Simulations in Plant Breeding

An emphasis on AlphaSimR

Matheus Dalsente Krause
Major advisors: Drs. William Beavis and Asheesh Singh

Ph.D. Student in Plant Breeding
Department of Agronomy - Iowa State University

February 7, 2022

Outline

Why do we use simulations in plant breeding?

Plant breeding is a complex system

[Podlich and Cooper, 1998, Sun et al., 2011]

"To investigate the implications of relaxing assumptions that are commonly made in quantitative genetics." These assumptions (usually) are not met in reality

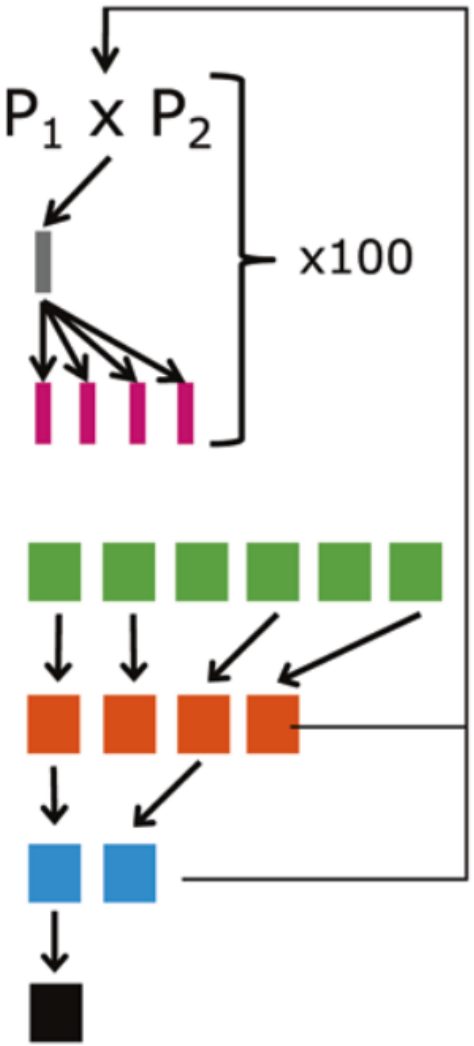
[Li et al., 2012]

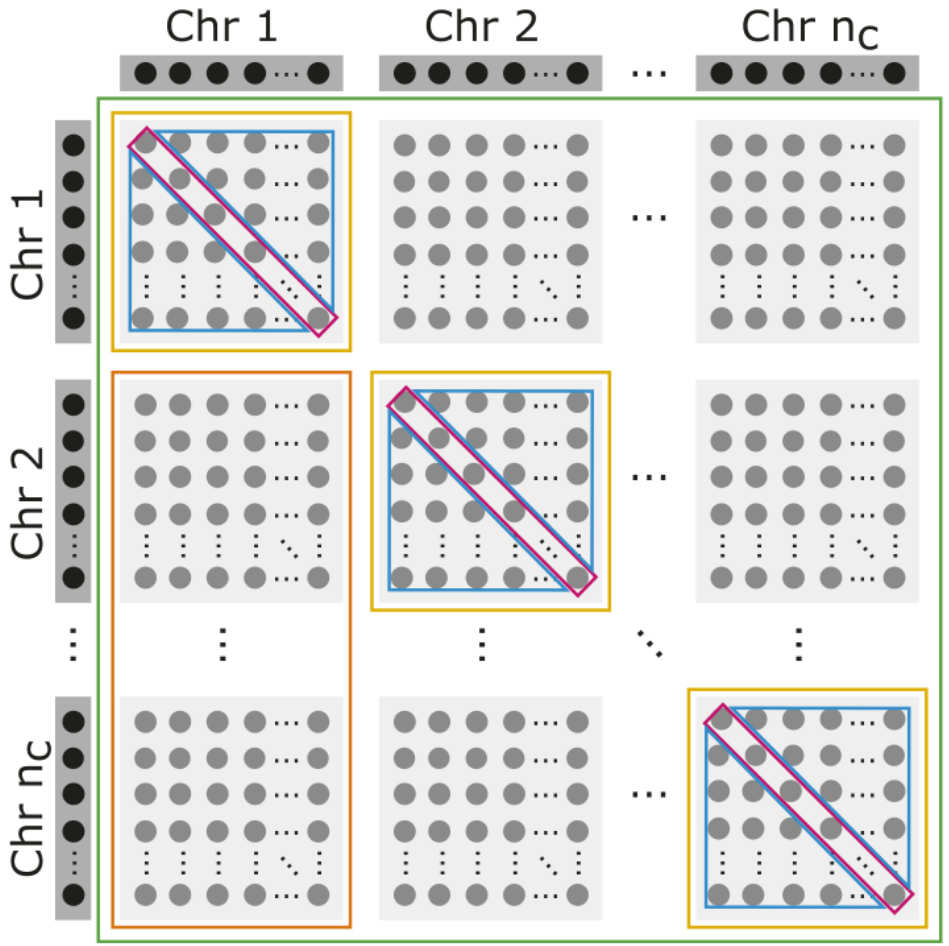
"... validation of theories but also guidelines for empirical experiments... Computer simulation can lay out the breeding process in silico and identify optimal candidates for various scenarios; empirical validation can then follow." Impact of new tools

[Faux et al., 2016]

"Simulation is the ideal tool to develop optimal breeding strategies while assessing costs and benefits."

Some examples [Lara et al., 2022]





Some examples [Aguate et al., 2019]

