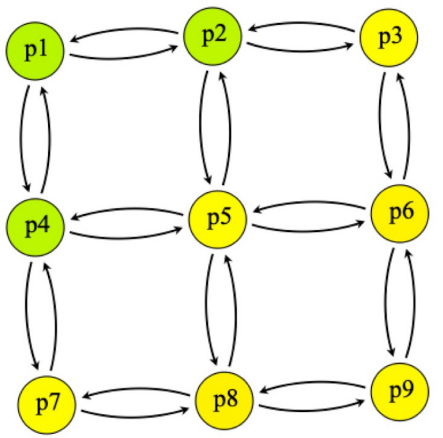
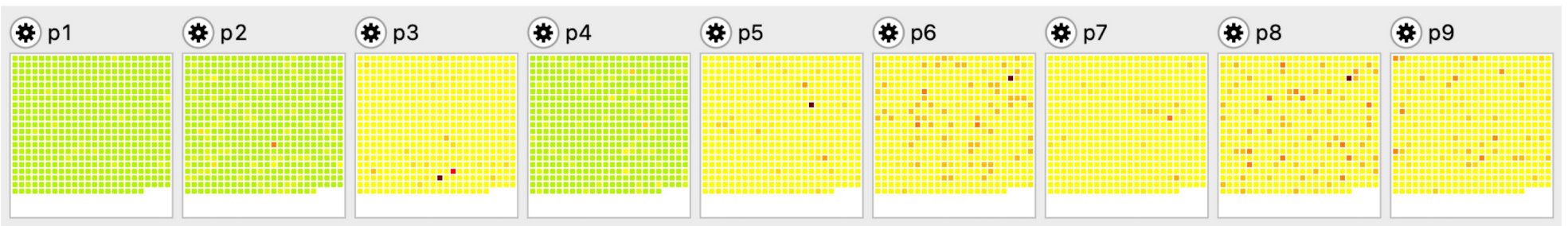
Modelling locally adaptive inversions





Alex Pinch, Jan. 30 2023

Stopping recombination

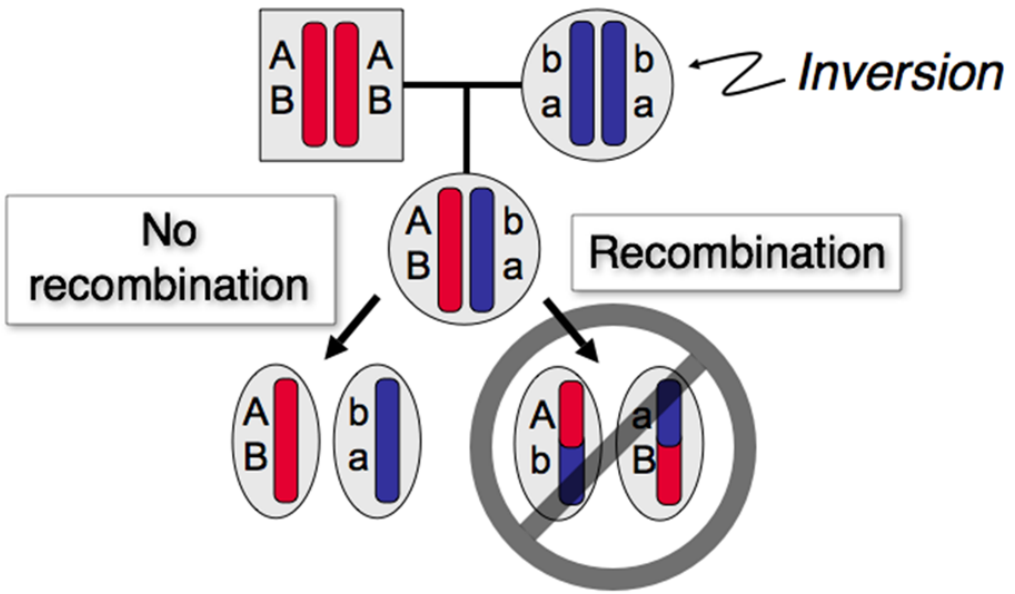


Fig. 2, Kirkpatrick 2010

- When offspring are heterozygous for an inversion, recombination does not occur as a broken segment cannot find their homologous pair

Why are inversions important?

- By stopping recombination, harmful mutations aren’t purged, resulting in their increase over time (Kirkpatrick 2010)
- They can persist through populations and homozygotes can accumulate more deleterious mutations over time (Huang, 2022)
- If genetic information is almost identical (only linear order changed), how do they spread?

Local adaptation theory

- An inversion's frequency can change across a geographical cline (Krimbas and Powell, 1992)
- Suggests that they could be locally adaptive and contain genes that are environmentally favourable
- If an inversion traps genes that become ecologically favoured, this could explain why they persist

What we’re testing

- Across different environments, individuals that possess a locally advantageous inversion will have an increase in mutational load
- Homozygous individuals will have a greater accumulation of deleterious mutations than heterozygotes

