

Simulating traits

Athens COST InsectIMP Course 2025

Alireza Ehsani, Jana Obšteter, and Gregor Gorjanc

2025-01-30

Introduction

What is a ‘trait’? Take any individual of any species from the entire animal or plant kingdom, including ourselves, and just look at them. Any characteristic that you can see or measure on them is called a trait. For example, their weight, their height, or their colour, their blood type. Any trait that we observe is called the “phenotype” of that individual. Derived from the Greek ‘pheno’ meaning “to show” and ‘type’ meaning, well, ‘type’.

Some traits are affected only by the genotype of an individual. We call such traits Mendelian traits. Mendelian traits are commonly affected by only one DNA locus or maybe a small number of DNA loci. When a trait is influenced by one locus, we call it a monogenic trait, indicating that one gene impacts that trait. When a trait is influenced by several loci, we call it an oligogenic trait, indicating that likely several genes are impacting that trait. Most traits are affected by many DNA loci. We call such traits polygenic (meaning many genes). Some traits, especially polygenic traits, are often affected by an individuals’ genotype and by the environment in which individuals live. We call such traits, complex traits. This complex situation is the basis for the so-called ‘nature and nurture’ concept. In mathematical terms, we say that the observed expression of a trait, or the phenotype value of an individual (P), is a function of the genetic value of the individual (G) and the value of the environment in which the individual lives (E); $P = f(G, E)$. The form of this function is largely unknown and is a subject of intense research. Following Fisher (1919), **AlphaSimR** assumes a linear approximation to this function: $P = G + E + G \times E$. Furthermore, in the simplest setting, it also assumes that genetic value of an individual (G) is also a linear function of individual DNA locus values: $G = G_1 + G_2 + \dots + G_k$, where k is the number of DNA loci affecting the trait.

We will now look at how to simulate a trait in **AlphaSimR**. To do this we will:

- Simulate founding genomes and set simulation parameters,
- Define a trait,
- Simulate a base population,
- Show genetic values of individuals,
- Sample phenotype values of individuals,
- Introduce some quantitative genetics functionality, and
- EXTRA: Point to additional resources on traits and quantitative genetics in AlphaSimR.

Founding genomes and simulation parameters

We will first simulate a population of individuals and their genomes. To speed up this session, we will only simulate founding genomes for 50 cattle, with only 2 pairs of chromosomes, and 10 loci (segregating sites) per chromosome.

```
# Clean the working environment  
rm(list = ls())
```

```

# Set the default plot layout
par(mfrow = c(1, 1))

library(AlphaSimR)

## Loading required package: R6

# Simulate founding genomes (this will run for longer, because we are simulating more genomes)
founderGenomes = runMacs(nInd = 50,
                        nChr = 2,
                        segSites = 10,
                        species = "CATTLE")

# Create the object holding simulation parameters
SP = SimParam$new(founderGenomes)

# Allocate sex at random
SP$setSexes("yes_rand")

```

Trait definition

We add a trait to the simulation via the `SimParam` object (here named `SP`) using the `addTraitA()` function (see `help(SimParam)` and scroll to `SimParam$addTraitA()`). This function simulates additive effects for the DNA loci that affect the trait. This function will only define the genetic component of the trait. We will come to the environmental component later. In the `addTraitA()` function we must first specify how many DNA loci affect the trait. We do this by using the `nQtlPerChr` argument. Here, `Qtl` stands for Quantitative Trait Loci (QTL), that is, DNA loci that affect a quantitative trait. As mentioned, these quantitative trait loci will have additive effects. This means that homozygote 0 will have genetic value $mean + 0$, heterozygote 1 will have genetic value $mean + a$, and homozygote 2 will have genetic value $mean + 2a$ - the effect of substituting ancestral allele with a mutation is additive. As just indicated, we also need to specify the mean for the trait using the `mean` argument and the genetic variance for the trait using the `var` argument. Variance will control the spread of genetic values around the mean. The higher the variance, the bigger the spread. And finally, we need to specify a distribution from which we will sample additive effects. Usually, we specify a normal (Gaussian) distribution for additive effects, which is the default option. If we know that some DNA loci have large effects, which is rare with the normal distribution, we can change the distribution from normal to gamma. We do this by setting the `gamma` argument to `TRUE` and controlling the shape of the gamma distribution with the `shape` argument.