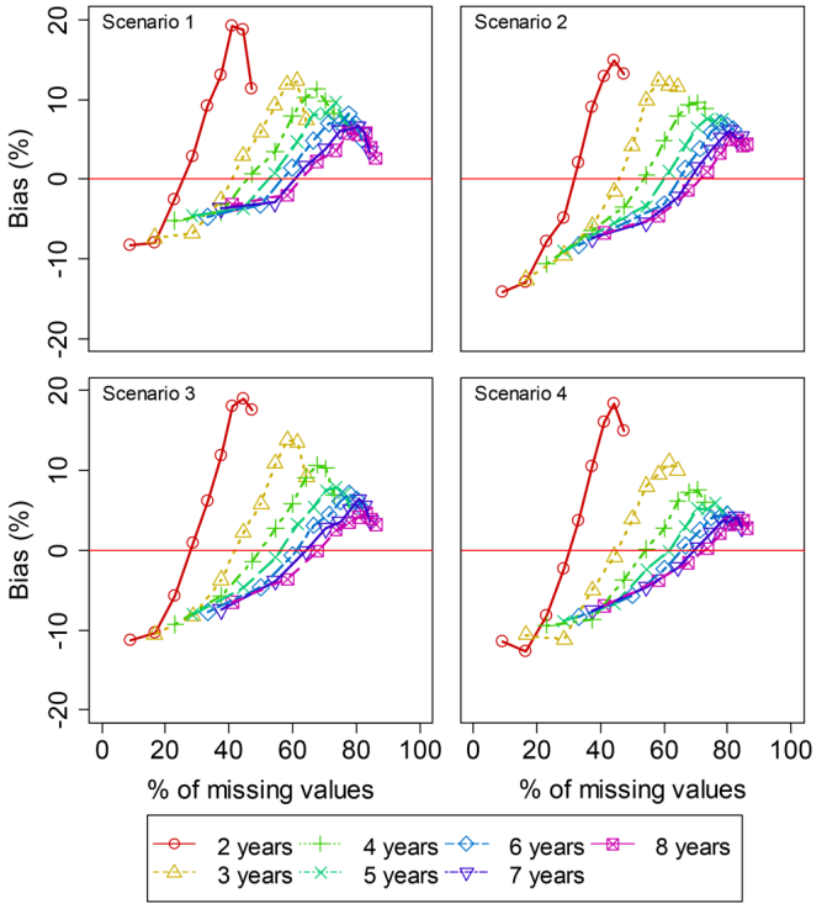


Fig. 1. Average bias (%) of genotypic variance estimates in relation

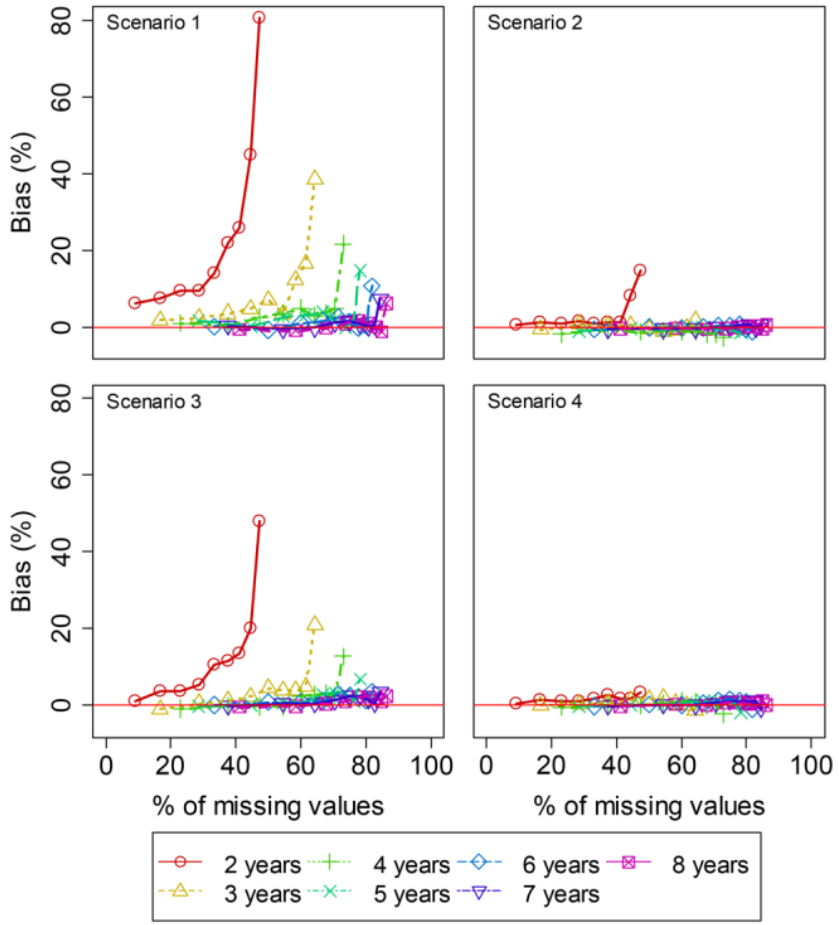
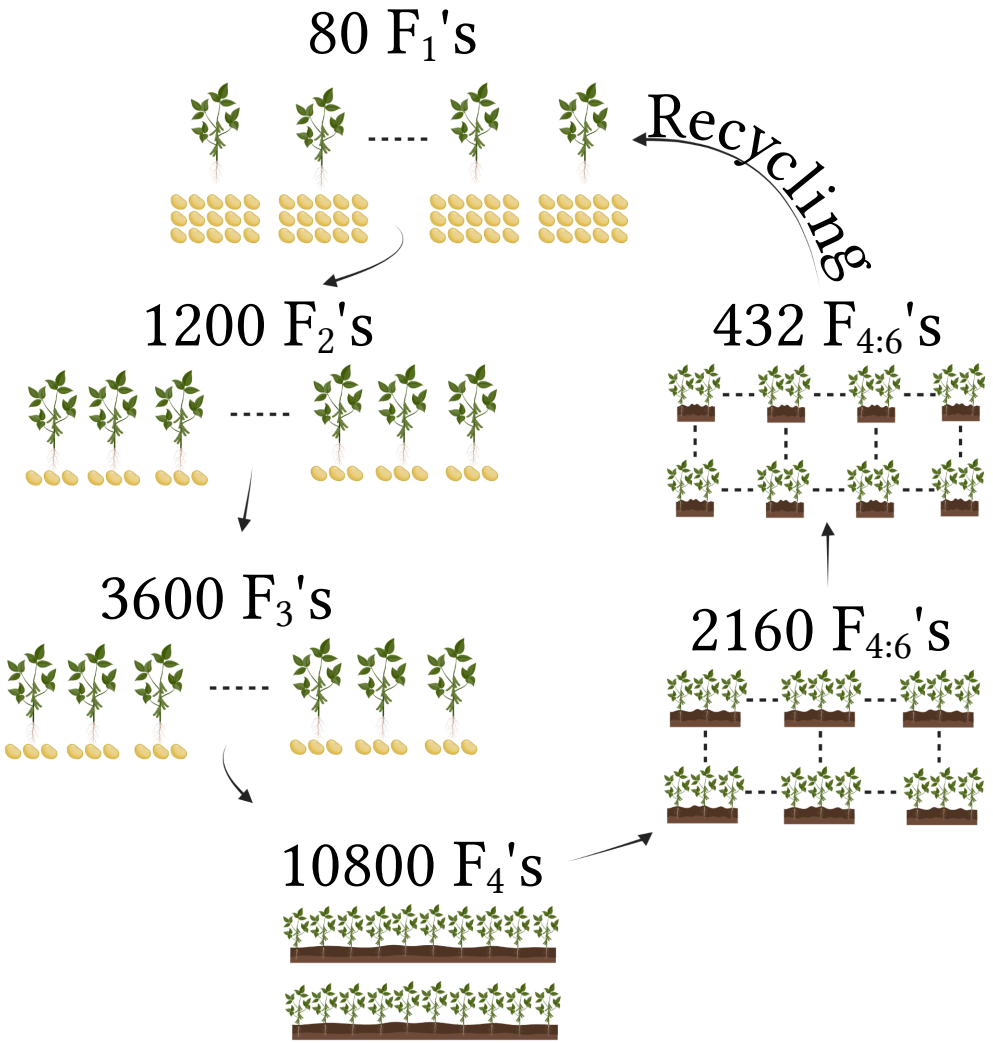
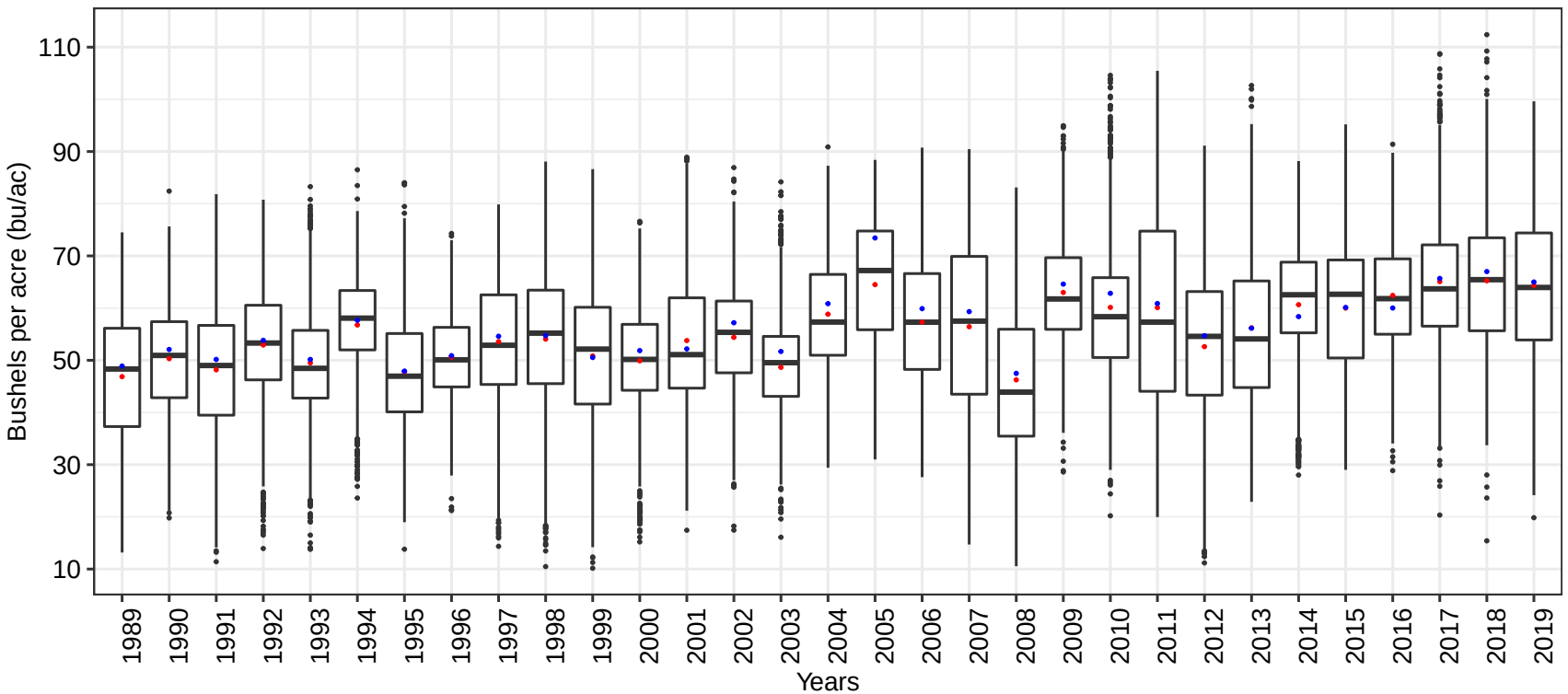


Fig. 2. Relative bias (%) of genotype × location interaction

Some examples [Part of my research project]



Some examples [Part of my research project]



Learn

Why bother with R?

Statistical analysis

Data management

Publishers Love R - Amazing graphs

It offers powerful simulation packages!

Learn

Why bother with R?

Statistical analysis

Data management

Publishers Love R - Amazing graphs

It offers powerful simulation packages!

Learn

We are going to use functions and loops

```r
# A toy example of a function
power <- function(x, y = 2) {
        result <- x^y
        return(result)
}

# 3 raised to the power 2 is 9
power(3)

## [1] 9

# 3 raised to the power 5 is 9
power(3, y = 5)

## [1] 243
```

Learn

We are going to use functions and loops

```
# A vector of numbers to be used
my_numbers <- c(2, 3, 4, 5, 6, 7)

# Power function applied to a vector, when y = 2
power(my_numbers)

## [1]  4  9 16 25 36 49

# Power function applied to a vector, when y = 5
power(my_numbers, 5)

## [1]    32   243  1024  3125  7776 16807
```

Learn

We are going to use functions and loops

Suppose we have a matrix of markers:

SNP/ID	Geno A	Geno B
SNP 1	0	2
SNP 2	1	2

```
# The marker matrix
M <- matrix(c(0, 2, 1, 2), 2, 2, byrow = TRUE)
M
```

```
##      [,1] [,2]
## [1,]    0    2
## [2,]    1    2
```

Learn

We are going to use functions and loops

Suppose we are interested in the number of copies of the reference allele:

```
nr <- nrow(M)

for(i in 1:nr){
        print(sum(M[i,]))
}

## [1] 2
## [1] 3

# Alternative ways
apply(M, 1, sum)

## [1] 2 3
```

Learn

Installing needed packages for today

```
install.packages(c("AlphaSimR", "ggplot2"))

#check.packages <- function(pkg){
#new.pkg <- pkg[!(pkg%in%installed.packages()[,"Package"])]
#if (length(new.pkg))
#install.packages(new.pkg, dependencies = TRUE)
#sapply(pkg, require, character.only = TRUE)
#}
#packages<-c("AlphaSimR", "ggplot2", "AGHmatrix")
#check.packages(packages)
```

