# Simulation of breeding programs with the Modular Breeding Program Simulator (MoBPS)

Torsten Pook

30/03/2023

WAGENINGEN
UNIVERSITY & RESEARCH

- You do not learn how to use a software by listening
  - Most of this course will be practical sessions
- Form small groups (~1 – 4)
  - In particular when you have limited programming background
- Joint discussion & sample solution
- All sample solutions and slides will be shared

# Agenda

- Thursday & Friday:
  - Program: 9:00 – 17:00
  - Lunch break: 12:30 – 13:30
  - Coffee breaks at ~10.30, 15.00
- Snacks & drinks at TheSpot after the workshop today

# Agenda - Thursday

- General introduction of the MoBPS framework
- Basic functionality of the web-interface
- More advanced features of the web-interface
- Scenario comparison in the web-interface
- Basic functionality of the R-package
- Trait generation in the R-package

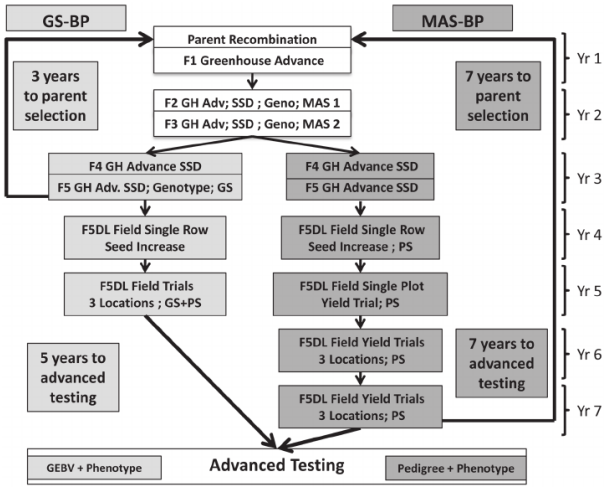
# Agenda - Friday

- Breeding value estimation in the R-package
- Implementing own methodology within R
- Use of offspring phenotypes in the R-package
- Setting up a simulation with multiple generation cycles in the R-package
- Simulating multiple scenarios in the R-package
- Evaluating different simulation scenarios in R

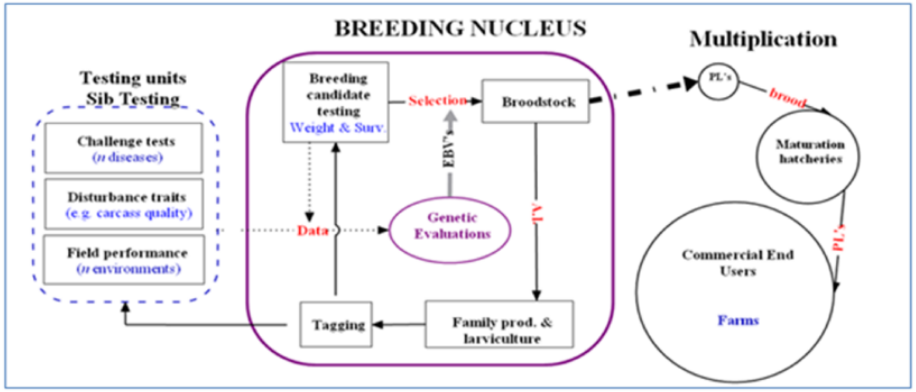
# General introduction

# What is a breeding program?

- Planned breeding of a group of animals or plants over several generations
- Goal: Change characteristics of animals/plant through careful selection of breeding partners
- General language to describe breeding programs (Simianer et al. 2021)




(Wheat breeding: Heffner et al., 2010)




(Fish breeding: Rye 2012)

# What are we interested in?

- What is our breeding objective?
  - Genetic progress
  - Maintenance of genetic diversity
  - Risk (variability of the outcome)
  - Economic efficiency

# How to control it?

- How many animals/plants to use
- Generate genotype / phenotype data
  - How many? Which?
- Mating scheme
- Selection technique
- Use of biotechnology
- And much more...

→ Complex optimization problem!

# Possible ways to answer this?

- Experience of the breeder
- Simulation study
- Cohort-based deterministic (ZPLAN+, Täubert et al. 2010)

$$R = i \cdot h \cdot \sigma_a$$

- Good approximation but formulas are limited to specific application and are constructed to handle „easy“ scenarios

  → Stochastic simulation

# What is a stochastic simulation?

- Every breeding action in a breeding scheme is simulated
  - Low number of assumptions
  - Flexibility
  - High level of detail
  - High computational demands & work!
- Results of a simulation will not be an expected gain but the realization of a stochastic process

# What is the MoBPS?

- Environment for the simulation
- The software takes care of backend-"stuff" that you have to account for but is not main part of analysis
  - Meiosis
  - Trait simulation
  - Efficient data storage
- Pre-implemented functions common breeding actions
  - Breeding value estimation
  - Phenotyping
  - Selection
- Function to help with down-stream analysis

# About the R-package

- Mainly distributed via GitHub
- Design philosophy:
  - Generate a framework that is able to simulate all breeding programs
  - When something is not yet possible and we see a general value in it, we are going to add it

**Version 1.10.48 (29.03.23)**

Documentation overhaul (?breeding.diploid, ?creating.diploid + Guidelines)

Added function to add additional genetic diversity to existing population (add.diversity)

Added options to avoid multiple recombinations in small areas of the genome in breeding.diploid()

Added options to generate traits in multiple locations / GxE in creating.diploid()

Improved efficeny for high number of traits (e.g. recalculate.manual() )

Removed sequenceZ functionality

Renamed shuffle.cor / shuffle.traits to trait.cor / trait.cor.include in creating.diploid()

Added plotting parameters to founder.simulation / ld.decay

Added function to print computing times of individuals steps of a simulation (get.computing.time)

Added function to calculate allele frequency / minor allele frequencies (get.allele.freq / get.maf)

MoBPSweb: Manually selected nodes for BVE that are not possible to be generated before are automatically excluded fr
(manual.select.check in json.simulation)

**Version 1.10.06 (05.12.22)**

Added MiXBLUP implementation for breeding value estimation

Added tracking of founder pools

Added size.scaling to creating.diploid / breeding.diploid

Added function to optimize the number of cores used for generation of individuals (optimize.cores() )

Added function to visualize the pedigree (get.pedigree.visual)

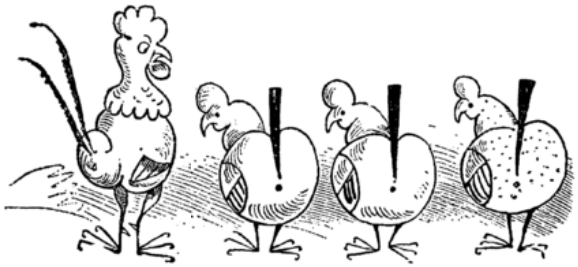
Added functionality for the generation of subpopulation specific traits including merging of traits

