



THE UNIVERSITY
of EDINBURGH



Biotechnology and
Biological Sciences
Research Council



Introduction to simulations of breeding programmes

Gregor Gorjanc, Chris Gaynor, Jon Bancic, Daniel Tolhurst

UNE, Armidale
2024-02-05

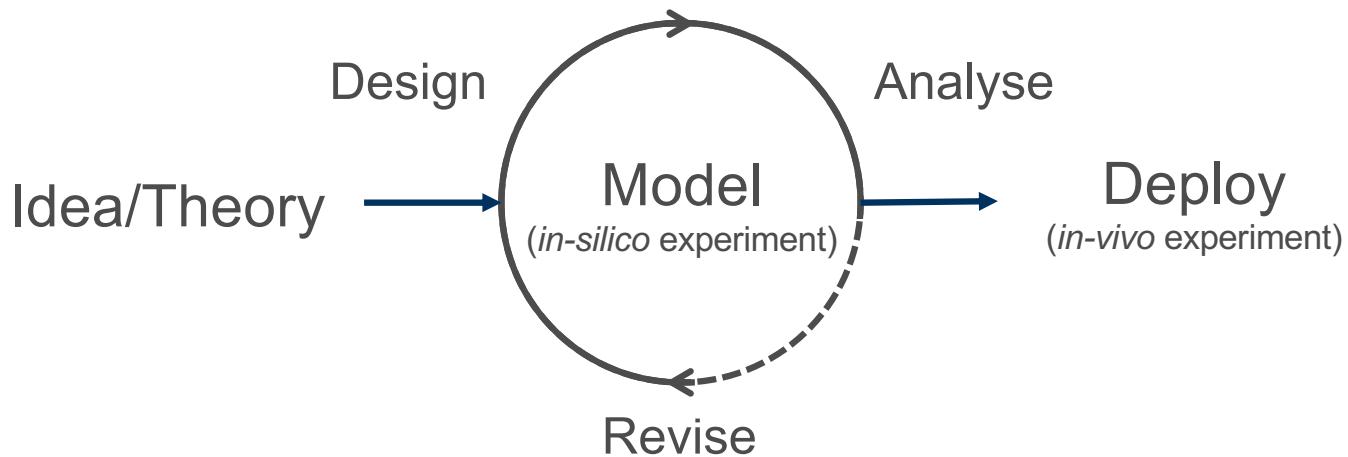


Learning objectives

- Introduce the concept of breeding simulations
- Differentiate deterministic and stochastic simulations
- Showcase one AlphaSimR simulation
- Differentiate backward- & forward-in-time simulations

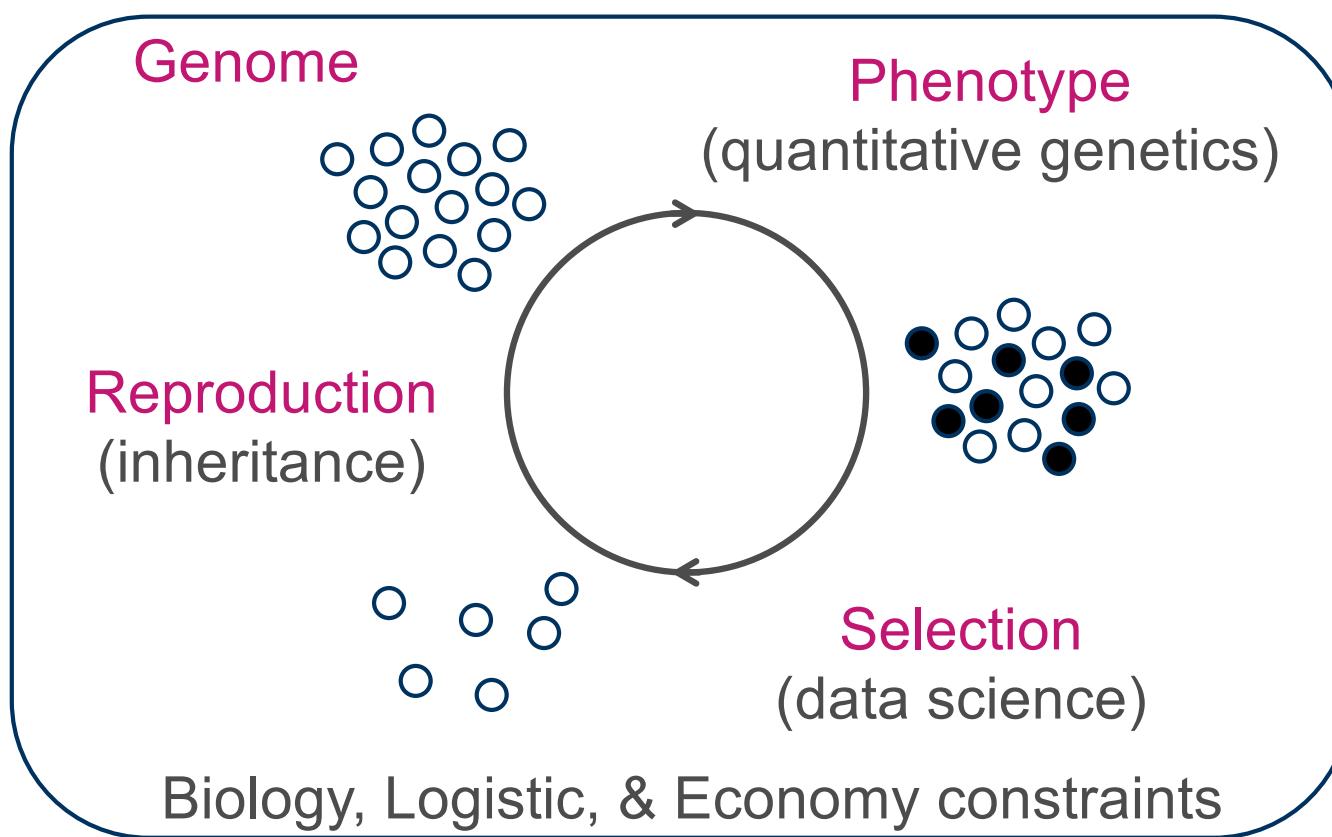
Modelling/simulation mindset in breeding

- Breeding programs are complex, costly, and can be slow!
(genetics, reproduction, production, disease, data, statistics, ...)
- A need for an *in-silico modelling sandbox*



- Capture major components to identify key drivers of population management and improvement

Basic elements of a breeding programme simulation



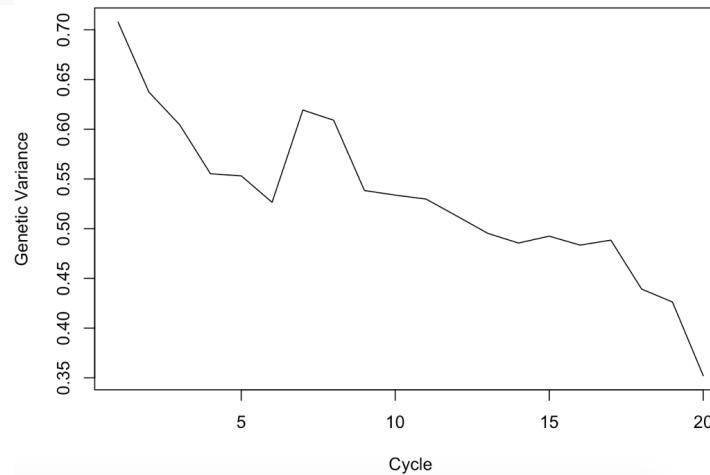
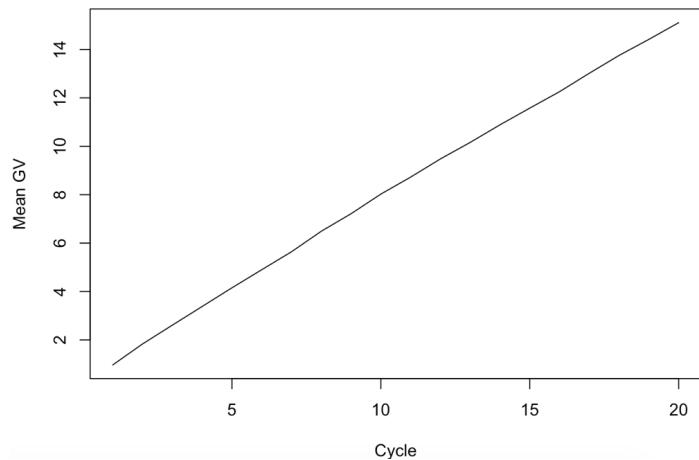
Evolution of our modelling/simulation platform

- ...
- 2006 – Multi-trait polygenic breeding values in livestock
- **2008 – Genomes and markers**
- 2010 – First attempt at complex populations under selection
- 2012 – Released a simple package (*AlphaDrop*)
- **2012 – Plant breeding features and adding complexity** (mating structures, etc.)
- 2014 – Released a complex package (*AlphaSim*)
- 2015 – Complex simulations required lots of “glue-scripting”
- **2017 – Migrate from Fortran to R/C++ and pop. objects - *AlphaSimR*!!!**
- 2017 – Realistic dominance model (total genetic, breeding, and dominance values)
- 2018 – Complete migration, develop many blueprints, EiB use, industry, ...
- 2022 – *AlphaSimR* course
- 2023 – Collection of plant breeding simulations published (animal version in progress)
- ...

AlphaSimR scripting



```
SP = SimParam$new(founderPop)
SP$addTraitA(nQtlPerChr=1000, mean=0, var=1)
SP$setGender("yes_sys")
pop = newPop(founderPop)
popMean = popVar = numeric(20)
for(cycle in 1:20){
  pop = selectCross(pop=pop, nFemale=500, nMale=25, use="gv", nCrosses=1000)
  popMean[cycle] = meanG(pop)
  popVar[cycle] = varG(pop)
}
plot(popMean, type="l", xlab="Cycle", ylab="Mean GV")
plot(popVar, type="l", xlab="Cycle", ylab="Genetic Variance")
```

