

# Animal breeding programme

Athens COST InsectIMP Course 2025

Ivan Pocrnic and Gregor Gorjanc

2025-01-30

## Introduction

Many breeding programmes have a more complex structure compared to what we have seen up to now in this course. Here we will look into an animal breeding programme that is still quite simple, but shows more complexity. Specifically, in animal breeding, selection intensity often differs substantially between males and females; its often higher in males. This is driven by the need to have a substantial number of dams (mothers) to generate selection candidates and dam replacements. Also, sires (fathers) can generate many progeny, either through natural mating or artificial insemination. Another complexity in real breeding programmes is that parents can be used across multiple generations, which generates a breeding programme with overlapping generations.

In this vignette, we will show how to simulate an animal breeding programme with overlapping generations and different selection intensities in males and females. To this end, we will simulate a population with 1000 cows of different ages mated with 50 bulls. The key trait of interest will be weaning weight of calves. We will also show how to record data across generations for a later analysis. We will achieve all this by:

- Simulating a base population,
- Storing miscellaneous information,
- Creating initial parent populations,
- Recording data,
- Simulating multiple years, and
- Analysing response to selection.

## Base population

We will start the simulation by simulating founder genomes, allocating sex of individuals, defining a trait, and initiating a base population. We will simulate a cattle genome with 30 chromosomes for 2000 founders. The trait under selection will be weaning weight of calves, with a mean of 250 kg, phenotype variance of 400 kg<sup>2</sup>, and heritability of 0.3. The trait will have a simple additive genetic architecture with 100 QTL on each chromosome. In cattle, males are on average heavier than females, due to sexual dimorphism. While such an effect can also be added in AlphaSimR, we will here ignore this effect for the sake of simplicity.

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

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

# Load AlphaSimR
library(AlphaSimR)
```

```
## Loading required package: R6
```

```

# Simulate founder genomes
# ... this runMacs() call will take quite a bit of time!
# founderGenomes = runMacs(nInd = 2000,
#                           nChr = 30,
#                           segSites = 100,
#                           species = "CATTLE")
# ... therefore, we will speed up the demonstration with the quickHaplo()
# (we recommend use of runMacs() for research purposes!)
founderGenomes = quickHaplo(nInd = 2000,
                           nChr = 30,
                           segSites = 100)

# Global simulation parameters
SP = SimParam$new(founderGenomes)

SP$setSexes("yes_sys")

phenoVar = 400
heritability = 0.3
genVar = phenoVar * heritability
SP$addTraitA(nQtlPerChr = 100, mean = 250, var = genVar)

# Base population
founders = newPop(founderGenomes)

# Phenotype the base population
founders = setPheno(pop = founders, h2 = heritability)

```

## Miscellaneous information

In this simulation we will work with individuals born in different years. To keep track of these individuals, we will save them in different R objects. But we will also combine the individuals with different birth years into one object. To keep track of the different years or birth for these individuals, we will store this additional information in the `misc` (=miscellaneous) slot of AlphaSimR populations. This slot is a `list` (type of R object) of length equal to the number of quantities stored (=node) and each of these nodes is expected to have the length equal to the number of individuals. We can store in the `misc` slot any information we want. You can use standard R list code to work with the `misc` object.

```

# Assign year of birth for the founders
year = 0
founders@misc$yearOfBirth = rep(year, times = nInd(founders))
founders@misc$yearOfBirth[1:10]

```