Let's now add a trait, for example body weight, affected by 5 QTLs per chromosome with their additive effects drawn from a normal distribution, and setting the trait mean to 500 kg and genetic variance to 450 kg².

```

# Add (define) a trait with additive genetic effects of QTL
SP$addTraitA(nQtlPerChr = 5,
            mean = 500,
            var = 450)

```

Note that all the information stored in the `SP` object will be used for all the downstream AlphaSimR populations that we will create.

We can access the trait information, including the QTL effects, from the `SP` object.

```

# Inspect the traits element of the SP object
SP$traits # essential information

```

```
## [[1]]
```

```
## An object of class "TraitA"
## Slot "addEff":
## [1] 19.233799 -12.747557 16.344603 -14.700370 -7.163622 -4.658308
## [7] 8.824405 13.956669 25.900451 17.802499
##
## Slot "intercept":
## [1] 505.4089
##
## Slot "nLoci":
## [1] 10
##
## Slot "lociPerChr":
## [1] 5 5
##
## Slot "lociLoc":
## [1] 1 2 4 6 7 1 2 3 7 8
##
## Slot "name":
## [1] "Trait1"
```

Base population

Let us now create a new population `basePop` and inspect genetic values of individuals in this population. When we create a population in a simulation with a defined trait, `AlphaSimR` automatically calculates genetic values for all individuals in the population. This is done by applying the QTL effects to each QTL genotype and summing these effects over the entire genome of each individual.

```
# Create a base population
basePop = newPop(founderGenomes)
basePop
```

```
## An object of class "Pop"
## Ploidy: 2
## Individuals: 50
## Chromosomes: 2
## Loci: 20
## Traits: 1
```

Genetic values

Now we can inspect genetic values of individuals in the `basePop` through the `gv` element or by using the `gv()` function (they both return the same values).

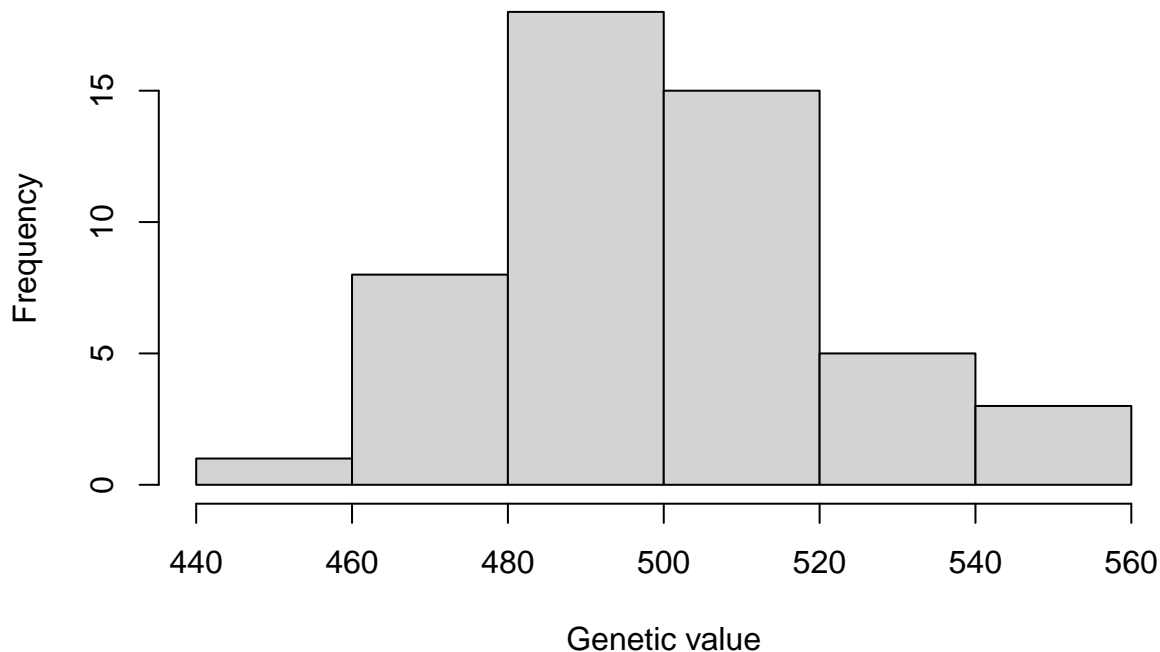
```
# Inspect genetic values of individuals through the gv element
# (just for the first 10 individuals)
basePop@gv[1:10]
```

```
## [1] 458.7084 494.7823 505.3902 471.2741 479.3633 466.6158 491.4335 531.8142
## [9] 497.5497 491.2113
```

```
# ... or by using the gv() function
gv(basePop)[1:10]
```

```
## [1] 458.7084 494.7823 505.3902 471.2741 479.3633 466.6158 491.4335 531.8142
## [9] 497.5497 491.2113
```

```
# Plot histogram of all genetic values
hist(gv(basePop), xlab = "Genetic value", main = "")
```



```
# Calculate variance of genetic values in this population
# (this is genetic variance in this population)
var(gv(basePop))
```

```
##          Trait1
## Trait1 459.1837
```