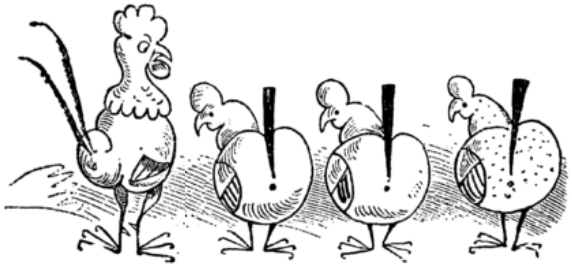
# About the R-package

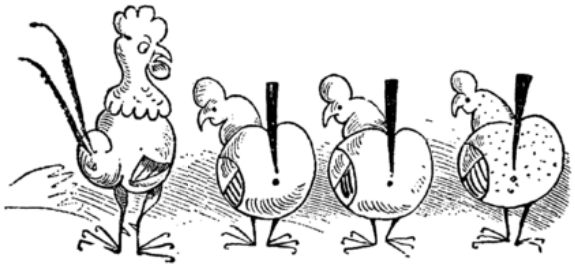
- The parameters in MoBPS are used to perform specific breeding actions
- You can not tell the tool to simulate a population for you with given effective population size / levels of LD / a population specific trait
    - You yourself have to set up a mating scheme to obtain those target values
        - Simulate 100 generations of random mating in a population
        - Analyze the final population
        - If levels of LD are too low
            - simulate more generations / reduce the number of individuals

