Modeling Population and Quantitative Genetics in SLiM

forward genetic simulation software

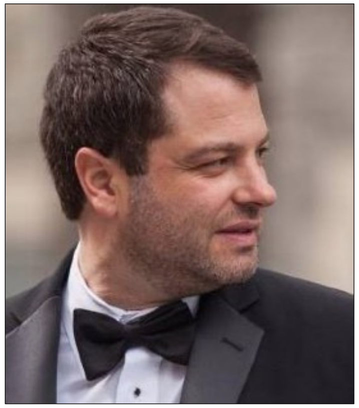
KITP, 2022

Benjamin C. Haller

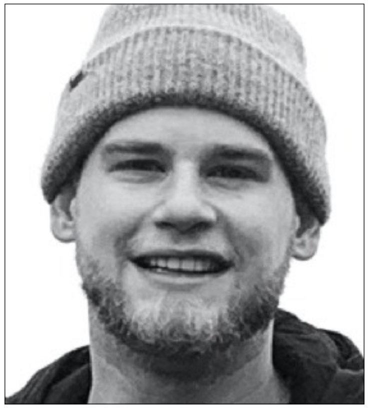
Messer Lab, Cornell University

SLiM Contributors

Ben Haller


Philipp Messer

Peter Ralph


Jared Galloway

Jerome Kelleher

Ben Jeffery

… *and many more!*

Talk outline

- What is SLiM? Why use SLiM?
- An introduction to Eidos and SLiM
- A survey of example models
- Population-genetic models in SLiM
- Quantitative-genetic models in SLiM
- Closing remarks

Talk outline

- What is SLiM? Why use SLiM?
- An introduction to Eidos and SLiM
- A survey of example models
- Population-genetic models in SLiM
- Quantitative-genetic models in SLiM
- Closing remarks

Forward genetic simulation

- "**Forward**":
  - runs forward from an initial state
    (vs. coalescent methods)
- "**Genetic**":
  - explicit loci on a chromosome
    (vs. phenotypic simulations)
- "**Simulation**":
  - individual-based modeling
    (vs. analytical modeling)

Why do simulations?

- Fitting **empirical** population genomic data
  - analyzing past evolutionary forces
  - predicting future evolution
- Analyzing **theoretical** evolutionary models
  - predicting the consequences of a new theory
  - comparing multiple theories
- Developing **statistical** methods
  - testing a method's accuracy, bias, or power
  - generating datasets for machine learning

Why do *forward* simulations?

- **Evolution is complex**:
  - complex demography and population structure
  - non-random mating and complex mating systems
  - spatial structure and non-random dispersal
  - spatial and temporal variation in selection
  - frequency-dependent selection, kin selection, etc.
  - realistic genetic/chromosomal structure
  - epistasis, polygenic traits, pleiotropy
  - multiple loci under selection
  - variable recombination / mutation rate

Why use SLiM?

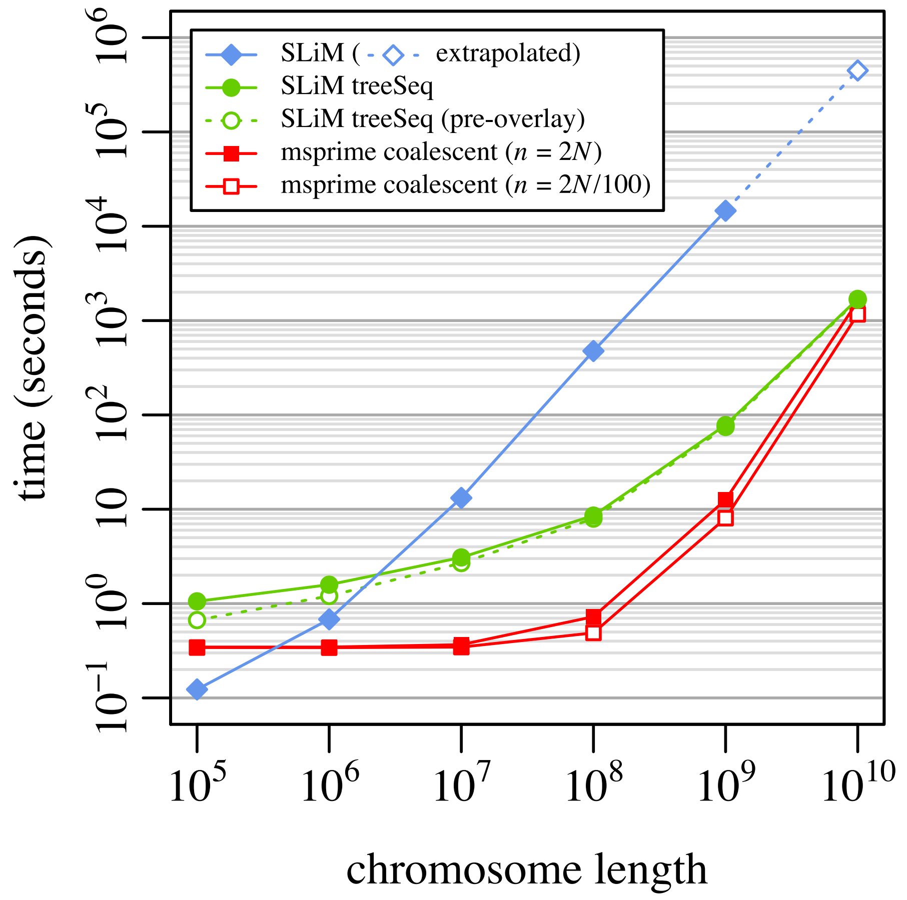
Why use SLiM?

- Flexible and customizable with Eidos
  - scriptability means there are (almost) no limits
  - similar to R in its syntax and function names
  - ends statements with semicolons; zero-based!

```
x = 0;
for (i in 1:10)
    x = x + i;
print(x);
```

Why use SLiM?

- Very fast: SLiM is highly optimized



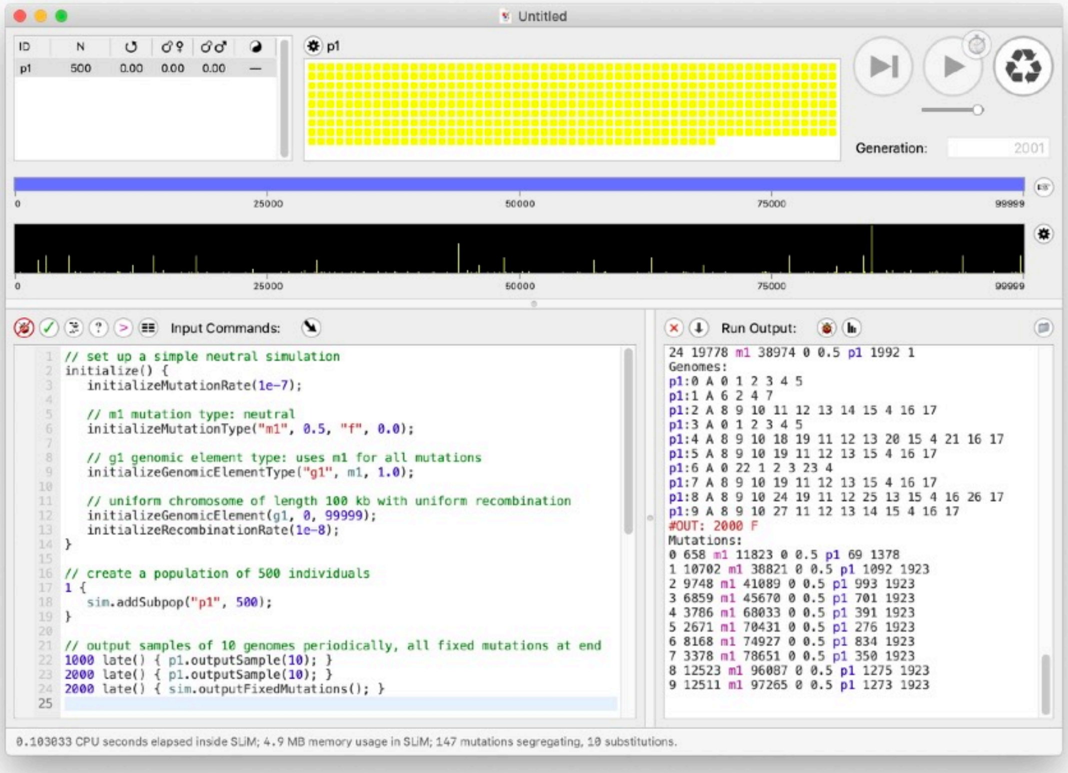
neutral simulation
mean over 10 replicates
N = 500 diploids
$r = 10^{-8}$
$\mu = 10^{-7}$

run to the expected time for coalescence (~3N to ~15N)

Haller et al., Mol. Ecol. Res (2019)

Why use SLiM?

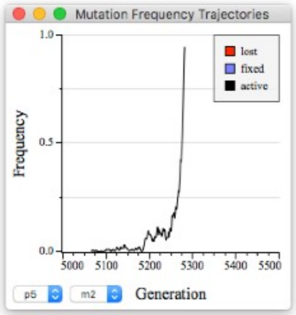
- Interactive and graphical
  - – easy to visualize / debug / explore

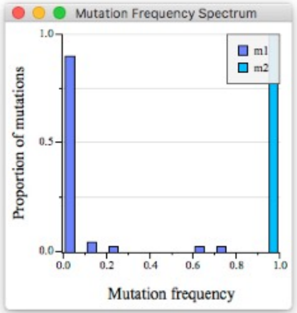


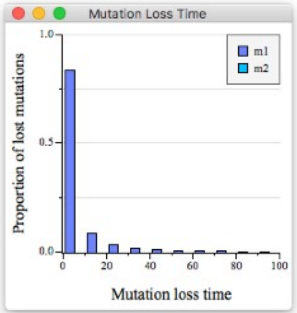
```
// set up a simple neutral simulation
initialize() {
   initializeMutationRate(1e-7);

   // m1 mutation type: neutral
   initializeMutationType("m1", 0.5, "f", 0.0);

   // g1 genomic element type: uses m1 for all mutations
   initializeGenomicElementType("g1", m1, 1.0);

   // uniform chromosome of length 100 kb with uniform recombination
   initializeGenomicElement(g1, 0, 99999);
   initializeRecombinationRate(1e-8);
}

// create a population of 500 individuals
1 {
   sim.addSubpop("p1", 500);
}

// output samples of 10 genomes periodically, all fixed mutations at end
1000 late() { p1.outputSample(10); }
2000 late() { p1.outputSample(10); }
2000 late() { sim.outputFixedMutations(); }
```

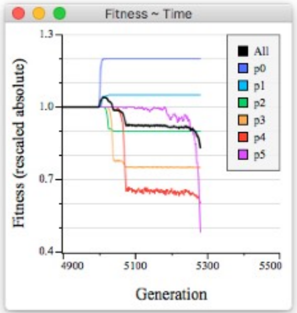
```
24 19778 m1 38974 0 0.5 p1 1992 1
Genomes:
p1:0 A 0 1 2 3 4 5
p1:1 A 6 2 4 7
p1:2 A 8 9 10 11 12 13 14 15 4 16 17
p1:3 A 0 1 2 3 4 5
p1:4 A 8 9 10 18 19 11 12 13 20 15 4 21 16 17
p1:5 A 8 9 10 19 11 12 13 15 4 16 17
p1:6 A 0 22 1 2 3 23 4
p1:7 A 8 9 10 19 11 12 13 15 4 16 17
p1:8 A 8 9 10 24 19 11 12 25 13 15 4 16 26 17
p1:9 A 8 9 10 27 11 12 13 14 15 4 16 17
#OUT: 2000 F
Mutations:
0 658 m1 11823 0 0.5 p1 69 1378
1 10702 m1 38821 0 0.5 p1 1092 1923
2 9748 m1 41089 0 0.5 p1 993 1923
3 6859 m1 45670 0 0.5 p1 701 1923
4 3786 m1 68033 0 0.5 p1 391 1923
5 2671 m1 70431 0 0.5 p1 276 1923
6 8168 m1 74927 0 0.5 p1 834 1923
7 3378 m1 78651 0 0.5 p1 350 1923
8 12523 m1 96087 0 0.5 p1 1275 1923
9 12511 m1 97265 0 0.5 p1 1273 1923
```

0.103033 CPU seconds elapsed inside SLiM; 4.9 MB memory usage in SLiM; 147 mutations segregating, 10 substitutions.









Why use SLiM?

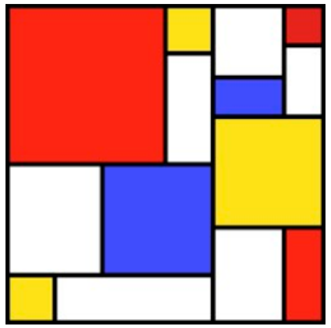
- Open source on GitHub, GPL license
  - free, reusable, shareable, debugged, easy!

GPL 3 Free Software

Free as in Freedom

Talk outline

- What is SLiM? Why use SLiM?
- An introduction to Eidos and SLiM
- A survey of example models
- Population-genetic models in SLiM
- Quantitative-genetic models in SLiM
- Closing remarks



live demo

An introduction to Eidos

Data types

- `NULL`: no explicit value
- `logical`: a Boolean true/false value (`T`/`F`)
- `integer`: a 64-bit signed integer (`10`, `–27`)
- `float`: a floating-point number (`10.0`, `–2.7`)
- `string`: a sequence of characters (`"foo"`)
- `object`: an instance of a class (`Individual`)

Operators

• Arithmetic: `+, -, *, /, %, ^, :`	`6 + 2*7`
• Logical: `&, |, !`	`T & !F`
• Comparison: `<, >, <=, >=, ==, !=`	`2+2 == 4`
• Assignment: `=`	`x = 8`
• Precedence and function call: `()`	`(6+2) * 7`
• Subset: `[]`	`x[5]`
• Property access and method call: `.`	`foo.bar`

Control flow

- **if-else**: conditional execution
  - `if (condition) statement; else statement;`
- **while**: loop on a condition, 0+ times
  - `while (condition) statement;`
- **do-while**: loop on a condition, 1+ times
  - `do statement; while (condition);`
- **for**: loop over the values in a vector
  - `for (i in vector) statement;`

Built-in functions

- **Math**: `abs()`, `ceil()`, `log()`, `setUnion()`, …
- **Statistics**: `max()`, `mean()`, `sd()`, `cov()`, …
- **Distributions**: `rnorm()`, `rpois()`, `runif()`, …
- **Vectors**: `c()`, `rep()`, `seq()`, `sample()`, …
- **Values**: `all()`, `any()`, `identical()`, `sort()`, …
- **Output**: `cat()`, `print()`, `paste()`, `str()`, …
- **Types**: `isFloat()`, `asFloat()`, …
- **Filesystem**: `readFile()`, `writeFile()`, …

Objects, properties, methods

- Objects represent **entities**:
  - e.g., individuals, mutations, subpopulations
- Objects have **properties**:
  - attributes like `age`, `sex`, `spatialPosition`
  - `individual.age` returns the age of `individual`
  - `individual.age = 10;` changes its age
- Objects have **methods**:
  - methods perform complex operations
  - `individual.containsMutations(muts)`

An introduction to SLiM

SLiM Eidos classes

- Chromosome hierarchy:
  - Chromosome
  - MutationType
  - GenomicElementType
  - GenomicElement

- Other classes:
  - InteractionType
  - LogFile
  - SLiMEidosBlock
  - SLiMgui

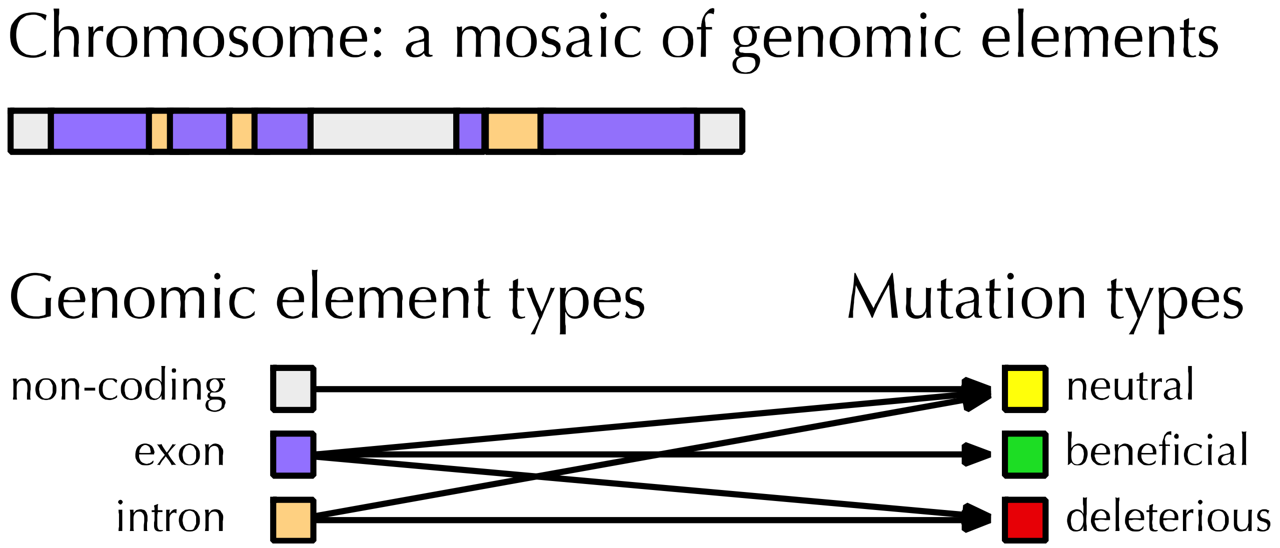
- Community hierarchy:
  - Community
  - Species
  - Subpopulation
  - Individual
  - Genome
  - Mutation
  - Substitution

- Popgen utilities, nucleotide utilities, etc.