Phenotype values

Unfortunately, in real life we can not observe genetic values. We can only observe phenotype values. As discussed, phenotypes of complex traits are affected by genes and environment. To simulate such phenotype values we need to simulate environmental effects in addition to the QTL effects. In **AlphaSimR**, we simulate phenotype values for individuals in a population with the `setPheno()` function. When calling this function, we either have to specify variance of environmental effects through the `varE` argument or we have to specify the heritability through the `h2` argument.

Heritability is the ratio between genetic variance and phenotype variance (V_P), which is a function of genetic variance (V_G) and environmental variance (V_E): $h^2 = V_G/V_P = V_G/(V_G + V_E)$ (assuming there is no interaction between genotype and environment and there are no other sources of phenotype variation).

Let's phenotype our population assuming that genetic and environmental variance are equal in size. So, $V_G = V_E = 450kg^2$, $V_P = V_G + V_E = 450kg^2 + 450kg^2 = 900kg^2$ and $h^2 = V_G/V_P = 1/2$.

```
# Phenotype the basePop
basePop = setPheno(basePop,
                    h2 = 0.5)
```

Similarly to genetic values, we can access phenotype values through the `pheno` element or by using the `pheno()` function (they both return the same values).

```
# Inspect phenotype values of individuals through the pheno element
# (just for the first 10 individuals)
```

```
basePop@pheno[1:10]
```

```
## [1] 492.4973 485.6547 533.6681 489.5718 502.7902 473.0622 471.0353 558.5958  
## [9] 533.7319 467.4406
```

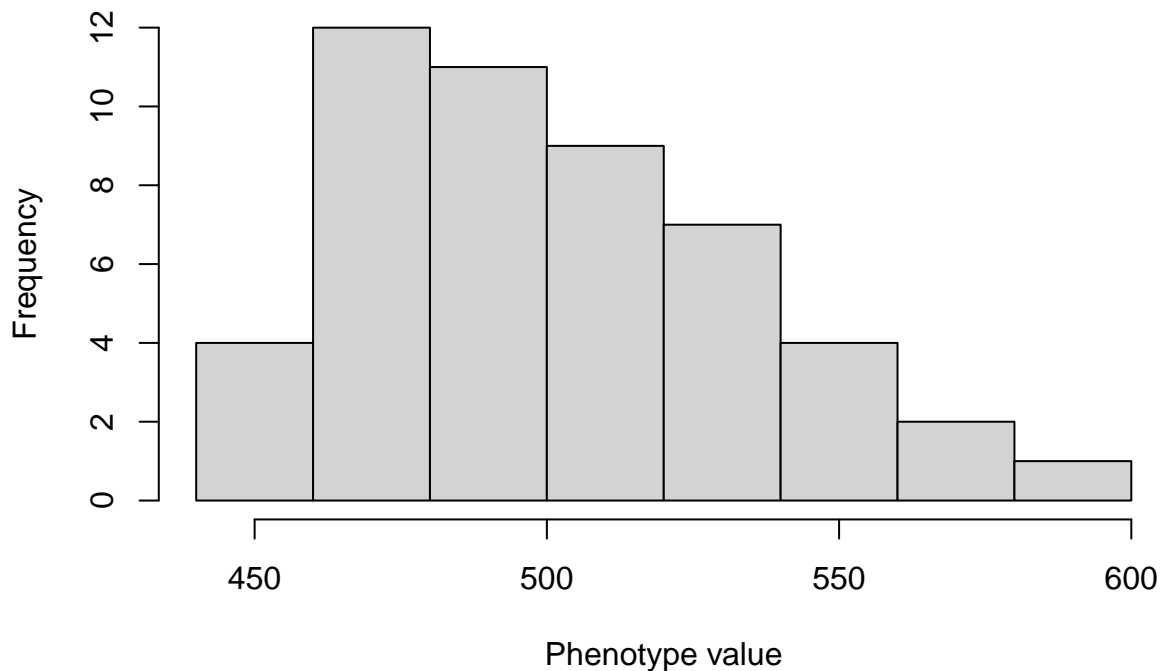
```
# ... or or by using the pheno() function
```

```
pheno(basePop)[1:10]
```

```
## [1] 492.4973 485.6547 533.6681 489.5718 502.7902 473.0622 471.0353 558.5958  
## [9] 533.7319 467.4406
```

```
# Plot histogram of phenotype values
```

```
hist(pheno(basePop), xlab = "Phenotype value", main = "")
```



```
# Calculate variance of phenotype values in this population
```

```
# (this is phenotypic variance in this population)
```

```
var(pheno(basePop))
```