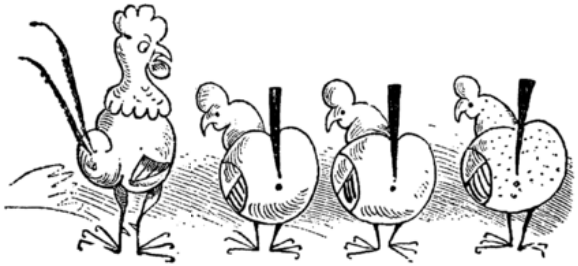
# Some simulations to inspire

# Cock rotation





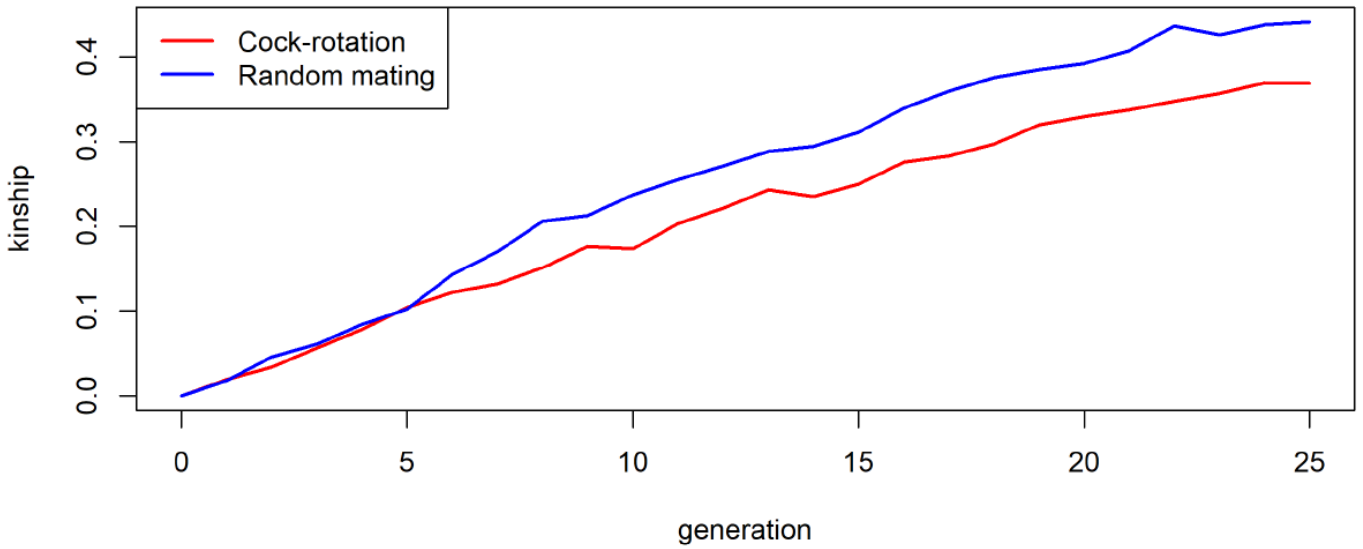




WAGENINGEN
UNIVERSITY & RESEARCH

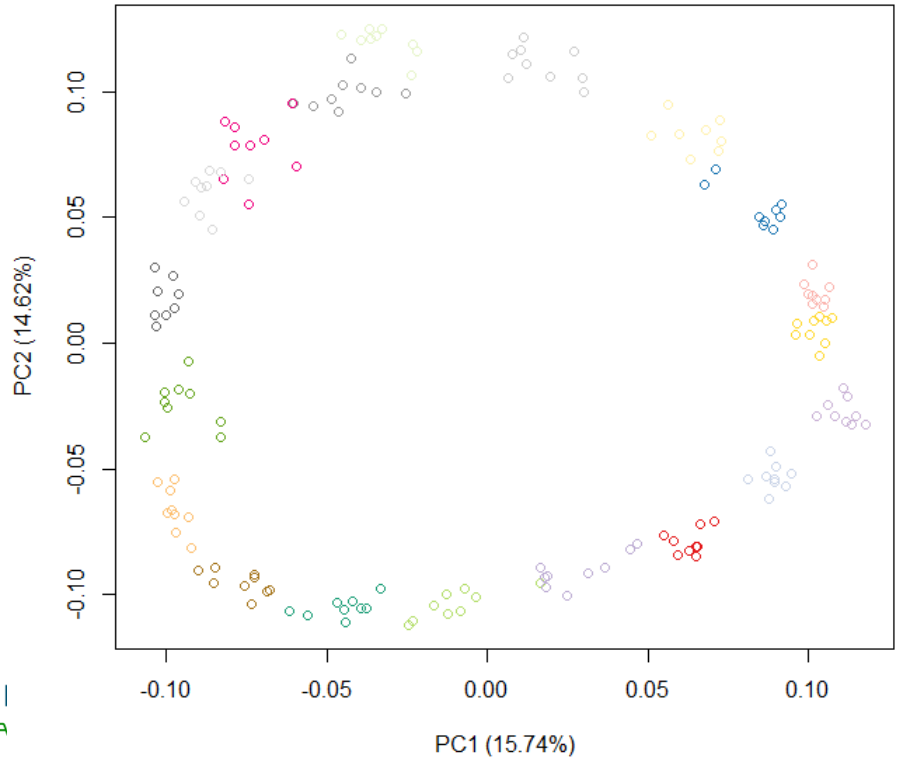
# Cock rotation

- Breeding scheme to reduce loss of genetic diversity



# Cock rotation

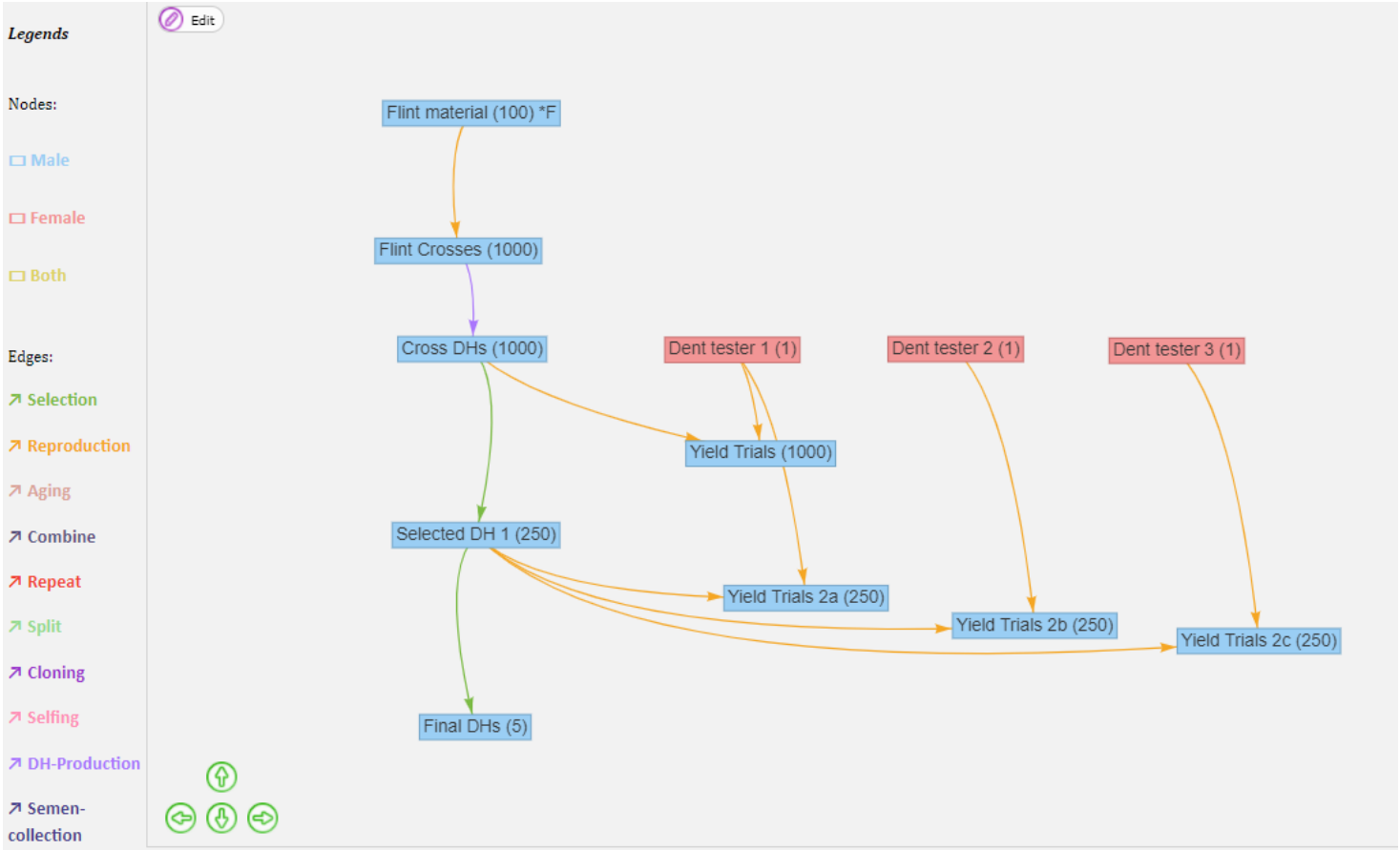
- Genetic distances between animals according to a principle component analysis




Each color represents one box

WAGENINGEN
UNIVERSITY & RESEARCH

# Test crossing scheme in maize



# Test crossing scheme in maize

- Phenotypes are collected in multiple different environments
- Different number of repetitions / plots

**Genetic Correlation**

| | Grain Yield Loc1 | Grain Yield Loc2 | Grain Yield Loc3 |
|---|---|---|---|
| Grain Yield Loc1 | 1 | 0.8 | 0.8 |
| Grain Yield Loc2 | 0.8 | 1 | 0.8 |
| Grain Yield Loc3 | 0.8 | 0.8 | 1 |

**Phenotype Information**

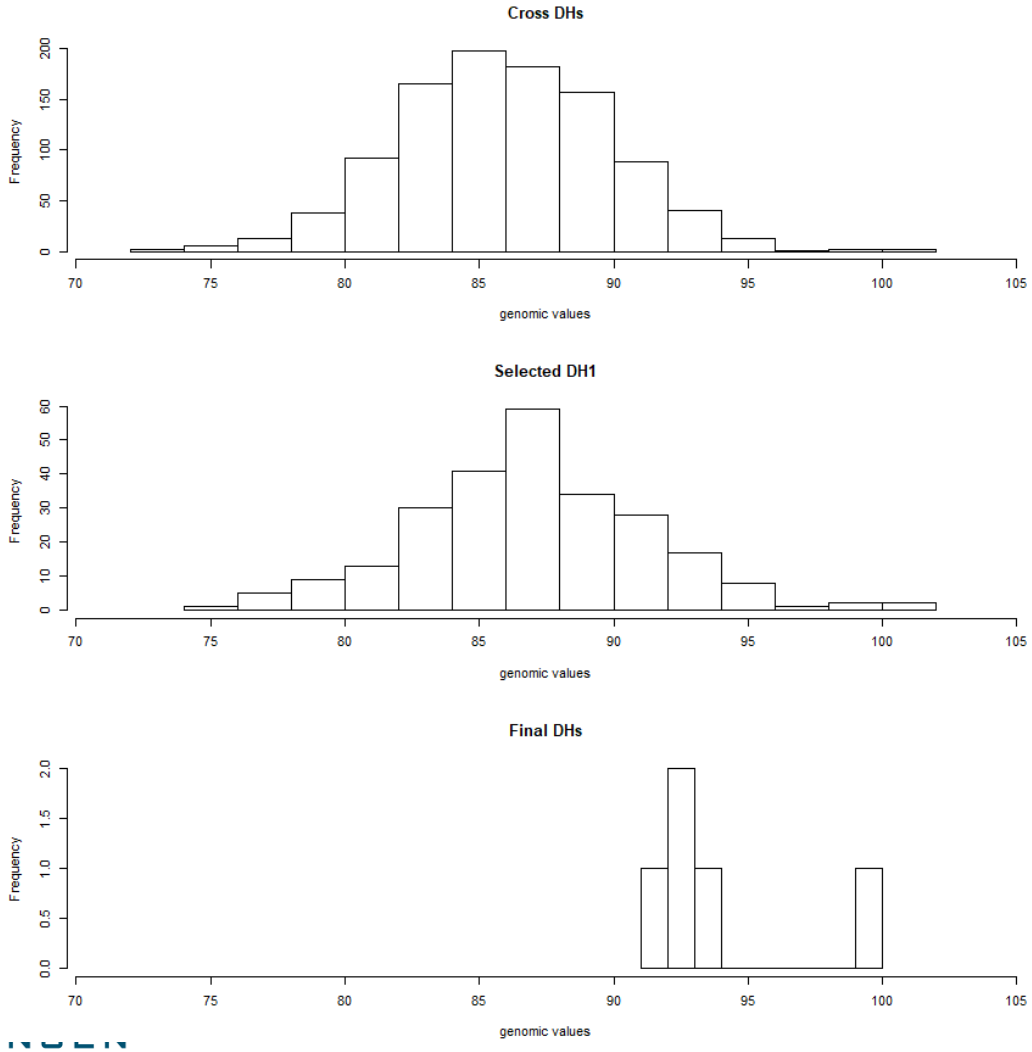
*Press here to get info on how this module works!*

