

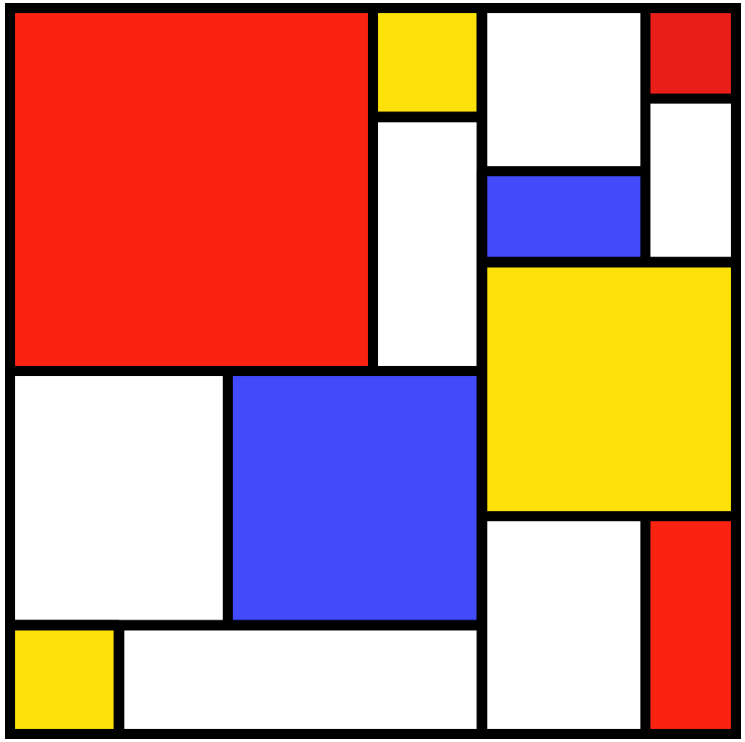
PLOS | COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

Efficient pedigree recording for fast population genetics simulation

Jerome Kelleher[1], **Kevin R. Thornton**[2], **Jaime Ashander**[3], **Peter L. Ralph**[4]*

1 Big Data Institute, University of Oxford, Oxford, United Kingdom, **2** Ecology and Evolutionary Biology, University of California, Irvine, Irvine, California, United States of America, **3** Ecology and Evolutionary Biology, University of California, Los Angeles, Los Angeles, United States of America, **4** Institute for Ecology and Evolution, University of Oregon, Eugene, Oregon, United States of America



SLiM Workshop Exercise #20