Basic simulation theory | P = G + E

$$P_{ijk} = \mu + E_j + G_i + \epsilon_{ijk}$$

Geno	Rep	Env	μ	E_j	G_i	ϵ_{ijk}
G1	1	Ames				
G2	1	Ames				
G3	1	Ames				
G1	2	Ames				
G2	2	Ames				
G3	2	Ames				
G1	1	Boone				
G2	1	Boone				
G3	1	Boone				
G1	2	Boone				
G2	2	Boone				
G3	2	Boone				

Basic simulation theory | P = G + E

$$P_{ijk} = \mu + E_j + G_i + \epsilon_{ijk}$$

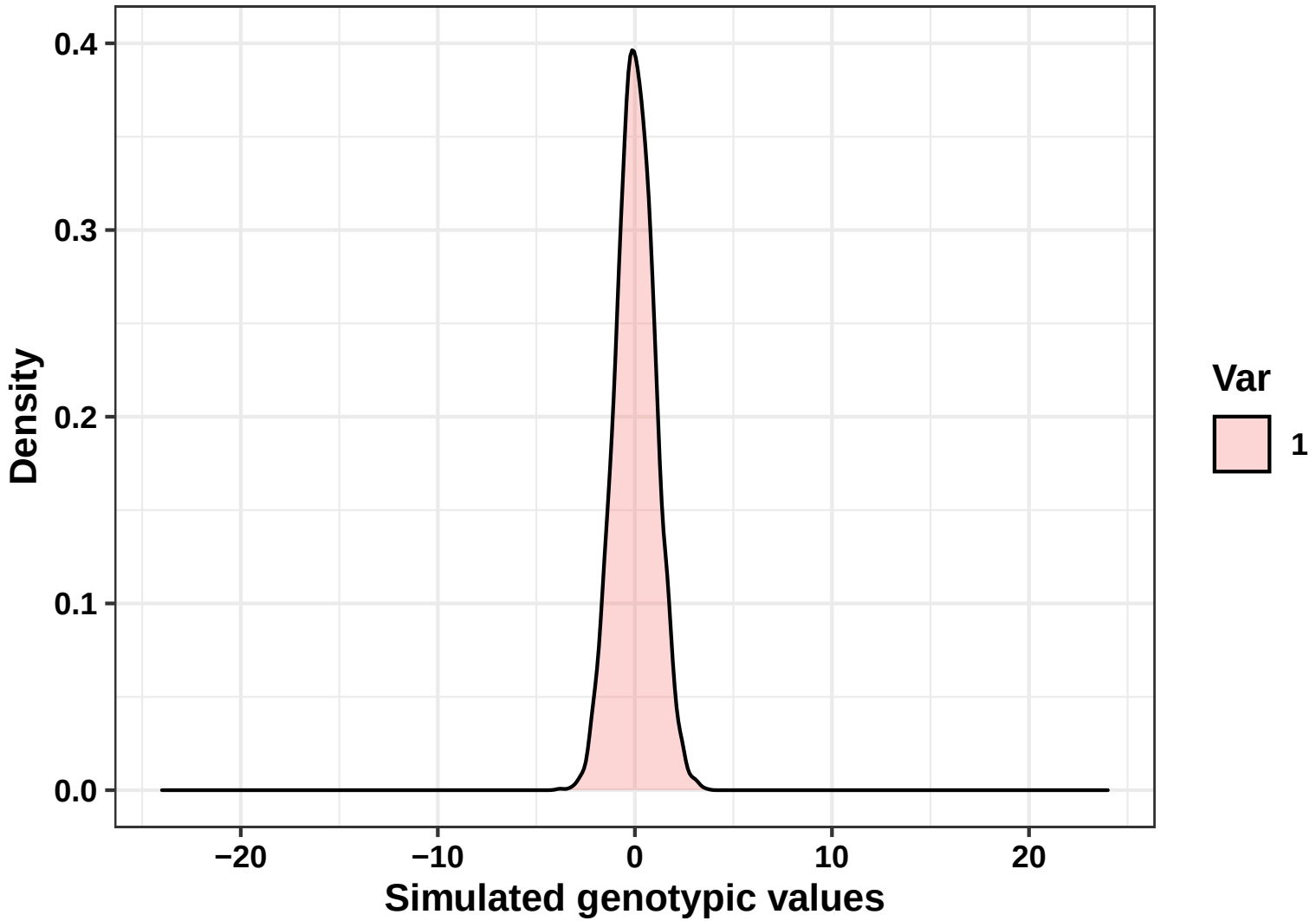
Geno	Rep	Env	μ	E_j	G_i	ϵ_{ijk}
G1	1	Ames	100			
G2	1	Ames	100			
G3	1	Ames	100			
G1	2	Ames	100			
G2	2	Ames	100			
G3	2	Ames	100			
G1	1	Boone	100			
G2	1	Boone	100			
G3	1	Boone	100			
G1	2	Boone	100			
G2	2	Boone	100			
G3	2	Boone	100			

Basic simulation theory | P = G + E

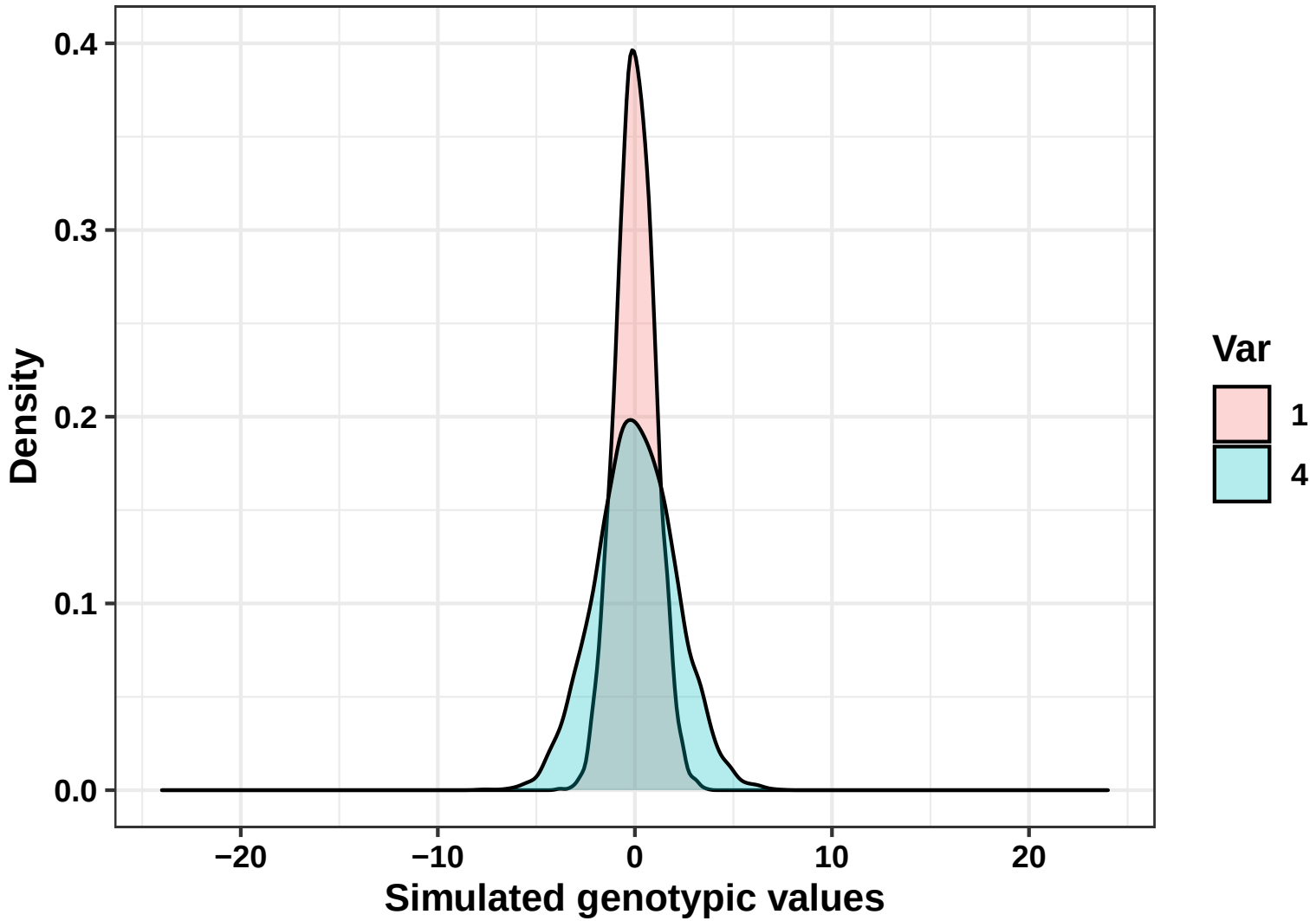
$$P_{ijk} = \mu + E_j + G_i + \epsilon_{ijk}$$

Geno	Rep	Env	μ	E_j	G_i	ϵ_{ijk}
G1	1	Ames	100	+20		
G2	1	Ames	100	+20		
G3	1	Ames	100	+20		
G1	2	Ames	100	+20		
G2	2	Ames	100	+20		
G3	2	Ames	100	+20		
G1	1	Boone	100	-15		
G2	1	Boone	100	-15		
G3	1	Boone	100	-15		
G1	2	Boone	100	-15		
G2	2	Boone	100	-15		
G3	2	Boone	100	-15		