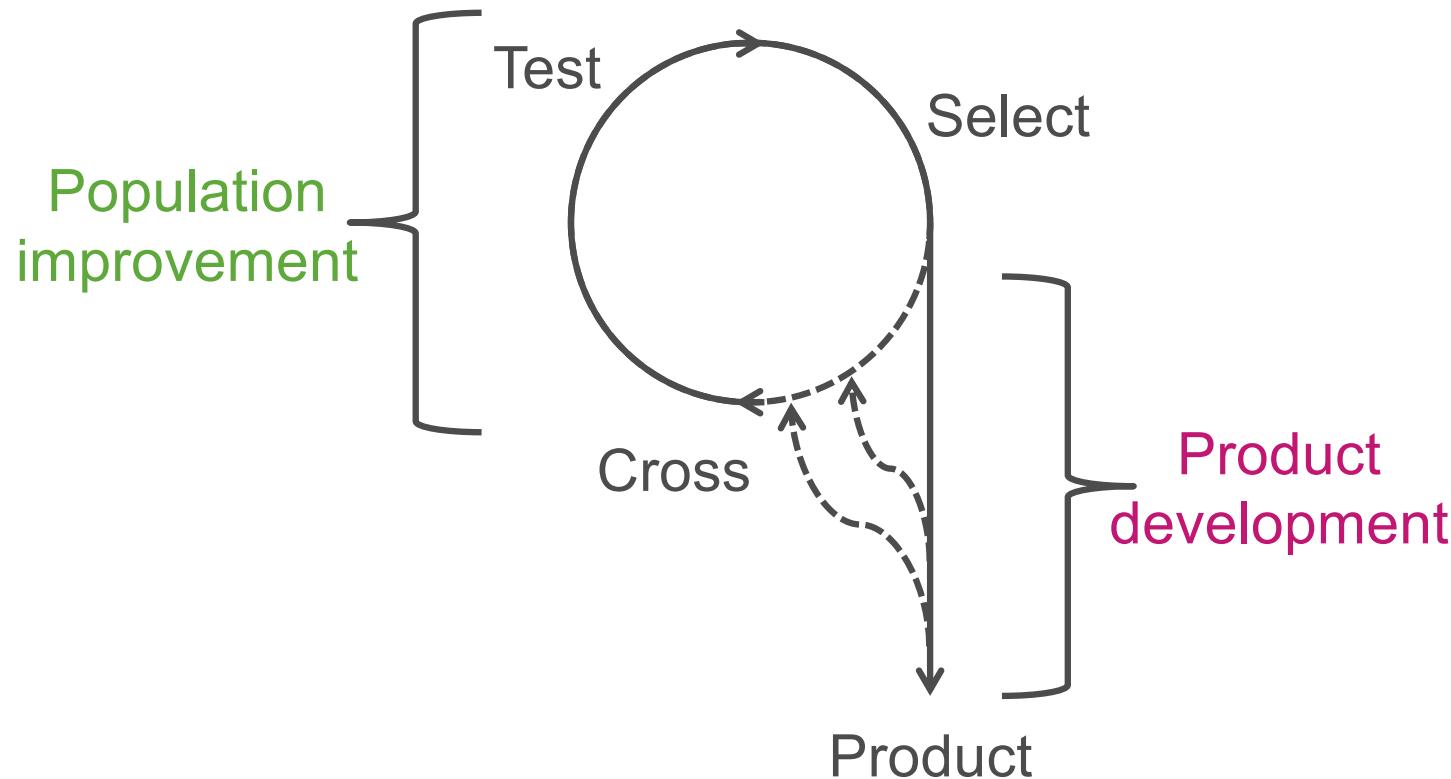


Architecture & Construction analogy



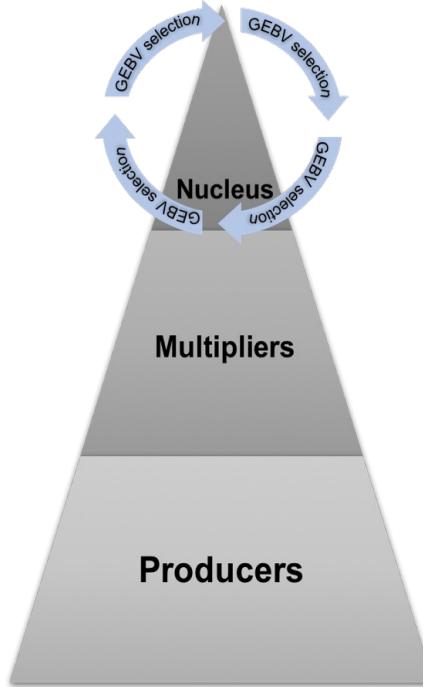
<https://www.dreamstime.com/illustration/home-architecture-project-completion.html>

Two core areas of every breeding programme

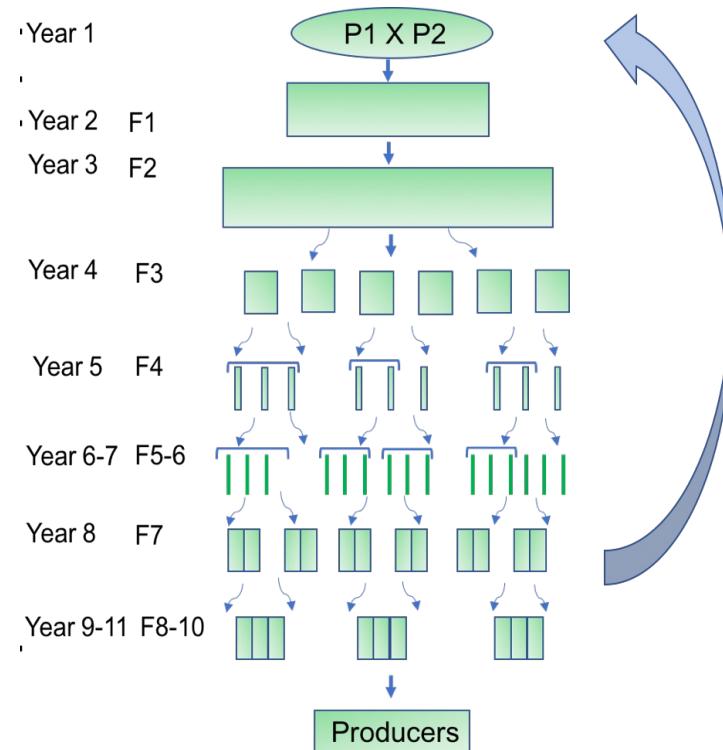


Examples

Animal breeding



Plant breeding



Used across many species



CGIAR - Excellence in Breeding Platform



About Us ▾ Modules ▾ Toolbox Projects ▾ News & outreach ▾



Breeding by the numbers

EiB Annual Report 2020-21

[Read the report](#)

• • •

CGIAR Excellence in Breeding (EiB) is accelerating the modernization of crop breeding programs that serve farmers in low- and middle-income countries. To combat hunger, poverty and climate change, farmers need diverse and **continually improving crop varieties**.

EiB provides system-level **coordination, shared services, expert guidance, resources, and access to cutting-edge innovations** to support CGIAR breeding programs to deliver on **six funder requests**.



Breeding program excellence



Optimizing breeding schemes



Genotyping / sequencing



Phenotyping tools and services

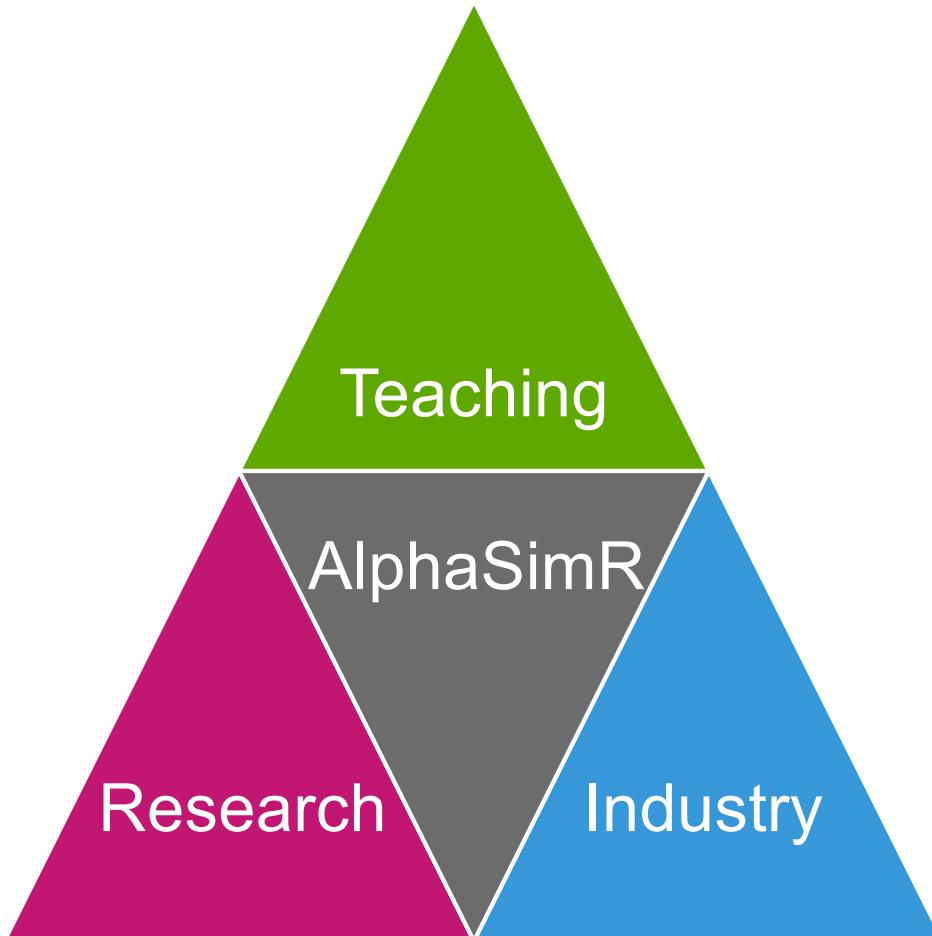


Bioinformatics and data management



Excellence in
Breeding
Platform

AlphaSimR is our core tool!



www.edx.org/course/breeding-programme-modelling-with-alphasimr

 Courses ▾ Programs & Degrees ▾ Schools & Partners 

Catalog > Data Analysis & Statistics Courses



Breeding Programme Modelling with AlphaSimR

Breeding programmes are key to the genetic improvement of plant varieties and animal breeds used in agriculture. This unique course shows how to model an existing or new breeding programme and the evaluation of alternative breeding scenarios.





 **5 weeks**
3–5 hours per week

 **Self-paced**
Progress at your own speed

 **Free**
Optional upgrade available

There is one session available:

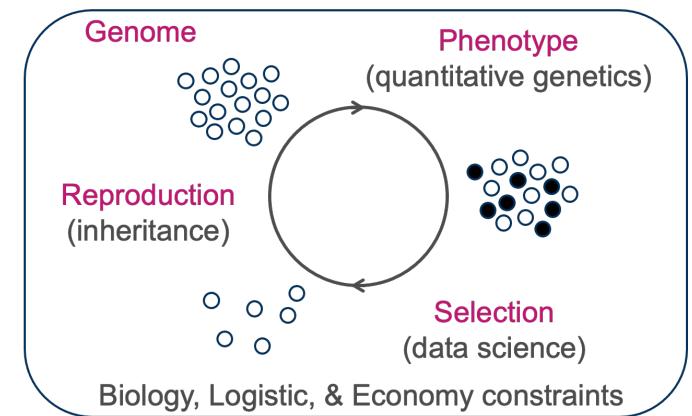
After a course session ends, it will be archived .

Starts Sep 19

Enroll

www.edx.org/course/breeding-programme-modelling-with-alphasimr

- 5 weeks, ~20h (with huge variance!)
 - Week 1: Introduction
 - Week 2: Simulation of DNA and traits
 - Week 3: DNA lottery
 - Week 4: Selection
 - Week 5: Complex breeding programmes
- Open “indefinitely” – share with colleagues & students



www.edx.org/course/breeding-programme-modelling-with-alphasimr



What is AlphaSimR?

- R package for stochastic simulation of genetics & breeding
- Two types of simulations
 - Stochastic (AlphaSimR and other fine simulators)
 - Deterministic
- Deterministic simulations are common
 - Breeder's equation
 - Optimizing multistage selection
 - Simple, until maths became unwieldy

Breeder's equation (R code)

Deterministic simulation

```
h2 = 0.5
```

```
Va = 1
```

```
i = dnorm( qnorm(0.9) ) / 0.1 # top 10%
```

```
i * sqrt(h2) * sqrt(Va) # 1.240961
```

Stochastic simulation

```
a = rnorm(10000, sd=1) # additive genetic values, Va = 1
```

```
e = rnorm(10000, sd=1) # environmental and non-additive genetic values
```

```
p = a + e # phenotype, h2 = 0.5
```

```
best = order(p, decreasing=TRUE)[1:1000] # top 10%
```

```
mean(a[best]) - mean(a) # ~1.240961
```

Why use stochastic simulation?