The chromosome hierarchy

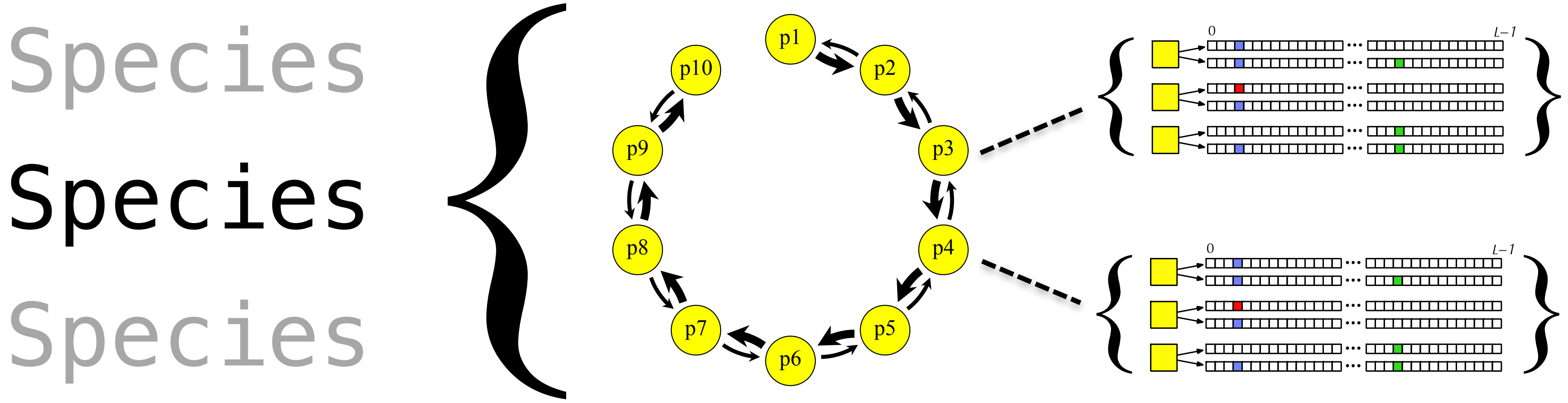
Genomic elements, genomic element types, and mutation types

- The chromosome defines the genetic structure
- It is a sequence of elements (`GenomicElement`)
- Each element has a type (`GenomicElementType`)
- Each type draws from a DFE (`MutationType`)



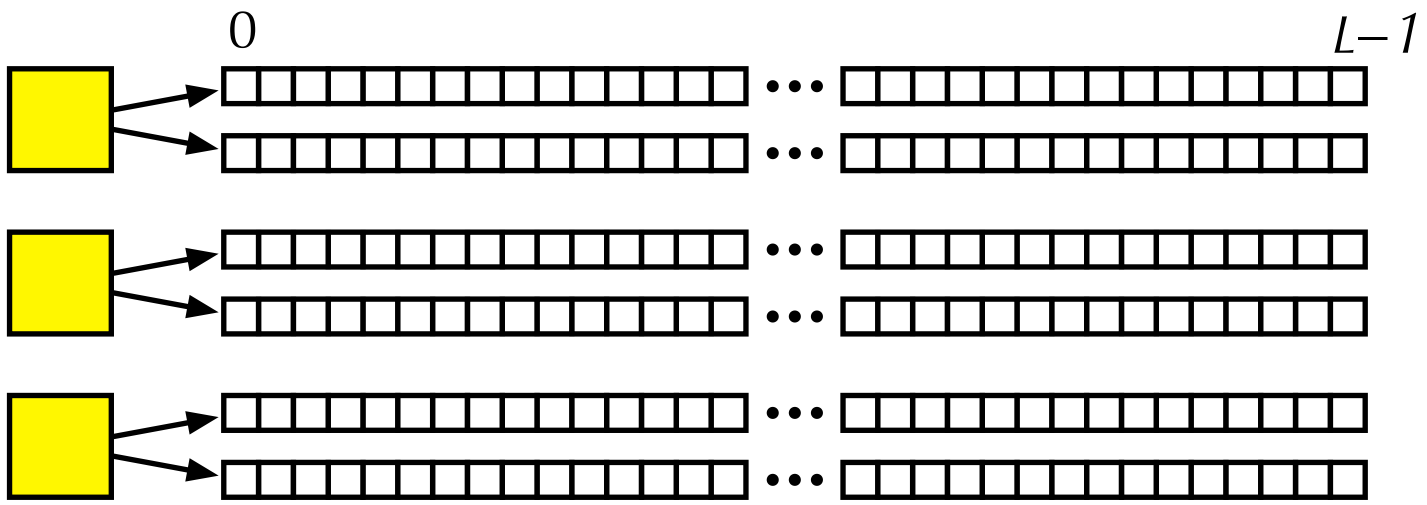
The community hierarchy

- The *community* contains one or more *species*
- Each species contains *subpopulations*
- Each subpopulation contains *individuals*
- Each individual contains *genomes*
- Each genome contains *mutations*



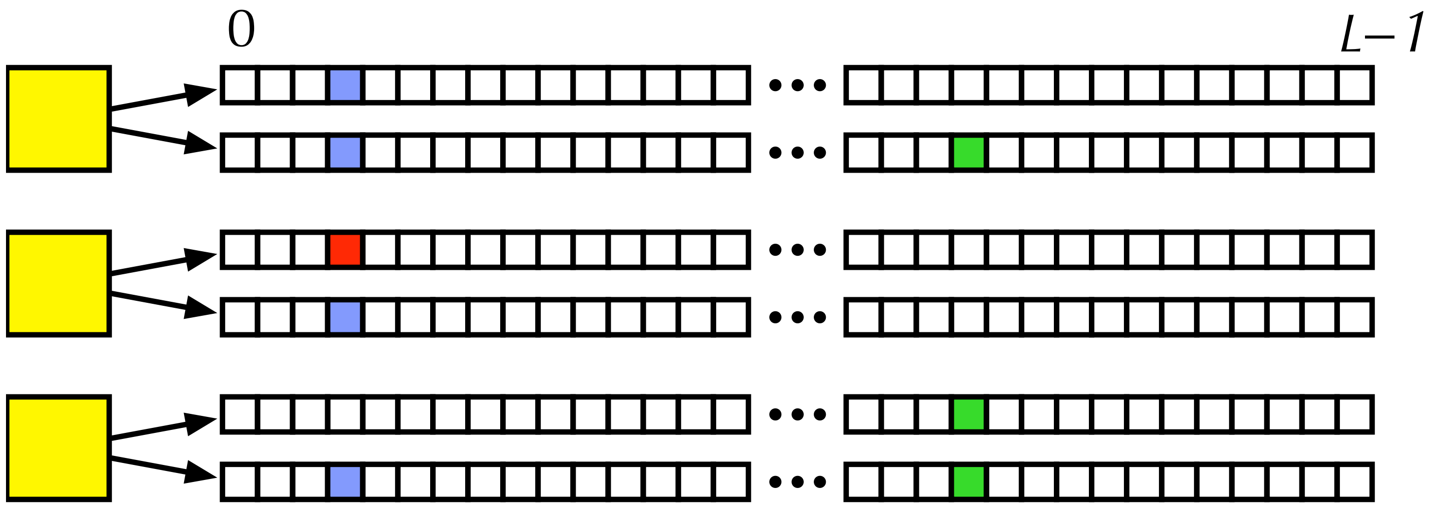
Individuals and genomes

- Individuals (class `Individual`) are organisms
- Individuals are born, mate, die, …
- Each individual has two genomes (class `Genome`)
- Each genome has L discrete base positions



Mutations

- Mutations (class `Mutation`) live in genomes
- Genomes begin empty (the *ancestral* state)
- Mutations represent a *non-ancestral* allele (SNP)
- Mutations have properties: selection coefficient



The “SLiM core”

- SLiM is divided into the “core”…
  - Optimized C++ to run the tick cycle
  - Default behaviors for simple models:
    - Fitness, mate choice, gamete generation, mutation, recombination, mortality, …
- … and everything else, in Eidos script:
  - Events (`first()`/`early()`/`late()`)
  - Callbacks

SLiM callbacks

- `fitness()`: individual-based fitness effects
- `mateChoice()`: non-random mating effects in WF models
- `reproduction()`: scripted reproduction in nonWF models
- `modifyChild()`: individual customization of offspring

- `mutation()`: customization of the mutational process
- `recombination()`: customization of recombination
- `survival()`: influencing mortality in nonWF models
- `interaction()`: spatial interactions (competition, mating)

WF versus nonWF

- SLiM supports two different model types:
  - **WF**: Wright–Fisher
  - **nonWF**: non-Wright–Fisher
- Differences:
  - tick cycles
  - offspring generation
  - population regulation
  - age structure
  - fitness models
  - demography
  - mate choice
  - migration

WF Models

- Population size is a parameter
- Population regulation is automatic
- Fitness affects mating probability
- Selection is soft (relative fitness)
- Non-overlapping generations
- No age structure
- Migration is due to parameters

nonWF Models

- Population size is emergent
- Population regulation is scripted
- Fitness affects viability/survival
- Selection is hard (absolute fitness)
- Generations can overlap
- Age structure is emergent
- Migration is scripted

Fitness

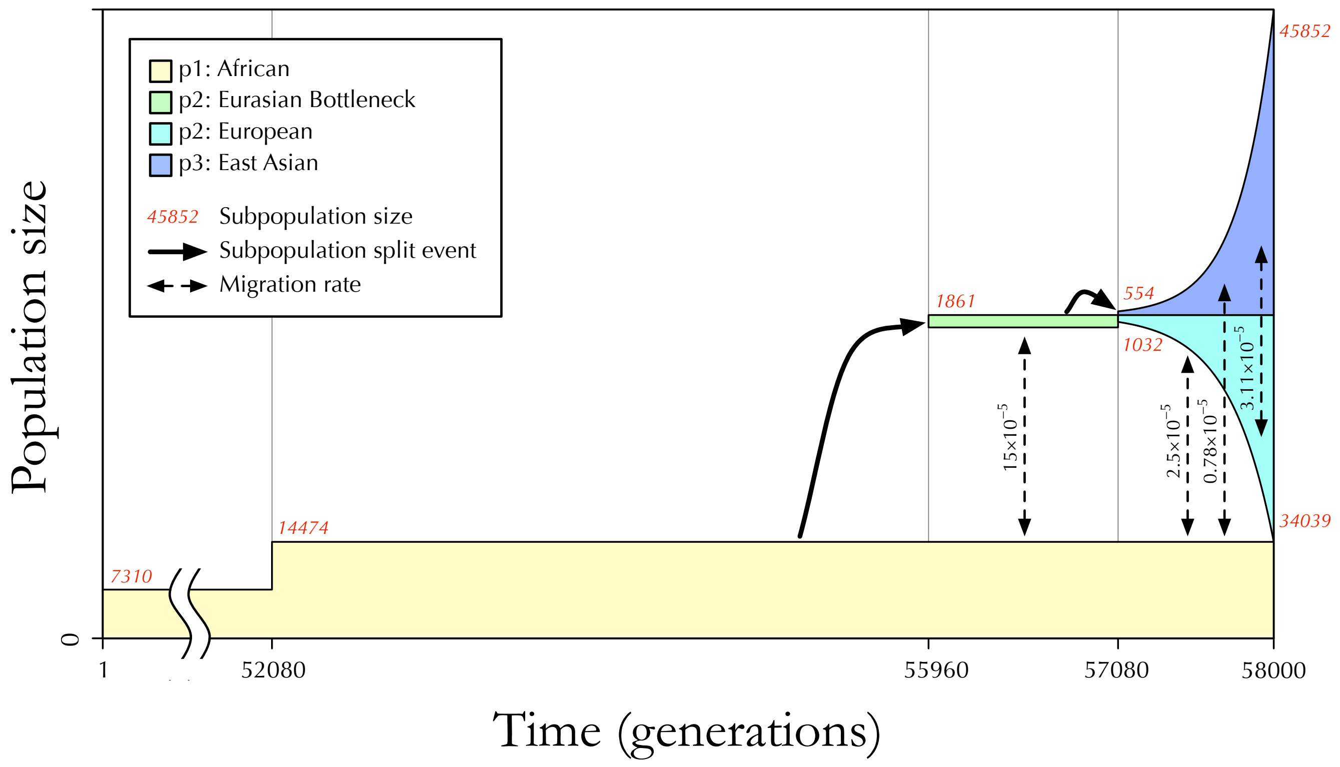
- Fitness is a multiplicative combination of *effects*
- Fitness effects come from:
  - mutations (1 / 1+*hs* / 1+*s*)
  - `fitness()` callbacks – *per mutation*
  - individual `fitnessScaling` values
  - subpopulation `fitnessScaling` values
- This allows:
  - QTLs, behavior, spatiotemporal variation…

Talk outline

- What is SLiM? Why use SLiM?
- An introduction to Eidos and SLiM
- A survey of example models
- Population-genetic models in SLiM
- Quantitative-genetic models in SLiM
- Closing remarks

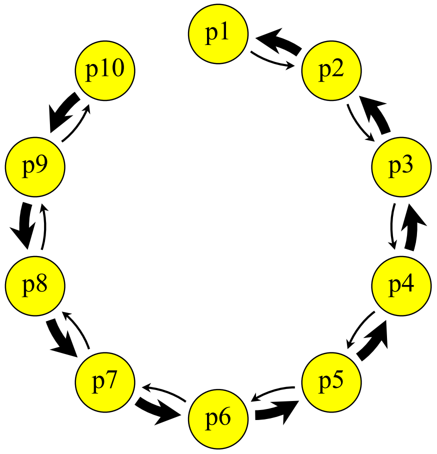
The Gravel model (5.4)