Add new phenotype | Show/Hide 3 phenotypes | Show/Hide QTLs | Show/Hide residual correlation | Show/Hide genetic correlation

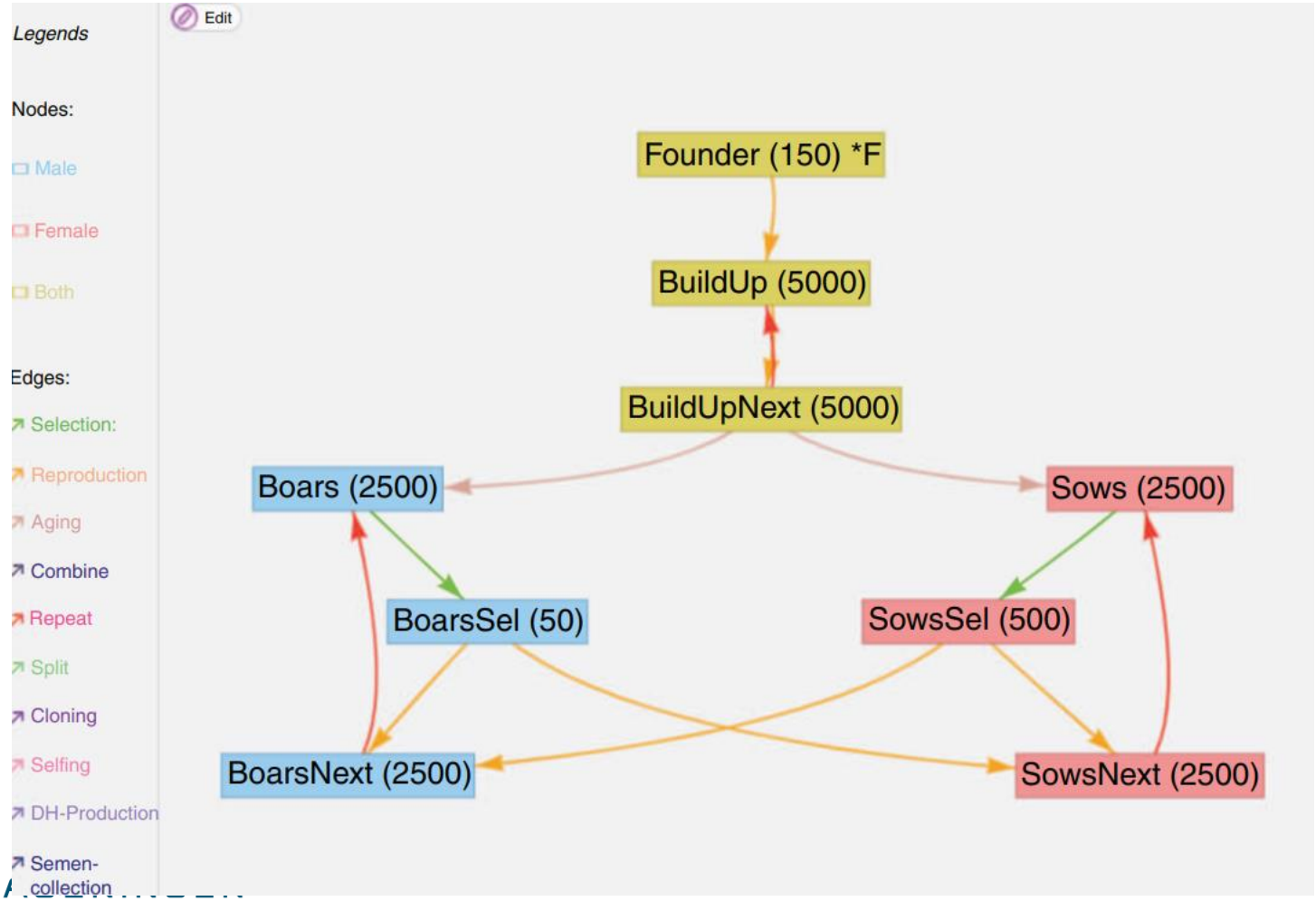
| Phenotype | Unit | Pheno. Mean | Pheno. SD | Heritability | Repeatability | # additive QTL | # dominant QTL | # qualitative epistatic QTL | # quantitative epistatic QTL | Major QTL | Value per unit (€) | Show Cor | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Grain Yield Lc | | 100 | 10 | 0.3 | 0.5 | 500 | 500 | 0 | 0 | 0 | 0 | ☑ | X |
| Grain Yield Lc | | 100 | 10 | 0.3 | 0.5 | 500 | 500 | 0 | 0 | 0 | 0 | ☑ | X |
| Grain Yield Lc | | 100 | 10 | 0.3 | 0.5 | 500 | 500 | 0 | 0 | 0 | 0 | ☑ | X |