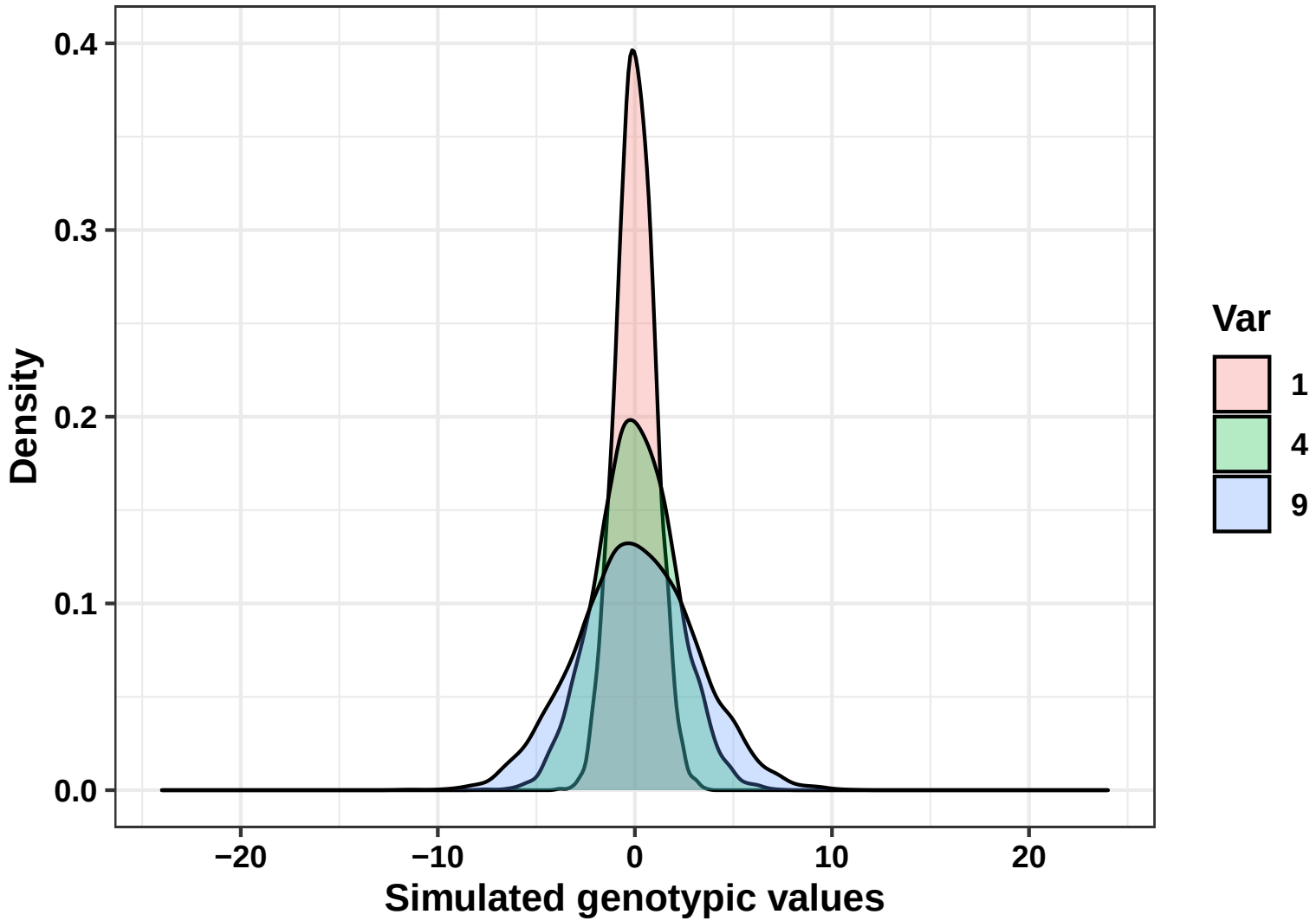
Basic simulation theory | $G_i \overset{i.i.d}{\sim} N(\mu = 0, \sigma^2_G)$



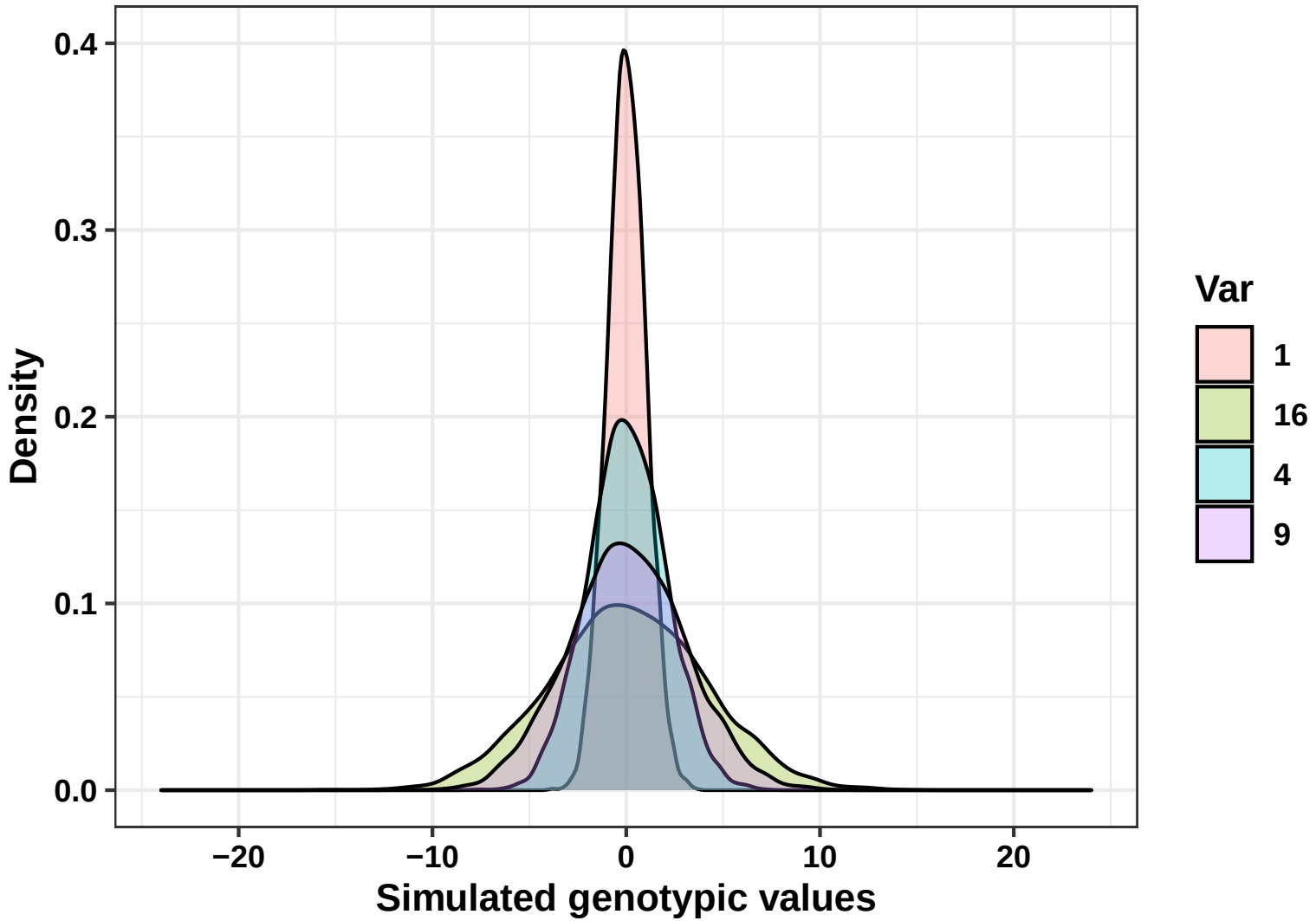
Basic simulation theory | $G_i \overset{i.i.d}{\sim} N(\mu = 0, \sigma^2_G)$



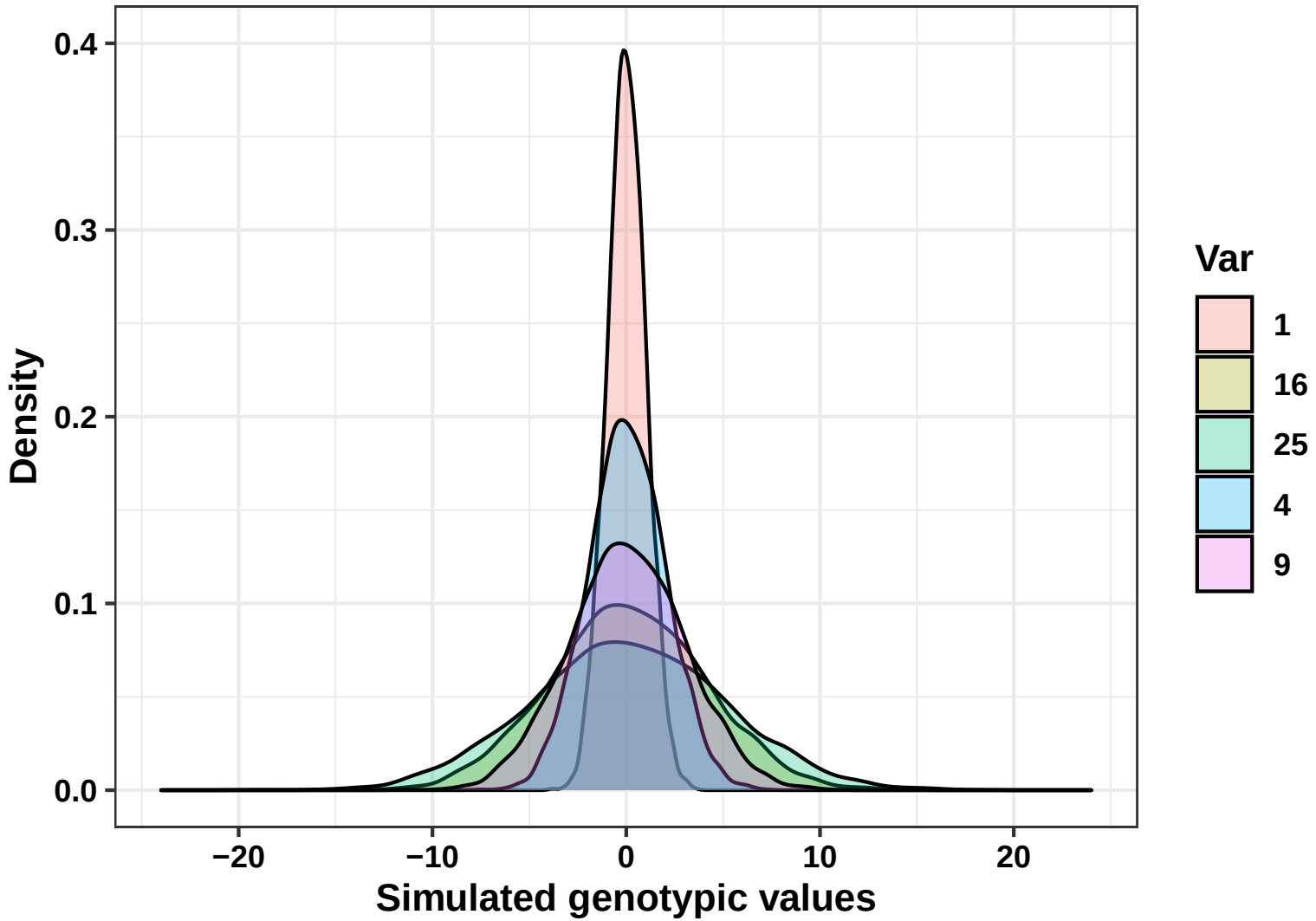
Basic simulation theory | $G_i \overset{i.i.d}{\sim} N(\mu = 0, \sigma^2_G)$