How inversions spread

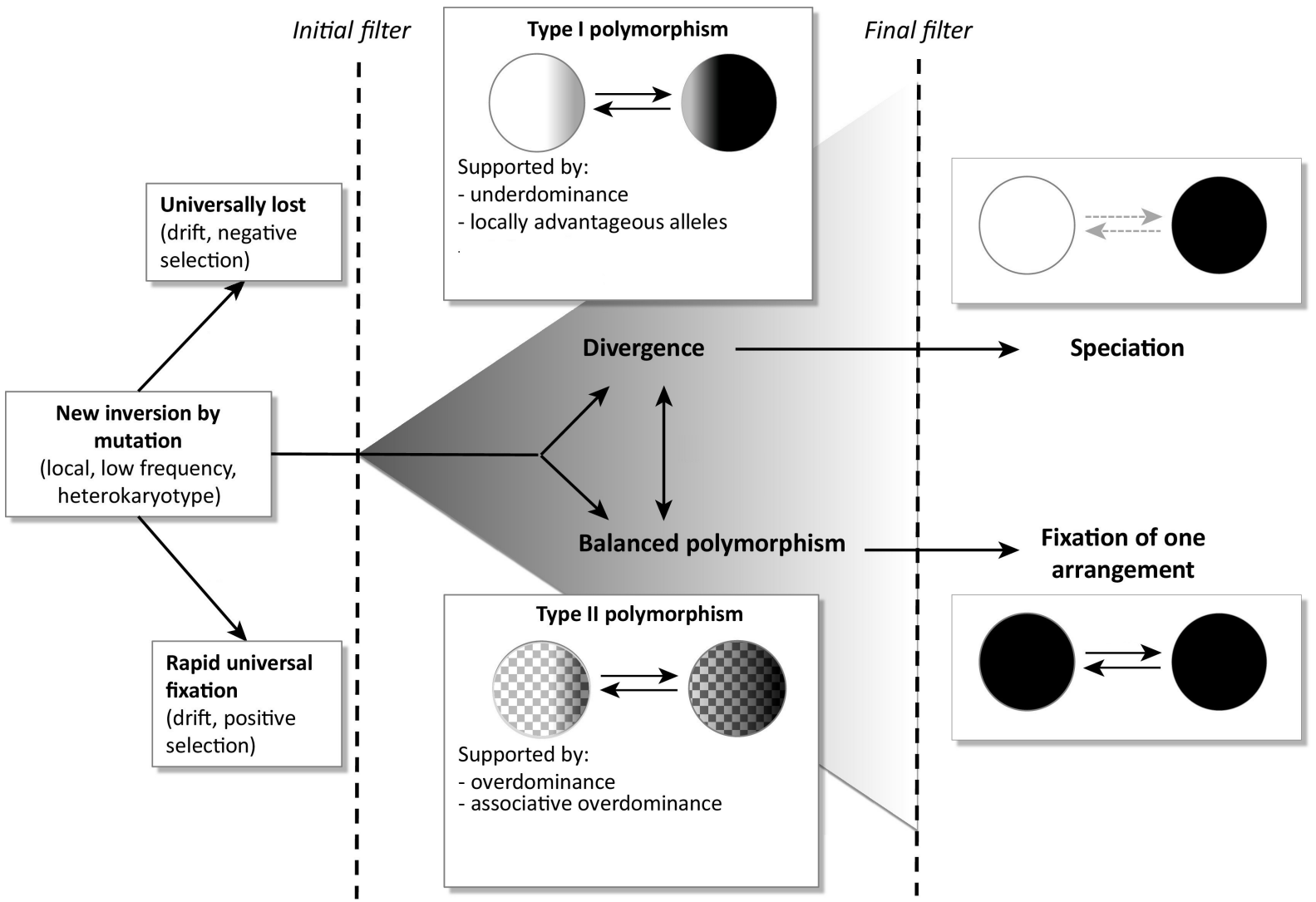


Fig. 1, Faria et al. 2019 (Modified)