```
## [1] 0 0 0 0 0 0 0 0 0 0
```

## Initial parent populations

Structure of the simulated breeding programme is shown in Figure 1. In the following we explain this breeding structure and show how to simulate it.

Now we will take founder males and select 50 superior males as sires (fathers) of the next generation. These 50 sires will be represented by 40 young sires (1 year old) and 10 older sires (2 years old). The rationale here is that better sires will be used for 2 years (2 generations), while we will replace most of the sires with a new

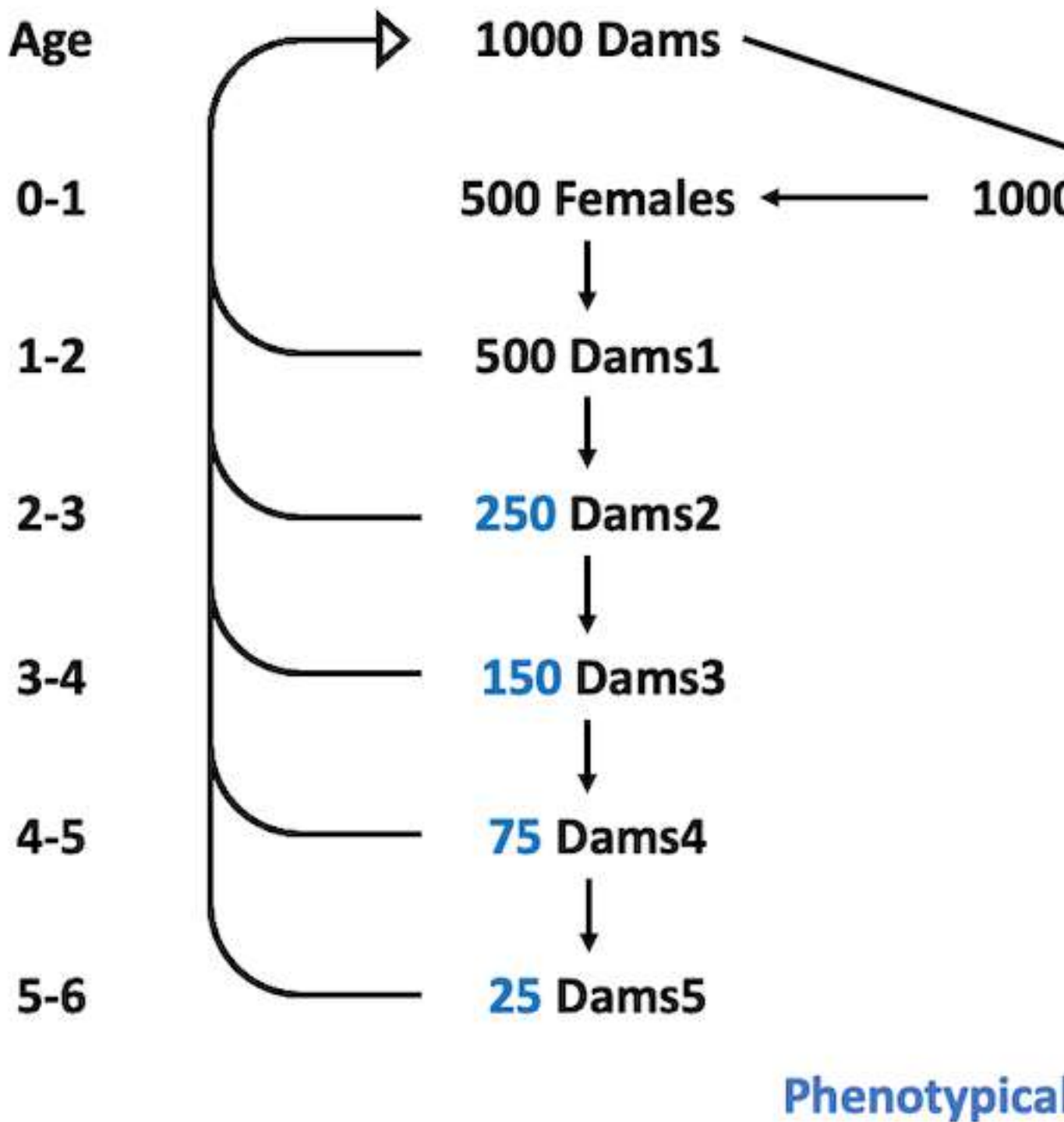


Figure 1: Figure 1: Simulated beef breeding programme with 1000 cows (dams) and 50 bulls (sires) of different ages and different selection intensity.

generation of genetically improved males. This “male selection path” is shown in the right part of Figure 1.

```
males = selectInd(pop = founders, nInd = 50, use = "pheno", sex = "M")
sires2 = males[ 1:10]
sires2@misc$yearOfBirth = rep(-1, times = nInd(sires2))

sires1 = males[11:50]
sires1@misc$yearOfBirth = rep(0, times = nInd(sires1))

sires = c(sires2, sires1)
nInd(sires)
```

```
## [1] 50
```

Let’s inspect year of birth for the sires.