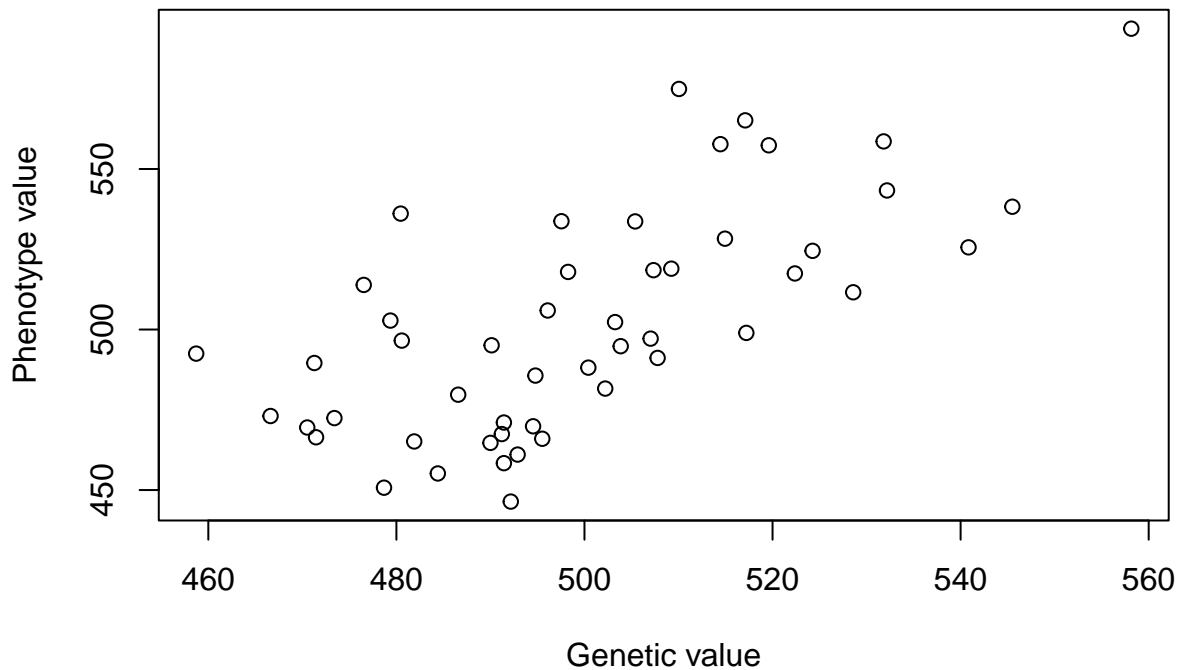
```
##          Trait1
```

```
## Trait1 1244.53
```

Since phenotype values are a function of genetic values we should see a positive correlation between these two sets of values.

```
# Relationship between genetic values and phenotype values
```

```
plot(x = gv(basePop), y = pheno(basePop),  
     xlab = "Genetic value", ylab = "Phenotype value")
```



```
# Correlation between genetic and phenotype values
cor(gv(basePop), pheno(basePop))
```

```
##           Trait1
## Trait1 0.6846265
```