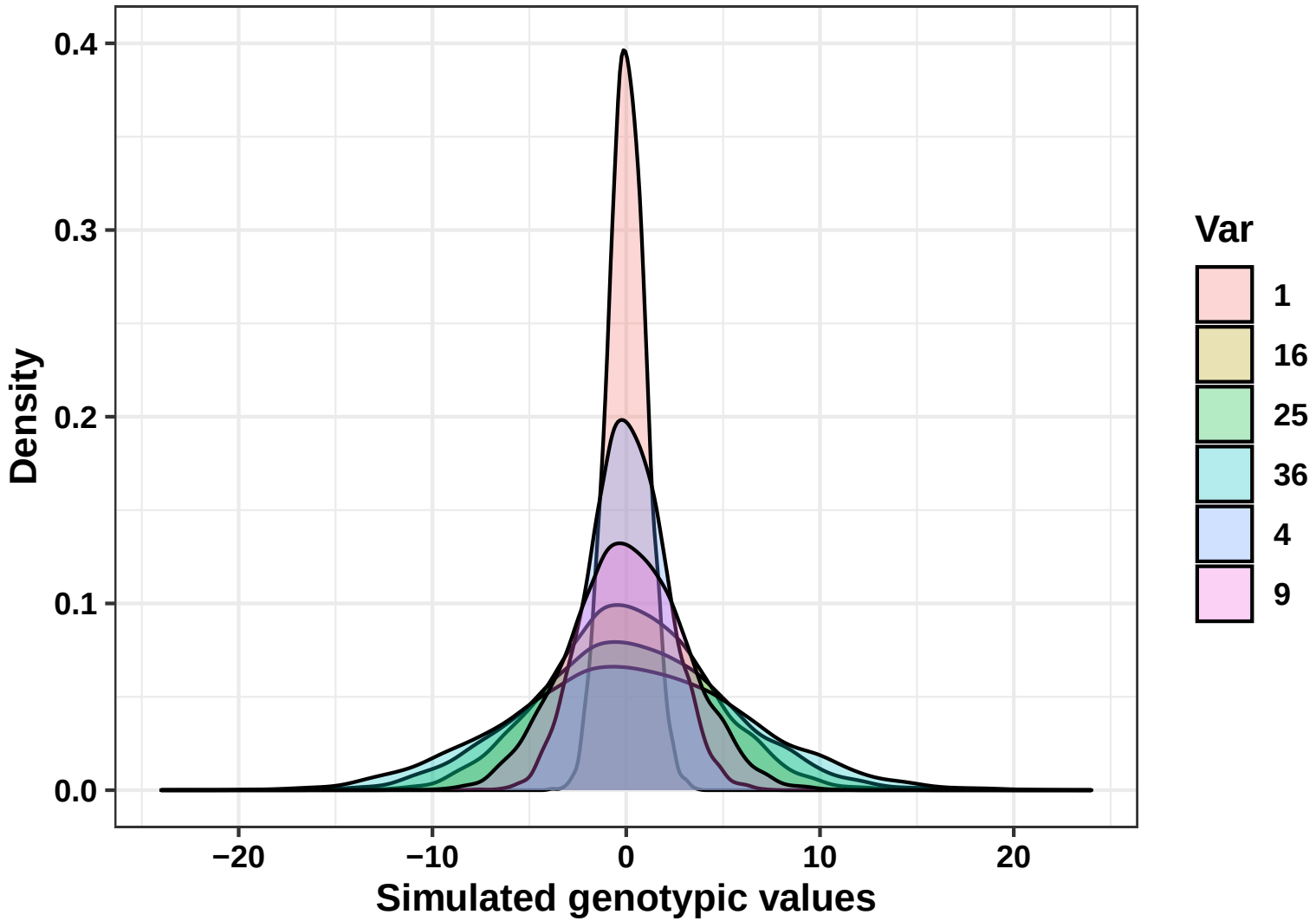
Basic simulation theory | $G_i \overset{i.i.d}{\sim} N(\mu = 0, \sigma^2_G)$



Basic simulation theory | $G_i \overset{i.i.d}{\sim} N(\mu = 0, \sigma^2_G)$



Basic simulation theory | $G_i \overset{i.i.d}{\sim} N(\mu = 0, \sigma^2_G)$



Basic simulation theory | $G_i \overset{i.i.d}{\sim} N(\mu = 0, \sigma^2_G)$

```r
# Sampling truth genotypic values
set.seed(8)
GV <- qnorm(runif(3), 0, 10) # inverse CDF
round(GV,2)

## [1] -0.85 -8.14  8.40
```

Geno	Rep	Env	μ	E_j	G_i	ϵ_{ijk}
G1	1	Ames	100	+20	-0.85	
G2	1	Ames	100	+20	-8.14	
G3	1	Ames	100	+20	8.40	
⋮	⋮	⋮	⋮	⋮		
G3	2	Boone	100	-15	8.40	

Basic simulation theory | $\epsilon_{ijk} \overset{i.i.d}{\sim} N(\mu = 0, \sigma^2_\epsilon = 49)$

```
set.seed(9) ; residual <- qnorm(runif(12), 0, 7)
```

Geno	Rep	Env	μ	E_j	G_i	ϵ_{ijk}
G1	1	Ames	100	+20	-0.85	-5.37
G2	1	Ames	100	+20	-8.14	-13.81
G3	1	Ames	100	+20	8.40	-5.72
G1	2	Ames	100	+20	-0.85	-5.51
G2	2	Ames	100	+20	-8.14	-0.99
G3	2	Ames	100	+20	8.40	-7.75
G1	1	Boone	100	-15	-0.85	-1.94
G2	1	Boone	100	-15	-8.14	-2.34
G3	1	Boone	100	-15	8.40	3.05
G1	2	Boone	100	-15	-0.85	16.90
G2	2	Boone	100	-15	-8.14	-8.31
G3	2	Boone	100	-15	8.40	-16.72

Basic simulation theory | MET Model

```
data <- data.frame(geno = rep(c('G1','G2','G3'), 4),
                   env = rep(c('Ames','Boone'), each=6),
                   rep = rep(rep(c(1,2), each = 3),2),
                   y = c(100+20-0.85-5.37,
                         100+20-8.14-13.81,
                         100+20+8.40-5.72,
                         100+20-0.85-5.51,
                         100+20-8.14-0.99,
                         100+20+8.40-7.75,
                         100-15-0.85-1.94,
                         100-15-8.14-2.34,
                         100-15+8.40+3.05,
                         100-15-0.85+16.90,
                         100-15-8.14-8.31,
                         100-15+8.40-16.72))
```

Basic simulation theory | MET Model

```
##    geno   env rep      y
## 1    G1  Ames   1 113.78
## 2    G2  Ames   1  98.05
## 3    G3  Ames   1 122.68
## 4    G1  Ames   2 113.64
## 5    G2  Ames   2 110.87
## 6    G3  Ames   2 120.65
## 7    G1 Boone   1  82.21
## 8    G2 Boone   1  74.52
## 9    G3 Boone   1  96.45
## 10   G1 Boone   2 101.05
## 11   G2 Boone   2  68.55
## 12   G3 Boone   2  76.68
```

Basic simulation theory | MET Model

```
data$geno <- as.factor(data$geno)
data$env <- as.factor(data$env)
model <- lm(y ~ env + geno + env:geno, data = data)
anova(model)

## Analysis of Variance Table
##
## Response: y
##           Df  Sum Sq Mean Sq F value   Pr(>F)
## env        1 2706.30 2706.30 34.1873 0.001104 **
## geno       2  636.19  318.10  4.0183 0.078102 .
## env:geno   2   97.29   48.64  0.6145 0.571769
## Residuals  6  474.97   79.16
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
```

AlphaSimR

R package developed by **AlphaGenes** - Roslin Institute (University of Edinburgh)

- Used for stochastic simulations of breeding programs
- Contained is a wide range of functions for modeling common tasks in a breeding program, such as selection and crossing
- It uses the Markovian Coalescent Simulator [Chen et al., 2009]
- Interface R and C++: *Rcpp*, *RcppArmadillo*
- R ($\geq$ 3.3.0)
- Maintainer: **Dr.Chris Gaynor**
- Diploid individuals with biallelic loci

[Faux et al., 2016, Gaynor et al., 2021]

AlphaSimR - Simulation of traits

ADEG

Additive + Dominance + Epistatic + GE effects

$$GV(x, w) = \mu + A(x) + D(x) + E(x) + G(x, w) \quad (1)$$

where:

- $GV(x, w)$ represents an individual's genetic value
- x represents a vector of QTL genotype dosages
  - number of copies of the "1" allele at a locus (all biallelic)
- w represents an environmental covariate
- A, AD, AE, AG, ADE, ADG, AEG, and ADEG

AlphaSimR - Simulation of traits

ADEG

Additive + Dominance + Epistatic + GE effects

$$GV(x, w) = \mu + A(x) + D(x) + E(x) + G(x, w) \quad (1)$$

where:

- $GV(x, w)$ represents an individual's genetic value
- x represents a vector of QTL genotype dosages
  - number of copies of the "1" allele at a locus (all biallelic)
- w represents an environmental covariate
- A, AD, AE, AG, ADE, ADG, AEG, and ADEG

AlphaSimR - Splitting $GV(x, w)$

1. μ: intercept (trait mean); it **isn't** a function $\Longrightarrow$ user specified
2. $A(x)$: the function for additive effects

$$A(x) = \sum ax_A \tag{2}$$

where:

- a is the additive effect of the QTL (stage 1). Normal or Gamma

AlphaSimR - Splitting $GV(x, w)$

1. μ: intercept (trait mean); it **isn't** a function $\Longrightarrow$ user specified
2. $A(x)$: the function for additive effects

$$A(x) = \sum ax_A \tag{2}$$

where:

- a is the additive effect of the QTL (stage 1). Normal or Gamma

$A(x) = \sum ax_A$, the x_A part, stage 2

1. μ: intercept (trait mean); it **isn't** a function $\Longrightarrow$ user specified
2. $A(x)$: the function for additive effects

$$A(x) = \sum ax_A \tag{3}$$

where:

- a is the additive effect of the QTL (stage 1). Normal or Gamma
- x_A is the scaled additive dosage (00, 01, 10, 11) to achieve a user specified genetic variance. It uses all variance components; additive or total, it depends on the trait (stage 2). Scaling matches real-world estimates!

AlphaSimR - Splitting $GV(x, w)$

3. $D(x)$: the function for dominance effects

$$D(x) = \sum dx_D \tag{4}$$

$$d = \delta|a| \tag{5}$$

where:

- d is the dominance effect of a given QTL, which is a function of its additive effect (a) and a dominance degree δ

AlphaSimR - Splitting $GV(x, w)$

3. $D(x)$: the function for dominance effects

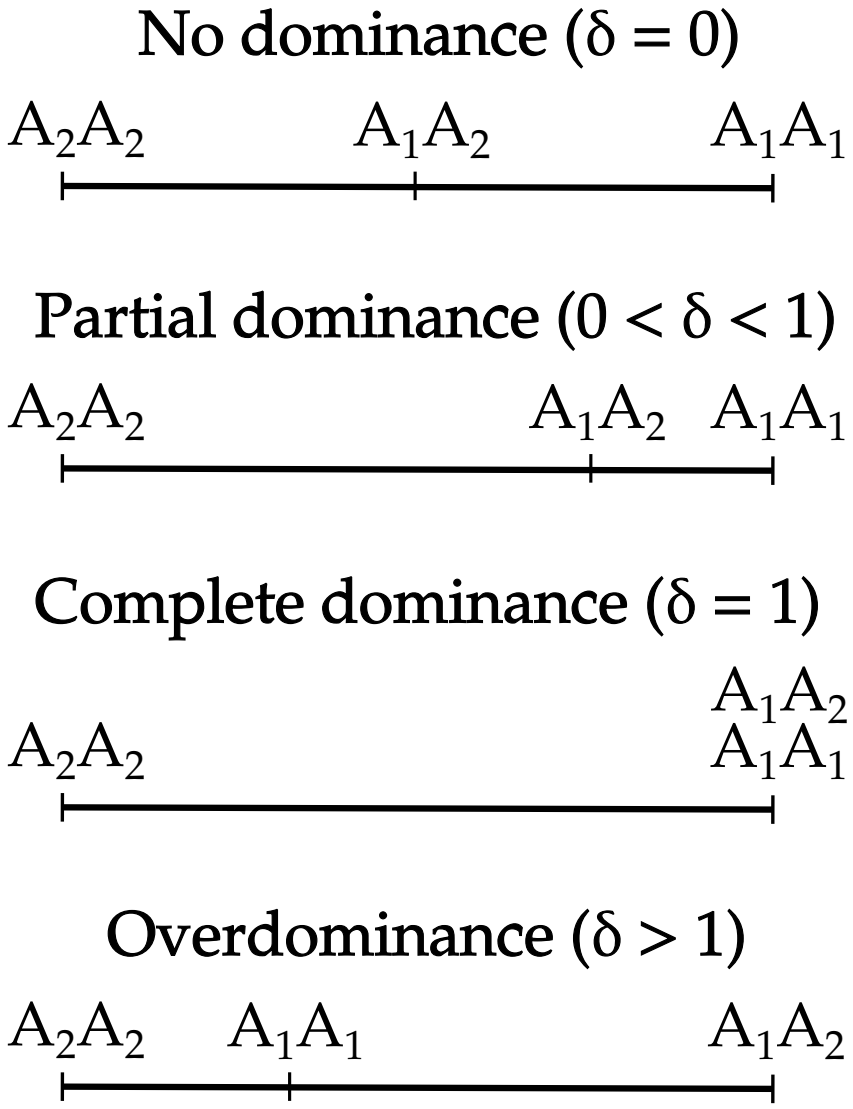
$$D(x) = \sum dx_D \tag{6}$$

$$d = \delta|a| \tag{7}$$

where:

- d is the dominance effect of a given QTL, which is a function of its additive effect ($|a|$) and a dominance degree δ
- $\delta \sim N(\mu_D, \sigma^2_D)$, user specified (stage 1)
- x_D is the scaled dominance dosage (00, 01, 10, 11) to achieve a user specified genetic variance. It uses all variance components (stage 2)

For δ, the dominance degree, the usual interpretation



Looking at the function

Usage:

```
SimParam$addTraitAD(
  nQtlPerChr,
  mean = 0,
  var = 1,
  meanDD = 0,
  varDD = 0,
  corA = NULL,
  corDD = NULL,
  useVarA = TRUE,
  gamma = FALSE,
  shape = 1,
  force = FALSE
)
```

```
addTraitAD = function(nQtlPerChr,mean=0,var=1,meanDD=0,
                      varDD=0,corA=NULL,corDD=NULL,useVarA=TRUE,
                      gamma=FALSE,shape=1,force=FALSE){
  if(!force){
    private$.isRunning()
  }
  if(length(nQtlPerChr)==1){
    nQtlPerChr = rep(nQtlPerChr,self$nChr)
  }
  nTraits = length(mean)
  if(length(meanDD)==1) meanDD = rep(meanDD,nTraits)
  if(length(varDD)==1) varDD = rep(varDD,nTraits)
  if(length(gamma)==1) gamma = rep(gamma,nTraits)
  if(length(shape)==1) shape = rep(shape,nTraits)
  if(is.null(corA)) corA=diag(nTraits)
  if(is.null(corDD)) corDD=diag(nTraits)
  stopifnot(length(mean)==length(var),
            isSymmetric(corA),
            isSymmetric(corDD),
            length(mean)==nrow(corA))
  qtlLoci = private$.pickLoci(nQtlPerChr)
  addEff = sampAddEff(qtlLoci=qtlLoci,nTraits=nTraits,
                      corr=corA,gamma=gamma,shape=shape)
  domEff = sampDomEff(qtlLoci=qtlLoci,nTraits=nTraits,addEff=addEff,
                      corDD=corDD,meanDD=meanDD,varDD=varDD)
  for(i in 1:nTraits){
    trait = new("TraitAD",
                qtlLoci,
                addEff=addEff[,i],
                domEff=domEff[,i],
                intercept=0)
    tmp = calcGenParam(trait, self$founderPop,
                       self$nThreads)
    if(useVarA){
      scale = sqrt(var[i])/sqrt(popVar(tmp$bv)[1])
    }else{
      scale = sqrt(var[i])/sqrt(popVar(tmp$gv)[1])
    }
    trait@addEff = trait@addEff*scale
    trait@domEff = trait@domEff*scale
    trait@intercept = mean[i]-mean(tmp$gv*scale)
    if(useVarA){
      private$.addTrait(trait,var[i],popVar(tmp$gv*scale)[1])
    }else{
      private$.addTrait(trait,popVar(tmp$bv*scale)[1],var[i])
    }
  }
  invisible(self)
},
```

AlphaSimR - Splitting $GV(x, w)$

4. $E(x)$: a **simplified** function for epistatic effects (add $\times$ add)

$$E(x) = \sum ex_{A_1}x_{A_2} \tag{8}$$

where:

- e is the epistatic effects, sampled from normal or gamma
- x_{A_1} is the scaled additive dosage for the first locus in a pair
- x_{A_2} is the scaled additive dosage for the second locus in a pair
- The epistatic variance is set according to the ratio $\frac{\text{add}\times\text{add}}{\text{add}}$, which is user specified

AlphaSimR - Splitting $GV(x, w)$

5. $G(x, w)$: the function for genotype-by-environment (GE) effects

$$G(x, w) = wb(x) \tag{9}$$

$$b(x) = \mu_G + \sum gx_A \tag{10}$$

where:

- w is an environmental covariate, where $w \sim N(0, \sigma^2_E = 1)$
- $b(x)$ is a genotype specific slope
- μ_G is the intercept (zero when $\sigma^2_E = 1$, one when $\sigma^2_E \neq 1$)
- g is a GE effect and x_A the scaled additive dosage, relative to the desired σ^2_{GE}. When $\sigma^2_E \neq 1$, the original σ^2_{GE} is scaled to $\frac{\sigma^2_{GE}}{\sigma^2_E}$. Note $b(x)$ is an additive trait

Hands-on example

Recurrent selection in an open pollinating maize population using different selection methods

1. True genetic value
2. Phenotype
3. Randomly selecting individuals
4. Estimated Breeding Value (EBV)
5. Selecting against the trait