```
table(sires@misc$yearOfBirth)
```

```
##
## -1  0
## 10 40
```

For generating the selection candidates and future production females we require a sufficient number of dams (mothers). Specifically, to generate 1000 progeny every year, we require 1000 dams. In this simulation we used option `SP$setSexes("yes_sys")` that will systematically assign sexes to newly created individuals, so we will get exactly 50% male and 50% female progeny. We will take the founder females and use most of them as dams. These dams will be assumed to have different ages because beef cows stay in the herd for multiple years and have multiple calves through their lifetime. In this simulation, we will assume that dams stay in the herd for up to 5 years, but that every year we only keep a certain number of phenotypically best dams. This “female selection path” is shown in the left part of Figure 1.

In reality the realised sex ratio is not strictly 50:50, and you might consider using option `SP$setSexes("yes_rand")` that will randomly assign a sex to each individual. In that case, the number of females will fluctuate from year to year and you will also have to modify the code to ensure a stable number of dams over the years. For simplicity, in this example we use the systematic assignment of sexes and a constant number of dams.

First, we need to select only female founders and set the number of dams kept in each year.

```
cat("Founder females\n")
```

```
## Founder females
```

```
(nFemales = sum(founders@sex == "F"))
```

```
## [1] 1000
```

```
females = selectInd(pop = founders, nInd = nFemales, use = "pheno", sex = "F")
```

```
# Here we define how many dams are kept in each year
```

```
nDams1 = 500
```

```
nDams2 = 250
```

```
nDams3 = 150
```

```
nDams4 = 75
```

```
nDams5 = 25
```

```
sum(nDams1, nDams2, nDams3, nDams4, nDams5)
```

```
## [1] 1000
```

Now we will select the oldest group of dams.

```

cat("Dams5\n")

## Dams5
(start = 1)

## [1] 1
(end = nDams5)

## [1] 25
dams5 = females[start:end]
dams5@misc$yearOfBirth = rep(-4, times = nDams5)
nInd(dams5)

## [1] 25
And second oldest group of dams.
cat("Dams4\n")

## Dams4
(start = end + 1)

## [1] 26
(end = start - 1 + nDams4)

## [1] 100
dams4 = females[start:end]
dams4@misc$yearOfBirth = rep(-3, times = nDams4)
nInd(dams4)

## [1] 75
And the other group of dams.
cat("Dams3\n")

## Dams3
(start = end + 1)

## [1] 101
(end = start - 1 + nDams3)

## [1] 250
dams3 = females[start:end]
dams3@misc$yearOfBirth = rep(-2, times = nDams3)
nInd(dams3)

## [1] 150
cat("Dams2\n")

## Dams2
(start = end + 1)

## [1] 251

```

```

(end = start - 1 + nDams2)

## [1] 500
dams2 = females[start:end]
dams2@misc$yearOfBirth = rep(-1, times = nDams2)
nInd(dams2)

## [1] 250
cat("Dams1\n")

## Dams1
(start = end + 1)

## [1] 501
(end = start - 1 + nDams1)

## [1] 1000
dams1 = females[start:end]
dams1@misc$yearOfBirth = rep(0, times = nDams1)
nInd(dams1)

## [1] 500
dams = c(dams5, dams4, dams3, dams2, dams1)
nInd(dams)

## [1] 1000
Let's inspect year of birth for the dams
table(dams@misc$yearOfBirth)

##
##  -4  -3  -2  -1   0
##  25  75 150 250 500

```

## Data recording

To record data from multiple populations, we will define a data recording function `recordData()`. As an input, the function will accept: 1) a data frame (`data` argument) that will collate the information from multiple AlphaSimR populations, 2) an AlphaSimR population (`pop` argument) whose data we will save, and 3) a year of use (`yearOfUse` argument) to denote parent usage.

In this example, we will be storing animal identification (`id`), parents' identification's (`father` and `mother`), sex (`sex`), genetic value (`gv`), phenotype value (`pheno`), year of birth (`yearOfBirth`), and year of use for parents (`yearOfUse`).

```

# Function to record and collate data
recordData <- function(data = NULL, pop, yearOfUse = NA) {
  popData = data.frame(id       = pop@id,
                        father   = pop@father,
                        mother   = pop@mother,
                        sex       = pop@sex,
                        gv       = pop@gv[, "Trait1"],
                        pheno     = pop@pheno[, "Trait1"],
                        yearOfBirth = pop@misc$yearOfBirth,

```

```

                                yearOfUse = yearOfUse)
# Manage first instance of calling this function, when data is NULL
if (is.null(data)) {
  ret = popData
} else {
  ret = rbind(data, popData)
}
return(ret)
}

```

We will create two data frames. The first one will be called `data4AllAnimals`, where we will store the data for all animals. The second one will be called `data4Parents`, where we will store data for parents.