How inversions spread

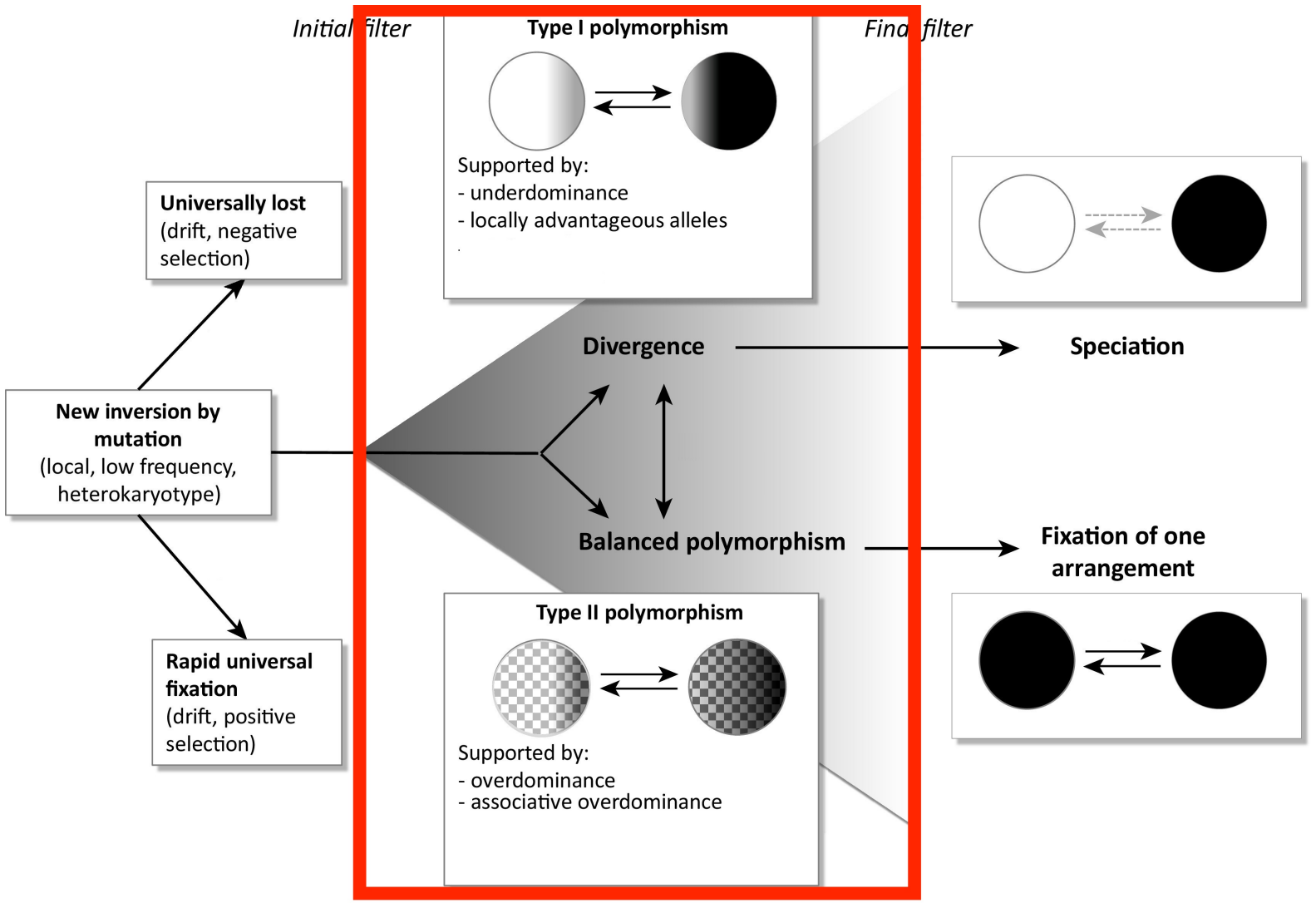


Fig. 1, Faria et al. 2019 (Modified)

How we’re testing it

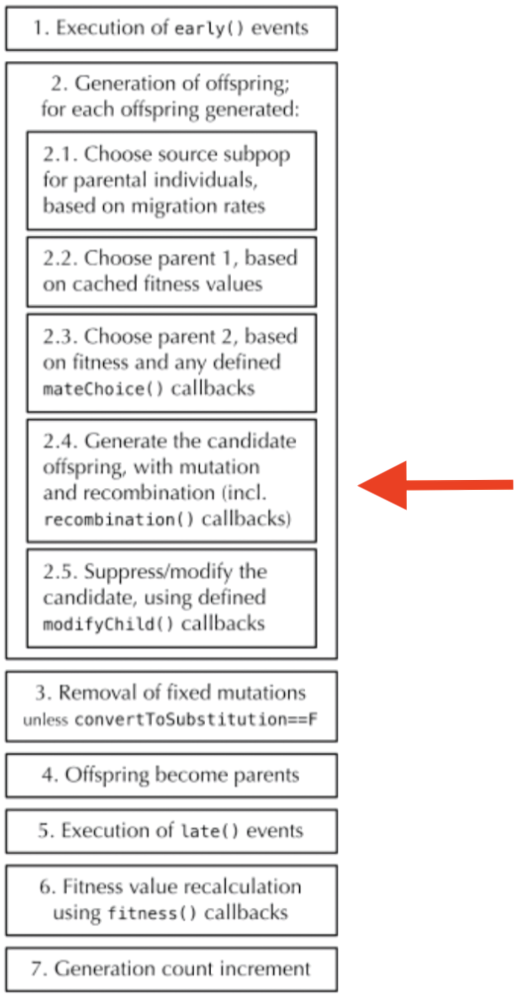
- By making a model that introduces a locally adaptive inversion across several populations we can keep track of fitness, number of mutations per genotype
- Can run the model many times to find trends

A SLiMtroduction

- We built a forward-genetics simulation using SLiM
- SLiM (**S**election on **Li**nked **M**utations) was developed by Ben Haller and Phillip Messer at Cornell
- SLiM provides a basic framework for modelling individuals over generations
- Computes at the gene level, not concerned with base pairs
- Uses a proprietary programming language similar to R, but as fast as C

A SLiMtroduction

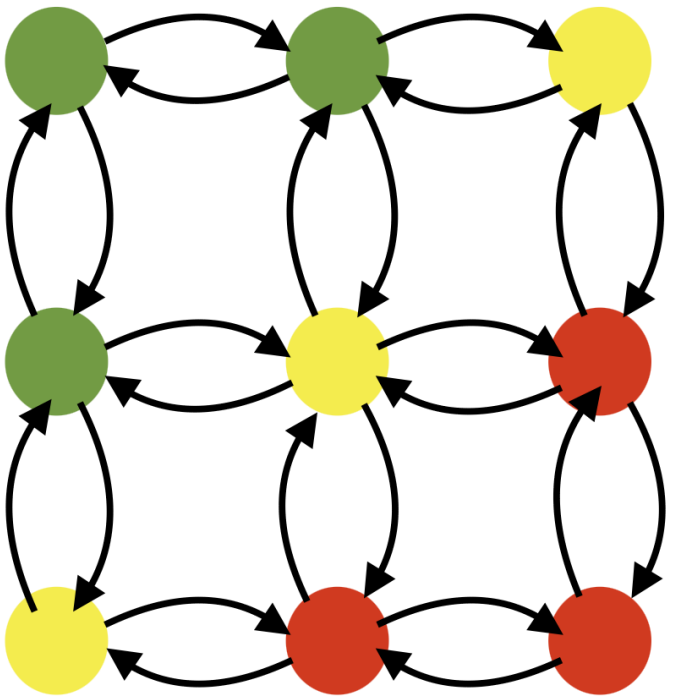
The sequence of events within one generation in a SLiM simulation.



Haller, 2016

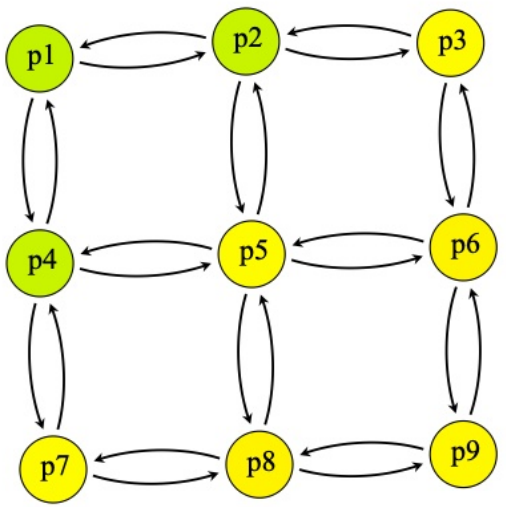
Our model

- At time = 0
  - Define a grid of subpopulations and establish a fitness gradient for inversion-carrying individuals



Our model

- After 50,000 generations:
  - Define a start and end point variable for the inversion
  - Give one individual two copies of a mutation at that start position, will serve as a marker