Hands-on example

AlphaSimR uses the MaCS software to create founder haplotypes

```
library(AlphaSimR)

# MaCS generates whole-chromosome founder haplotypes
founderPop <- runMacs(nInd = 20,
                      nChr = 10,
                      segSites = 1000, #segregating loci
                      species = 'MAIZE',
                      inbred = FALSE,
                      ploidy = 2L,
                      nThreads = 3)

SP <- SimParam$new(founderPop)
```

Hands-on example

Let's set simulation parameters

```
SP$addTraitA(nQtlPerChr = 500, mean = 0,
                 gamma = FALSE, var =  1)

SP$addSnpChip(nSnpPerChr = 200)

SP$setVarE(h2=0.4)

# sampling haplotypes from founders
pop <- newPop(founderPop, simParam=SP)

for(i in 1:100){
        pop <- selectOP(pop = pop, nInd = 20,
                        nSeeds = 10, use = 'rand',
                        probSelf = 0.01, simParam=SP)
        }
```

Hands-on example

AlphaSimR uses the MaCS software to create founder haplotypes

```
pop

## An object of class "Pop"
## Ploidy: 2
## Individuals: 200
## Chromosomes: 10
## Loci: 10000
## Traits: 1
```

Hands-on example

AlphaSimR uses the MaCS software to create founder haplotypes

```
str(pop)

## Formal class 'Pop' [package "AlphaSimR"] with 18 slots
##   ..@ id     : chr [1:200] "19821" "19822" "19823" "19824" ...
##   ..@ iid    : int [1:200] 19821 19822 19823 19824 19825 19826
##   ..@ mother : chr [1:200] "19687" "19687" "19687" "19687" ...
##   ..@ father : chr [1:200] "19781" "19738" "19778" "19680" ...
##   ..@ sex    : chr [1:200] "H" "H" "H" "H" ...
##   ..@ nTraits: int 1
##   ..@ gv     : num [1:200, 1] -1.478 2.759 -0.946 -1.662 0.744
##   ..@ pheno  : num [1:200, 1] -2.644 1.14 -0.124 0.379 4.075 ..
##   ..@ ebv    : num[1:200, 0 ]
##   ..@ gxe    :List of 1
##   .. ..$ : NULL
##   ..@ fixEff : int [1:200] 1 1 1 1 1 1 1 1 1 1 ...
##   ..@ reps   : num [1:200] 1 1 1 1 1 1 1 1 1 1 ...
##   ..@ misc   :List of 200
```

Hands-on example

Let's create 5 copies of the same population

```
## Initial population - 5 scenarios
popGV <- newPop(pop, simParam=SP)
popPHENO <- newPop(pop, simParam=SP)
popRAND <- newPop(pop, simParam=SP)
popEBV <- newPop(pop, simParam=SP)
popMeanA <- newPop(pop, simParam=SP)

## Getting initial means
genMeanGV <- meanG(popGV)
genMeanPHENO <- meanG(popPHENO)
genMeanRAND <- meanG(popRAND)
genMeanEBV <- meanG(popEBV)
genMeanA <- meanG(popMeanA)
```

Hands-on example

50 cycles of recurrent selection

```
#### Selection based on the true genetic value
for(generation in 1:50){

        popGV = selectOP(pop = popGV,
                         nInd = 20,
                         pollenControl = TRUE,
                         nSeeds = 10,
                         use = "gv",
                         probSelf = 0.01,
                         nCrosses = 20)

        genMeanGV = c(genMeanGV, meanG(popGV))
}
```

New haplotypes are created by modeling genetic recombination during meiosis with gamma model [McPeek and Speed, 1995]

Hands-on example

50 cycles of recurrent selection

```
#### Selection based on the phenotype

for(generation in 1:50){

        popPHENO = setPheno(popPHENO, varE = 5)

        popPHENO = selectOP(pop = popPHENO,
                            nInd = 20,
                            pollenControl = TRUE,
                            nSeeds = 10,
                            use = "pheno",
                            probSelf = 0.01,
                            nCrosses = 20)

genMeanPHENO = c(genMeanPHENO, meanG(popPHENO))
}
```

Hands-on example

50 cycles of recurrent selection

```
#### Randomly selecting individuals
for(generation in 1:50){
popRAND = selectOP(pop = popRAND, nInd = 20, nSeeds = 10,
pollenControl = TRUE,use="rand", probSelf = 0.01,
nCrosses=20)
genMeanRAND = c(genMeanRAND, meanG(popRAND))

#### TOP false - selecting against the trait
for(generation in 1:50){
popMeanA = setPheno(popMeanA, varE = 5)
popMeanA = selectOP(pop = popMeanA, nInd = 20, nSeeds = 10,
pollenControl = TRUE,use = "pheno", probSelf = 0.01,
nCrosses = 20, selectTop = FALSE)
genMeanA = c(genMeanA, meanG(popMeanA))
}
```

Hands-on example

50 cycles of recurrent selection

```
#### Selection based on Estimated Breeding Value (EBV)

for(generation in 1:50){

        popEBV = setPheno(popEBV, varE = 5)

        rrBLUP <- RRBLUP(popEBV, simParam=SP)

        popEBV <- setEBV(popEBV, rrBLUP, simParam=SP)

        popEBV = selectOP(pop=popEBV, nInd = 20, use = "ebv",
                          pollenControl = TRUE, nSeeds = 10,
                          probSelf=0.01, nCrosses=20)

genMeanEBV = c(genMeanEBV, meanG(popEBV))
}
```

Hands-on example

Let's look at the markers

```
snp <- pullSnpGeno(popEBV, snpChip = 1)
snp[1:5,1:5]
```

```
##       SNP_1 SNP_2 SNP_3 SNP_4 SNP_5
## 60821     2     0     0     2     0
## 60822     2     0     0     2     0
## 60823     2     0     0     2     0
## 60824     2     0     0     2     0
## 60825     2     0     0     2     0
```

```
summary(as.factor(snp))
```

```
##      0      1      2
## 248586    294 151120
```

```
# G <- AGHmatrix::Gmatrix(snp, method = "VanRaden")
```

Hands-on example

Let's look at the markers

```
snp <- pullSnpGeno(popPHENO, snpChip = 1)
snp[1:5,1:5]

##       SNP_1 SNP_2 SNP_3 SNP_4 SNP_5
## 40821     0     0     0     2     0
## 40822     0     0     0     1     1
## 40823     0     0     0     2     0
## 40824     0     0     0     2     0
## 40825     0     0     0     0     2

summary(as.factor(snp))

##      0      1      2
## 242954  11656 145390

# G <- AGHmatrix::Gmatrix(snp, method = "VanRaden")
```

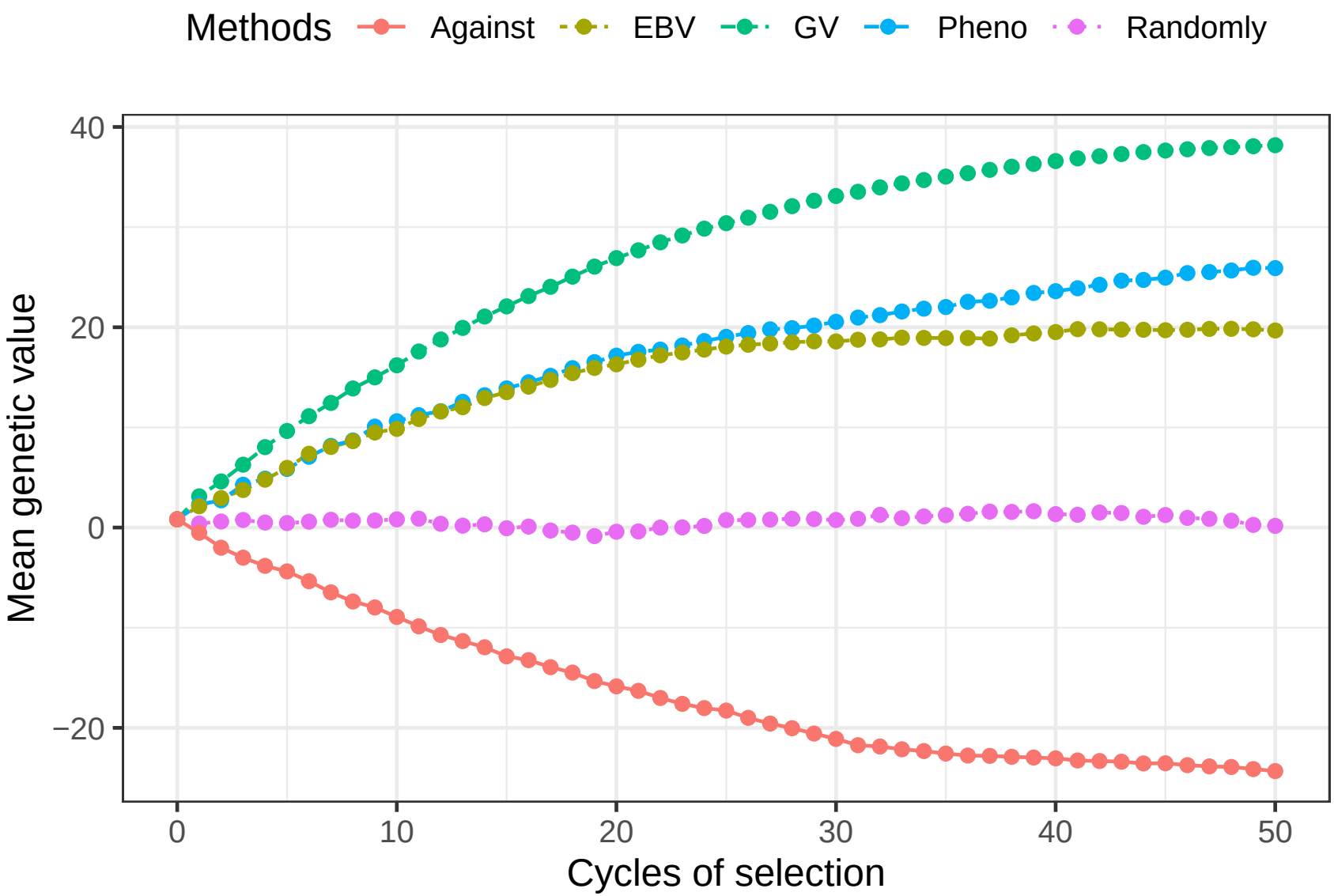
Hands-on example

Gathering the data

```
## all the data
results <- data.frame(
        Methods = rep(c("GV","Pheno","Randomly",
                        "EBV","Against"), each = 51),
        cycle = rep(0:50,5),
        means = c(genMeanGV,genMeanPHENO,genMeanRAND,
                  genMeanEBV,genMeanA))

## Plotting
library(ggplot2)
ggplot(results, aes(cycle, means, colour=Methods)) +
        geom_line(aes(linetype=Methods)) +
        geom_point()
```

Results - mean genetic values after 50 cycles of selection



Results - what about the genetic variance?