```

data4AllAnimals = recordData(pop = founders)
head(data4AllAnimals)

```

```

##   id father mother sex      gv      pheno yearOfBirth yearOfUse
## 1  1      0      0  M 270.7453 270.8095      0         NA
## 2  2      0      0  F 251.3719 260.8371      0         NA
## 3  3      0      0  M 241.4850 234.3260      0         NA
## 4  4      0      0  F 257.9352 262.0949      0         NA
## 5  5      0      0  M 249.5907 248.7881      0         NA
## 6  6      0      0  F 255.7261 280.4392      0         NA

```

```

data4AllParents = recordData(pop = c(sires, dams), yearOfUse = year) # year is 0 at this stage in the s
head(data4AllParents)

```

```

##   id father mother sex      gv      pheno yearOfBirth yearOfUse
## 1 1719      0      0  M 274.1191 328.4603      -1         0
## 2  459      0      0  M 263.3845 308.3033      -1         0
## 3 1705      0      0  M 281.5045 307.0642      -1         0
## 4  231      0      0  M 275.6281 305.0776      -1         0
## 5  121      0      0  M 270.2998 304.5244      -1         0
## 6 1437      0      0  M 267.3431 300.3913      -1         0

```

```

data4NewParents = recordData(pop = c(sires1, dams1))
head(data4NewParents)

```

```

##   id father mother sex      gv      pheno yearOfBirth yearOfUse
## 1 1815      0      0  M 265.2696 294.8080      0         NA
## 2 1025      0      0  M 268.2717 294.7255      0         NA
## 3 1607      0      0  M 255.3244 294.5358      0         NA
## 4 1089      0      0  M 267.7703 294.1674      0         NA
## 5  897      0      0  M 260.5203 294.0597      0         NA
## 6  179      0      0  M 258.3127 293.4198      0         NA

```

## Multiple years

We will simulate 20 years of the breeding programme. As mentioned before, we will select 50 phenotypically best males as sires, 40 young (1 year old) and 10 old (2 years old). We will be generating 1000 progeny. To generate these progeny we will require 1000 dams, that will be used up to 5 years, with only phenotypically best dams staying in the herd for longer.

During this simulation we will record the data, as before, for all the newly generated individuals, all parents, and newly selected parents.

```

for (year in 1:20) {
  cat("Working on the year:", year, "\n")

  # Generate progeny from current dams and sires
  candidates = randCross2(males = sires, females = dams, nCrosses = nInd(dams))
  candidates@misc$yearOfBirth = rep(year, times = nInd(candidates))
  candidates = setPheno(candidates, h2 = heritability)

  # Record data for all newborn animals
  data4AllAnimals = recordData(data = data4AllAnimals,
                              pop = candidates)

  # Record data for the used sires and dams (young and old)
  data4AllParents = recordData(data = data4AllParents,
                              pop = c(sires, dams),
                              yearOfUse = year)

  # Update and select sires
  sires2 = selectInd(pop = sires1, nInd = 10, use = "pheno")
  sires1 = selectInd(pop = candidates, nInd = 40, use = "pheno", sex = "M")
  sires = c(sires2, sires1)

  # Update and select dams
  dams5 = selectInd(pop = dams4, nInd = nDams5, use = "pheno")
  dams4 = selectInd(pop = dams3, nInd = nDams4, use = "pheno")
  dams3 = selectInd(pop = dams2, nInd = nDams3, use = "pheno")
  dams2 = selectInd(pop = dams1, nInd = nDams2, use = "pheno")
  dams1 = selectInd(pop = candidates, nInd = nDams1, use = "pheno", sex = "F")
  dams = c(dams5, dams4, dams3, dams2, dams1)

  # Record data for the newly selected sires and dams (just the new ones)
  data4NewParents = recordData(data = data4NewParents,
                              pop = c(sires1, dams1))
}

```