Note that phenotype values are a function of the genetic value, which is fixed for an individual, and the environment effects, which can vary over time. Hence, a repeated call of `setPheno(basePop)` will return different values!

```
pheno(setPheno(basePop, h2 = 0.5))[1:5]
```

```
## [1] 441.5806 494.3548 504.3772 478.8440 492.1363
```

```
pheno(setPheno(basePop, h2 = 0.5))[1:5]
```

```
## [1] 473.4784 481.9578 503.4331 457.9347 471.8421
```

Quantitative genetics

AlphaSimR provides basic functionality for quantitative genetic analysis of simulated populations, such as calculating means and variances of the simulated values. Let's show how these functions compare with the R functions.

```
# Calculate genetic variance using the standard R functionality
var(gv(basePop))
```

```
##           Trait1
## Trait1 459.1837
```

```
# Calculate phenotype variance using the standard R functionality
var(pheno(basePop))
```

```
##           Trait1
## Trait1 1244.53
```

```

# Calculate heritability using the standard R functionality
var(gv(basePop)) / var(pheno(basePop))

##           Trait1
## Trait1 0.3689615

# ... or using AlphaSimR functions for quantitative genetics
varG(basePop)

##           Trait1
## Trait1      450

varP(basePop)

##           Trait1
## Trait1 1219.639

varG(basePop) / varP(basePop)

##           Trait1
## Trait1 0.3689615

```

You might have noticed slight difference between result from `var(gv(basePop))` and `varG(basePop)`. This is because base R `var()` uses $n - 1$ as a denominator to calculate sample variance, while AlphaSimR `varG()` (or `varP()`) use n as a denominator to calculate population variance.

Note that while we have provided heritability of 0.5 in the `setPheno()` call, we will likely see some variation in the actual heritability of the simulated phenotypes (as shown above). This is so because AlphaSimR simulations are stochastic. Difference between the specified and actual heritability will be small in large populations, yet will increase as populations become smaller.

You can study more about AlphaSimR functions for quantitative genetics by reading help pages.

```

help(meanP)
help(meanG)

help(varP)
help(varG)

```

EXTRA: Additional resources on traits and quantitative genetics in AlphaSimR

We have shown how to simulate a trait affected by additive effects of QTL. Additive effects are arguably the most important genetic effects. They capture most of genetic variation even when traits are affected by additive and non-additive effects. Moreover, additive effects are passed from generation to generation and are thus key for selective breeding. Non-additive effects can be divided into dominance effects (interaction between alleles at one QTL) and epistasis effects (interaction between alleles at different QTL). In addition, there may also be an interaction between genotype and environment, referred to as genotype-by-environment interaction or GxE. AlphaSimR allows the simulation of all these effects through different `addTrait*()` functions. Details about these functions are available in a separate AlphaSimR vignette.

```
vignette(package = "AlphaSimR", topic = "traits")
```

In relation to the additive and non-additive effects see the additional AlphaSimR functions for quantitative genetics listed below. Quantitative genetics is a large and a very important field for breeding. See Hivert et al. (2021) for a recent introduction to gene action and genetic variation, while detailed reviews of quantitative genetics are in Falconer and Mackay (1996) and Lynch and Walsh (1998).

```
help(genParam)

help(gv)
help(bv)
help(dd)
help(aa)

help(varG)
help(varA)
help(varD)
help(varAA)

help(genicVarG)
help(genicVarA)
help(genicVarD)
help(genicVarAA)
```

References

- Fisher R.A. (1919) The correlation between relatives on the supposition of Mendelian inheritance. Transactions of The Royal Society of Edinburgh, LII:15, 399—433, <https://doi.org/10.1017/S0080456800012163>
- Hivert V., Wray N.R., Visscher P.M. (2021) Gene action, genetic variation, and GWAS: A user-friendly web tool. PLoS Genetics, 17:5, <https://doi.org/10.1371/journal.pgen.1009548>
- Falconer D.S., Mackay T.F.C. (1996) Introduction to quantitative genetics. 4th edition, Longman, <https://www.pearson.com/uk/educators/higher-education-educators/program/Falconer-Introduction-to-Quantitative-Genetics-4th-Edition/PGM432132.html>
- Lynch M., Walsh B. (1998) Genetics and Analysis of Quantitative Traits. Sinauer Associates, <https://global.oup.com/ushe/product/genetics-and-analysis-of-quantitative-traits-9780878934812>