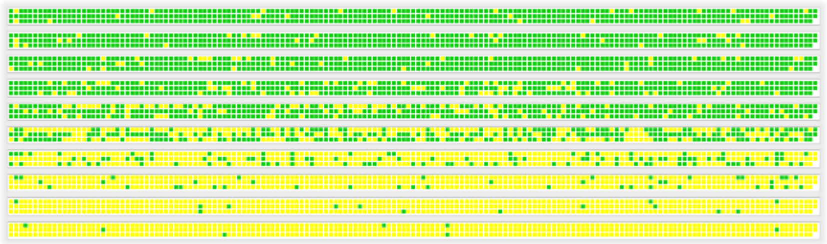
- Simulating human evolutionary history
- Demographic events, exponential growth

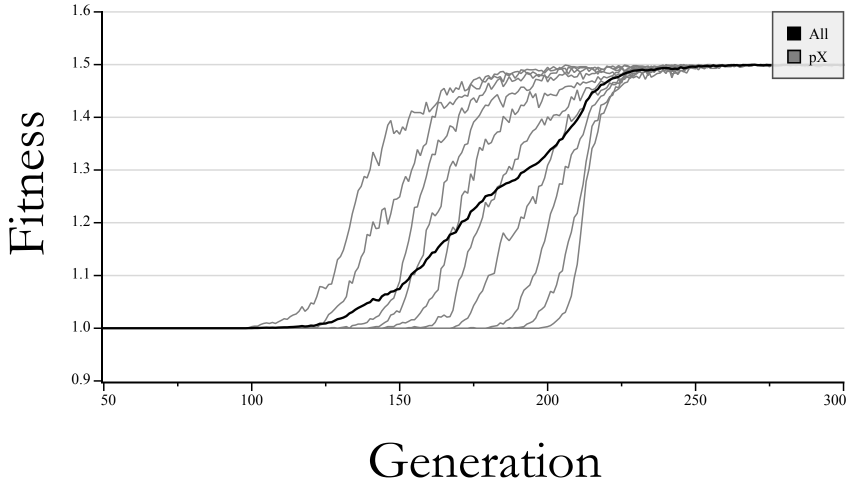


Introgression & sweeps (9.7)

- Introgression of a single introduced mutation
- Ten subpopulations connected by migration

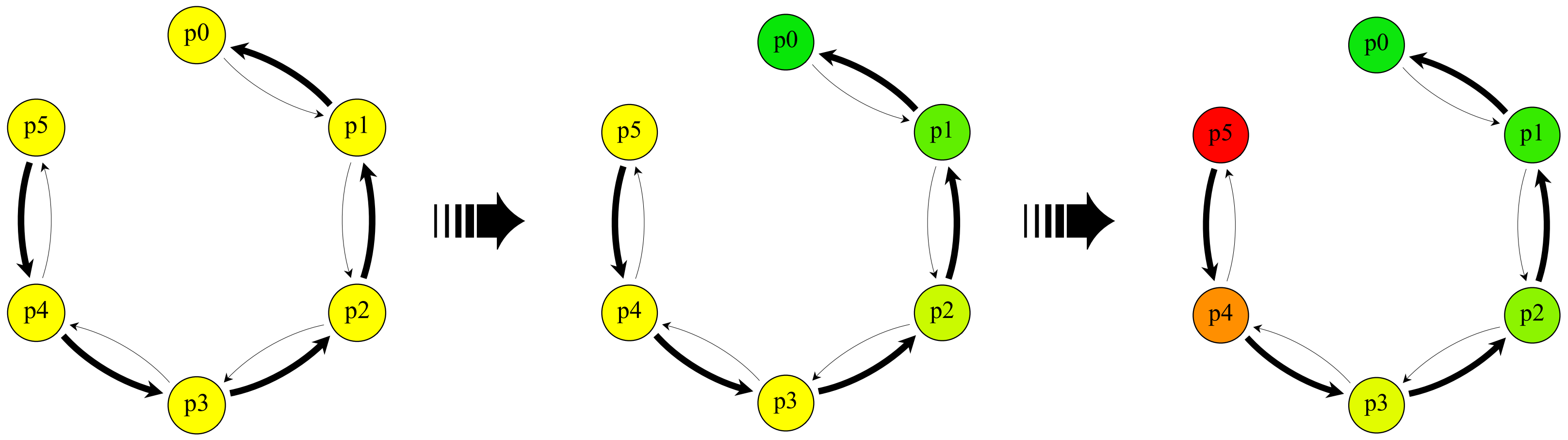






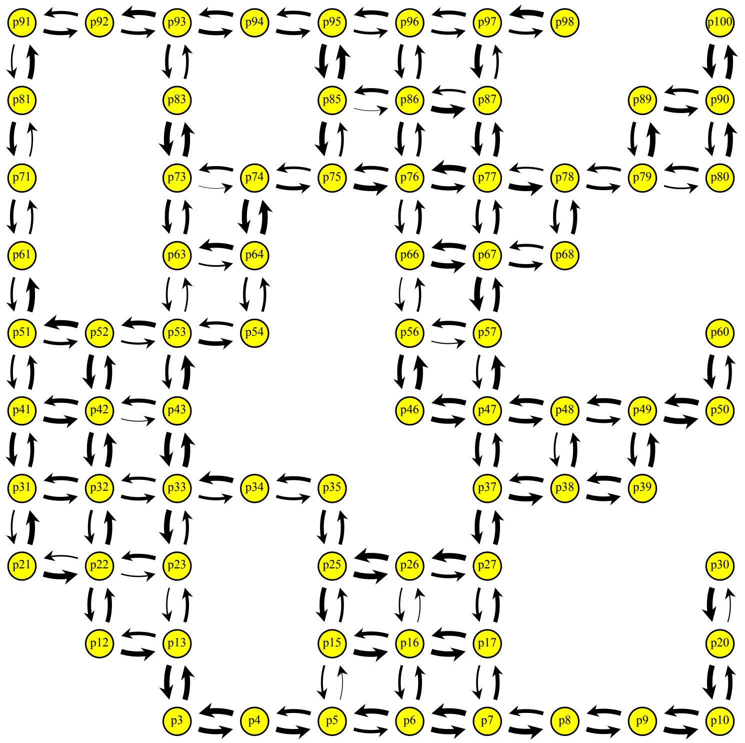
Gene drives (12.3)

- Simulating CRISPR gene drive
- Fixes despite negative fitness effects
- Fixes despite going against migration



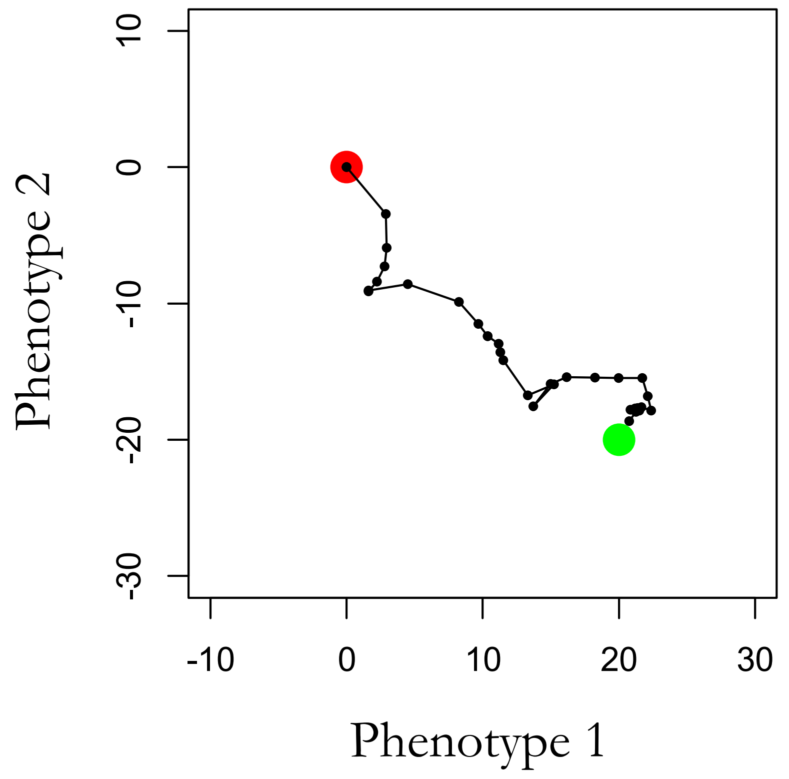
Metapopulations (5.3.4)

- Many subpopulations connected by migration
- The connection pattern can be spatial, or not


p91 p92 p93 p94 p95 p96 p97 p98 p100
p81 p83 p85 p86 p87 p89 p90
p71 p73 p74 p75 p76 p77 p78 p79 p80
p61 p63 p64 p66 p67 p68
p51 p52 p53 p54 p56 p57 p60
p41 p42 p43 p46 p47 p48 p49 p50
p31 p32 p33 p34 p35 p37 p38 p39
p21 p22 p23 p25 p26 p27 p30
p12 p13 p15 p16 p17 p20
p3 p4 p5 p6 p7 p8 p9 p10

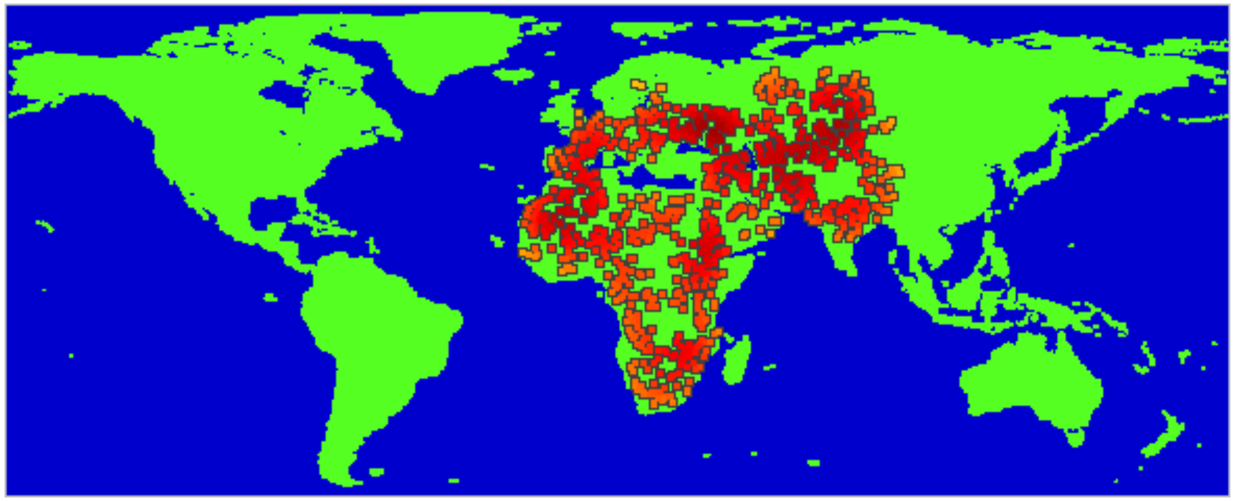
Adaptive walks (14.8)

- A QTL-based model with pleiotropy (M-matrix)
- Two phenotypic traits defined by additive QTLs



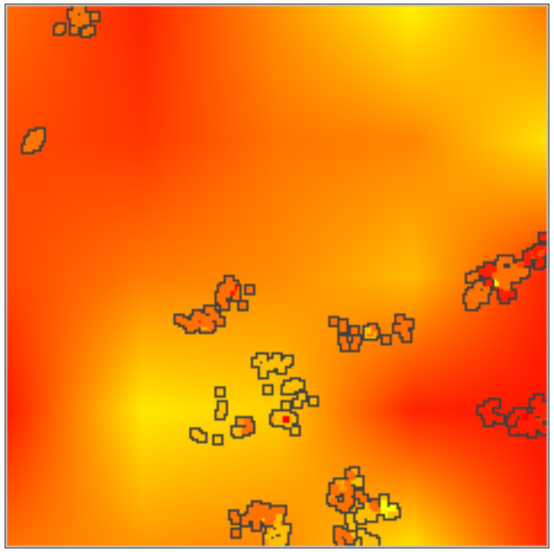
Continuous space (15.10)

- Individuals live in a continuous 2-D space
- A landscape map of the world is used
- Population expansion out of Africa



Local adaptation (15.11)

- Individuals live in a continuous 2-D space
- A map defines a heterogeneous environment
- Adaptation to the local environment results



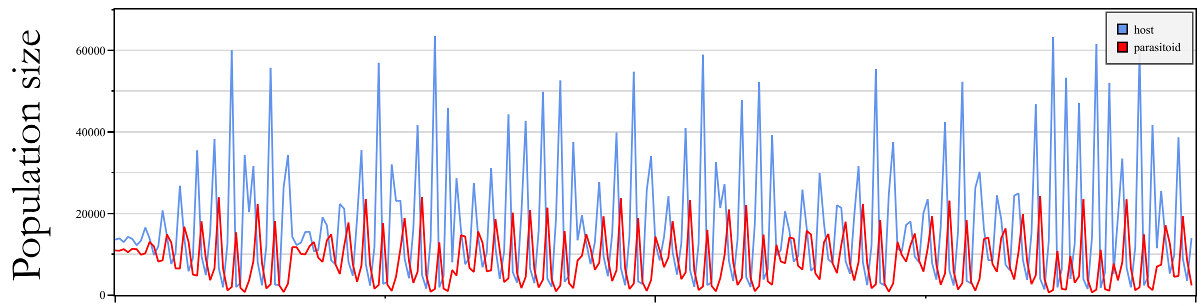
Nucleotide-based models (18.1)

- Track the nucleotide sequence of every genome
- Mutations have an associated nucleotide
- Mutation rates are sequence-dependent
- Realistic gene conversion, including gBGC

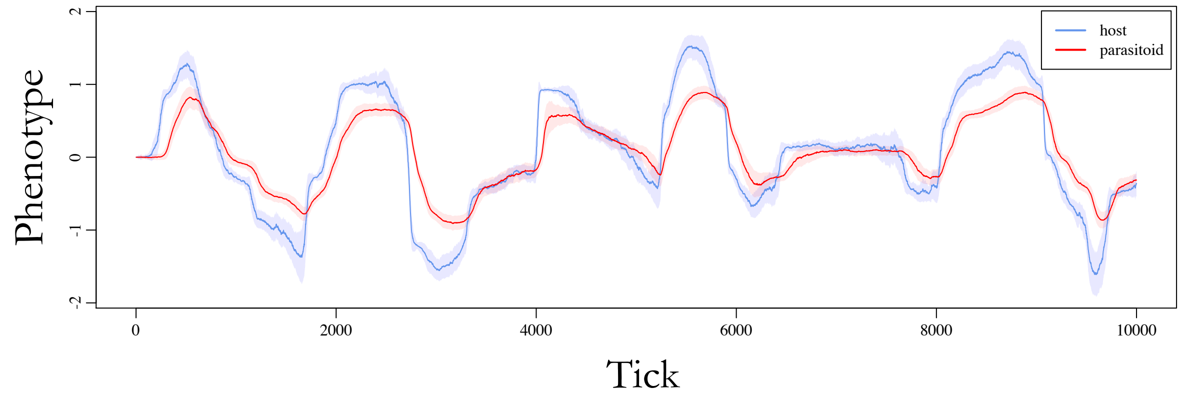
```
GAATGTCGGTTAGAGCAACCTAGCTTCTCAGATCGCAATA
GAATGTCCGTTAGAGCAACCTAGCTTCTCAGATGGCTATA
GAATGTCCGTTAGAGCAACCTAGCTTCTCAGATGGCCATA
GAATGTCGGTTAGAGCATCCTAGCTTCTCAGATCGCAATA
GAATGTCGGTTAGAGCAACCTAGCTTCTCAGATCGCAATA
GAATGTCCGTTAGAGCAACCTAGCTTCTCAGATGGCAATA
GAATGTCGGTTAGAGCATCCTAGCCTCTCAGATGGCAATA
GAATGTCGGTTAGAGCATCCTAGCTTCTCAGATCGCAATA
```