```

## Working on the year: 1
## Working on the year: 2
## Working on the year: 3
## Working on the year: 4
## Working on the year: 5
## Working on the year: 6
## Working on the year: 7
## Working on the year: 8
## Working on the year: 9
## Working on the year: 10
## Working on the year: 11
## Working on the year: 12
## Working on the year: 13
## Working on the year: 14
## Working on the year: 15
## Working on the year: 16
## Working on the year: 17
## Working on the year: 18
## Working on the year: 19

```

```
## Working on the year: 20
```

## Response to selection

To summarise response to selection over the years, we will show distributions of phenotype and genetic values for males and females separately because selection intensity is different between the two sexes in this breeding programme. Furthermore, we will show it for newborn animals (selection candidates), all parents, and for selected animals (parents). To do this, we will use two additional R packages: `ggplot2` and `ggridges` because they enable a quick generation of quite sophisticated plots. The `ggplot2` package and its wider ecosystem is very powerful and versatile. We warmly recommend you study it further. Here we will simply use the following plotting code, without delving into details.

```
# Install additional packages for plotting
install.packages(pkg = c("ggplot2", "gggridges"), repos = "https://cloud.r-project.org")

## Installing packages into '/Users/ggorjanc/Library/R/arm64/4.4/library'
## (as 'lib' is unspecified)

##
## The downloaded binary packages are in
## /var/folders/f_/58r_f9r97t3gmvlmt3_l834m0000gq/T//RtmpAUSefh/downloaded_packages

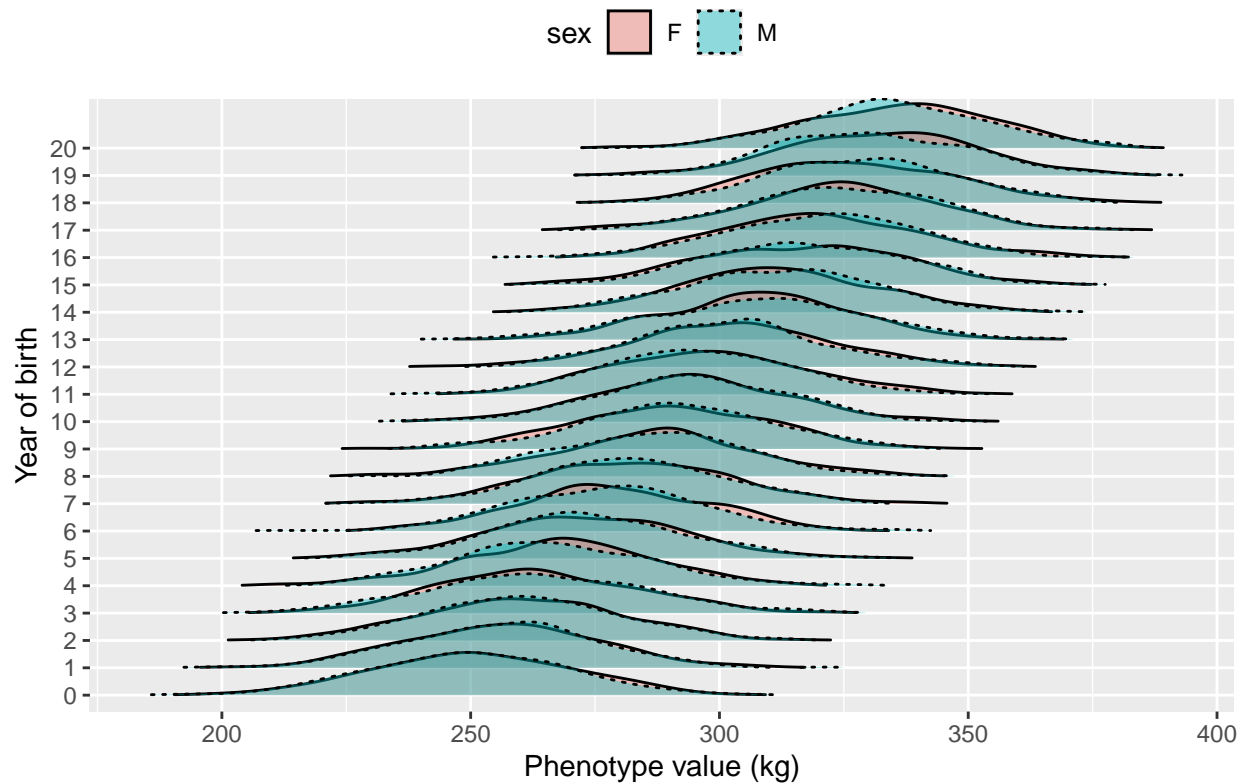
# Load the packages
library(ggplot2)
library(gggridges)