- Doesn't require a deterministic formula
 - Long-term selection
 - Genomic prediction accuracy
 - Other complex processes
- Handles very complicated simulations
 - Whole breeding programs

Take home message no. 1

Stochastic simulations are cool and powerful!

Let's whet your appetite!



- Genomic selection in wheat as an example
 - Template for our simulations
 - Example of interpreting results
 - Highlights strengths and weakness

Gaynor *et al.* (2017) A Two-Part Strategy for Using Genomic Selection to Develop Inbred Lines. *Crop Science* 57: 2372–2386.

Goal

- Evaluate standard & novel approaches to genomic selection
- Two-part strategy for genomic selection
 - Splits breeding program into two components
 - Population improvement
 - Product development
- Compare against more standard designs
 - Two-part design is risky
 - Does potential justify the risk?

Traditional strategies

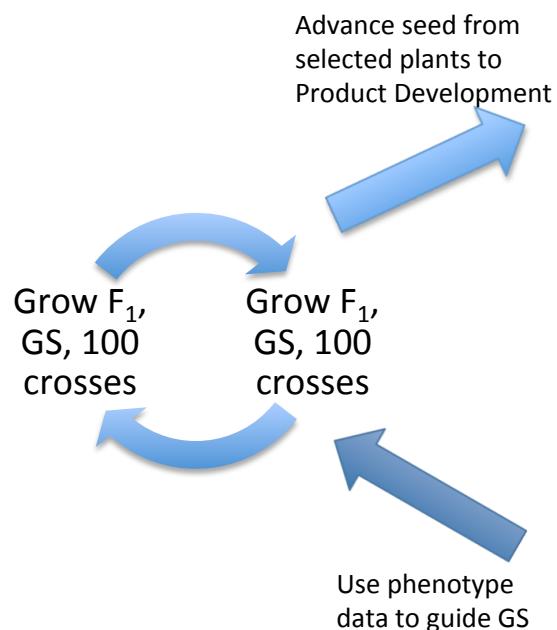
Year	Stage		Number of Plants	Action
1	Crossing	$P_1 \times P_2$	100 crosses	Make bi-parental crosses
1-2	F ₁ /DH	x100	100 full-sib families	Produce DH lines
3	Headrows		100 x N ⁺ DH lines	Advance 500 lines, genotype/cross (Head GS)
4	PYT		500 DH lines	Yield trial, genotype/cross (PYT GS)
5	AYT		50 DH lines	Yield trial, cross (Conv), genotype/cross (Conv GS)
6	EYT		10 DH lines	Yield trial
7	EYT		10 DH lines	Yield trial
8	Variety		1 DH line	Release variety

The diagram illustrates the traditional breeding strategy timeline across eight years:

- Year 1:** Crossing. Starts with parents $P_1 \times P_2$. 100 crosses are made.
- Year 1-2:** F₁/DH. 100 full-sib families are produced.
- Year 3:** Headrows. 100 x N⁺ DH lines are advanced. This stage is labeled **Head GS**.
- Year 4:** PYT. Yield trial, genotype/cross. This stage is labeled **PYT GS**.
- Year 5:** AYT. Yield trial, cross (**Conv**), genotype/cross (**Conv GS**). This stage is labeled **Conv/Conv GS**.
- Year 6:** EYT. Yield trial.
- Year 7:** EYT. Yield trial.
- Year 8:** Variety. 1 DH line is released.

Two-part strategy

Population Improvement



Product Development

Year	Stage	Number of Plants	Action
1-2	F ₁ /DH	200 half-sib families	Produce DH lines
3	Headrow	200 x N [†] DH lines	Advance 500 lines, genotype (2Part+H)
4	PYT	500 DH lines	Yield trial, genotype (2Part)
5	AYT	50 DH lines	Yield trial
6	EYT	10 DH lines	Yield trial
7	EYT	10 DH lines	Yield trial
8	Variety	1 DH line	Release variety

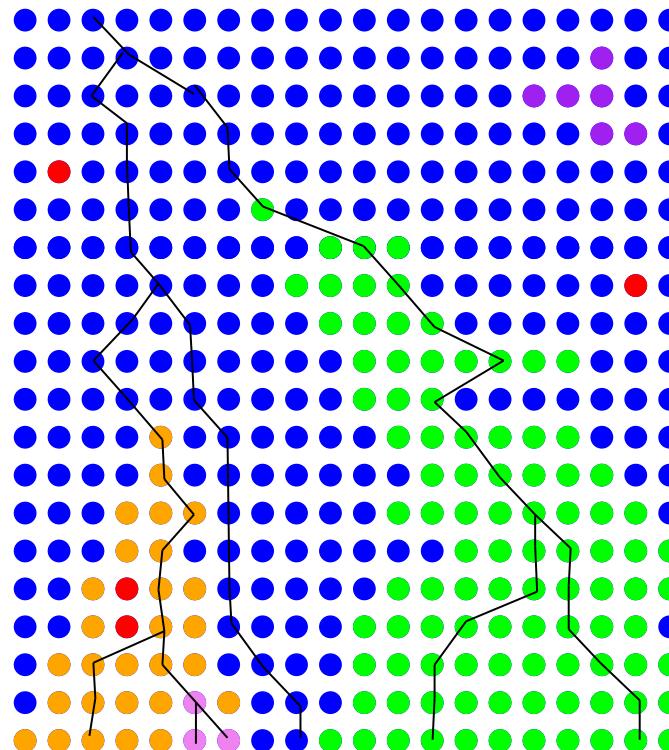
Basic simulation scheme

1. Coalescent simulation (MaCS)

- Backward-in-time
- Model species' genome

Backward-in-time simulation of DNA

Backward-in-time stochastic process
("progeny choose their parent chromosomes" → coalescent)



AlphaSimR uses
MaCS (SMC algorithm)

Basic simulation scheme

1. Coalescent simulation (MaCS)

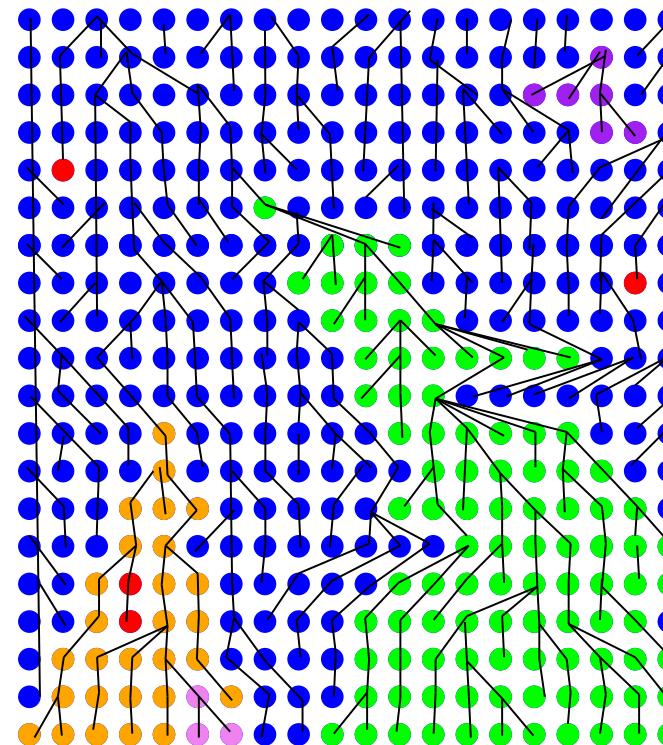
- Backward-in-time
- Model species' genome

2. Gene drop simulation (AlphaSimR)

- Forward-in-time
- Model traits and genetic recombination
- Model breeding programs

Forward-in-time simulation of DNA

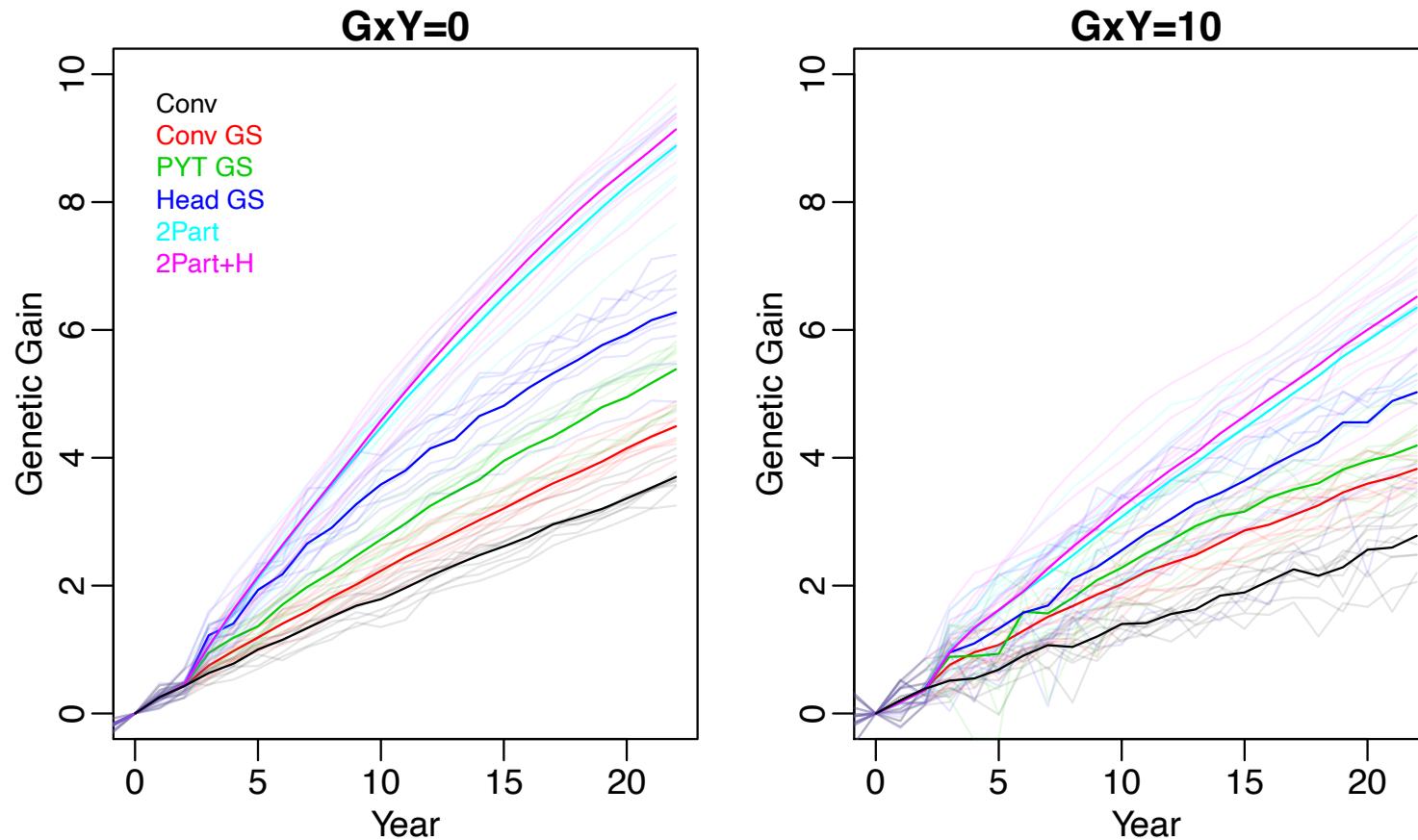
Forward-in-time stochastic process
("parents transmit chromosomes to their progeny")