Our model

- Every generation:
  - Check every individual for that marker mutation
  - If found: suppress recombination and manually alter fitness
  - Individuals have a chance to migrate to another subpopulation, where their inversion will either help or hurt them

Our model

- What this model is not doing:
  - It is not actually inverting a section of a simulated chromosome
  - Instead, we define the length and place a marker to keep track of it, changing the breakpoints and fitness manually
  - It is not keeping track of base pairs, codons, amino acids, proteins…

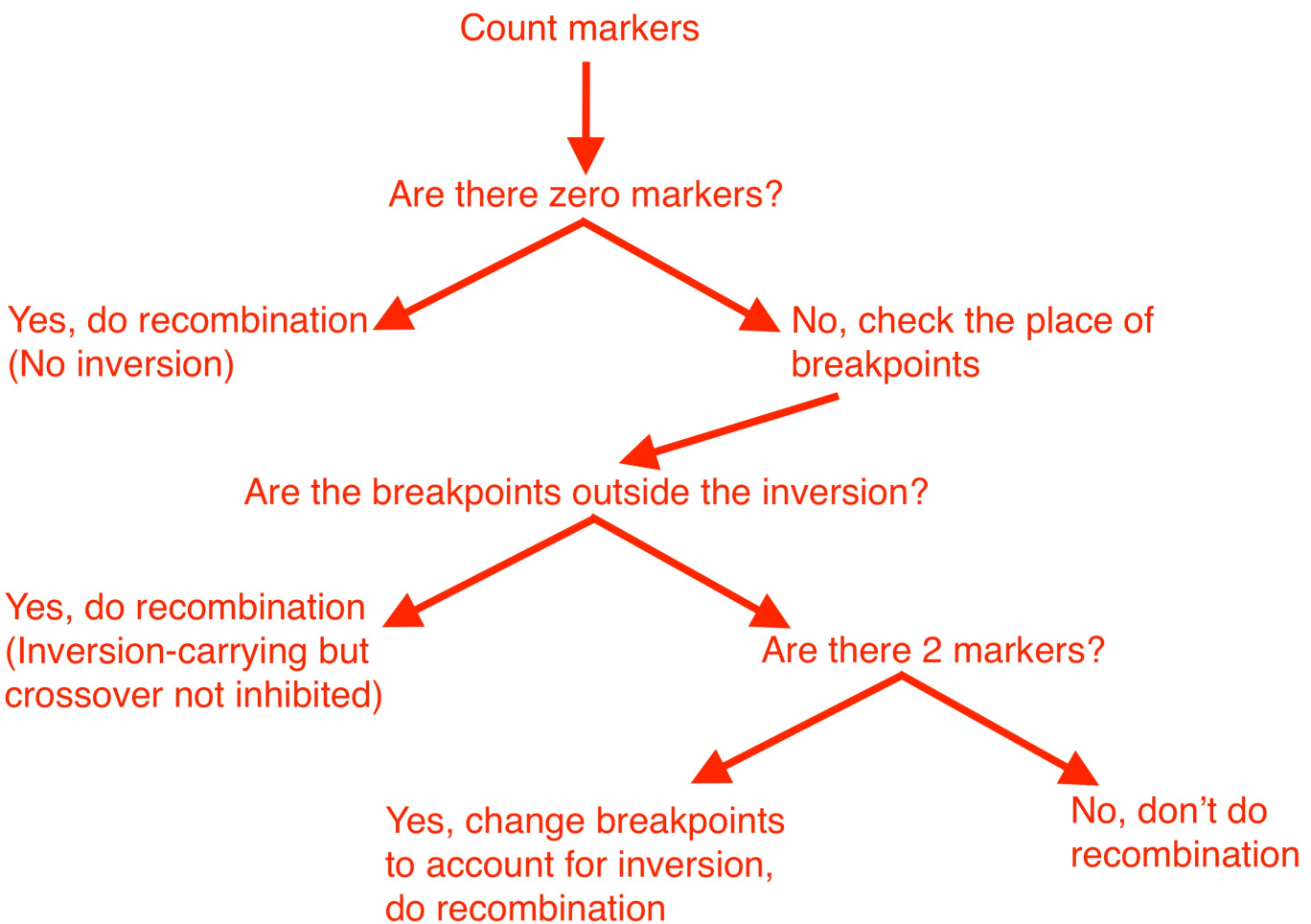
Altering recombination

- Preventing recombination is at the base of how inversions are physically maintained on the chromosome over time, this is the crux of the model
- In heterozygotes, recombination must be prevented
- In homozygotes, the breakpoints must be changed and told where the inversion is

Altering recombination

- SLiM does recombination automatically, so you must define a function named ‘recombination’ to supersede it
- If this function returns false, SLiM’s normal recombinaton function takes over
- If this function returns true, SLiM considers recombination complete and continues the loop

Altering recombination



Altering recombination

```
1  recombination() {
2    # Search both chromosomes of an individual for the inversion
3      gm1 = genome1.containsMarkerMutation(m2, invStart);
4      gm2 = genome2.containsMarkerMutation(m2, invStart);
5      # If neither chromosomes have the inversion, return false
6      if (!(gm1 | gm2)) {
7          return(F)
8      }
9      # Check if the breakpoints are outside the inversion
10     inInv = (breakpoints > invStart) & (breakpoints <= invEnd);
11     # If their sum is even, return false
12     if (sum(inInv) %% 2 == 0) {
13         return(F)
14     }
15     # If both chromosomes have the inversion (is homozygous)...
16     if (gm1 & gm2) {
17         left = (breakpoints == invStart);
18         right = (breakpoints == invEnd + 1);
```