# Range of values
phenoRange = range(c(data4AllAnimals$pheno, data4AllAnimals$gv))

# Plot phenotype values for all newborn animals per year and sex
p = ggplot(data4AllAnimals, aes(x = pheno, y = as.factor(yearOfBirth))) +
  geom_density_ridges(aes(fill = sex, linetype = sex), alpha = .4, rel_min_height = 0.01) +
  xlim(phenoRange) +
  ylab("Year of birth") +
  xlab("Phenotype value (kg)") +
  ggtitle("Newborn animals") +
  theme(legend.position = "top")
print(p)

## Picking joint bandwidth of 4.98
```

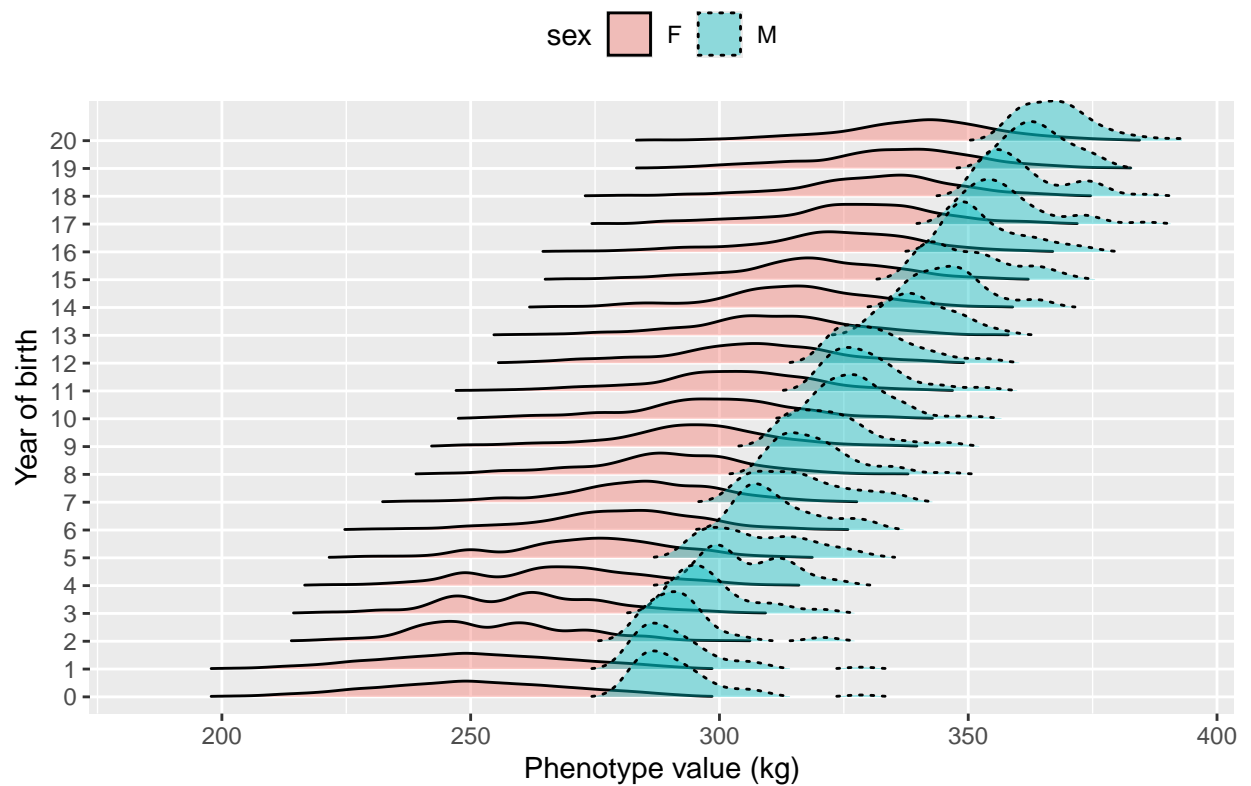
## Newborn animals



```
# Plot phenotype values for all parents per year and sex
p = ggplot(data4AllParents, aes(x = pheno, y = as.factor(yearOfUse))) +
  geom_density_ridges(aes(fill = sex, linetype = sex), alpha = .4, rel_min_height = 0.01) +
  xlim(phenoRange) +
  ylab("Year of birth") +
  xlab("Phenotype value (kg)") +
  ggtitle("Parents") +
  theme(legend.position = "top")
print(p)
```

