SLiM

Workshop Series

#13: Quantitative Traits

Benjamin C. Haller

Messer Lab, Cornell University

Quantitative Traits

- Mendelian traits:
  - governed by a single locus
  - produce discrete outcomes
  - can be modeled with selection coefficients

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

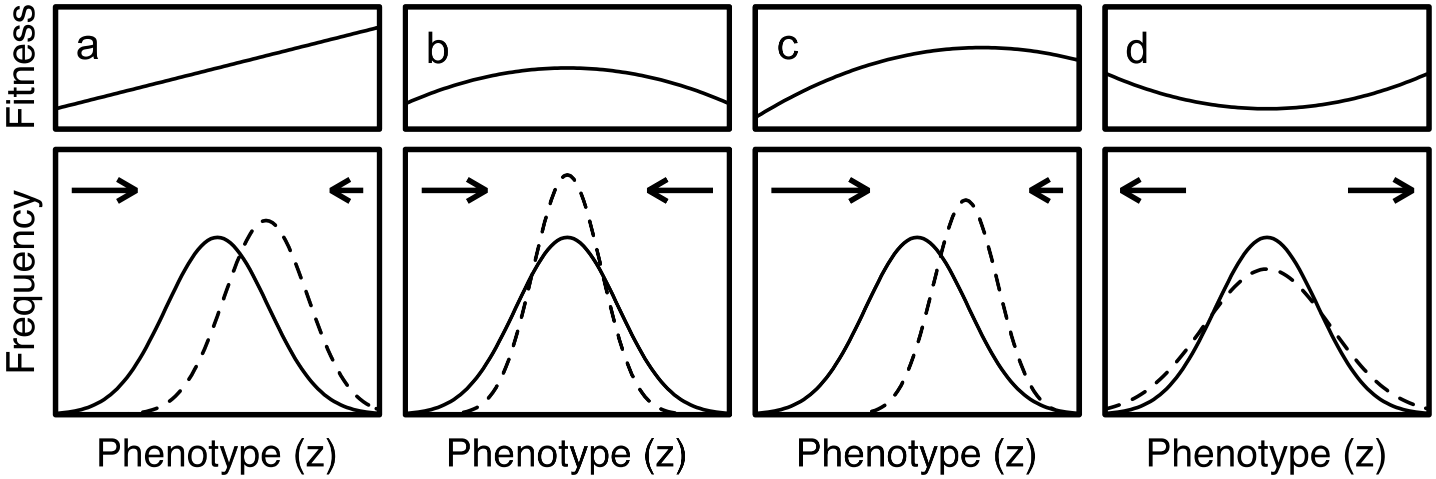
- Phenotype
  - calculated from QTL effects
  - **additive effects** are central (breeding value)
  - non-additive effects can also be modeled
    - dominance, epistasis
  - environmental noise can be added
  - phenotypic plasticity can be included
  - the final result: a **phenotypic trait value**

Quantitative Traits

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

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 a function of phenotype
- Fitness effects are assigned to individuals

Modeling Quantitative Traits

- QTL mutations:
  - have an additive effect size
  - do *not* have a selection coefficient
- Therefore:
  - **repurpose** the `selectionCoeff` property to hold the effect size!

Modeling Quantitative Traits

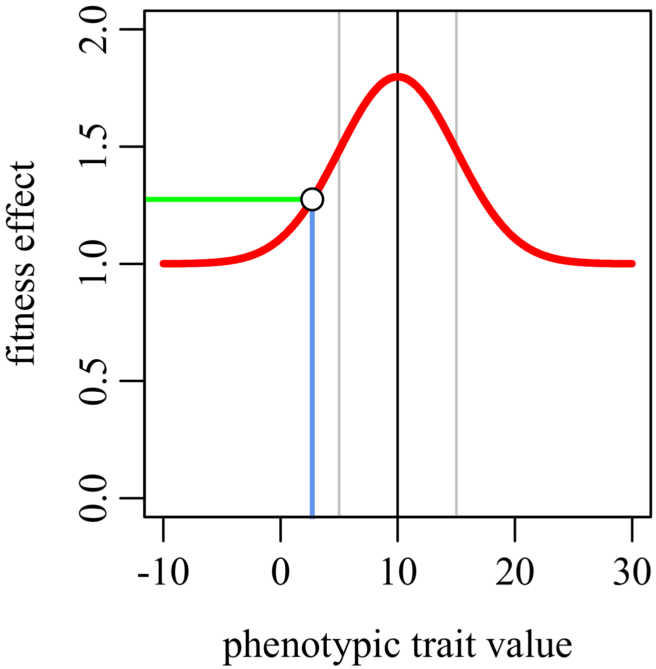
- QTL mutation types:
  - need a distribution of additive effect sizes
  - do *not* have a DFE (no selection coefficients)
- Therefore:
  - **repurpose** their DFE to instead be a distribution of additive effect sizes!
  - made neutral with a `fitness()` callback
  - use `convertToSubstitution=F`

Modeling Quantitative Traits

- Phenotypic trait values:
  - calculated in script
  - usually with `sumOfMutationsOfType()`
  - can be remembered in `tag` if needed
- Fitness:
  - also calculated in script
  - uses a fitness function (often `dnorm()`)
  - fitness effects applied with `fitnessScaling`

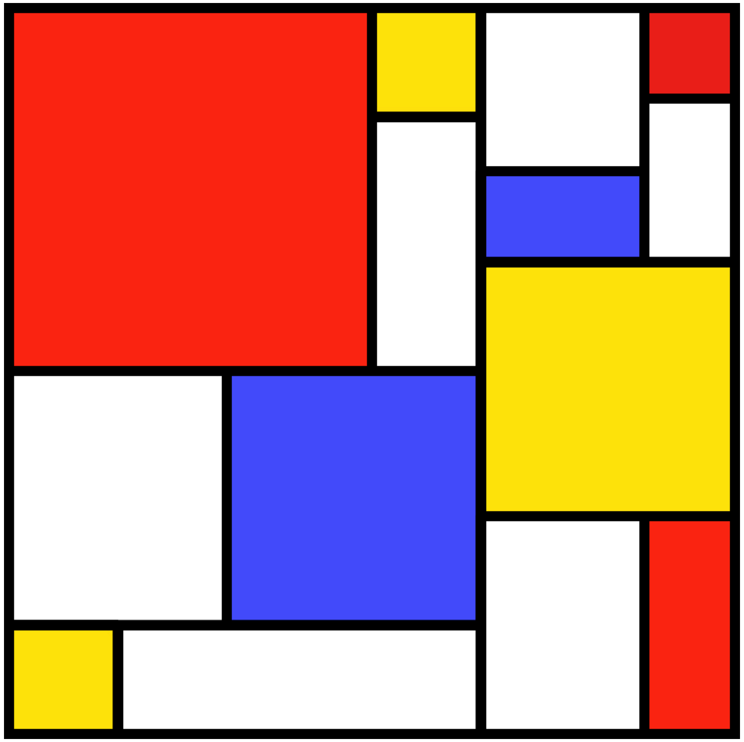
Why `dnorm()`?

- `dnorm()` provides a Gaussian fitness function
- Often good for modeling stabilizing selection
- Relatively neutral (skewness, kurtosis)



The Big Picture

- QTL mutations have an effect size
  - kept in `selectionCoeff`
- Phenotype is the sum of all effects
  - calculated with `sumOfMutationsOfType()`
- Fitness is a function of phenotype
  - for stabilizing selection, using `dnorm()`
- Fitness effects are assigned to individuals
  - using the `fitnessScaling` property



SLiM Workshop Exercise #13