Multispecies models (19.4 & 19.6)

- Simulate more than one species in a model
- Ecology: competition, predation, parasitism, …
- Coevolutionary and eco-evolutionary dynamics



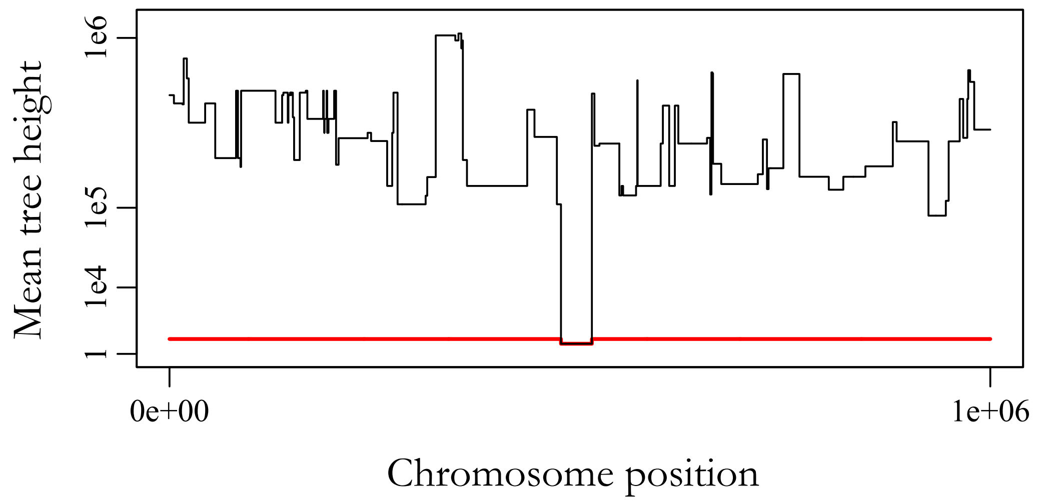
19.4: Host-parasitoid population size cycling



19.6: Red Queen coevolutionary dynamics

Tree sequences & ancestry (17.10)

- Tracking the ancestry tree at every position
- Mean tree height is a proxy for diversity at a site
- After a sweep, diversity is lowest near the sweep
- Recapitation constructs neutral burn-in history



Talk outline

- What is SLiM? Why use SLiM?
- An introduction to Eidos and SLiM
- A survey of example models
- Population-genetic models in SLiM
- Quantitative-genetic models in SLiM
- Closing remarks

Population-genetic models

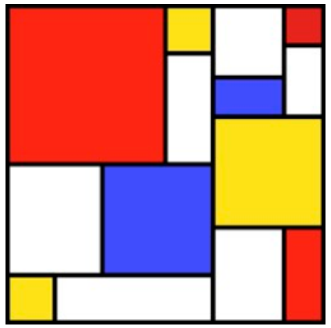
- Demography:
  - Discrete subpopulations with migration
  - Continuous space with dispersal
- Ploidy:
  - Diploid, haploid, haplodiploid*, …*
- Sex and mating:
  - Hermaphrodites, separate sexes, X/Y, …*
  - Biparental sexual mating
  - Cloning, selfing, horizontal gene transfer*, …*

Population-genetic models

- Any chromosome length
- Any number of chromosomes*
- Any number of loci and alleles
- Any recombination-rate map
- Any mutation-rate map
- Any distributions of fitness effects
- Any individual effects* on fitness, mortality, mate choice, recombination, migration, dispersal, mutation, …

Population-genetic models

- Selective sweeps
  - partial or full
  - conditional on fixation or establishment
  - soft and hard sweeps of various kinds
- Background selection
- Local adaptation
- Introgression
- Reproductive isolation (partial or full)
- Speciation



live demo

Talk outline

- What is SLiM? Why use SLiM?
- An introduction to Eidos and SLiM
- A survey of example models
- Population-genetic models in SLiM
- Quantitative-genetic models in SLiM
- Closing remarks

Quantitative Traits

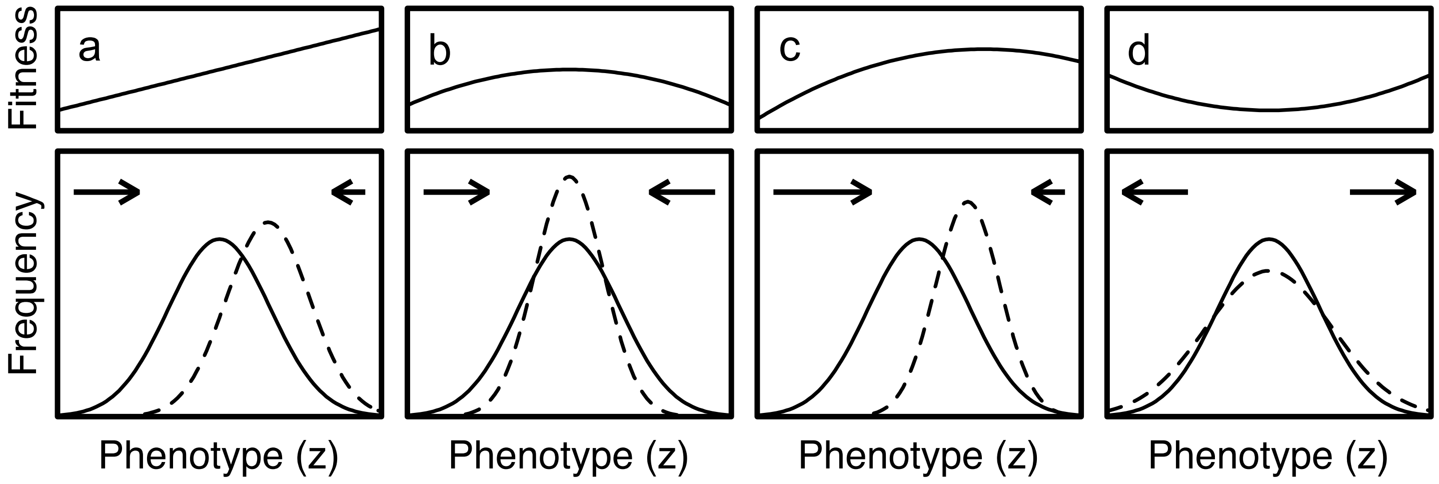
- Mendelian traits:
  - governed by a single locus
  - produce discrete outcomes
  - can be modeled with selection coefficients
- Quantitative traits:
  - governed by multiple loci: QTLs
  - produce continuous variation (e.g., height)
  - need to be modeled via **phenotype**

Quantitative Traits

- Phenotype
  - calculated from QTL effects
  - **additive effects** are central (breeding value)
  - non-additive effects can also be modeled
    - dominance, epistasis
  - environmental noise can be added
  - phenotypic plasticity can be included
  - the final result: a **phenotypic trait value**

Quantitative Traits

- Fitness
  - a function of phenotypic trait value
  - often modeled as a **fitness function**



directional stabilizing both disruptive

others:

balancing

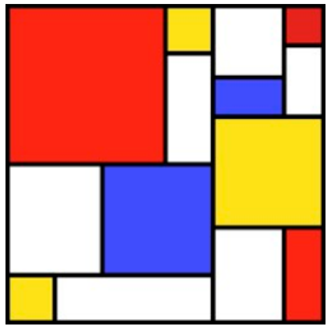
truncating

"squashed stabilizing"

etc.

The Big Picture

- QTL mutations have an effect size
- Phenotype is the sum of all effects
- Fitness is some function of phenotype
- Fitness effects are assigned to individuals

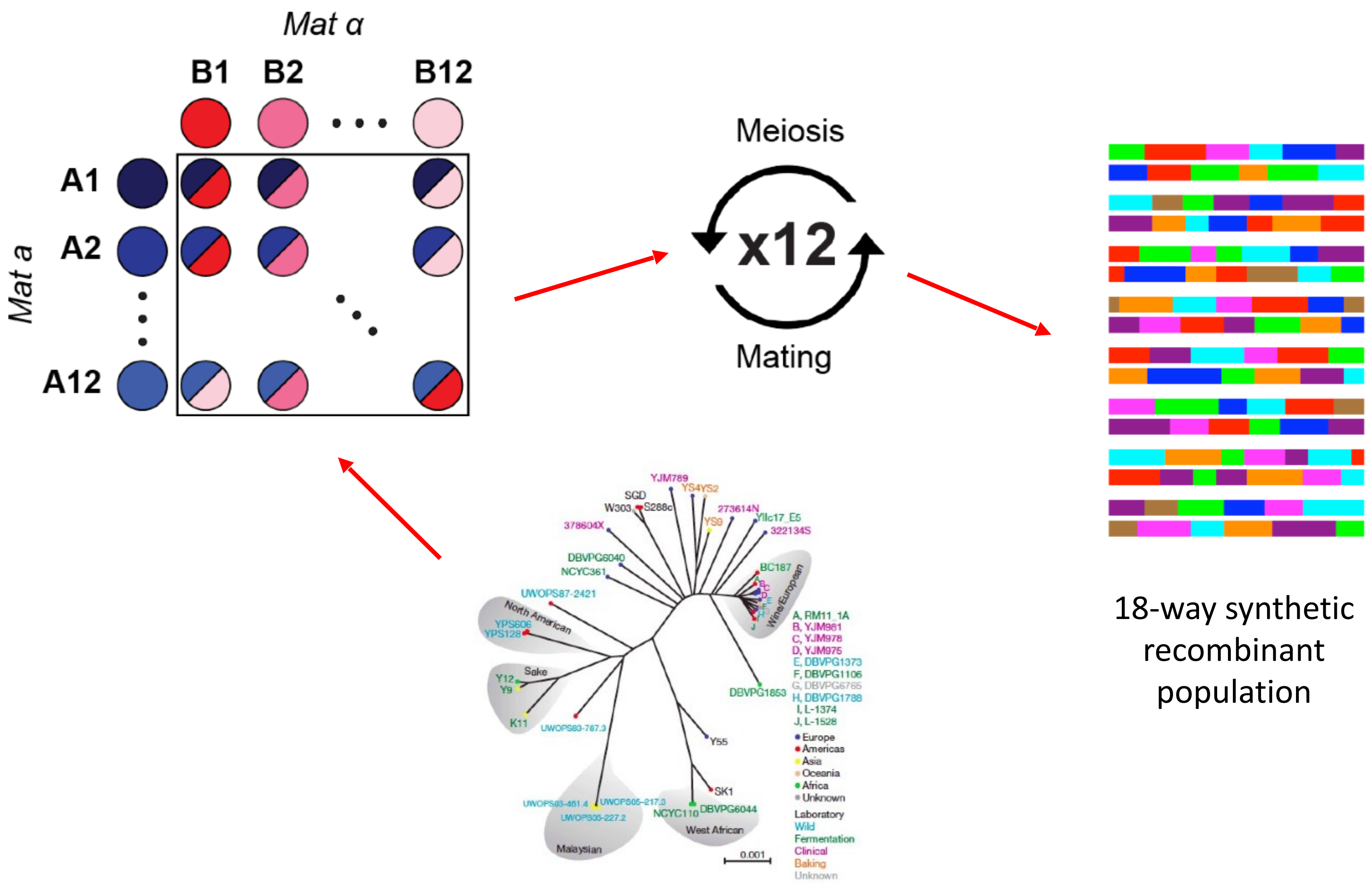


live demo

Surprise Guest Star

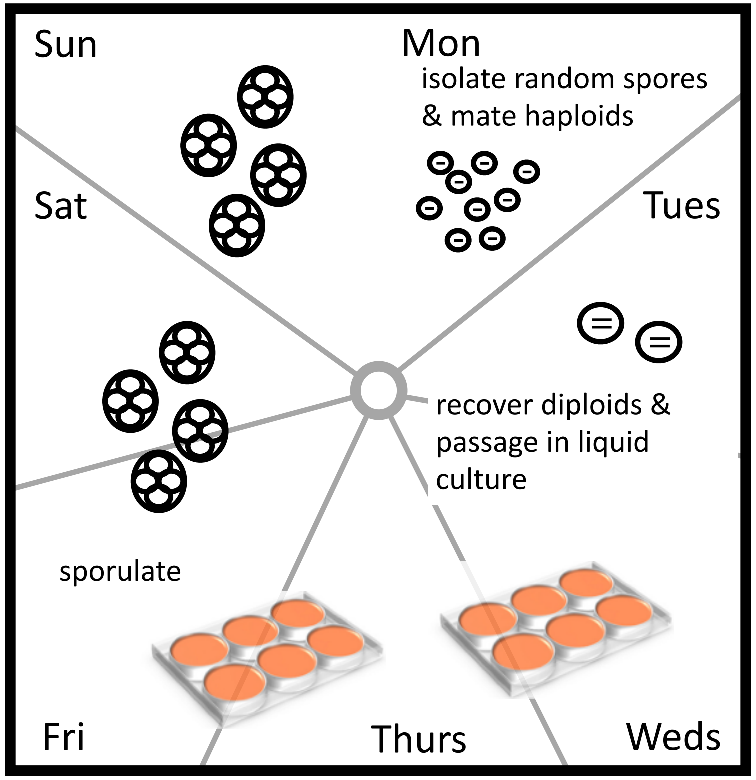
Tony Long!

Yeast E&R from a synthetic base population



18-way synthetic recombinant population

Experimental Evolution Strategy


Sun
Mon
isolate random spores & mate haploids
Sat
Tues
recover diploids & passage in liquid culture
sporulate
Fri
Thurs
Weds

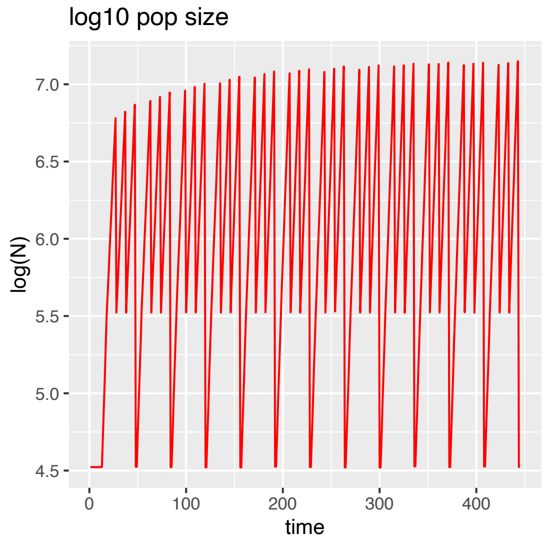
Complex E&R experiment

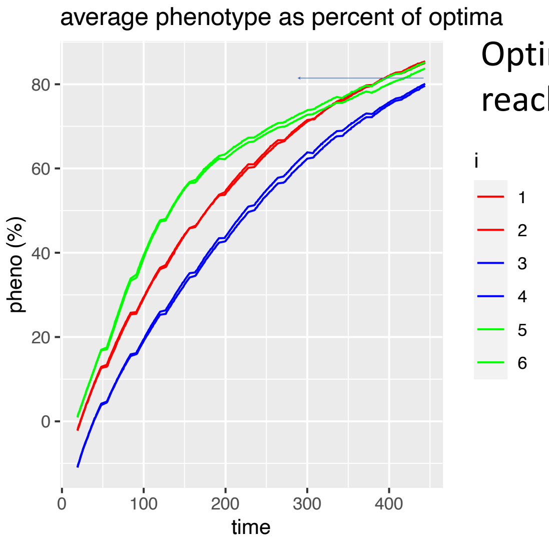
- Sex on Friday
- Recover spores and mate on Tuesday (bottleneck to 10^5 - 10^6 cells)
- Asexual growth in media with chemical challenge
- 3 transfers per week (100-fold dilution)
- 10^7-10^9 cells during selection/asexual growth

Would like quantitative predictions … but this experiment seemed painful to model

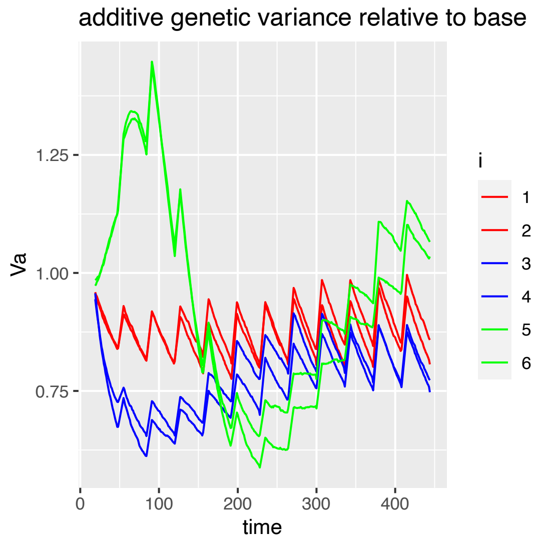
SLIM!