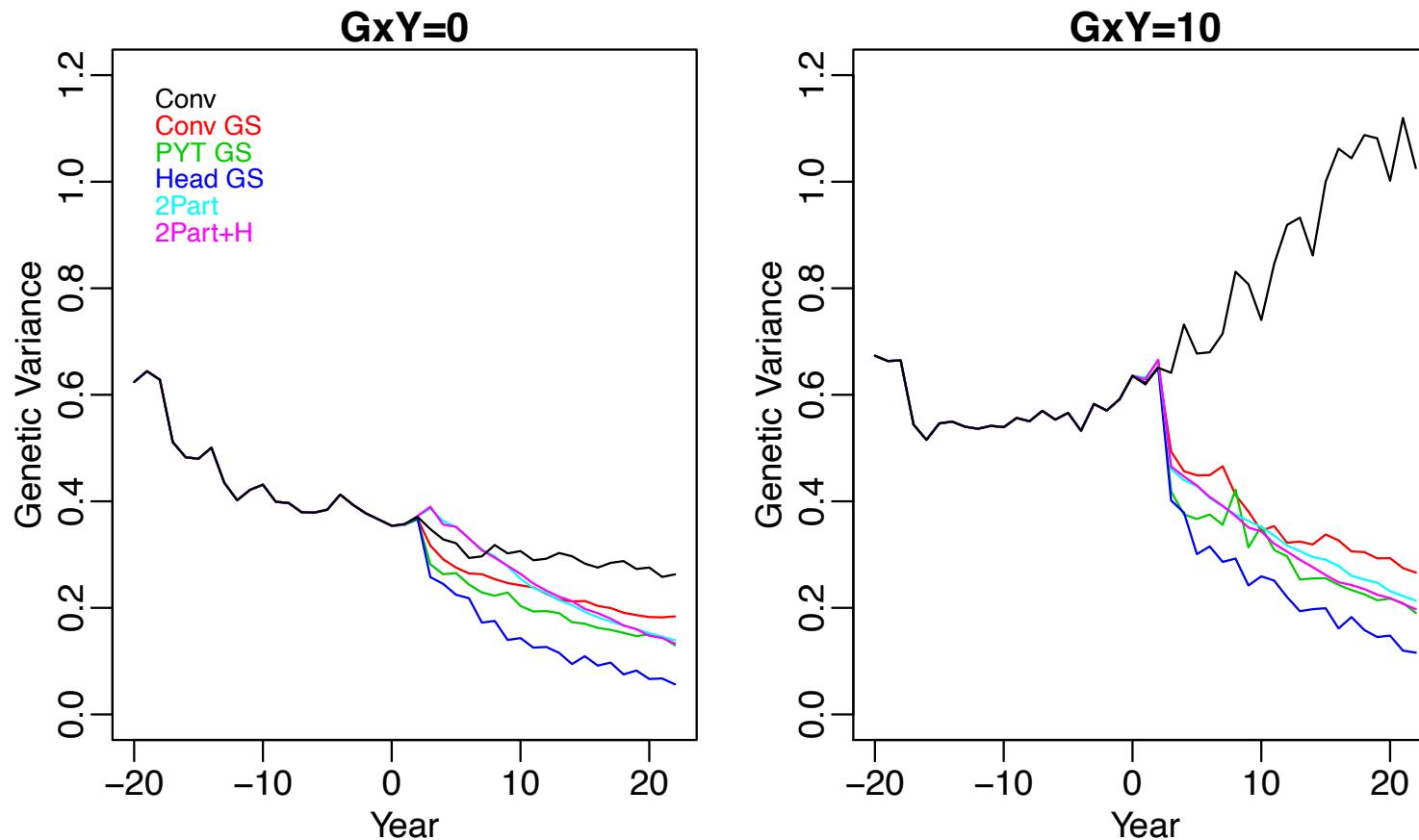
Detailed simulation scheme

Simulation Stage	Key Features
Burn-in	Genome Sequence
	100,000 generations of evolution
	Wheat historical effective population size
Recent Breeding	21 chromosome pairs
	1.43 Morgans per chromosome
	8×10^8 base pairs per chromosome
Future Breeding	2×10^{-9} mutation rate
	50 inbred founders
	21,000 SNP markers
Evaluation	21,000 QTN
	Normally distributed QTN effects
	20 years of modern breeding (-19 to 0)
Recent Breeding	Double haploid lines
	No genomic selection
	20 years of breeding (1 to 20)
Future Breeding	Testing alternative breeding programs
	Equal cost programs
	Ridge regression for genomic selection

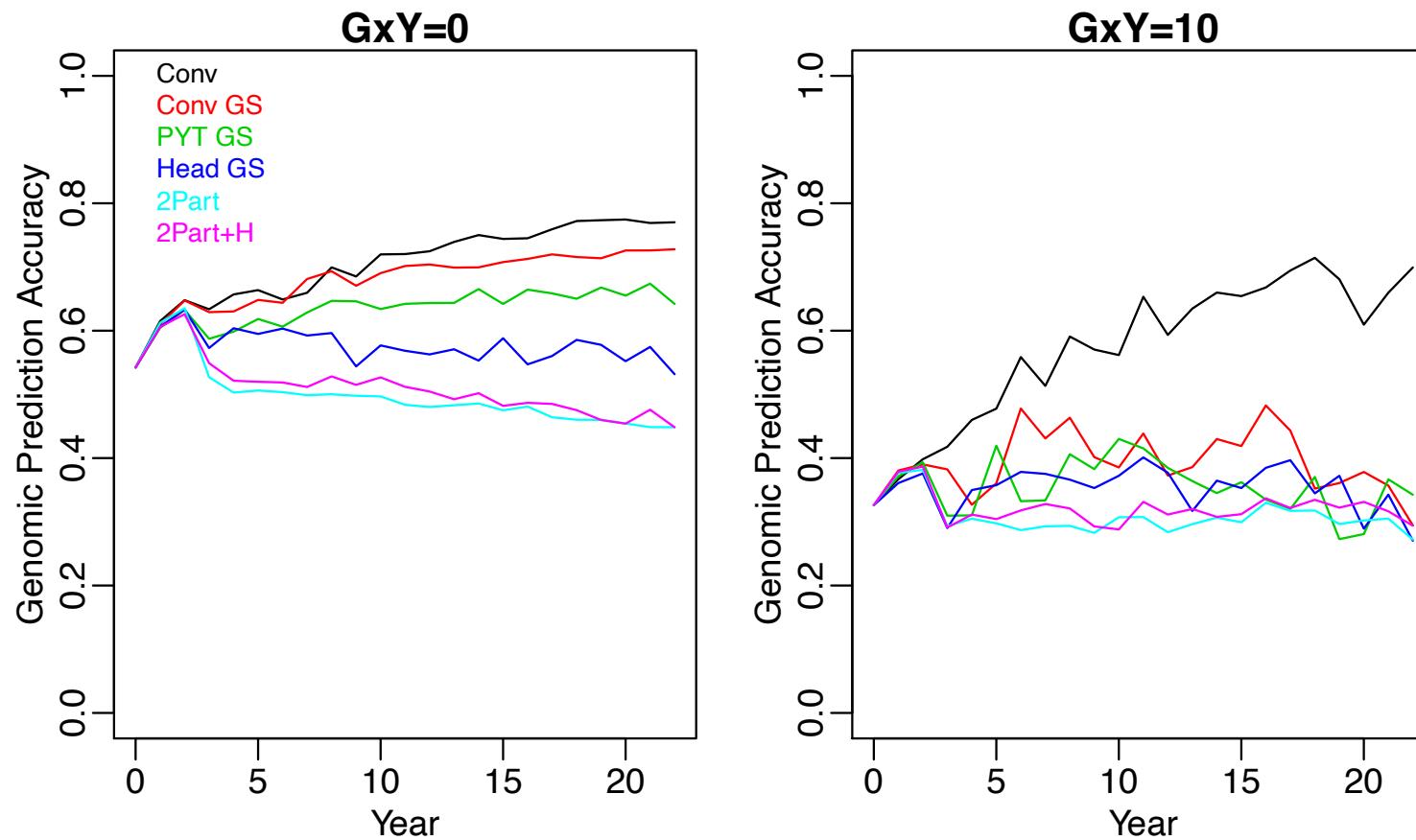
Genetic gain



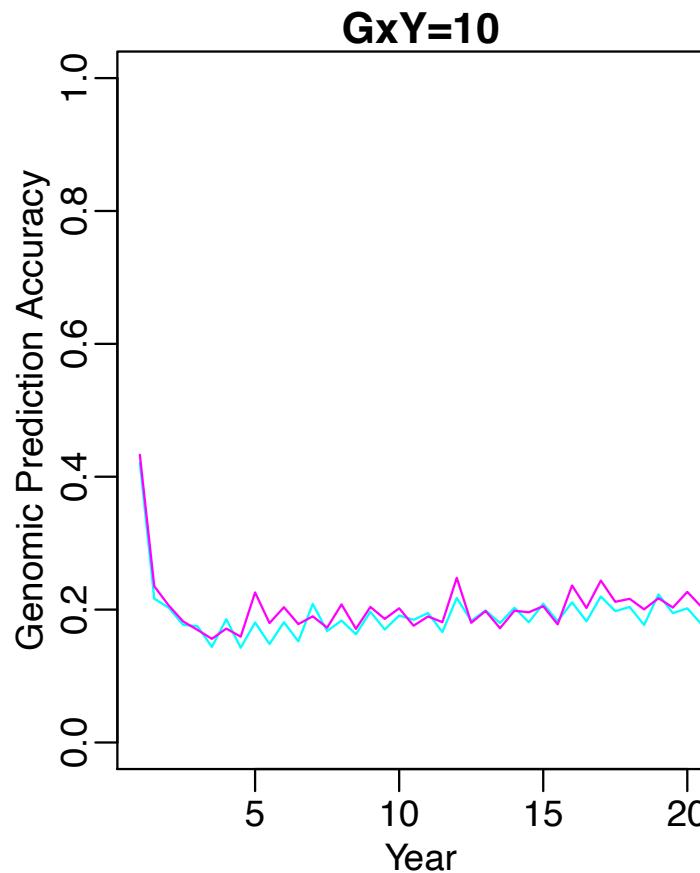
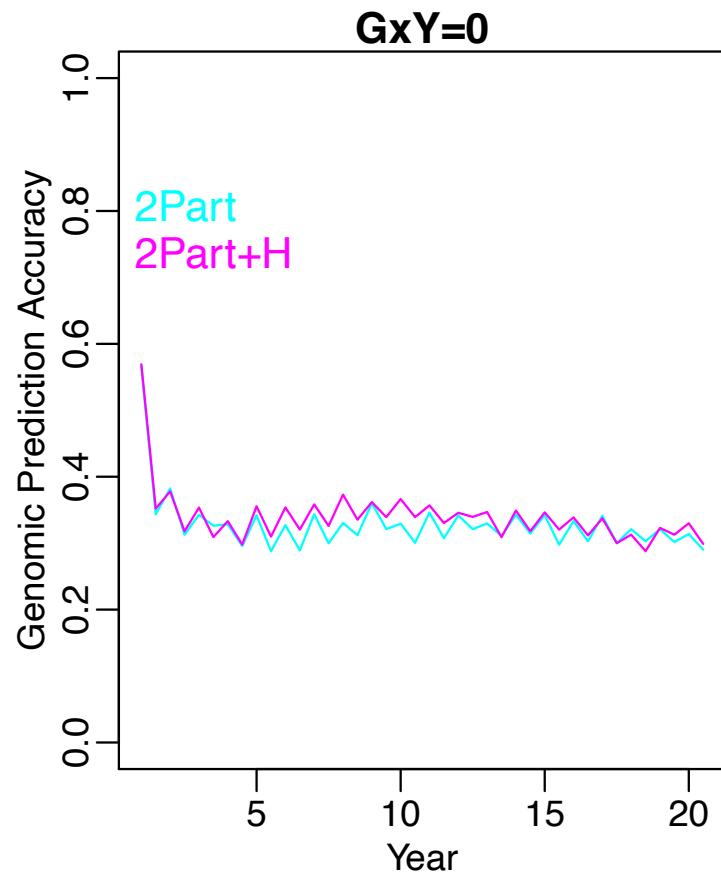
Genetic variance