Previous work in modelling inversions

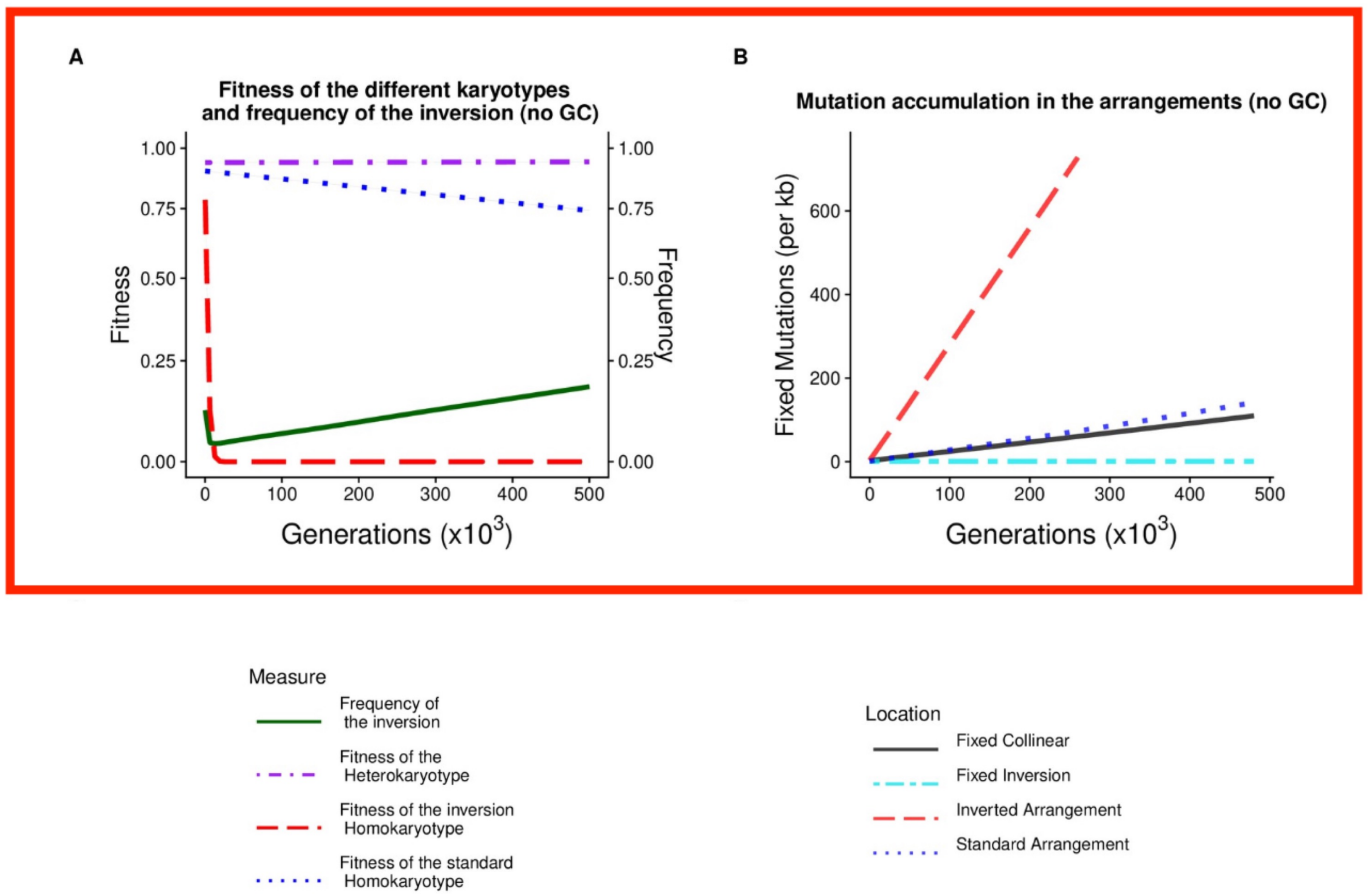