~30K, less than real





Optimum not reached yet



🟥 1800 SNPs

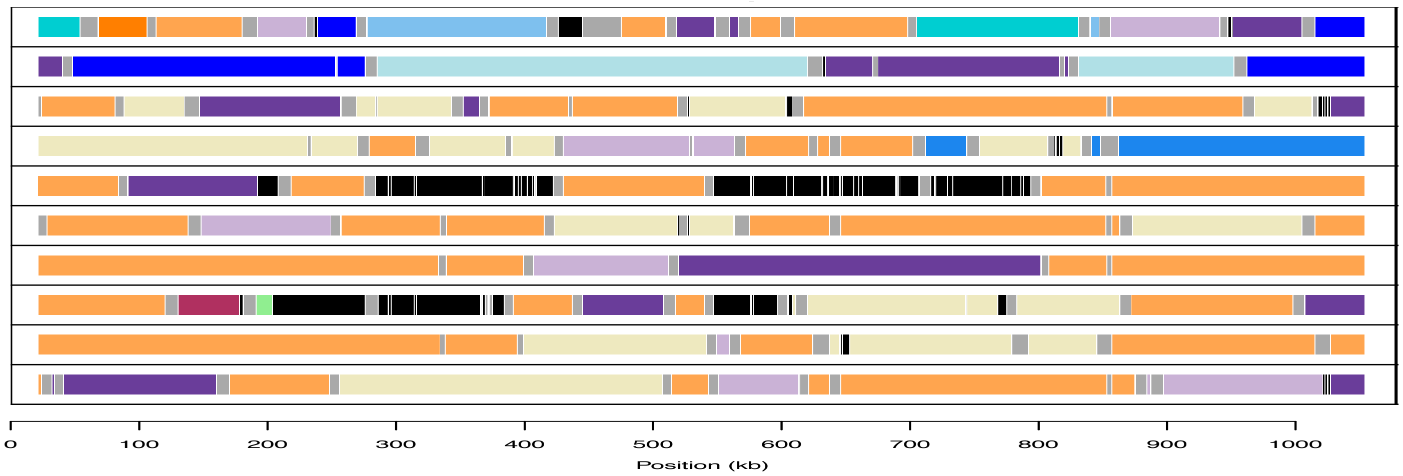
🟦 180 SNPs

🟩 18 SNPs

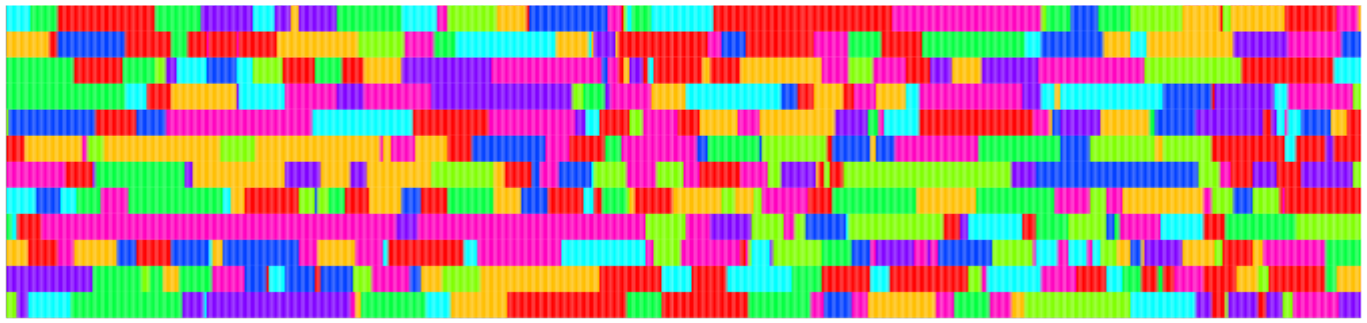
- 9 hours
- 25 Gb memory

haploid alleles for a single chromosome

real

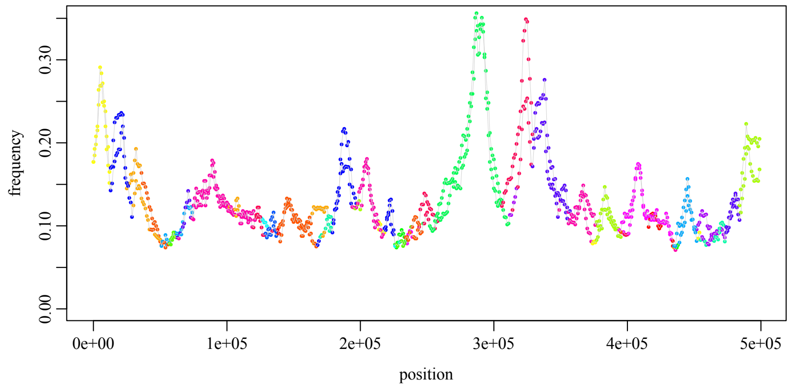


simulated

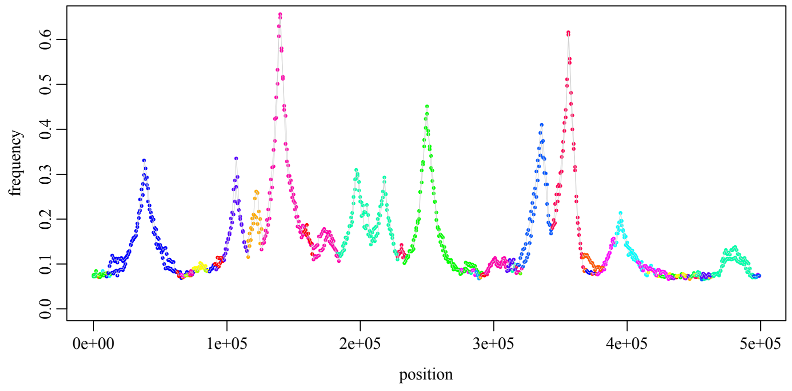


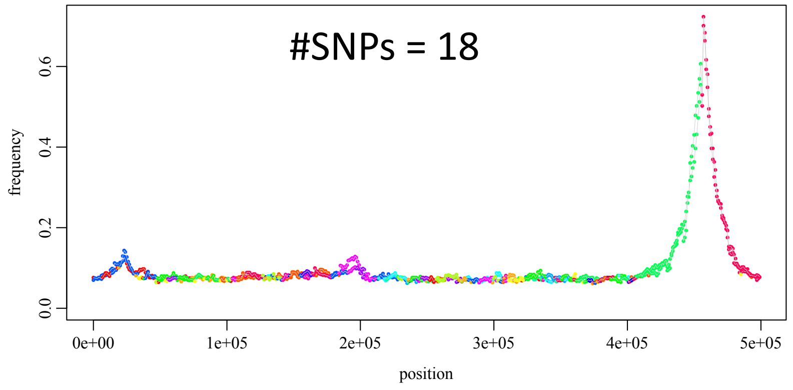
Pollock Plots

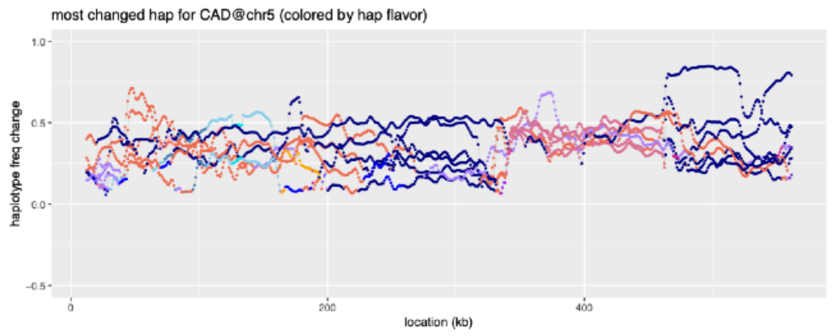
#SNPs = 1800



#SNPs = 180







Talk outline

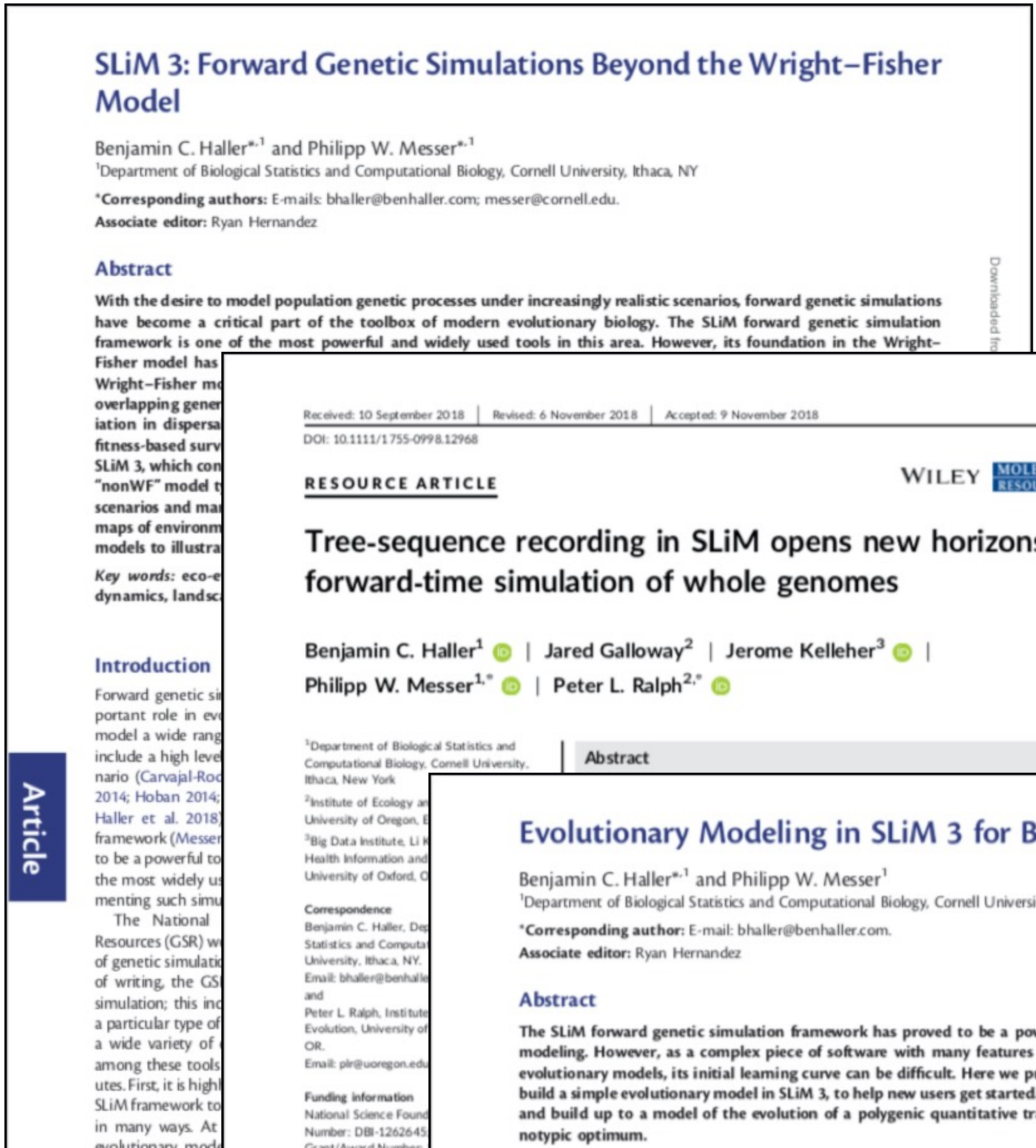
- What is SLiM? Why use SLiM?
- An introduction to Eidos and SLiM
- A survey of example models
- Population-genetic models in SLiM
- Quantitative-genetic models in SLiM
- Closing remarks

Resources

messerlab.org/slim/

bit.ly/slim-discuss

bhaller@mac.com



SLiM 3: Forward Genetic Simulations Beyond the Wright–Fisher Model

Benjamin C. Haller*[,1] and Philipp W. Messer*[,1]
[1]Department of Biological Statistics and Computational Biology, Cornell University, Ithaca, NY
*Corresponding authors: E-mails: bhaller@benhaller.com; messer@cornell.edu.
Associate editor: Ryan Hernandez

Abstract

With the desire to model population genetic processes under increasingly realistic scenarios, forward genetic simulations have become a critical part of the toolbox of modern evolutionary biology. The SLiM forward genetic simulation framework is one of the most powerful and widely used tools in this area. However, its foundation in the Wright–
Fisher model has
Wright–Fisher mo
overlapping gener
iation in dispersa
fitness-based surv
SLiM 3, which con
"nonWF" model t
scenarios and mar
maps of environm
models to illustra