GS accuracy, headrows



GS accuracy, population improvement



Main messages

- Reducing cycle time is key
 - 2-Part(+H) > Head GS > PYT GS > Conv GS
- Genomic selection can improve accuracy
 - Conv GS > Conv
- Genomic selection accuracy can rapidly decay
 - Primary limiter of two-part methods

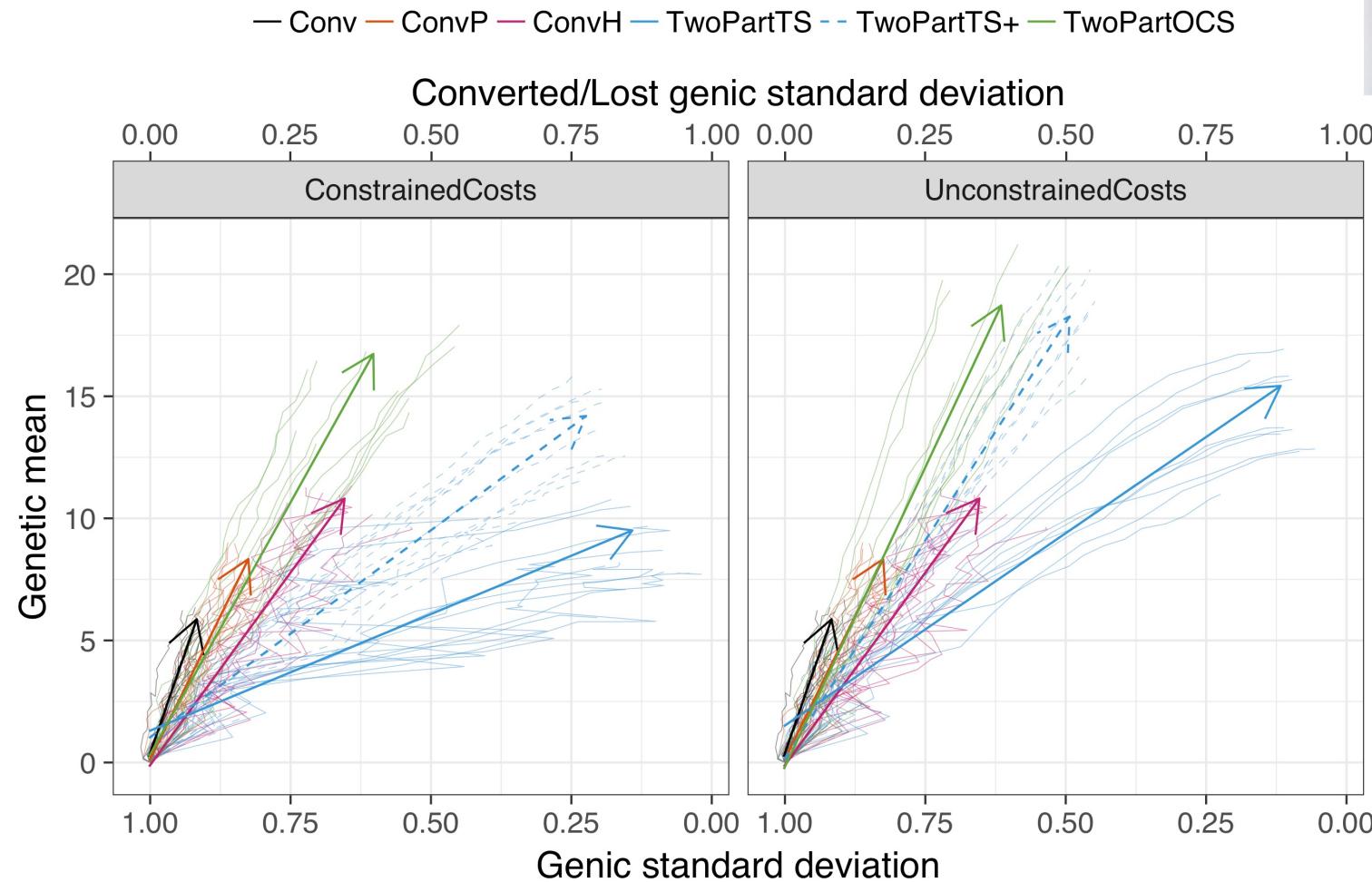
Limitations

- Simple trait model
 - High genomic selection accuracy and persistence
- Only one trait (yield)
- Open questions
 - Germplasm exchange
 - Conservation of diversity
 - Lots of fine tuning

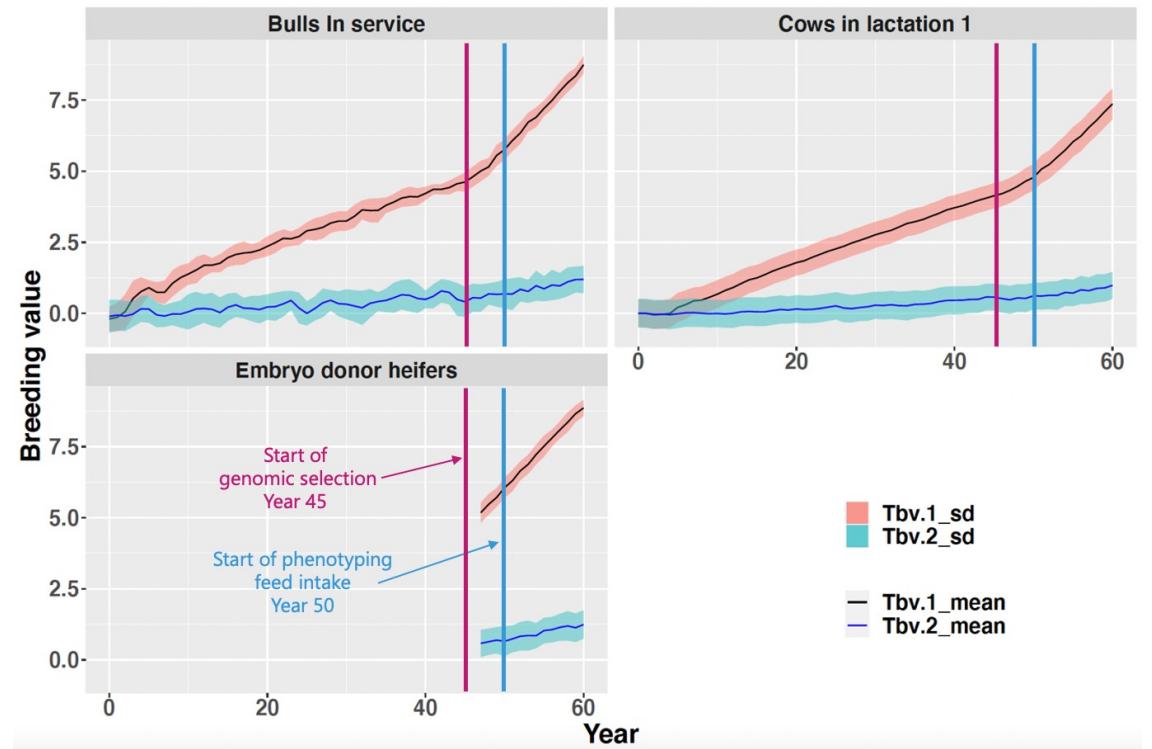
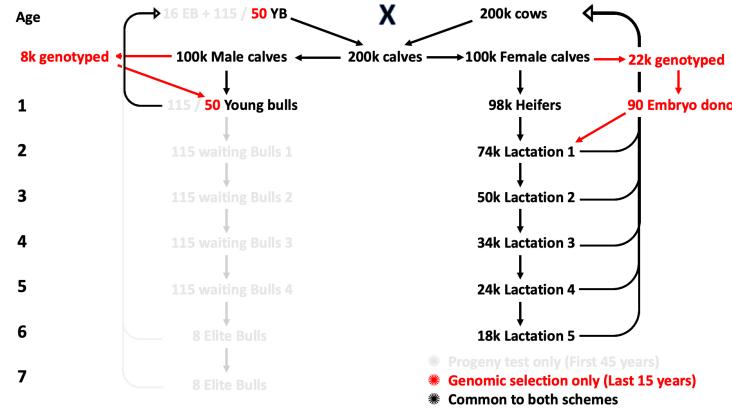
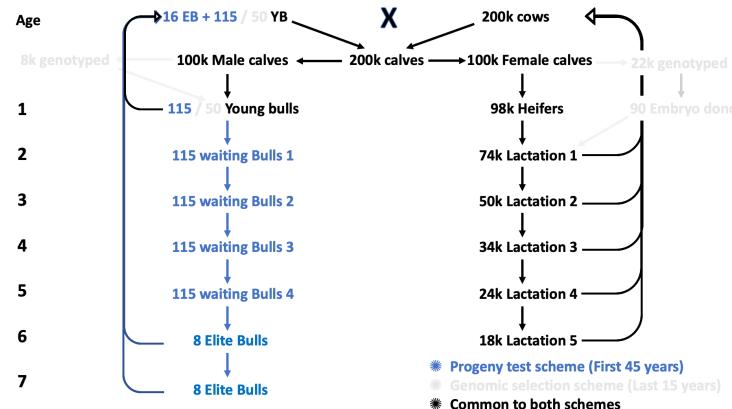
Suggestions

- Focus on reducing cycle time
 - Main driver of genetic gain
- Phased deployment of two-part strategy
 - Build infrastructure
 - Gain confidence
 - Mitigate risks