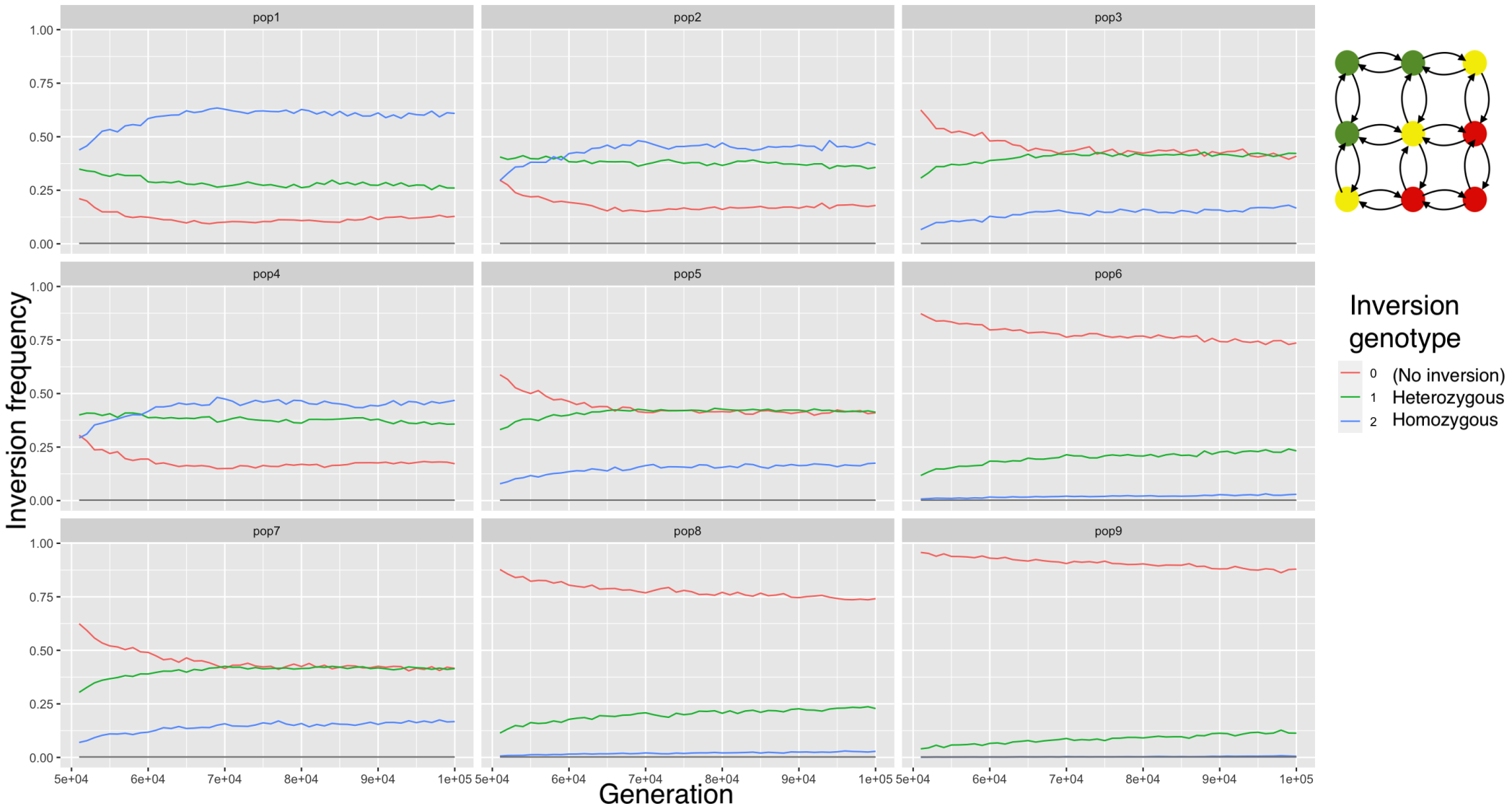
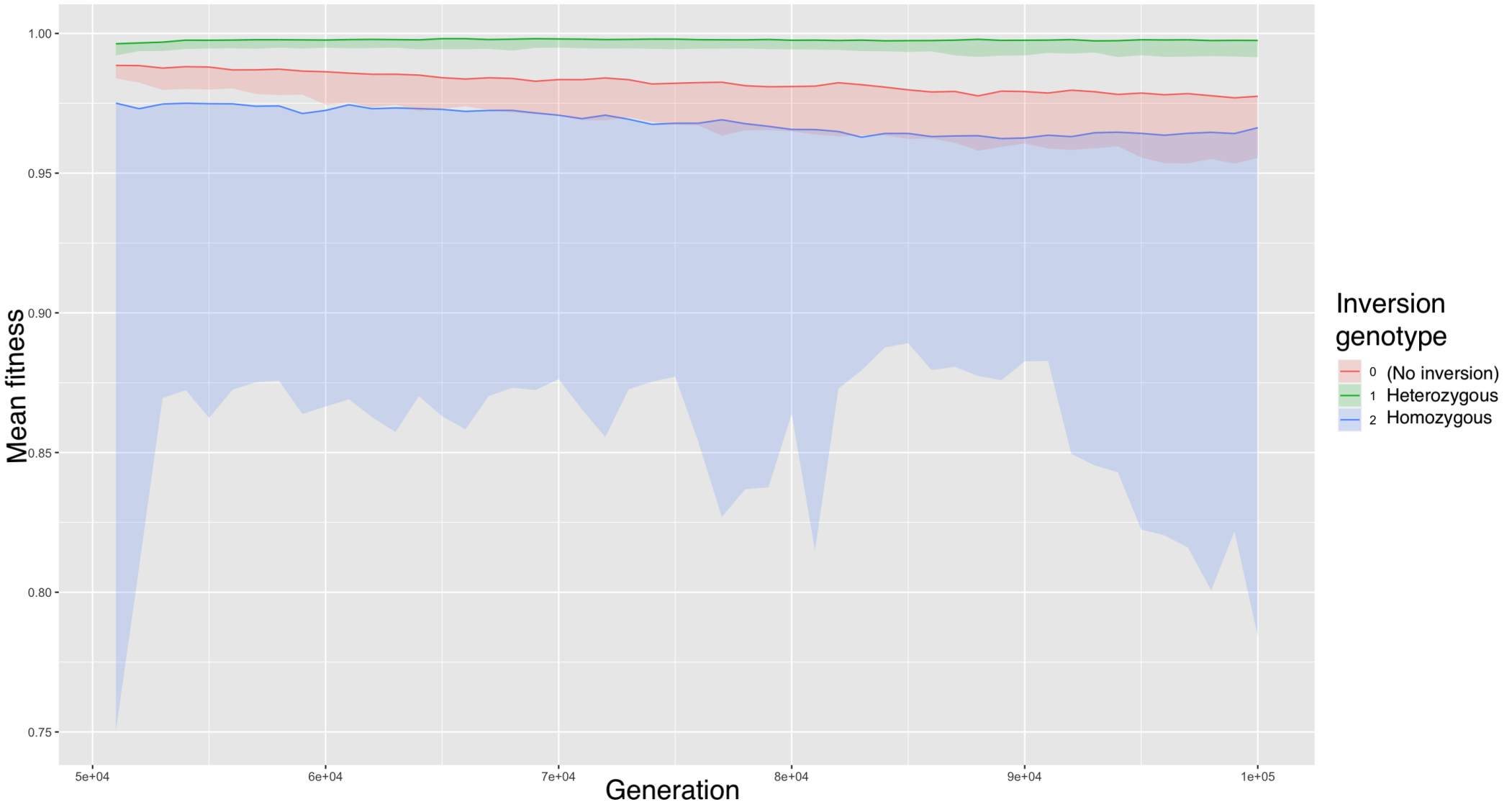