WAGENINGEN
UNIVERSITY & RESEARCH

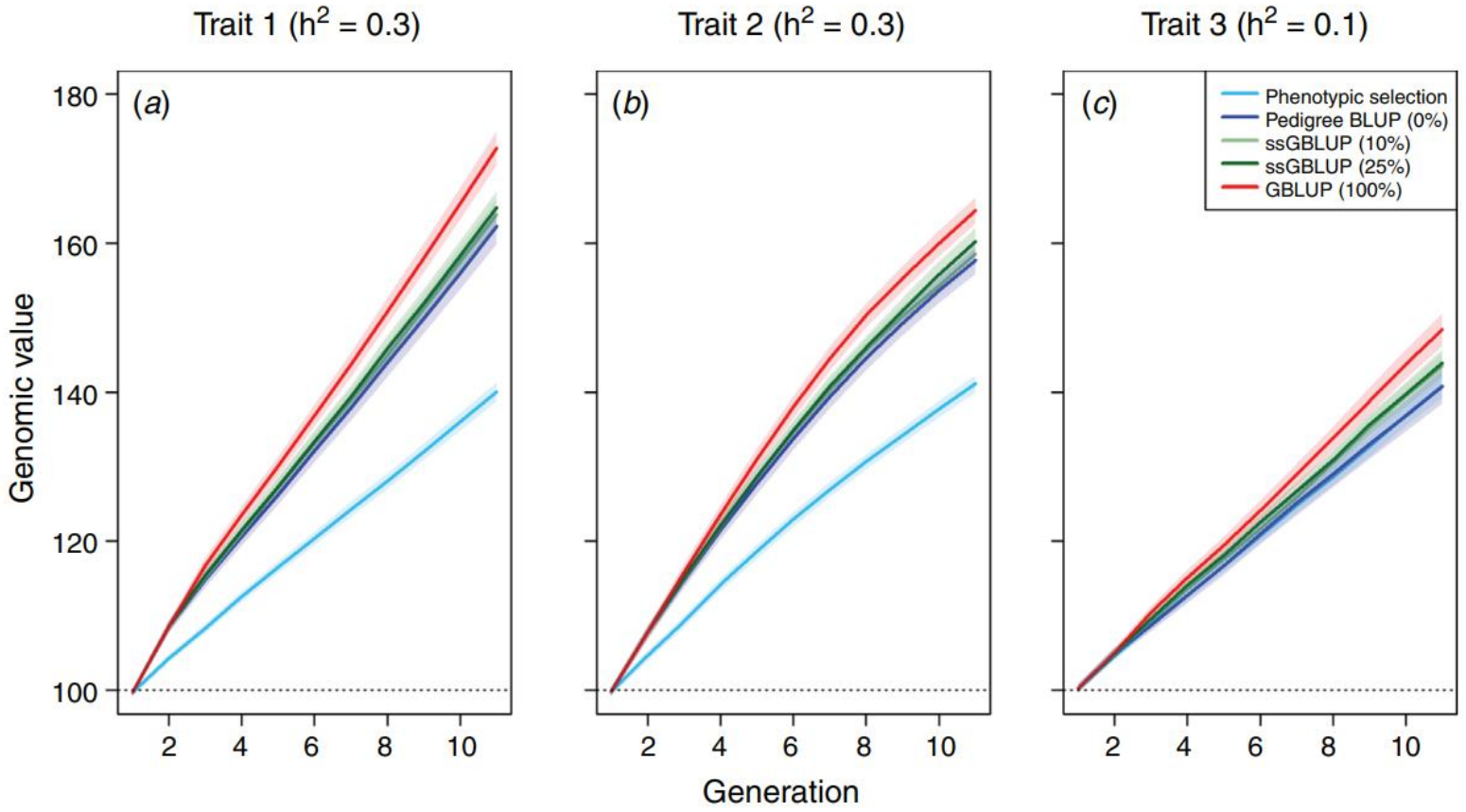
# Test crossing scheme in maize



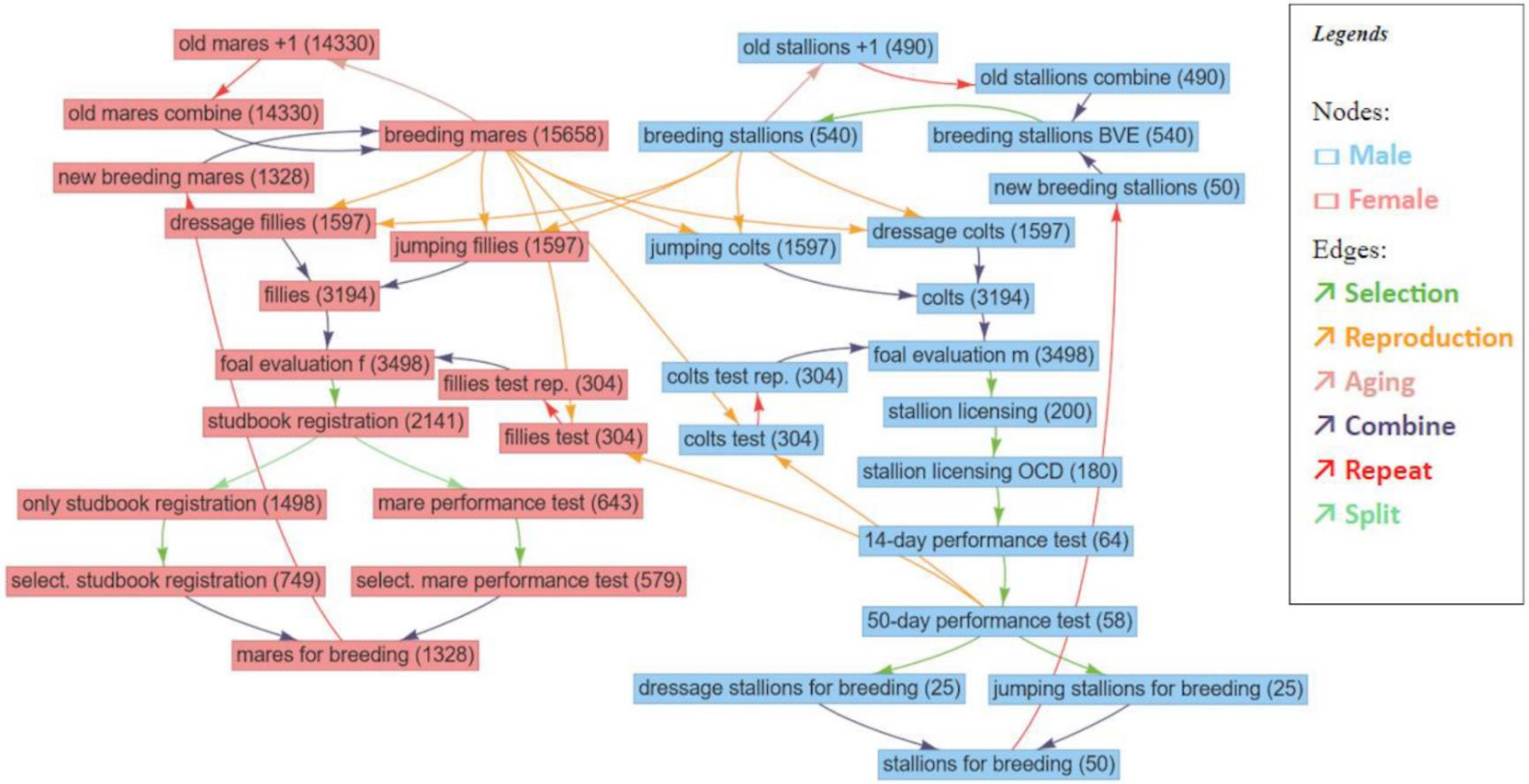
# Pig breeding (Pook et al. 2021)