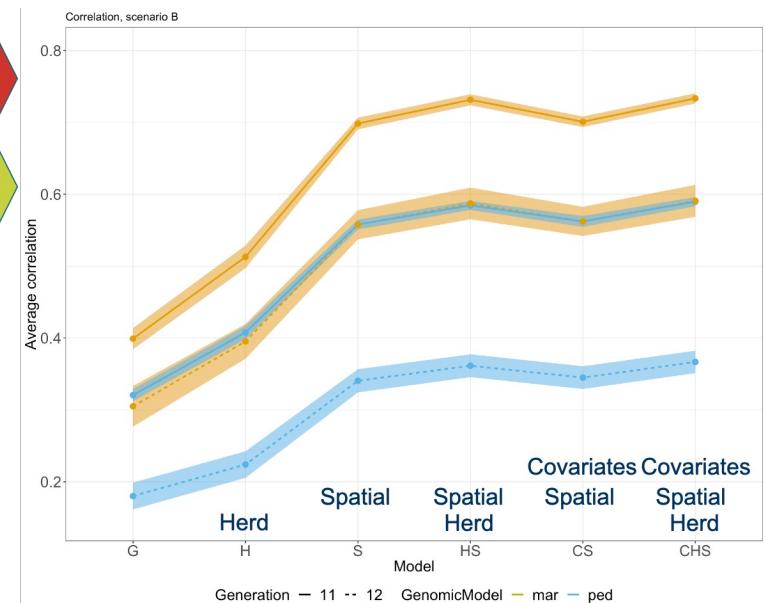
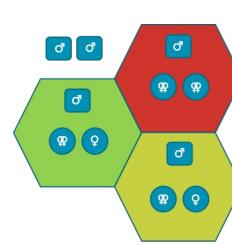
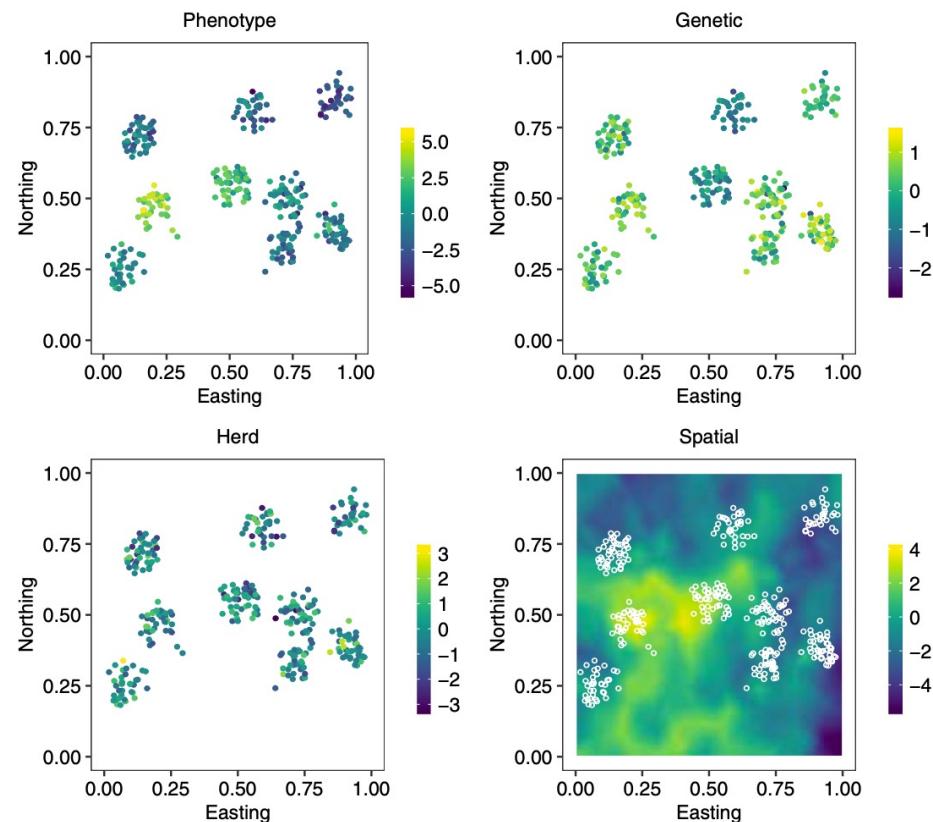
Example: Balancing selection & diversity



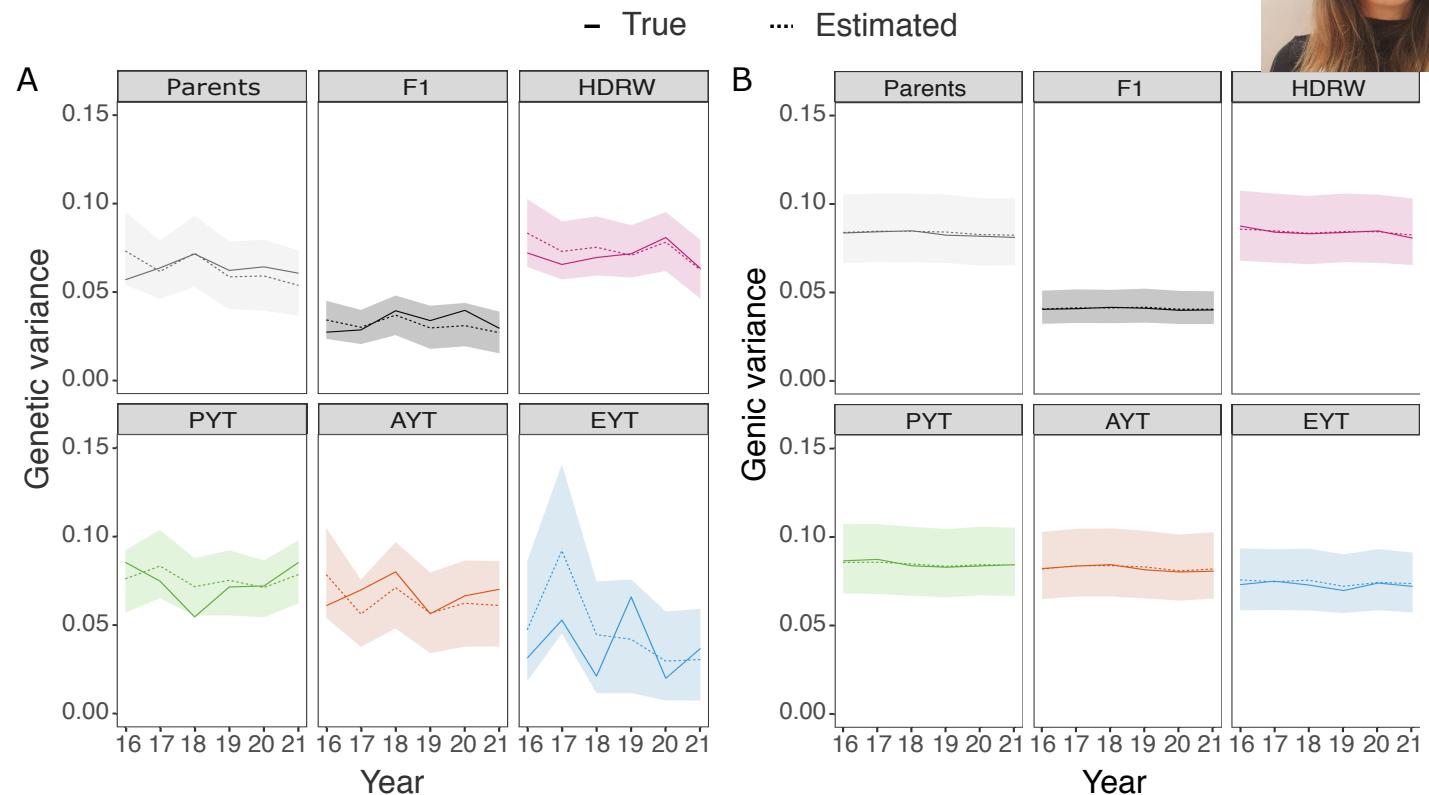
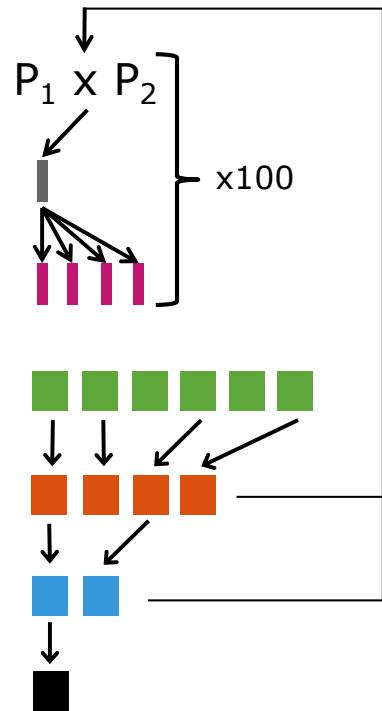
Example: Dairy cattle breeding programme



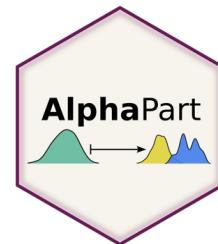
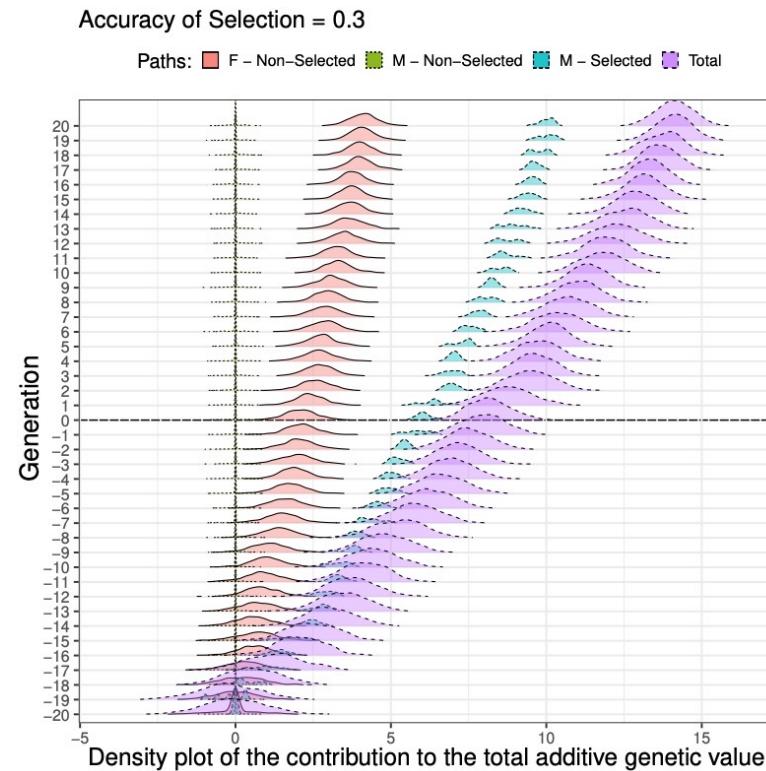
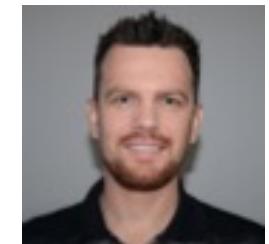
Example: Spatial modelling & smallholders