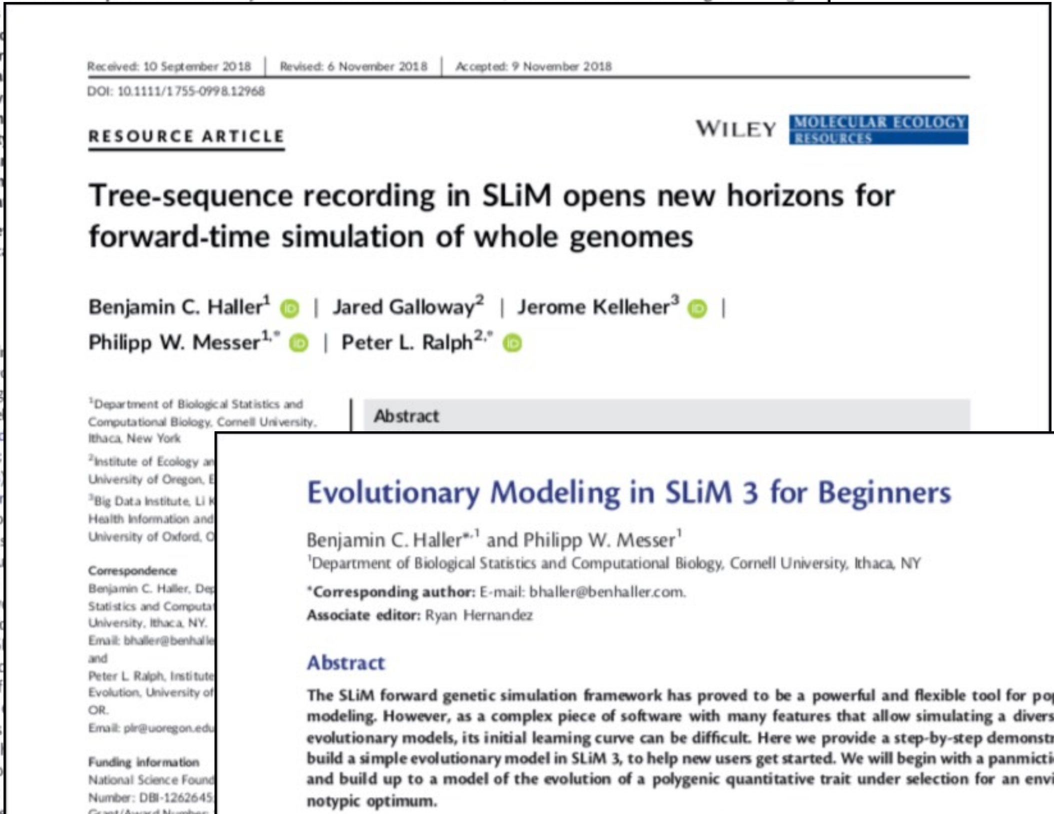
Key words: eco-e
dynamics, landsc

Introduction

Forward genetic si
portant role in ev
model a wide rang
include a high leve
nario (Carvajal-Ro
2014; Hoban 2014;
Haller et al. 2018)
framework (Messe
to be a powerful to
the most widely us
menting such simu
The National
Resources (GSR) w
of genetic simulati
of writing, the GS
simulation; this inc
a particular type of
a wide variety of
among these tools
utes. First, it is high
SLiM framework to
in many ways. At

Article

Downloaded fro



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

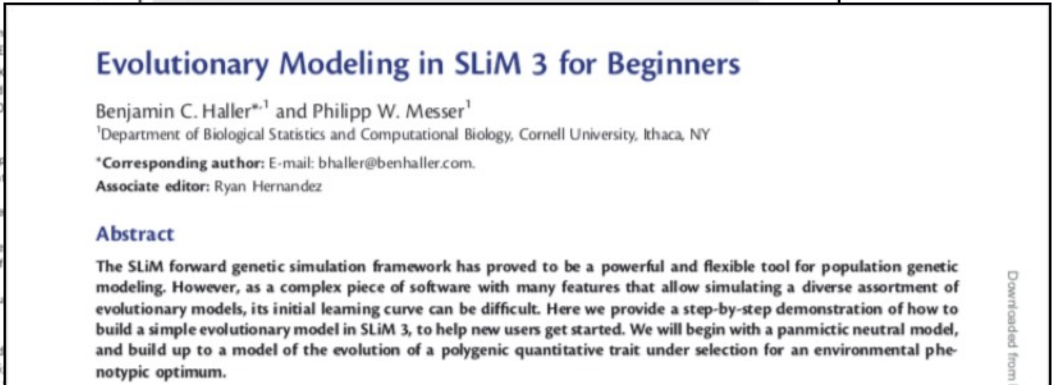
Benjamin C. Haller[1] | Jared Galloway[2] | Jerome Kelleher[3] | Philipp W. Messer[1,*] | Peter L. Ralph[2,*]

[1]Department of Biological Statistics and Computational Biology, Cornell University, Ithaca, New York
[2]Institute of Ecology an
University of Oregon, E
[3]Big Data Institute, Li K
Health Information and
University of Oxford, O

Correspondence
Benjamin C. Haller, Dep
Statistics and Computat
University, Ithaca, NY.
Email: bhaller@benhalle
and
Peter L. Ralph, Institute
Evolution, University of
OR.
Email: plr@uoregon.edu

Funding information
National Science Found
Number: DBI-1262645

Abstract



Evolutionary Modeling in SLiM 3 for Beginners

Benjamin C. Haller*[,1] and Philipp W. Messer[1]
[1]Department of Biological Statistics and Computational Biology, Cornell University, Ithaca, NY
*Corresponding author: E-mail: bhaller@benhaller.com.
Associate editor: Ryan Hernandez

Abstract

The SLiM forward genetic simulation framework has proved to be a powerful and flexible tool for population genetic modeling. However, as a complex piece of software with many features that allow simulating a diverse assortment of evolutionary models, its initial learning curve can be difficult. Here we provide a step-by-step demonstration of how to build a simple evolutionary model in SLiM 3, to help new users get started. We will begin with a panmictic neutral model, and build up to a model of the evolution of a polygenic quantitative trait under selection for an environmental phenotypic optimum.

Downloaded from

SLiM Workshops

Sweden, 2019

Iceland, 2020

UK, 2019

Now available online, free, at messerlab.org/slim/

Acknowledgements

Thanks to Simon Aeschbacher, Pamela Alamilla, Rodrigo Pracana Fragoso De Almeida, Jorge Amaya, Bill Amos, Chenling Antelope, Jaime Ashander, Hannes Becher, Emma Berdan, Jeremy Berg, Gertjan Bisschop, Tom Booker, Gideon Bradburd, Jason Bragg, Vince Buffalo, Yoann Buoro, Richard Burns, Ian Caldas, Bryce Carson, Sam Champer, Deborah Charlesworth, Jeremy Van Cleve, Jonathan Cocker, Zuxi Cui, Jean Cury, Michael DeGiorgio, A.P. Jason de Koning, Emily Dennis, Jordan Rohmeyer Dherby, Russell Dinnage, Alan Downey-Wall, Ian Dworkin, Julia Frank, Jared Galloway, Jesse Garcia, Kimberley Gilbert, Graham Gower, Alexandre Harris, Kelley Harris, Rebecca Harris, Matthew Hartfield, Ding He, Jody Hey, Marcus Hicks, Kathryn Hodgins, Christian Huber, Melissa Jane Hubisz, Emilia Huerta-Sanchez, Chaz Hyseni, Jacob Malte Jensen, Peter Keightley, Jerome Kelleher, Andy Kern, Bhavin Khatri, Bernard Kim, Isabel Kim, Jere Koskela, Athanasios Kousathanas, Peter Krawitz, Chris Kyriazis, Benjamin Laenen, Anna Maria Langmüller, Áki Láruson, Stefan Laurent, Clancy Lawler, Anita Lerch, Mitchell Lokey, Eugenio Lopez, Kathleen Lotterhos, Nicolas Lou, Andrew Marderstein, Sebastian Matuszewski, Mikhail Matz, Rupert Mazzucco, Christopher McAllester, Maéva Mollion, Miguel Navascués, Chase Nelson, Dominic Nelson, Bruno Nevado, Tram Nguyen, Etsuko Nonaka, Boyana Norris, Nathan Oakes, Nick O'Brien, Omar Eduardo Cornejo Ordaz, Matt Osmond, Greg Owens, Josephine Paris, Harvinder Pawar, Martin Petr, Denis Pierron, Fernando Racimo, Peter Ralph, Erik Regla, David Rinker, Murillo Fernando Rodrigues, Andrew Sackman, Aaron Sams, Kieran Samuk, Sara Schaal, Derek Setter, Elissa Sorojsrisom, Onuralp Söylemez, Fabian Staubach, Stefan Strütt, Michelle Su, Anastasia Teterina, Rob Unckless, Magnus Dehli Vigeland, Nathan Villiger, Christos Vlachos, Silu Wang, Clemens L. Weiß, James Whiting, Alexander Whitwam, Aaron Wolf, Yan Wong, Yannick Wurm, Alexander Xue, Justin Yeh, and Yulin Zhang. Special thanks also to the Messer Lab, the stackoverflow community, Kevin Thornton, Ryan Hernandez, and Dmitri Petrov.

Questions!

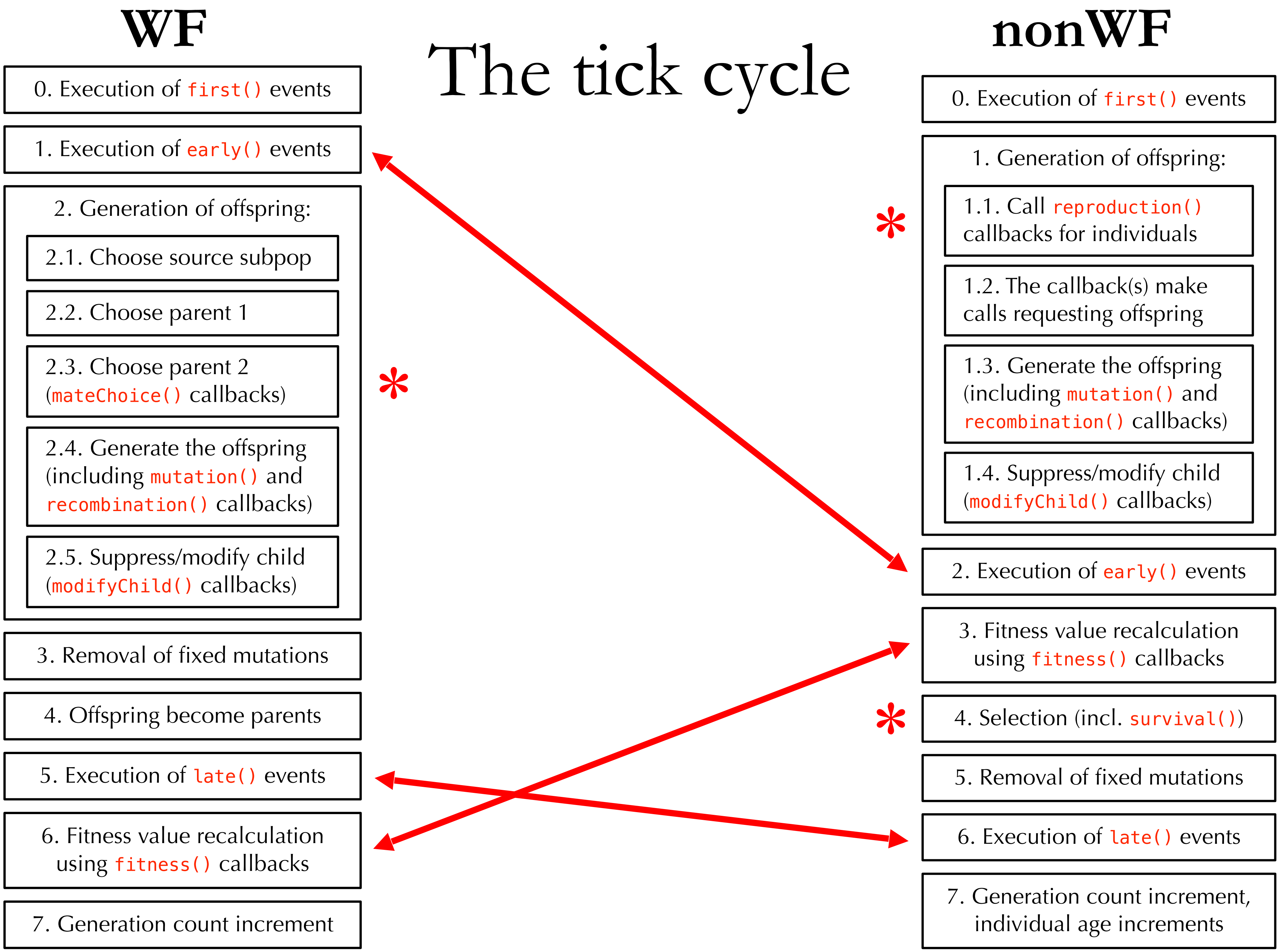
(and SLiM stickers!)

Extra slides

How to use SLiM?

- Do **initial modeling** in SLiMgui
  - interactive model development & visual debugging
  - syntax coloring, online docs, code completion
- Do **production runs** on the cluster
  - trivial to build and run on a computing cluster
  - do many replicate runs simultaneously, one per core
- Do **post-run analysis** in Eidos, Python, or R
  - a lot of analysis can be done in-script in Eidos
  - `.trees` output can be read in Python with `pyslim`

The tick cycle



WF

0. Execution of first() events
1. Execution of early() events
2. Generation of offspring:
   - 2.1. Choose source subpop
   - 2.2. Choose parent 1
   - 2.3. Choose parent 2 (mateChoice() callbacks)
   - 2.4. Generate the offspring (including mutation() and recombination() callbacks)
   - 2.5. Suppress/modify child (modifyChild() callbacks)
3. Removal of fixed mutations
4. Offspring become parents
5. Execution of late() events
6. Fitness value recalculation using fitness() callbacks
7. Generation count increment

*

nonWF

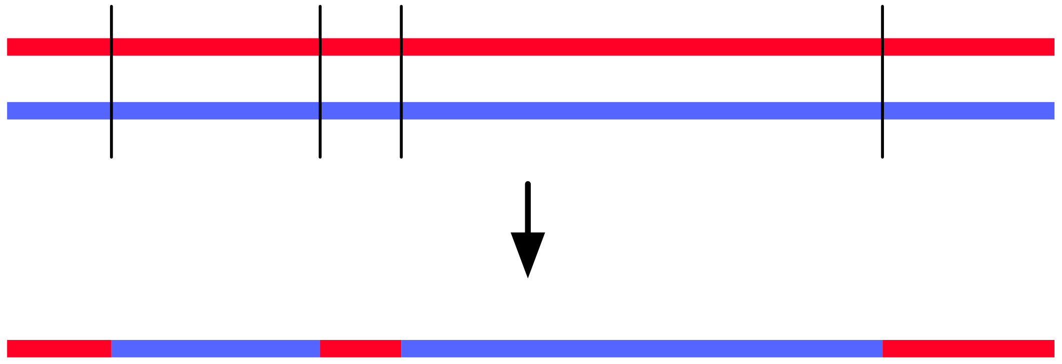
0. Execution of first() events
1. Generation of offspring:
   - 1.1. Call reproduction() callbacks for individuals
   - 1.2. The callback(s) make calls requesting offspring
   - 1.3. Generate the offspring (including mutation() and recombination() callbacks)
   - 1.4. Suppress/modify child (modifyChild() callbacks)
2. Execution of early() events
3. Fitness value recalculation using fitness() callbacks
4. Selection (incl. survival())
5. Removal of fixed mutations
6. Execution of late() events
7. Generation count increment, individual age increments

*

*

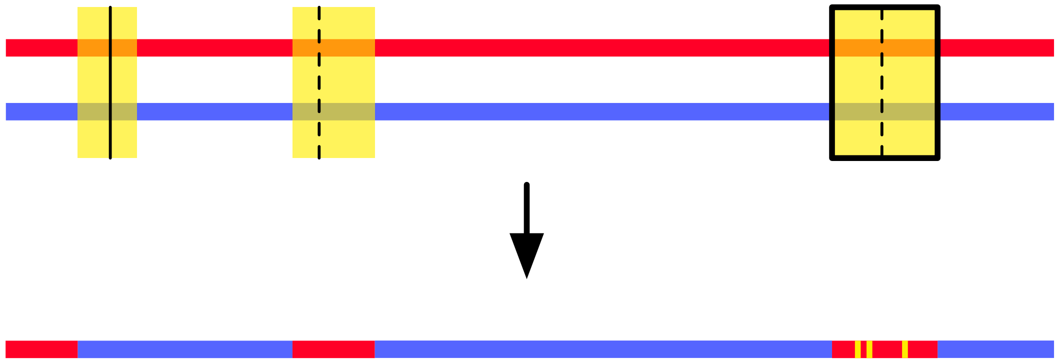
The Crossover Breakpoints Model

- During gamete generation:
  - breakpoints are drawn by probability
  - a simple crossover model is the default