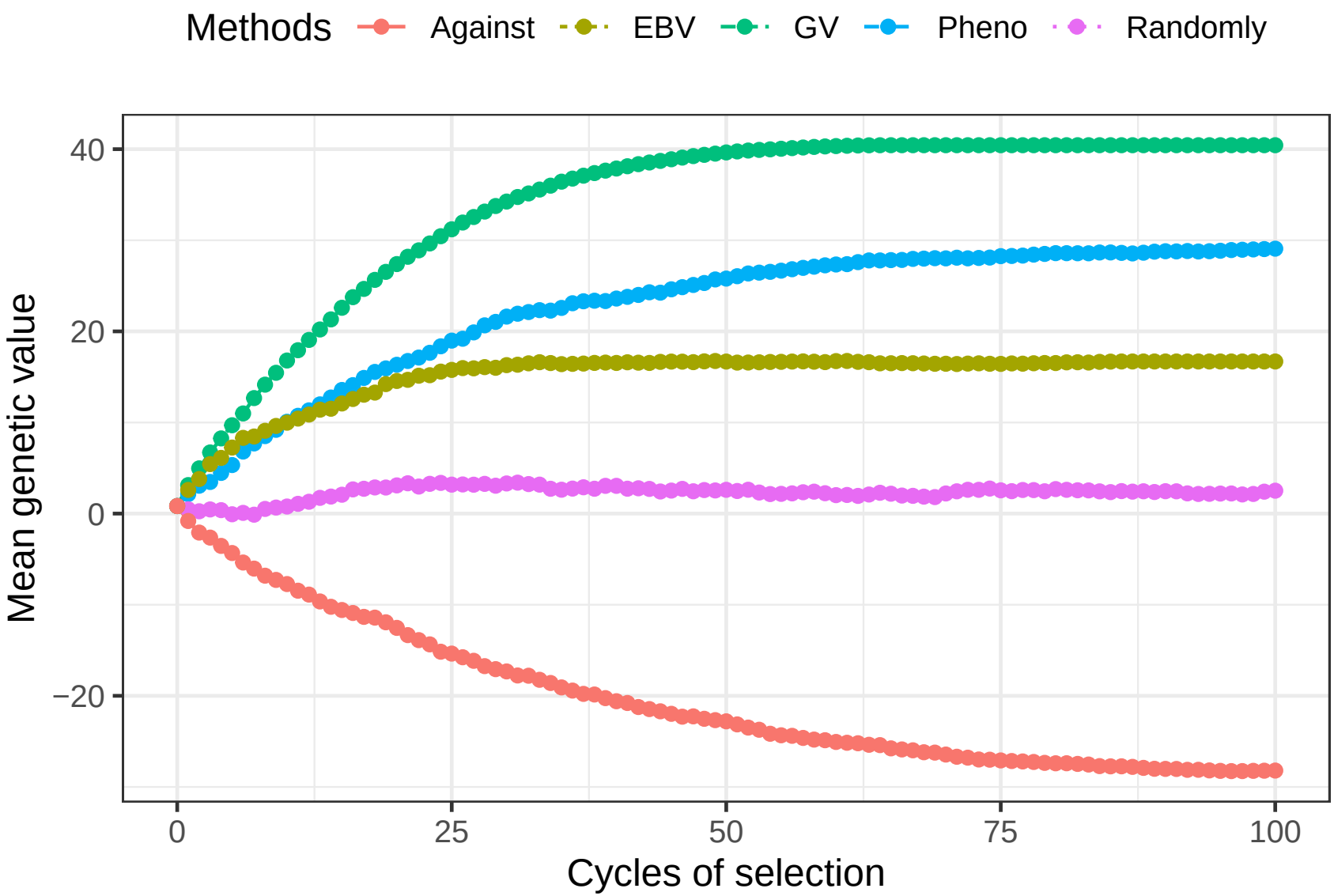
```
        varG(pop) ; varG(popMeanA) ; varG(popEBV)

##           [,1]
## [1,] 2.253115
##             [,1]
## [1,] 0.1291135
##              [,1]
## [1,] 0.02489582

        varG(popGV) ; varG(popPHENO) ; varG(popRAND)

##              [,1]
## [1,] 0.00279903
##             [,1]
## [1,] 0.1886122
##             [,1]
## [1,] 0.3652808
```

Results - mean genetic values after 100 cycles of selection



Final Remarks

To push the envelope, dig deeper

- The official repository, `https://cran.r-project.org/`
- AlphaSimR
- Plant breeding simulation workshop; QU-GENE
- R for Plant Breeders - Intro to R workshop
- Breeding scheme designer



Matheus Dalsente Krause
mdkrause@iastate.edu

References I

Aguate, F., Crossa, J., and Balzarini, M. (2019).

Effect of missing values on variance component estimates in multienvironment trials.
Crop Science, 59:508–517.

Chen, G. K., Marjoram, P., and Wall, J. D. (2009).

Fast and flexible simulation of dna sequence data.
Genome Research, 19:136–142.

Faux, A. M., Gorjanc, G., Gaynor, R. C., Battagin, M., Edwards, S. M., Wilson, D. L., Hearne, S. J., Gonen, S., and Hickey, J. M. (2016).

Alphasim: Software for breeding program simulation.
Plant Genome, 9:1–14.

Gaynor, R. C., Gorjanc, G., and Hickey, J. M. (2021).

Alphasimr: an r package for breeding program simulations.
G3 Genes|Genomes|Genetics, 11.

Lara, L. A. C., Pocrnic, I., de P. Oliveira, T., Gaynor, R. C., and Gorjanc, G. (2022).

Temporal and genomic analysis of additive genetic variance in breeding programmes.
Heredity, 128:21–32.

Li, X., Zhu, C., Wang, J., and Yu, J. (2012).

Computer simulation in plant breeding.

McPeek, M. S. and Speed, T. P. (1995).

Modeling interference in genetic recombination.
Genetics, 139:1031–44.

References II

Podlich, D. W. and Cooper, M. (1998).

Qu-gene: A simulation platform for quantitative analysis of genetic models.
Bioinformatics, 14:632–653.

Silva, E. D. B., Xavier, A., and Faria, M. V. (2021).

Joint modeling of genetics and field variation in plant breeding trials using relationship and different spatial methods: A simulation study of accuracy and bias.
Agronomy, 11:1397.

Sun, X., Peng, T., and Mumm, R. H. (2011).

The role and basics of computer simulation in support of critical decisions in plant breeding.
Molecular Breeding, 28:421–436.

What if your crop of interest is not included in AlphaSimR? What can you do? [Silva et al., 2021]

```
founderPop <-

# www.ncbi.nlm.nih.gov/pmc/articles/PMC2612967/pdf/136.pdf
runMacs2(nInd = 20,

nChr = 20,

# Genome sequence of the palaeopolyploid soybean
segSites = as.integer(round(46000/20,0)),

# Tsuda 2015 BMC genomics
# github.com/alenxav/Lectures/blob/master/Manuscripts/09_PGR_2018.

Ne = 106,
```

What if your crop of interest is not included in AlphaSimR? What can you do? [Silva et al., 2021]

```
ploidy = 2L,

# Choi et al, 2007 TAG
bp = 5.5e+07,

# Choi et al, 2007 TAG
genLen = round(2550.3/20,0)/100,

# Lavin, M., Herendeen, P. S. & Wojciechowski,
# M. F. Evolutionary rates

mutRate = 2.5e-08,
```

What if your crop of interest is not included in AlphaSimR? What can you do? [Silva et al., 2021]

```
# T. E. CarterJr.T. HymowitzR. L. Nelson &
# Tracing soybean domestication
# history: From nucleotide to genome 2012

histNe = c(500, 1500, 6000, 12000, 1e+05),

# T. E. CarterJr.T. HymowitzR. L. Nelson &
# Tracing soybean domestication
# history: From nucleotide to genome 2012

histGen = c(100, 1000, 10000, 1e+05, 1e+06),

inbred = TRUE)
```

What if your crop of interest is not included in AlphaSimR? What can you do?

Create a new MapPop-class from user supplied genetic maps and haplotypes from real data!

```
# Create genetic map for two chromosomes, each 1 Morgan long
# Each chromosome contains 11 equally spaced segregating sites
genMap = list(seq(0,1,length.out=11),
         seq(0,1,length.out=11))

# Create haplotypes for 10 outbred individuals
chr1 = sample(x=0:1,size=20*11,replace=TRUE)
chr1 = matrix(chr1,nrow=20,ncol=11)
chr2 = sample(x=0:1,size=20*11,replace=TRUE)
chr2 = matrix(chr2,nrow=20,ncol=11)
haplotypes = list(chr1,chr2)
founderPop = newMapPop(genMap=genMap, haplotypes=haplotypes)
```