# Pig breeding (Pook et al. 2021)

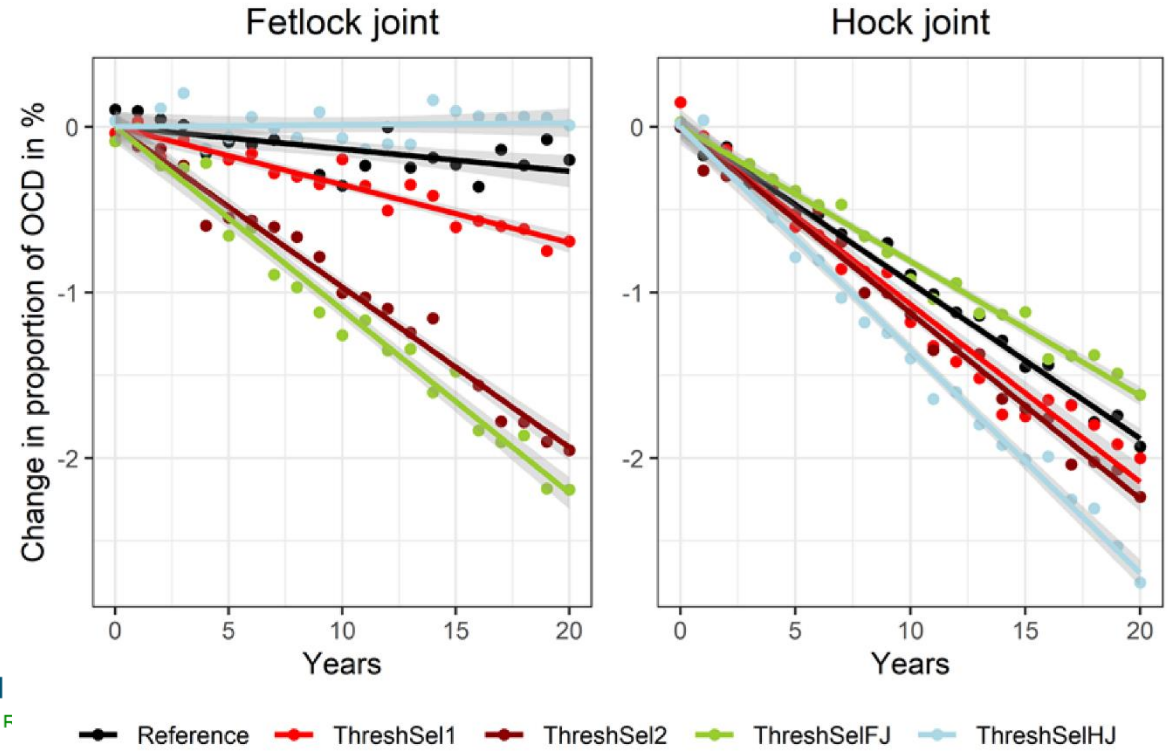


# Horse breeding scheme (Buettgen et al. 2020)

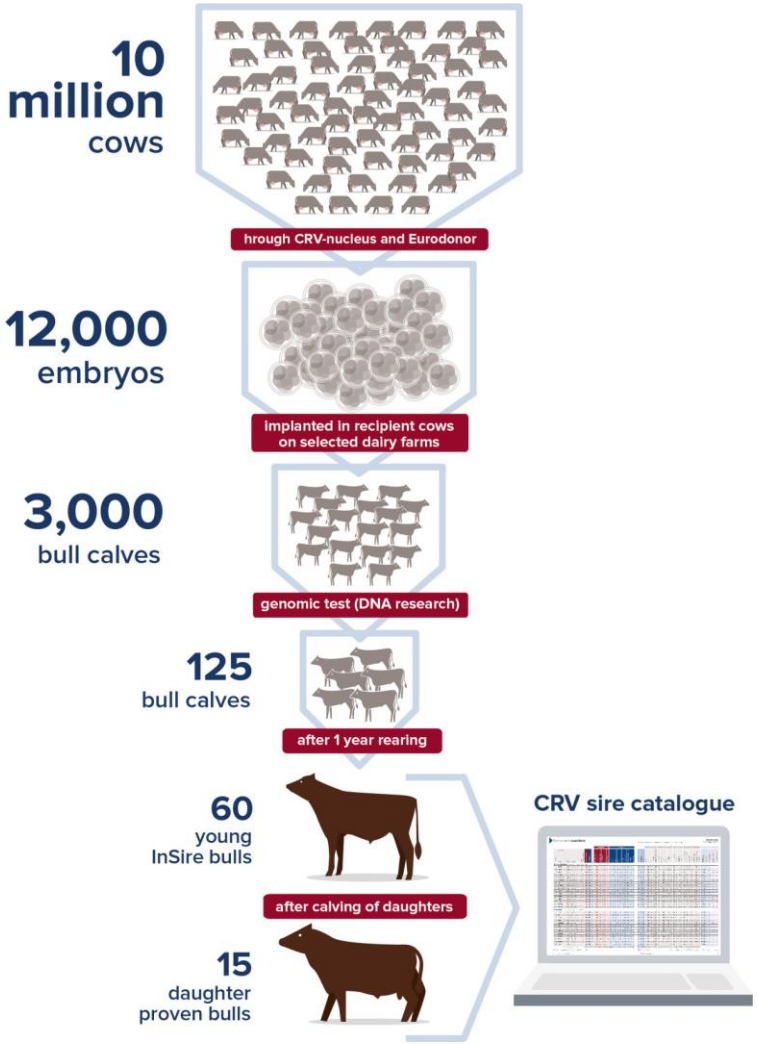


# Horse breeding scheme (Buettgen et al. 2020)

- Impact of changes to the breeding program
- Exclude horses with osteochondritis dissecans from selection