The DSB Model

- Each breakpoint initiates *gene conversion*
  - DSBs can be crossovers or non-crossovers
  - gene conversion tracts can be simple or complex
  - heteroduplex mismatch repair, GC biased repair

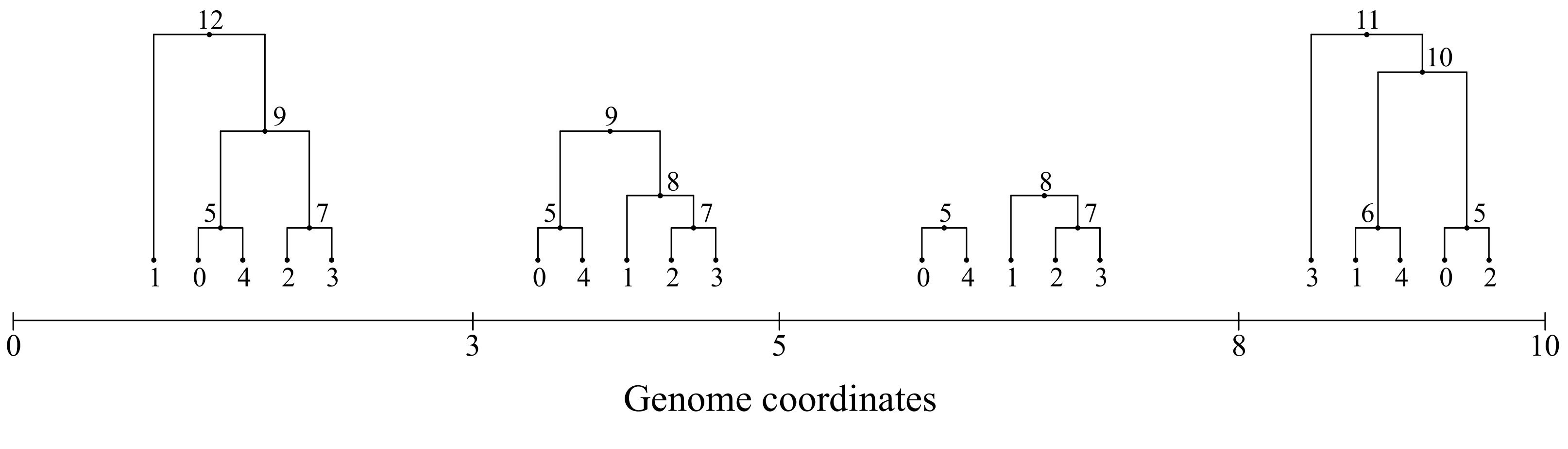


Nucleotide-based Models

- Optional facilities
  - Trinucleotide-based mutation rates
  - Reading and writing FASTA, VCF files
  - Getting the nucleotide sequence
  - Getting the codon sequence
  - Getting the amino acid sequence
  - Hotspot maps (variable mutation rate)
  - GC-biased gene conversion (gBGC)

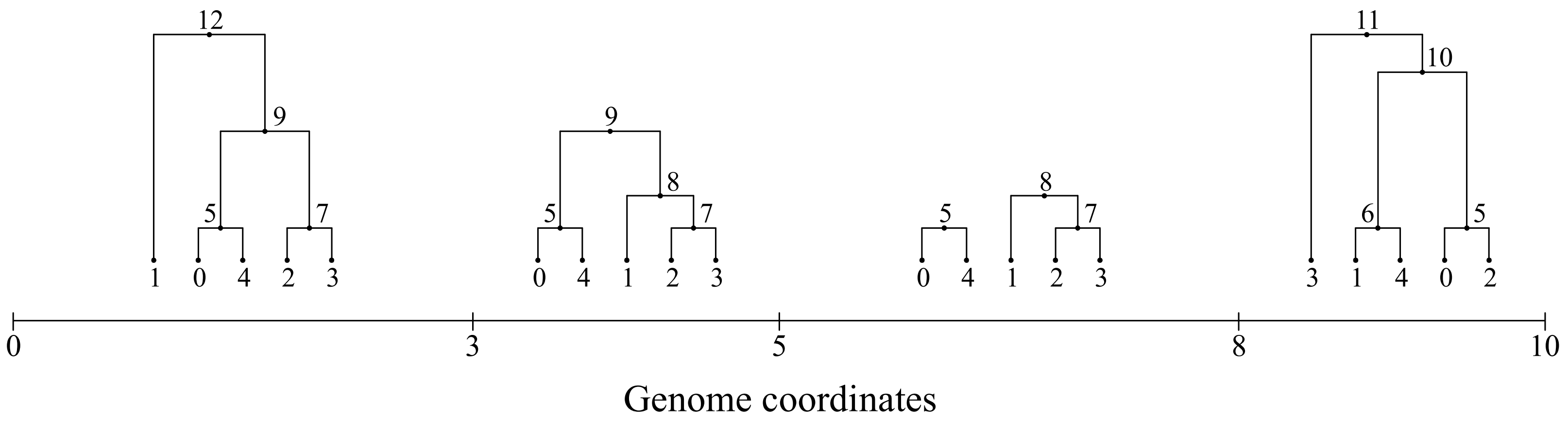
Tree sequences

- 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



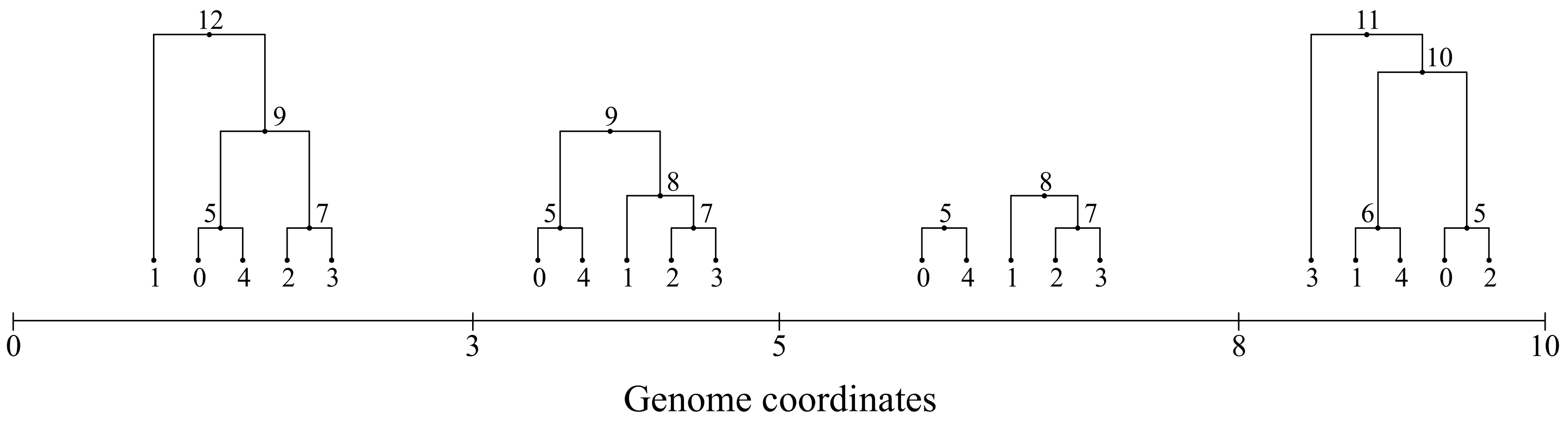
Tree sequences

- 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
- 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

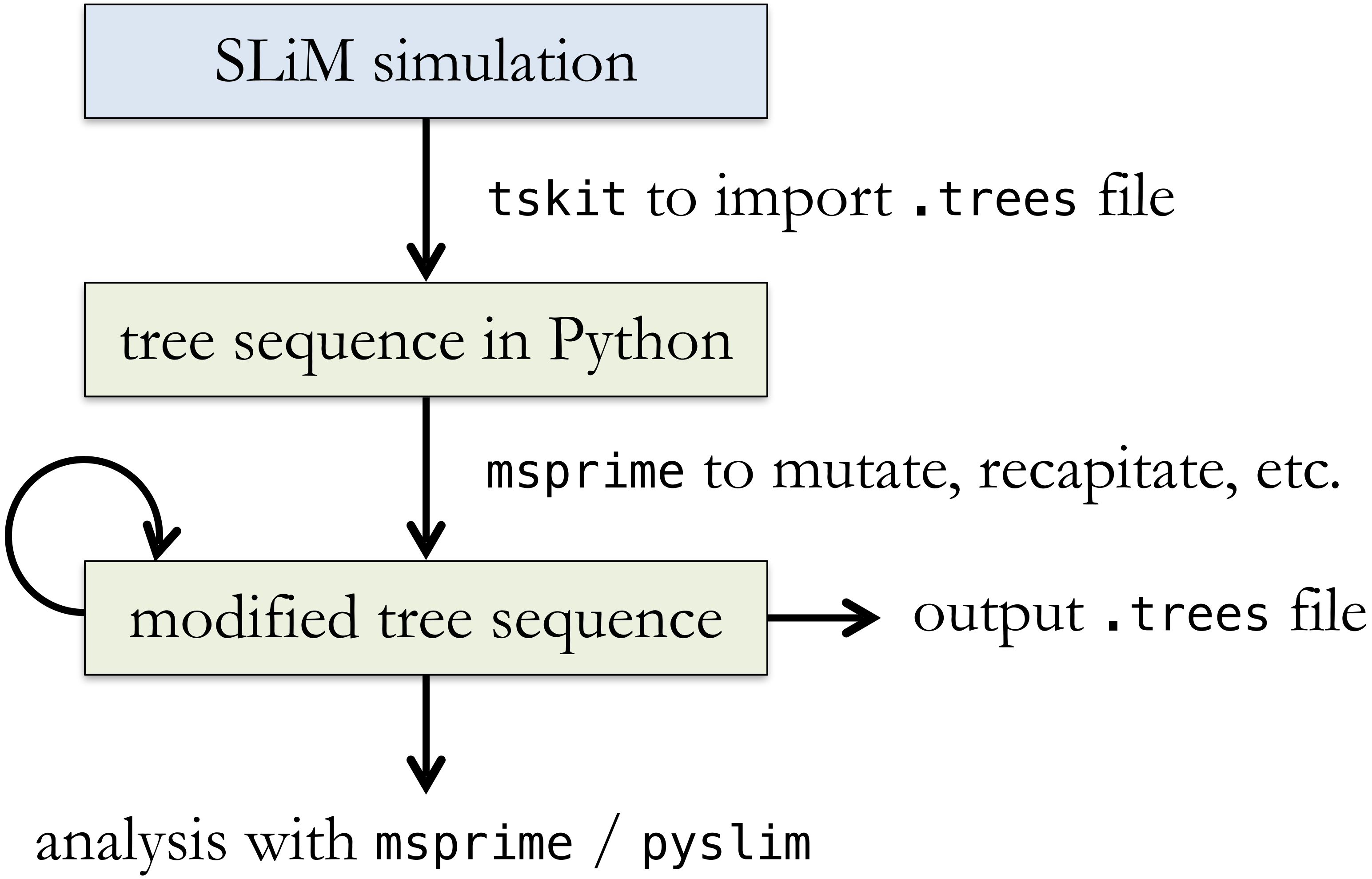
Tree-sequence analysis in Python

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

Tree-sequence analysis in Python

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

Tree-sequence analysis in Python


SLiM simulation
tskit to import .trees file
tree sequence in Python
msprime to mutate, recapitate, etc.
modified tree sequence
output .trees file
analysis with msprime / pyslim