Example: Analysis of genetic variance



Example: Partitioning genetic trends



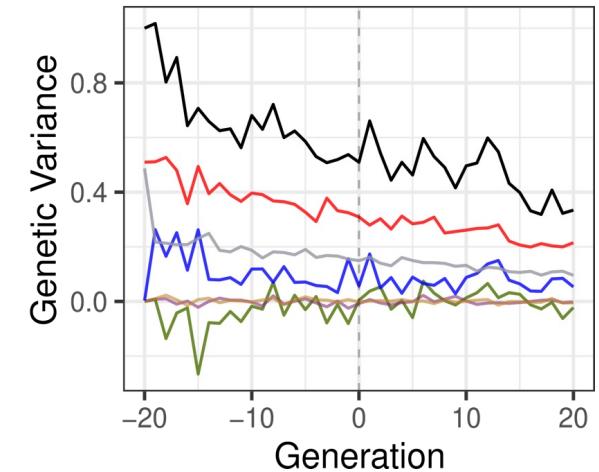
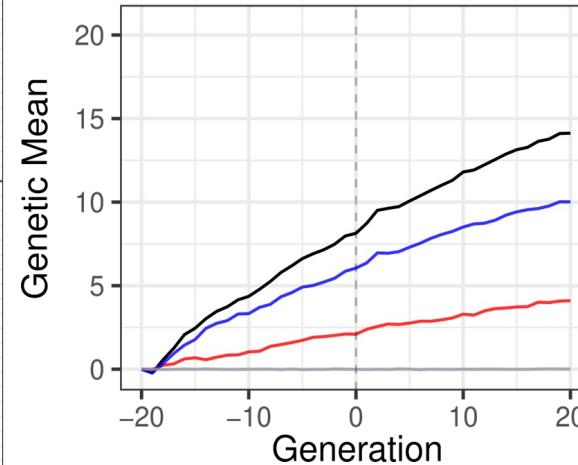
Path:

— Sum
— F

— M(N)
— M(S)

— F:M(S)
— F:M(N)

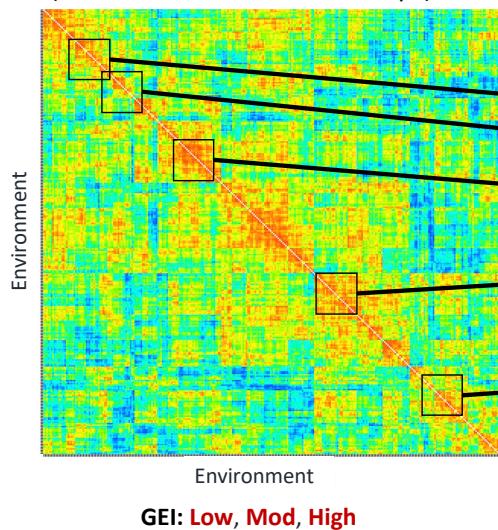
— M(S):M(N)



Example: GxE simulations



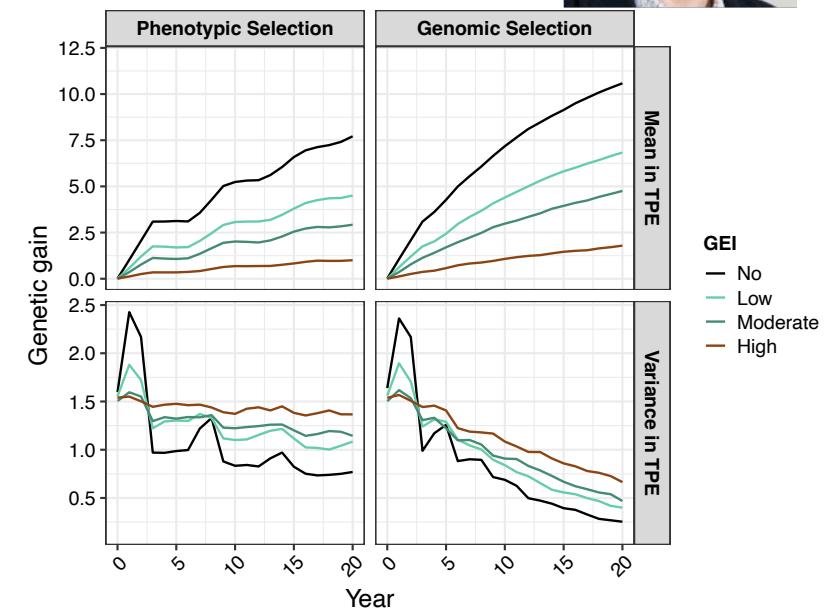
1. Simulate 1000×1000 TPE (constant across simulation reps)



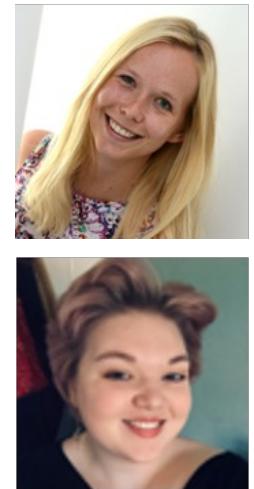
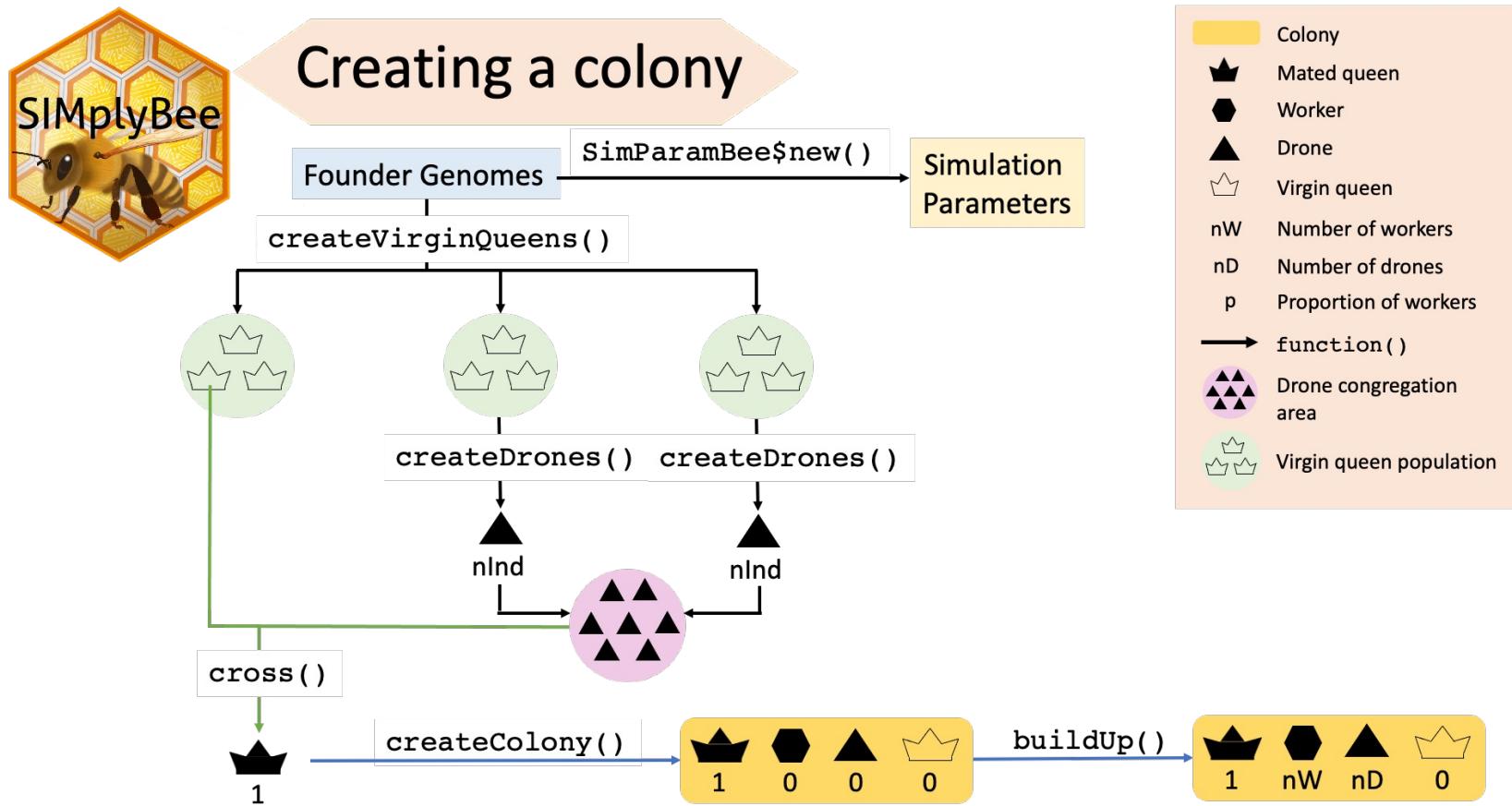
 Simulation of TPE genetic effect
→ True performance

2. Sample for each simulation year

Simulation of MET genetic effects
→ Estimated/observed performance
(e.g. Stage 1 ~ 1 env,
Stage 4 ~ 20 env)



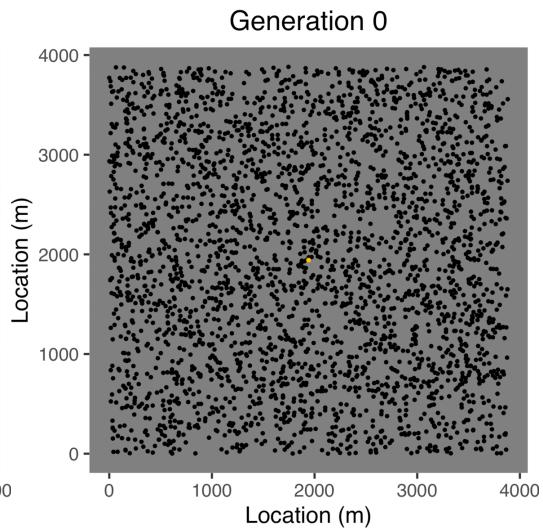
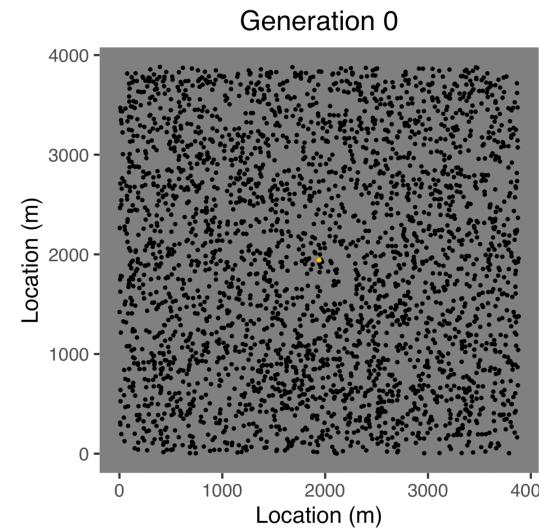
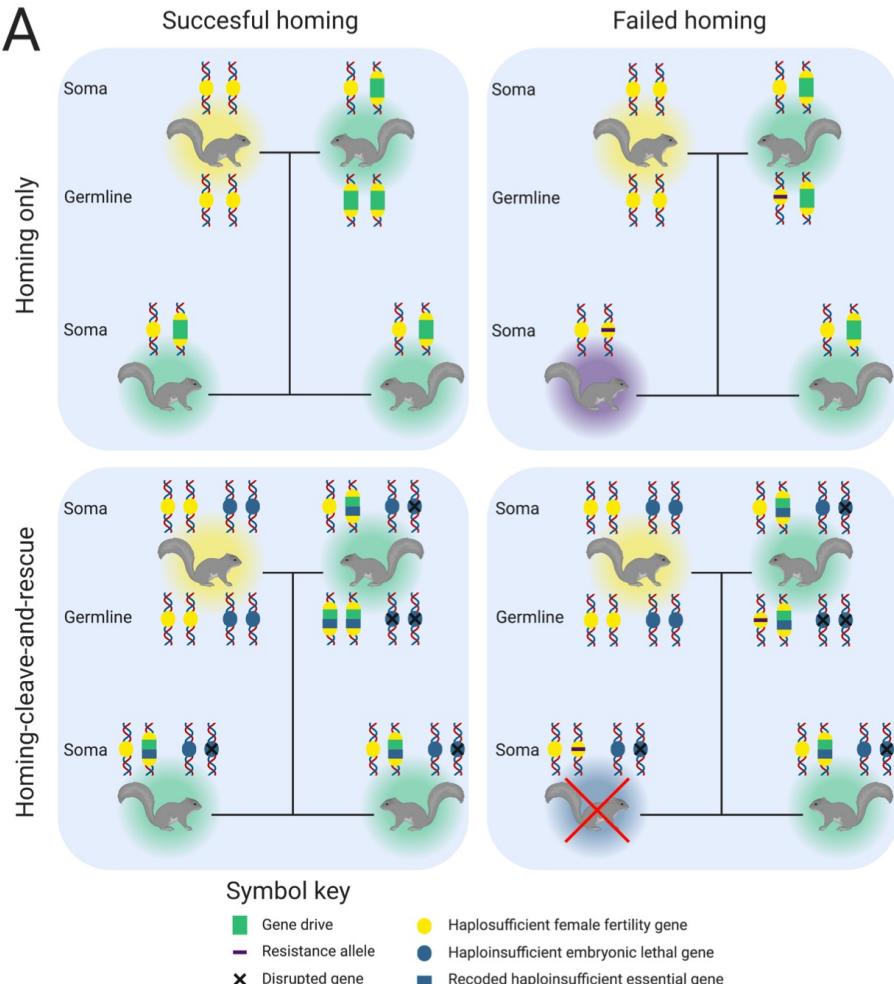
Example: Honey bee extension