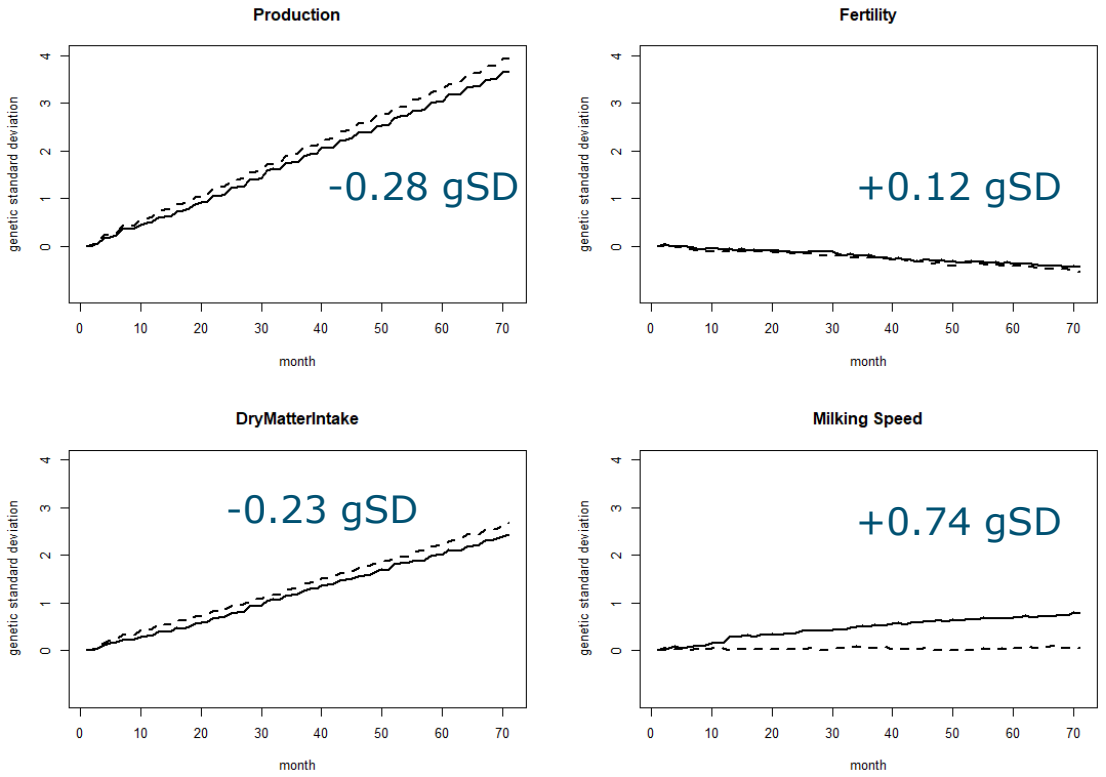
# Dairy cattle breeding (CRV)




WAGENINGEN
UNIVERSITY & RESEARCH

# Dairy cattle breeding (CRV)

- Threshold selection for fertility & milking speed
- Fertility negatively correlated with production traits



- Various small exemplary script in Guidelines (section 6)
- Exemplary script on my GitHub:
  - https://github.com/tpook92/MoBPS/tree/master/Exemplary_scripts
- Tobias Github:
  - https://github.com/tobiasniehoff/exploring_MoBPS

# Questions?

# Getting started with MoBPS

- MoBPS is mostly used as an R-package
- Most flexible & efficient
- Not the easiest to get started!
- User-Manual: https://github.com/tpook92/MoBPS/blob/master/Guidelines%20to%20MoBPS.pdf

```
breeding.diploid <- function(population, mutation.rate = 10^-8, remutation.rate = 10^-5, recombination.rate = 1,
          selection.m = "random", selection.f = NULL, new.selection.calculation = TRUE, selection.function.matrix = NULL,
          selection.size = 0, ignore.best = 0, breeding.size = 0, breeding.sex = NULL, breeding.sex.random = FALSE,
          used.generations.m = 1, used.generations.f = NULL, relative.selection = FALSE, class.m = 0, class.f = 0,
          add.gen = 0, recom.f.indicator = NULL, recom.f.polynom = NULL, duplication.rate = 0,
          duplication.length = 0.01, duplication.recombination = 1, new.class = 0L, bve = FALSE, sigma.e = NULL, sigma.g = 100,
          new.bv.child = "mean", computation.A = "vanRaden", delete.haplotypes = NULL, delete.individuals = NULL,
          fixed.breeding = NULL, fixed.breeding.best = NULL, max.offspring = Inf, store.breeding.totals = FALSE, forecast.sigma.g = TRUE,
          multiple.bve = "add", multiple.bve.weights = 1, store.bve.data = FALSE, fixed.assignment = FALSE,
          reduce.group = NULL, reduce.group.selection = "random", selection.critera = c(TRUE,TRUE), selection.criteria.type = c("bve", "bve"),
          same.sex.activ = FALSE, same.sex.sex = 0.5, same.sex.selfing = TRUE, selfing.mating = FALSE, selfing.sex = 0.5,
          praeimplantation = NULL, heritability = NULL, multiple.bve.scale = FALSE, use.last.sigma.e = FALSE,
          save.recombination.history = FALSE, martini.selection = FALSE, BGLR.bve = FALSE, BGLR.burnin = 500,
          BGLR.iteration = 5000, copy.individual = FALSE, dh.mating = FALSE, dh.sex = 0.5, n.observation = 1,
          bve.0isNA = TRUE, phenotype.bv = FALSE, standardize.bv = FALSE, standardize.bv.level = 100,
          standardize.bv.gen = 1, delete.same.origin = FALSE, remove.effect.position = FALSE, estimate.u = FALSE,
          BGLR.print = FALSE, new.phenotype.correlation = NULL, new.breeding.correlation = NULL, estimate.add.gen.var = FALSE,
          estimate.pheno.var = FALSE, best1.from.group = NULL, best2.from.group = NULL, best1.from.cohort = NULL,
          best2.from.cohort = NULL, add.class.cohorts = TRUE, store.comp.times = TRUE, store.comp.times.bve = TRUE,
          store.comp.times.generation = TRUE, special.comb = FALSE, max.auswahl = Inf, predict.effects = FALSE,
          SNP.density = 10, use.effect.markers = FALSE, use.effect.combination = FALSE, import.position.calculation = NULL,
          special.comb.add = FALSE, BGLR.save = "RKHS", BGLR.save.random = FALSE, ogc = FALSE, ogc_cAc = NA, emmreml.bve = FALSE,
          sommer.bve = FALSE, breedR.bve = FALSE, breedR.groups = NULL, nr.edits = 0, gene.editing.offspring = FALSE,
          gene.editing.best = FALSE, gene.editing.offspring.sex = c(TRUE,TRUE), gene.editing.best.sex = c(TRUE,TRUE),
          gwas.u = FALSE, approx.residuals = TRUE, sequenceZ = FALSE, maxZ = 5000, maxZtotal = 0, delete.sex = 1:2,
          gwas.group.standard = FALSE, y.gwas.used = "pheno", gen.architecture.m = 0, gen.architecture.f = NULL,
          add.architecture = NULL, ncore = 1, ncore.generation = 1, Z.integer = FALSE, store.effect.freq = FALSE,
          backend = "doParallel", randomSeed = NULL, randomSeed.generation = NULL, Rprof = FALSE, miraculix = FALSE,
          miraculix.mult = NULL, fast.compiler = 0, miraculix.cores = 1, store.bve.parameter = FALSE, miraculix.chol = TRUE,
          best.selection.ratio.m = 1, best.selection.ratio.f = NULL, best.selection.criteria.m = "bv", best.selection.criteria.f = NULL,
          best.selection.manual.ratio.m = NULL, best.selection.manual.ratio.f = NULL, bve.class = NULL, parallel.generation = FALSE,
          name.cohort = NULL, display.progress = TRUE, max.ticks = Inf, combine = FALSE, repeat.mating = 1, time.point = 0,
          creating.type = 0, multiple.observation = FALSE, new.bv.observation = NULL, new.bv.observation.gen = NULL,
          new.bv.observation.cohorts = NULL, new.bv.observation.database = NULL, bve.gen = NULL, bve.cohorts = NULL,
          bve.database = NULL, sigma.e.gen = NULL, sigma.e.cohorts = NULL, sigma.e.database = NULL, sigma.g.gen = NULL,
          sigma.g.cohorts = NULL, sigma.g.database = NULL, gwas.gen = NULL, gwas.cohorts = NULL, gwas.database = NULL,
          bve.insert.gen = NULL, bve.insert.cohorts = NULL, bve.insert.database = NULL, reduced.selection.panel.m = NULL,
          reduced.selection.panel.f = NULL, breeding.all.combination = FALSE, depth.pedigree = Inf, copy.individual.keep.bve = TRUE,
          bve.avoid.duplicates = TRUE, report.accuracy = TRUE, share.genotyped = 1, singlestep.active = FALSE,
          remove.non.genotyped = TRUE, added.genotyped = 0, fast.uhat = FALSE, offspring.bve.parents.gen = NULL,
          offspring.bve.parents.database = NULL, offspring.bve.parents.cohort = NULL, offspring.bve.offspring.gen = NULL,
          offspring.bve.offspring.database = NULL, offspring.bve.offspring.cohort = NULL){
```