Fig. 2A-B, Berdan et al. 2021 (Adapted)

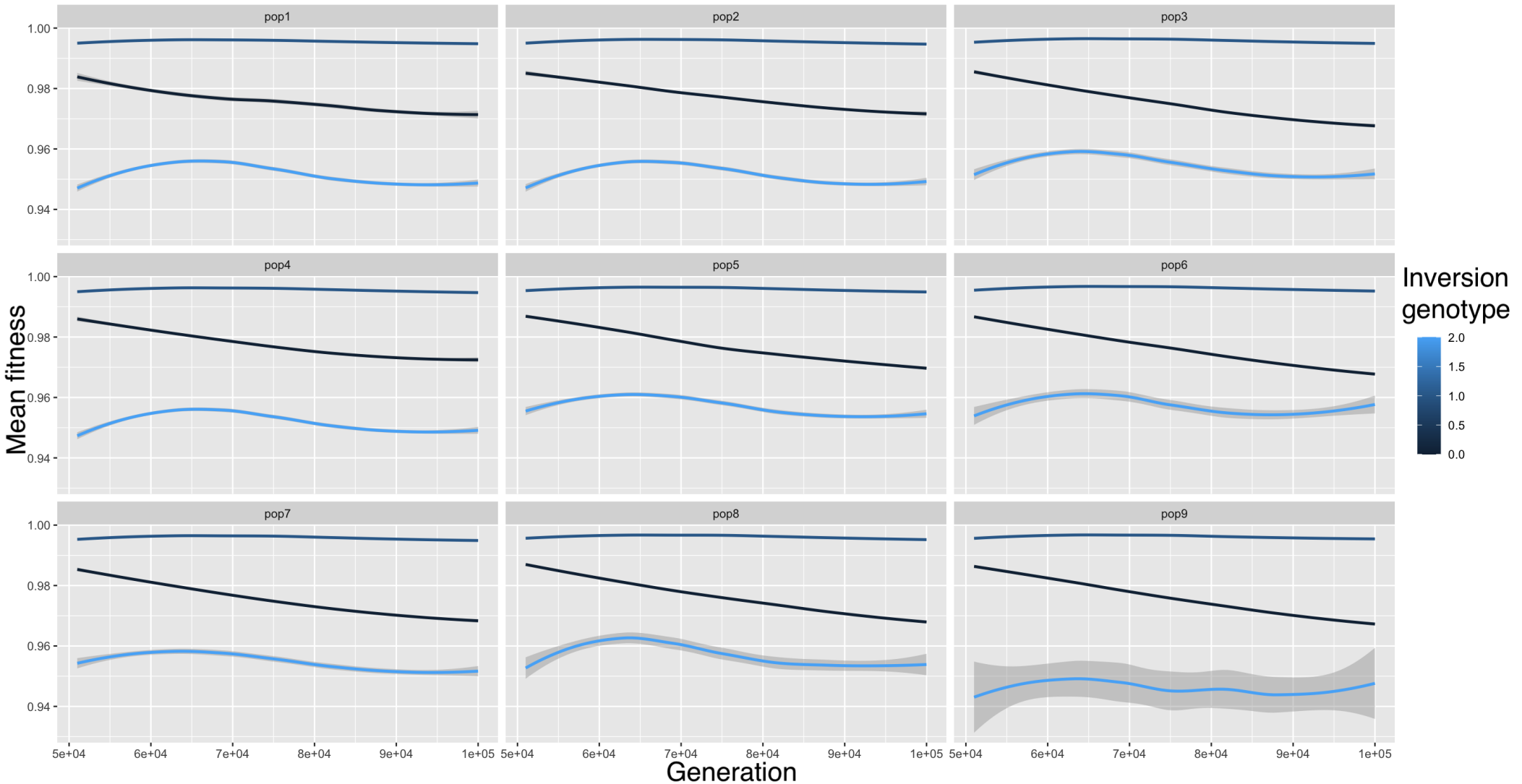
Preliminary results



Preliminary results

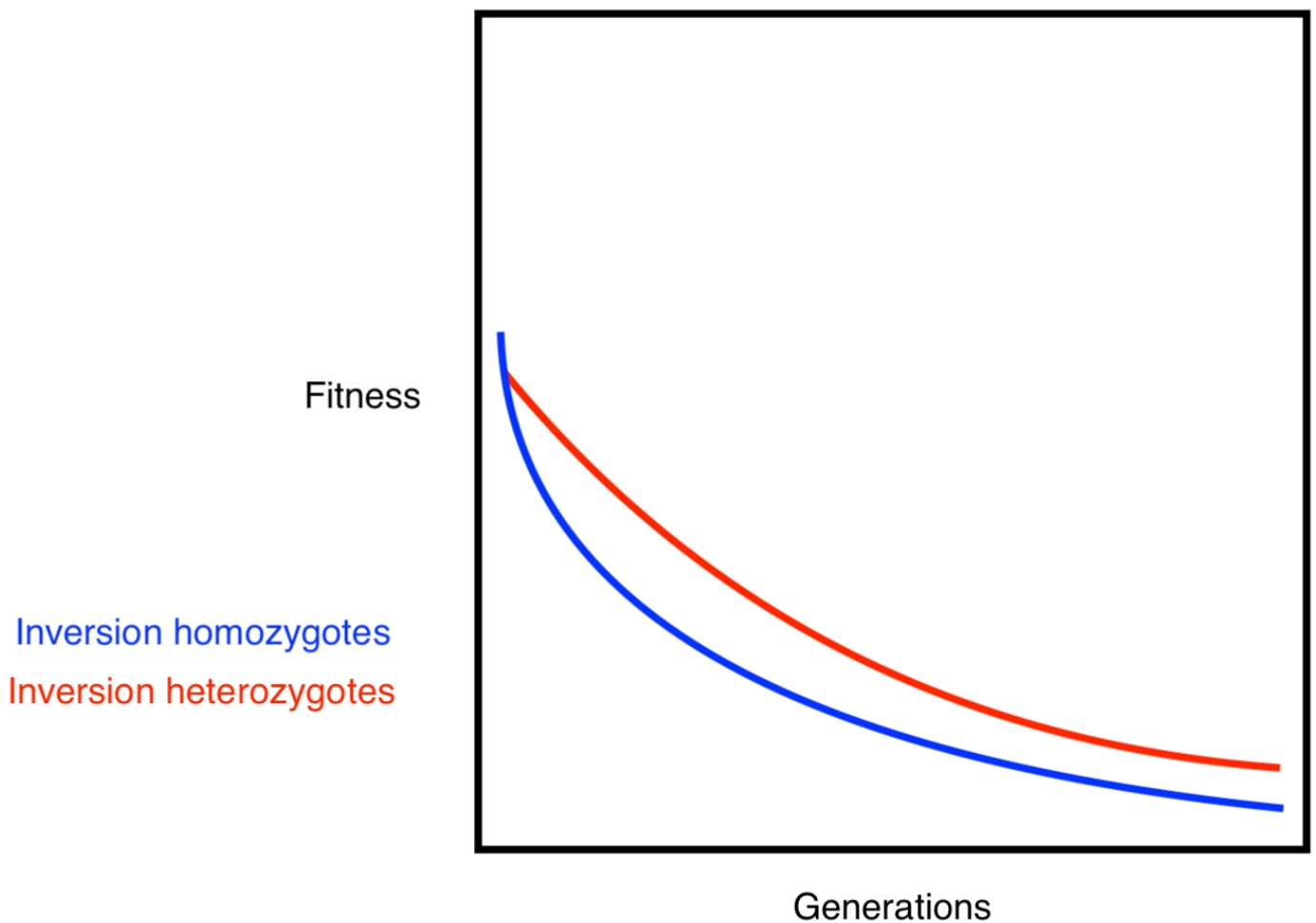


Preliminary results



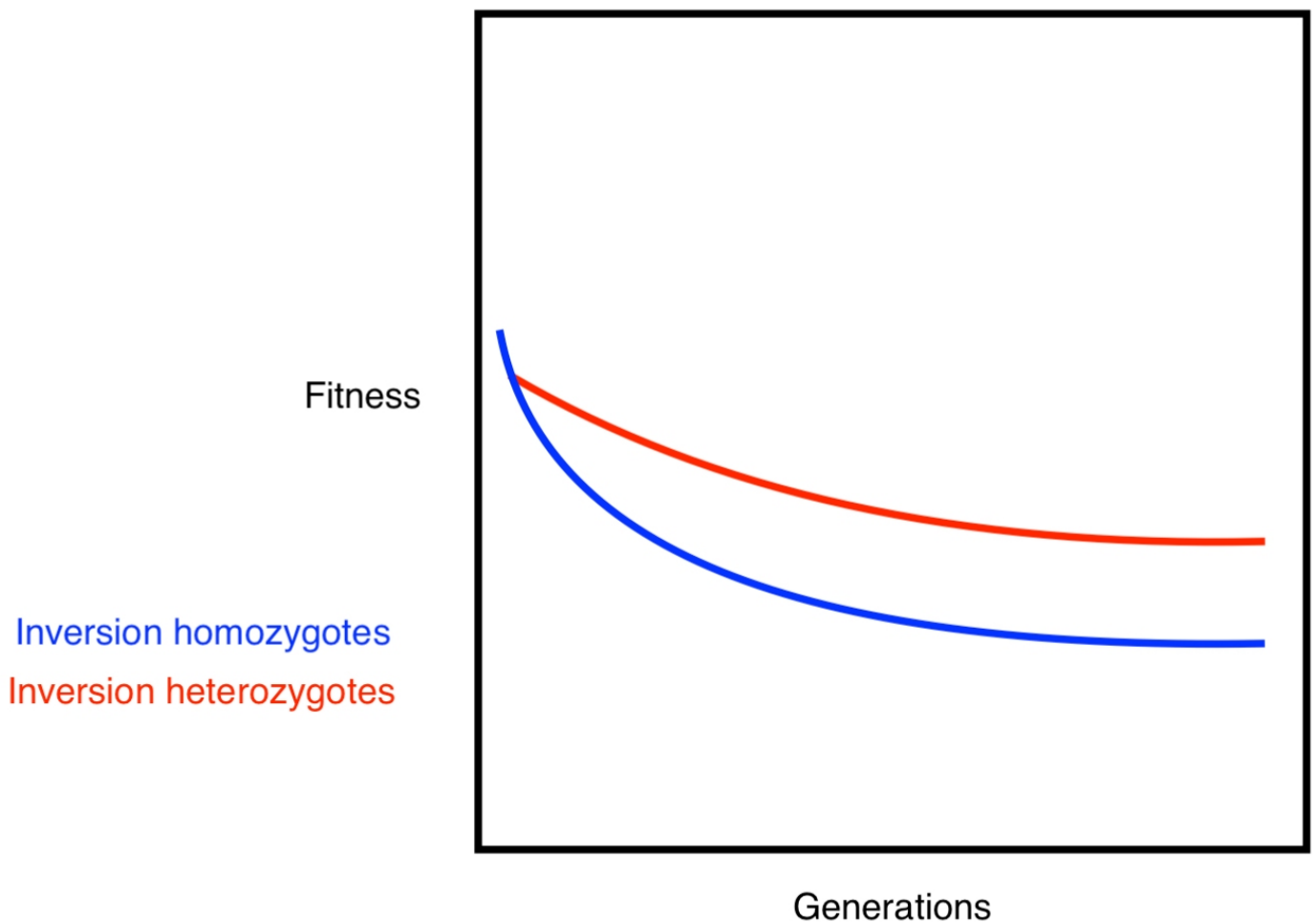
Expected results

In populations where inversion is deleterious:



Expected results

In populations where inversion is advantageous:



Next steps

- Results could be unexpected because of scaling issues, or the recombination function
- Could look at mutations trapped in the inversion next, plot their contents across runs
- Runs with an inversion with deleterious mutations locked in it may be outweighing the strict fitness benefit we give them

Thank you!

Acknowledgements: SLiM development team (Ben Haller, Phillip Messer) as well as their slim-discuss Google forum

Dr. Greg Owens, whose guidance and work in sunflowers has spurred this interest in modelling inversions

Questions?