Example: Gene drives in gray squirrels



A



Take home message no. 2

**Stochastic simulations support development of
new breeding programs / methods / theory / ...!**

AlphaSimR's strengths

- Very flexible
 - Not limited to preset designs
 - Access to inner workings
- Fast
 - Leverages modern computing techniques
- Alignment with classic quantitative genetics
 - Strong foundations
 - Genetic effects, values, and variance components

AlphaSimR's weaknesses

- Limitations of all simulations
 - Simplified versions of reality
- Moderate learning curve
 - Designed for scripters
 - Limited documentation of complex tasks
- Output on demand
 - Must rerun simulations if you forget something

AlphaSimR hands-on

- All examples will use R Markdown
 - We'll briefly explain how they work
- We will work through R Markdown files in folder
 - Day_1_Intro_AlphaSimR
 - 01Practical_DNA
 - 02Practical_Selection
 - 03Practical_Breeding program

Questions?!

Other fine simulators

- MoPBS (R) – a competitor!
 - PyBrOpS & ChromaX (Python)
 - XSim & GEAS (Julia)
 - Many other fine simulators!
→ Spend time on a few more or contribute to the existing? ;)
-
- msprime (Python/C)
 - SLiM (Eidos/C++)
 - stdpopsim (Python)

msprime (backward-in-time simulator)

<https://pypi.org/project/msprime>

The screenshot shows the 'Introduction' page of the msprime manual. At the top right are navigation icons: back, forward, search, and download. The title 'Introduction' is centered above a detailed paragraph about the msprime simulator. Below the paragraph is a bulleted list of resources for learning about tskit and msprime. A yellow callout box labeled 'Important' contains a note about citing the software. At the bottom is a 'Contents' section with links to 'Getting started', 'Running simulations', and other sections.

← ⌂ ⌓ ⌚ ⌚ ⌚

Msprime manual

Q Search this book...

Introduction

GETTING STARTED

Quickstart
Installation

RUNNING SIMULATIONS

Ancestry simulations
Mutation simulations
Demographic models
Randomness and replication

INTERFACES

API Reference
Command line interface

UTILITIES

Rate Maps
Pedigrees
Computing likelihoods
Logging

MISCELLANEOUS

Legacy (version 0.x) APIs
Switching from other simulators
Development
Citing msprime
Changelog

Introduction

This is the manual for **msprime**, a population genetics simulator of ancestry and DNA sequence evolution based on **tskit**. **msprime** can simulate **ancestral histories** for a sample of individuals, consistent with a given **demography** under a range of different models and evolutionary processes. It can also simulate **mutations** on a given ancestral history (which can be produced by **msprime** ancestry simulations or other programs supporting **tskit**) under a variety of different **models** of genome sequence evolution.

Besides this manual, there are a number of other resources available for learning about **tskit** and **msprime**:

- The [tskit tutorials](#) site contains in-depth tutorials on different aspects of **msprime** simulations as well as how to analyse simulated **tskit** tree sequences.
- Our [Discussions board](#) is a great place to ask questions like "how do I do X" or "what's the best way to do Y". Please make questions as clear as possible, and be respectful, helpful, and kind.
- The book chapter [Coalescent simulation with msprime](#) is a comprehensive introduction to running coalescent simulations with **msprime**, and provides many examples of how to run and use coalescent simulations. **Note however** that the chapter uses the deprecated [legacy 0.x API](#), and so does not follow current best practices.
- If you would like to understand more about the underlying algorithms for **msprime**, please see the [2016 PLoS Computational Biology paper](#). For more information on the [Discrete Time Wright-Fisher model](#), please see the [2020 PLoS Genetics paper](#).

Important

If you use **msprime** in your work, please remember to cite it appropriately: see the [citations](#) page for details.

Contents

Getting started

- [Quickstart](#)
- [Installation](#)

Running simulations

SLiM (forward-in-time simulator)

<https://messerlab.org/slim>

About SLiM

SLiM is an evolutionary simulation framework that combines a powerful engine for population genetic simulations with the capability of modeling arbitrarily complex evolutionary scenarios. Simulations are configured via the integrated Eidos scripting language that allows interactive control over practically every aspect of the simulated evolutionary scenarios. The underlying individual-based simulation engine is highly optimized to enable modeling of entire chromosomes in large populations. We also provide a graphical user interface on macOS, Linux, and Windows, for easy simulation set-up, interactive runtime control, and dynamical visualization of simulation output.

A 4-5 day SLiM Workshop is [now available online](#). The SLiM Workshop is also offered in person from time to time; see the SLiM Workshops subsection below for more information.

Downloads (version 4.0.1)



[macOS Installer](#)



[Source Code](#)



[SLiM Manual](#)



[Eidos Manual](#)



[Ref Sheets](#)

SLiMgui

With the SLiMgui graphical modeling environment (compatible with macOS, Linux, and Windows), you can visualize your simulation as it runs and examine its parameters in real-time, allowing for much easier simulation development.

The screenshot shows the SLiMgui interface. At the top, there's a header "Recipe 12.3 - Simulating gene drive". Below it is a table with columns ID, N, U, ♂♀, ♂♂, and a row of population icons labeled p0 through p5. To the right of the table is a "Generation" slider set to 158. Below the table are two horizontal bars: one for population size (ranging from 0 to 99999) and one for mutation rate (ranging from 0 to 99999). On the left, there's an "Input Commands:" section with a code editor containing Eidos script. The script initializes parameters like migration rates, mutation types, and population sizes. On the right, there's a "Run Output:" section with a code editor showing log messages and a small circular graph showing population transitions between p0, p1, p2, p3, p4, and p5. The bottom status bar indicates "0.737700 CPU seconds elapsed inside SLiM; 38.7 MB memory usage in SLiM; 435 mutations segregating, 0 substitutions."

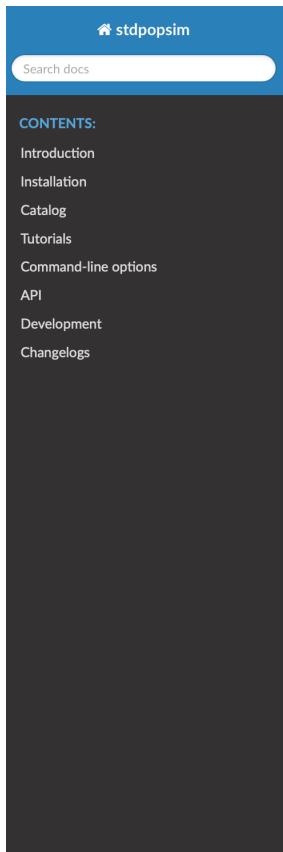
```
1 // Keywords: migration, dispersal, CRISPR gene drive, fitness()
2
3 initialize() {
4     initializeMutationRate(1e-7);
5     initializeMutationType("m1", 0.5, "+", 0.0); // neutral
6     initializeMutationType("m2", 0.5, "-", 0.0); // MCR complex
7     initializePopulation("p0", 500, 0.0, 0.0, 0.0, 0.0);
8     initializeGenomicElement(g1, 0, 99999);
9     initializeRecombinationRate(1e-8);
10 }
11
12 for (i in 0:5) {
13     sim.addSubpopulation(i, 500);
14     for (j in 1:5) {
15         sim.subpopulations[i].setMigrationRates(i-1, 0.001);
16         sim.subpopulations[i].setMigrationRates(i+1, 0.1);
17     }
18 }
19 late() {
20     pb.genomes[0:49].addNewDrawnMutation(m2, 10000);
21 }
22
23 100:10000 late() {
24     if (sim.countOfMutationsOfType(m2) == 0)
25     {
26         fixed = any(sim.substitutions.mutationType == m2);
27         if (fixed, "FIXED", "LOST\n");
28     }
29 }
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
99
100 }
```

stdpopsim (standard population genetics library)

<https://pypi.org/project/stdpopsim>

Frontend for:

- msprime
- SLiM



Welcome to stdpopsim's documentation!

Contents:

- Introduction
 - First steps
 - Getting involved
 - Citations
 - Licence and usage
- Installation
 - Requirements
 - Conda
 - Pip
 - Running the CLI
- Catalog
 - Arabidopsis thaliana
 - Canis familiaris
 - Drosophila melanogaster
 - Escherichia coli
 - Homo sapiens
 - Pongo abelii
- Tutorials
 - Running `stdpopsim` with the command-line interface (CLI)
 - Running stdpopsim with the Python interface (API)
 - Example analyses with `stdpopsim`
- Command-line options
 - Positional Arguments
 - Named Arguments
 - msprime specific parameters
 - SLiM specific parameters
 - Sub-commands:

Take home message no. 3

AlphaSimR is cool & there are additional fine genetics simulators!

Takeaways

- Learning objectives
 - Introduce the concept of breeding simulations
 - Differentiate deterministic and stochastic simulations
 - Showcase one AlphaSimR simulation
 - Differentiate backward- & forward-in-time simulations
- Take home messages
 - Stochastic simulations are cool and powerful!
 - Stochastic simulations support development of new breeding programs / methods / theory / ...!
 - AlphaSimR is cool & there are additional fine genetics simulators!

Questions?!



THE UNIVERSITY
of EDINBURGH



Biotechnology and
Biological Sciences
Research Council



Introduction to simulations of breeding programmes

Gregor Gorjanc, Chris Gaynor, Jon Bancic, Daniel Tolhurst

UNE, Armidale
2024-02-05