```
## Picking joint bandwidth of 3.27
```

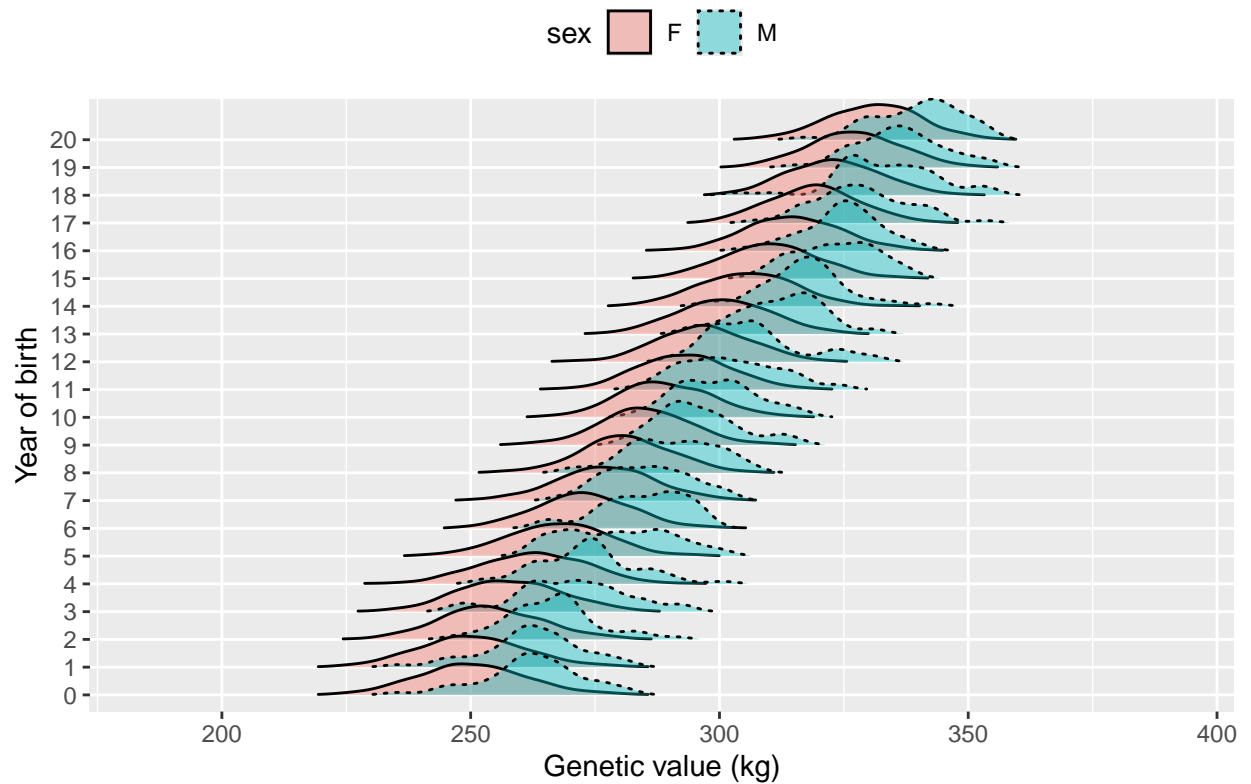
## Parents



```
# Plot genetic values for all parents per year and sex
p = ggplot(data4AllParents, aes(x = gv, y = as.factor(yearOfUse))) +
  geom_density_ridges(aes(fill = sex, linetype = sex), alpha = .4, rel_min_height = 0.01) +
  xlim(phenoRange) +
  ylab("Year of birth") +
  xlab("Genetic value (kg)") +
  ggtitle("Parents") +
  theme(legend.position = "top")
print(p)
```