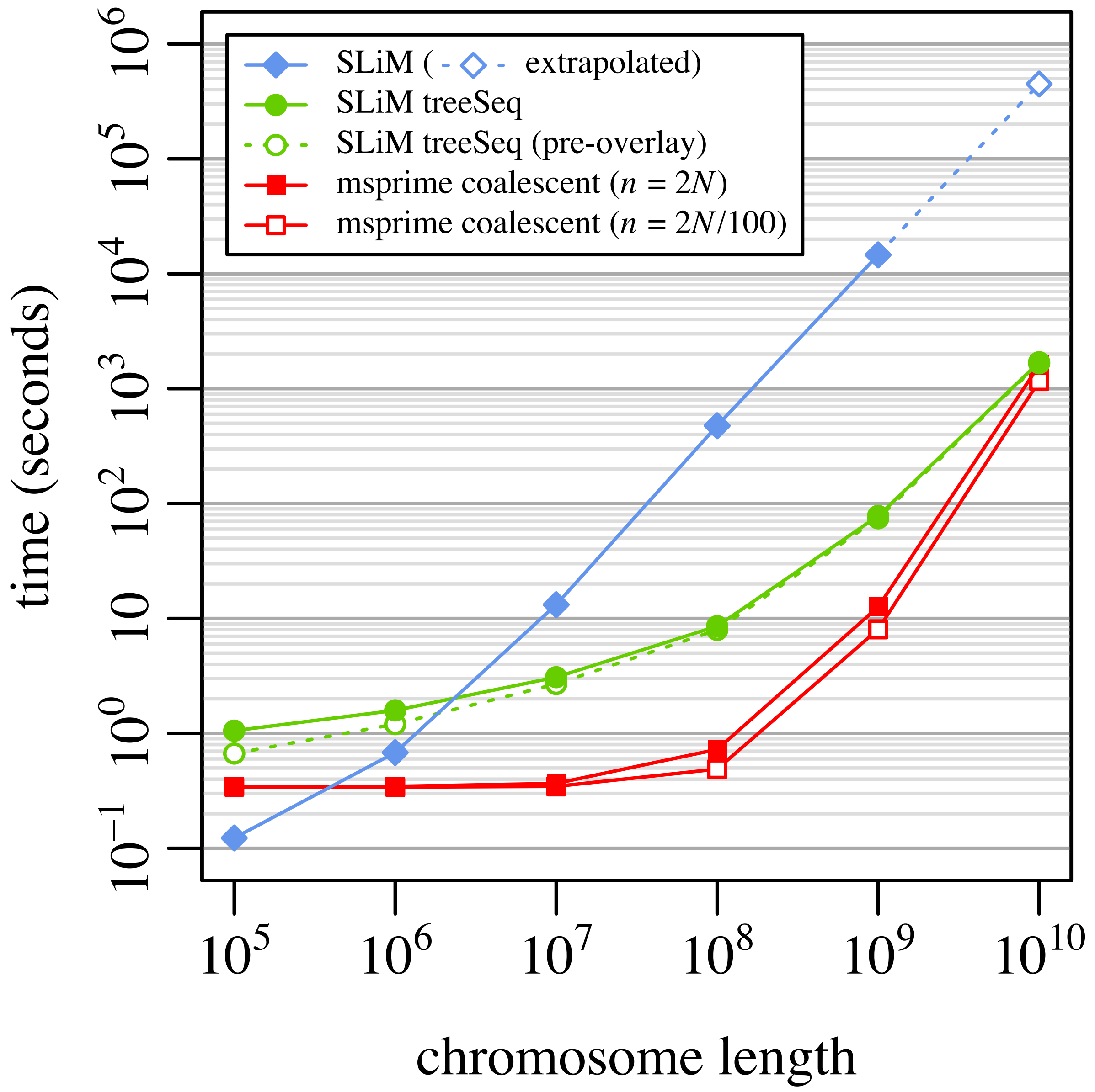
A complete Python analysis script

```python
import msprime, tskit

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

- Import `msprime` and `tskit` packages
- Load the saved `.trees` file with `tskit`
- 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

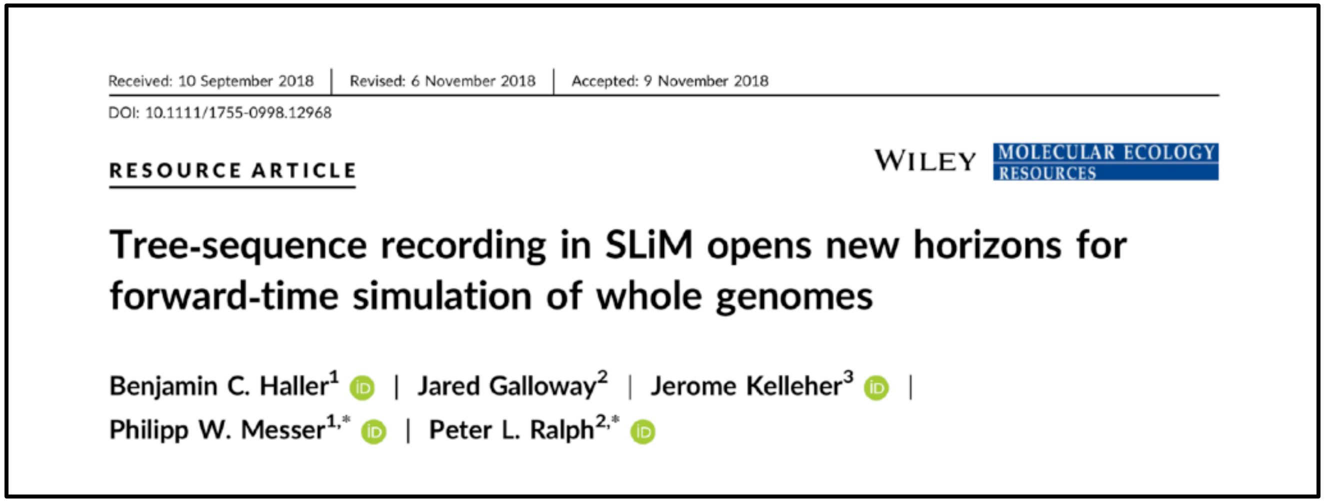


Haller et al. (2019), Molecular Ecology Resources

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

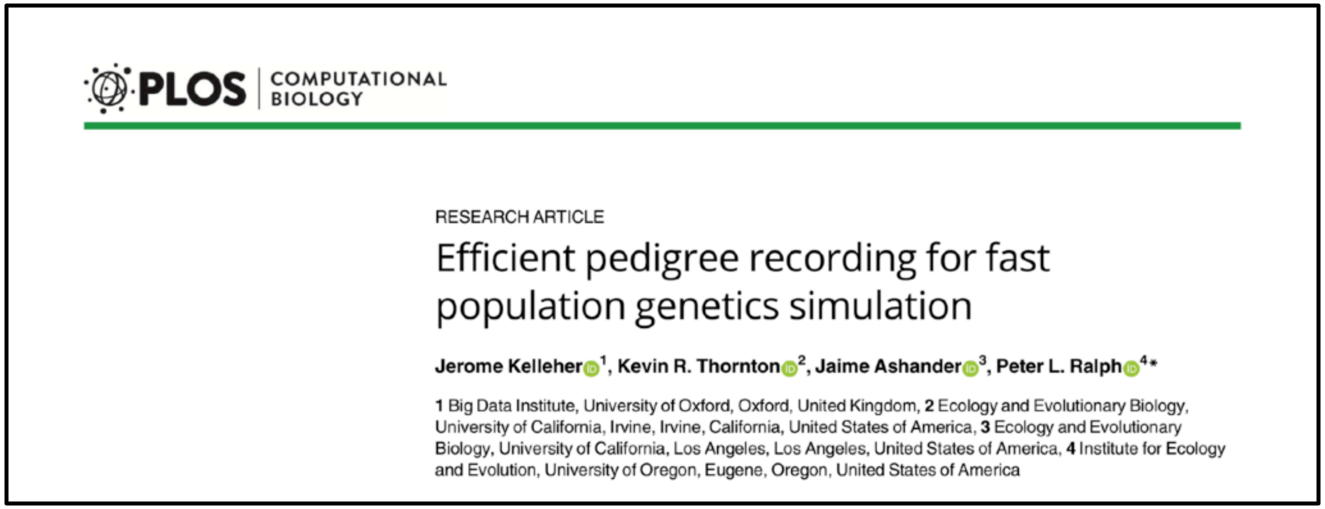
DOI: 10.1111/1755-0998.12968

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*]

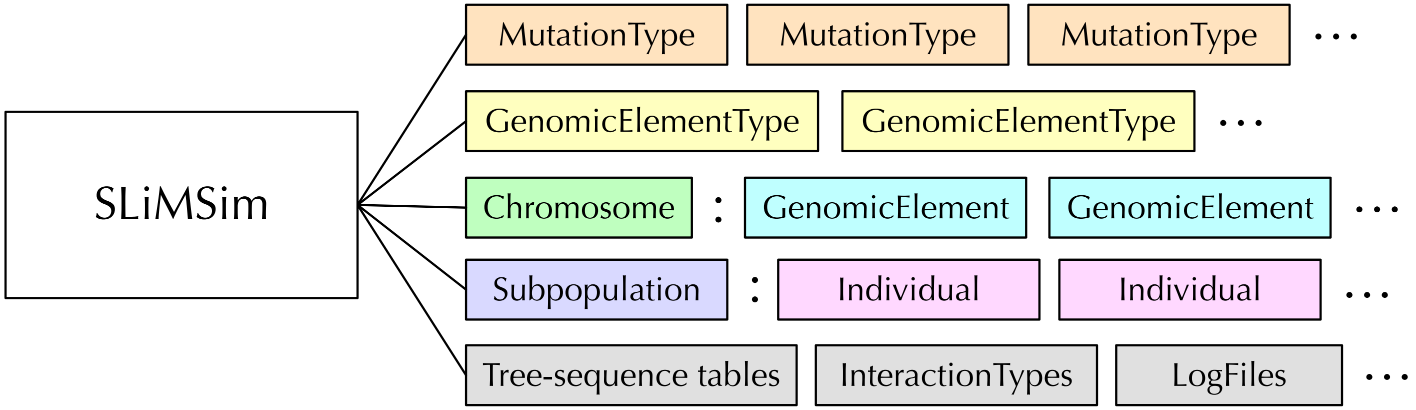
1 Big Data Institute, University of Oxford, Oxford, United Kingdom, **2** Ecology and Evolutionary Biology, University of California, Irvine, Irvine, California, United States of America, **3** Ecology and Evolutionary Biology, University of California, Los Angeles, Los Angeles, United States of America,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

Multispecies modeling

- SLiM 4 adds support for multiple species
- Ecological interactions:
    - predation, competition, parasitism, mutualism, within-host evolution, …
    - individual-based spatial interactions between species (local prey search, local host search, local resource competition)
- Eco-evolutionary dynamics
- Coevolutionary dynamics

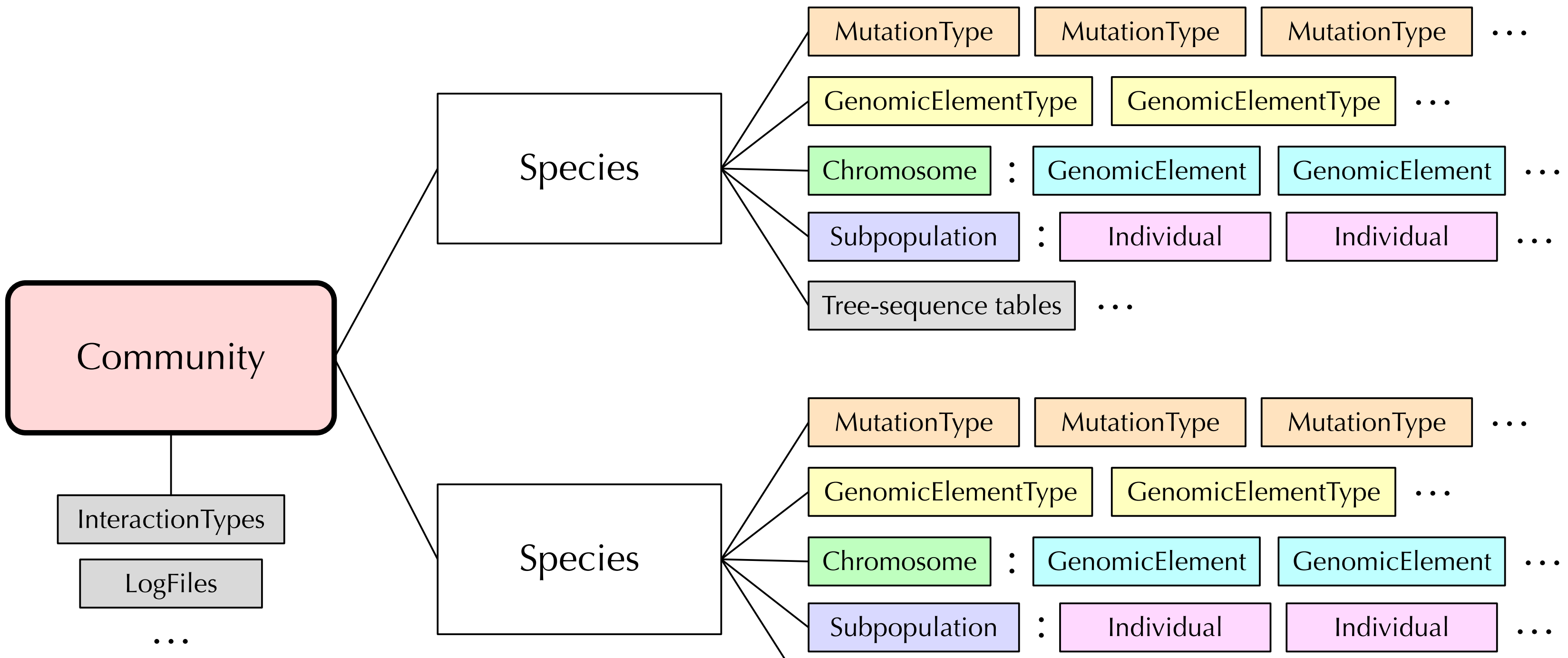
Multispecies modeling

- Now, SLiMSim represents one *simulation*:



Multispecies modeling

- In SLiM 4, SLiMSim represents one *species*:



Multispecies modeling

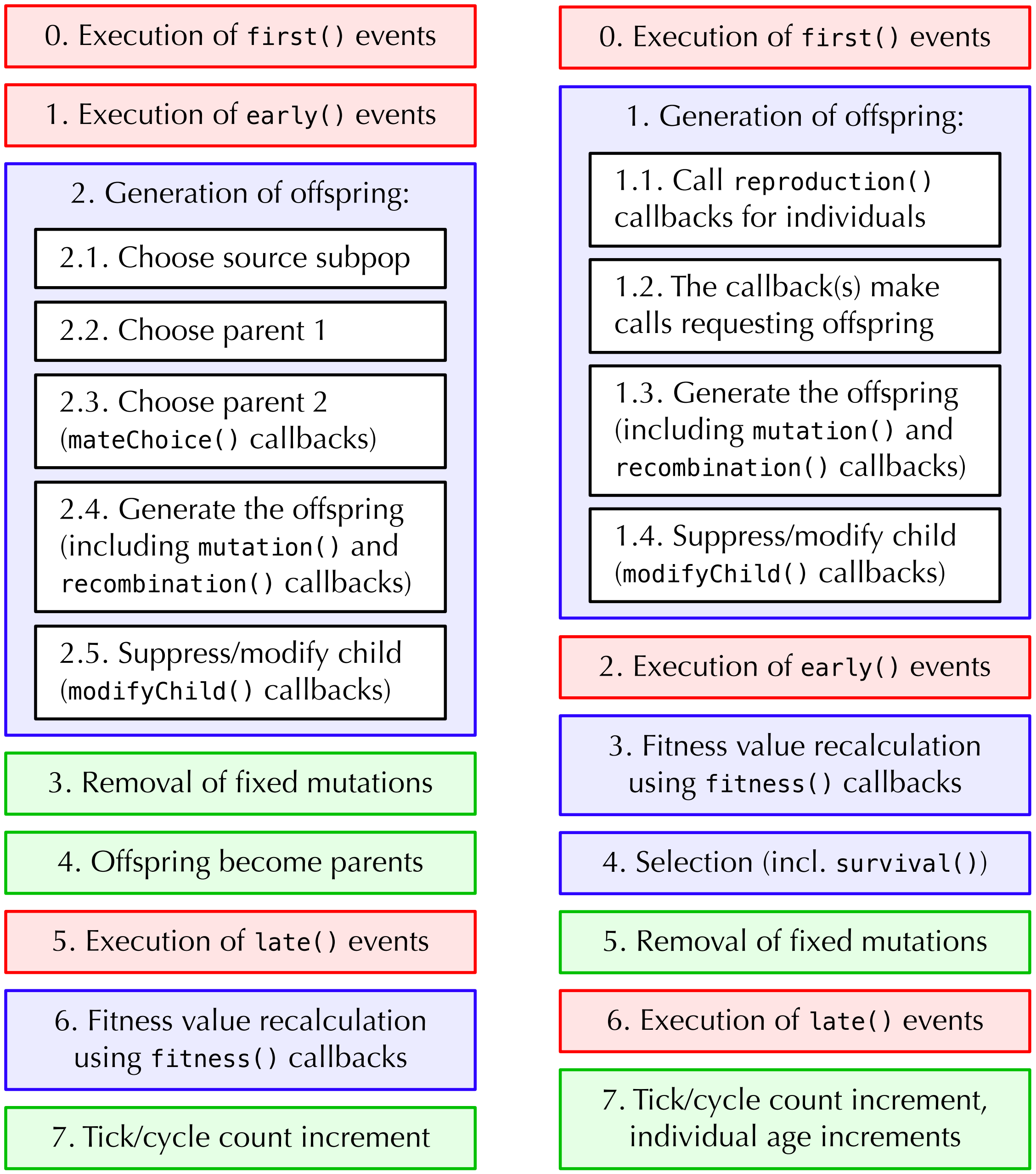
- SLiMSim has become Species
- Community has been added on top
- Each species is independent:
  - separate genetics & behavior
  - separate scripting and callbacks
  - separate tree-sequence recording
- Separate timescales; but their execution in each tick is interleaved!

Multispecies modeling

- Time is now represented in *ticks* and *cycles*:

tick	**species fox** modulo 3 phase 5	**species mouse** modulo 1 phase 1
1		cycle 1
2		cycle 2
3		cycle 3
4		cycle 4
5	cycle 1	cycle 5
6		cycle 6
7		cycle 7
8	cycle 2	cycle 8
9		cycle 9

Intereaved tick cycles


The sequence of events within one cycle in WF models.
0. Execution of `first()` events
1. Execution of `early()` events
2. Generation of offspring:
2.1. Choose source subpop
2.2. Choose parent 1
2.3. Choose parent 2 (`mateChoice()` callbacks)
2.4. Generate the offspring (including `mutation()` and `recombination()` callbacks)
2.5. Suppress/modify child (`modifyChild()` callbacks)
3. Removal of fixed mutations
4. Offspring become parents
5. Execution of `late()` events
6. Fitness value recalculation using `fitness()` callbacks
7. Tick/cycle count increment
The sequence of events within one cycle in nonWF models.
0. Execution of `first()` events
1. Generation of offspring:
1.1. Call `reproduction()` callbacks for individuals
1.2. The callback(s) make calls requesting offspring
1.3. Generate the offspring (including `mutation()` and `recombination()` callbacks)
1.4. Suppress/modify child (`modifyChild()` callbacks)
2. Execution of `early()` events
3. Fitness value recalculation using `fitness()` callbacks
4. Selection (incl. `survival()`)
5. Removal of fixed mutations
6. Execution of `late()` events
7. Tick/cycle count increment, individual age increments