```
## Picking joint bandwidth of 2.81
```

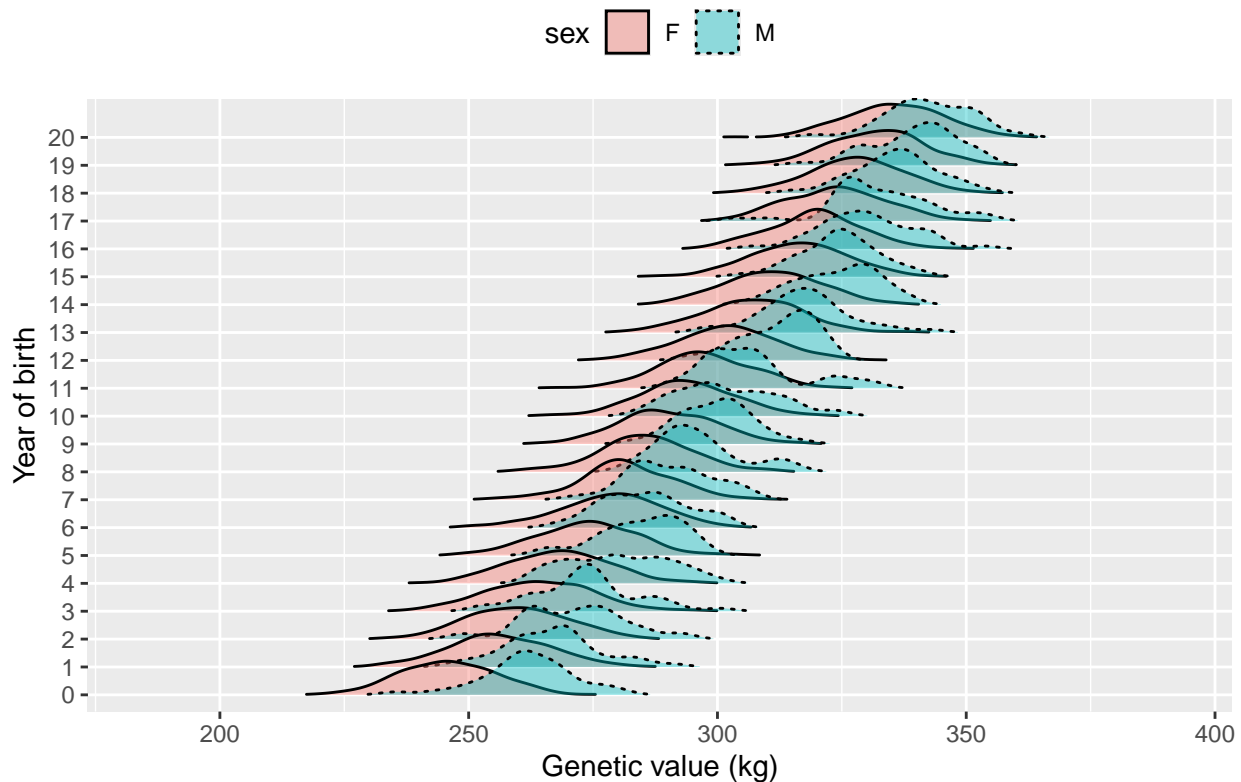
## Parents



```
# Plot genetic values for newly selected parents per year and sex
p = ggplot(data4NewParents, aes(x = gv, y = as.factor(yearOfBirth))) +
  geom_density_ridges(aes(fill = sex, linetype = sex), alpha = .4, rel_min_height = 0.01) +
  xlim(phenoRange) +
  ylab("Year of birth") +
  xlab("Genetic value (kg)") +
  ggtitle("New parents") +
  theme(legend.position = "top")
print(p)
```

```
## Picking joint bandwidth of 3
```

## New parents



The first plot showed distribution of *phenotype values of newborn animals* for each year, separated by sex. We clearly saw a response to selection across years and no difference between sexes. This is expected, because, in the absence of sexual dimorphism, DNA lottery and environment generate similar distributions of genetic and phenotype values for each sex.

The second plot showed distribution of *phenotype values of parents* for each year, separated by sex. As before, response to selection was clearly seen across years, but we also saw a large difference between distributions for females and males. Males had larger values due to higher selection intensity in males.

The third plot showed distribution of *genetic values of parents* for each year, separated by sex. Compared to phenotype values, we saw narrower distribution because phenotype values are more dispersed due to environmental effects. Also, the difference between sexes was not as large as we might have thought based on phenotype values.

The fourth plot showed distribution of *genetic values of newly selected parents* for each year, separated by sex. Compared to all parents, we saw that newly selected parents had higher genetic values due to genetic progress year on year.