**Table of Contents**



WA UNIVE

# MoBPSweb

- Web-based application to enter a breeding program in a more intuitive way (www.mobps.de)

# Some general tips

- Change your password
- Frequently press “Save”
- Exemplary templates can provide inspiration
- Do not ignore these buttons:

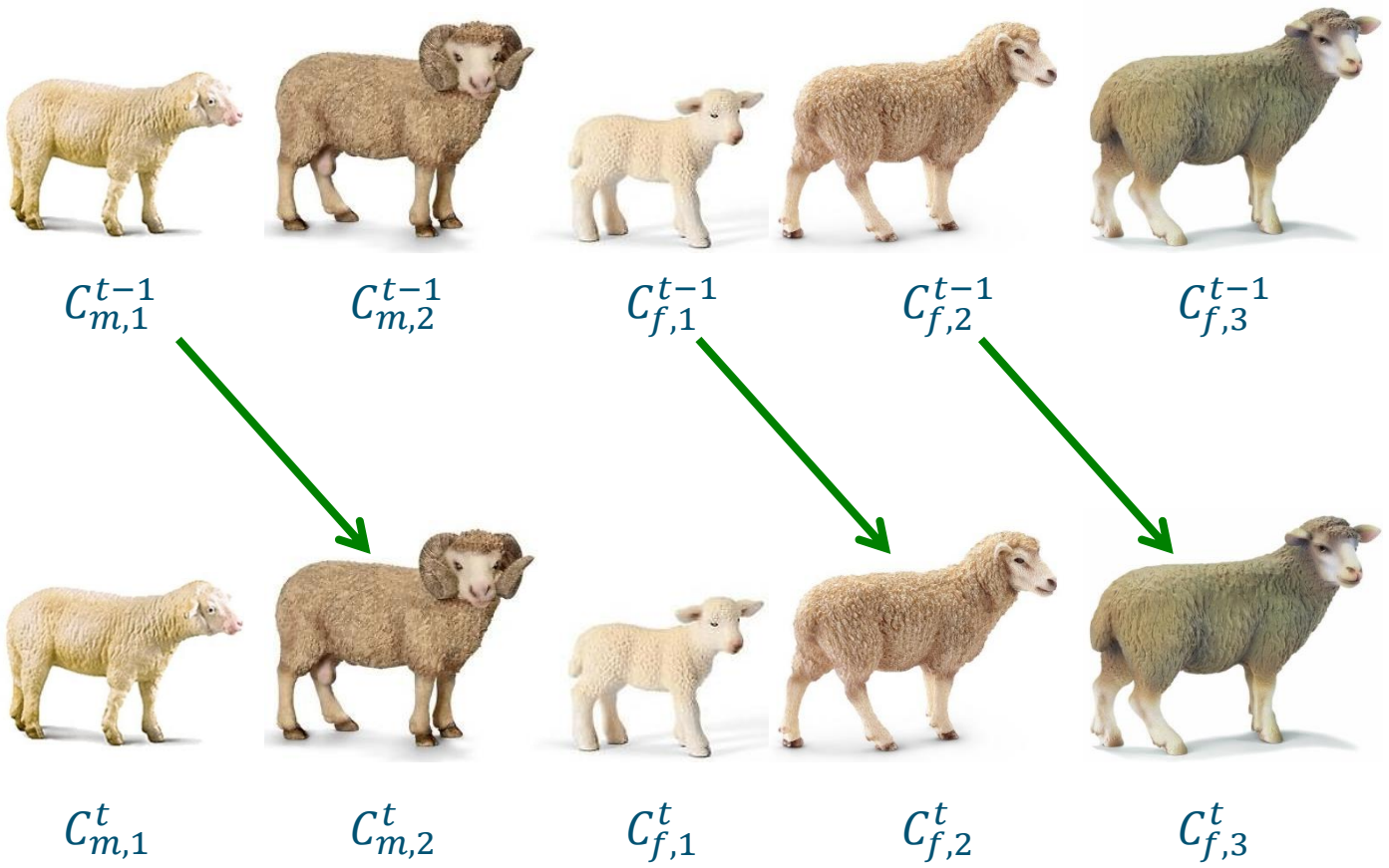
- Do not ignore warnings:

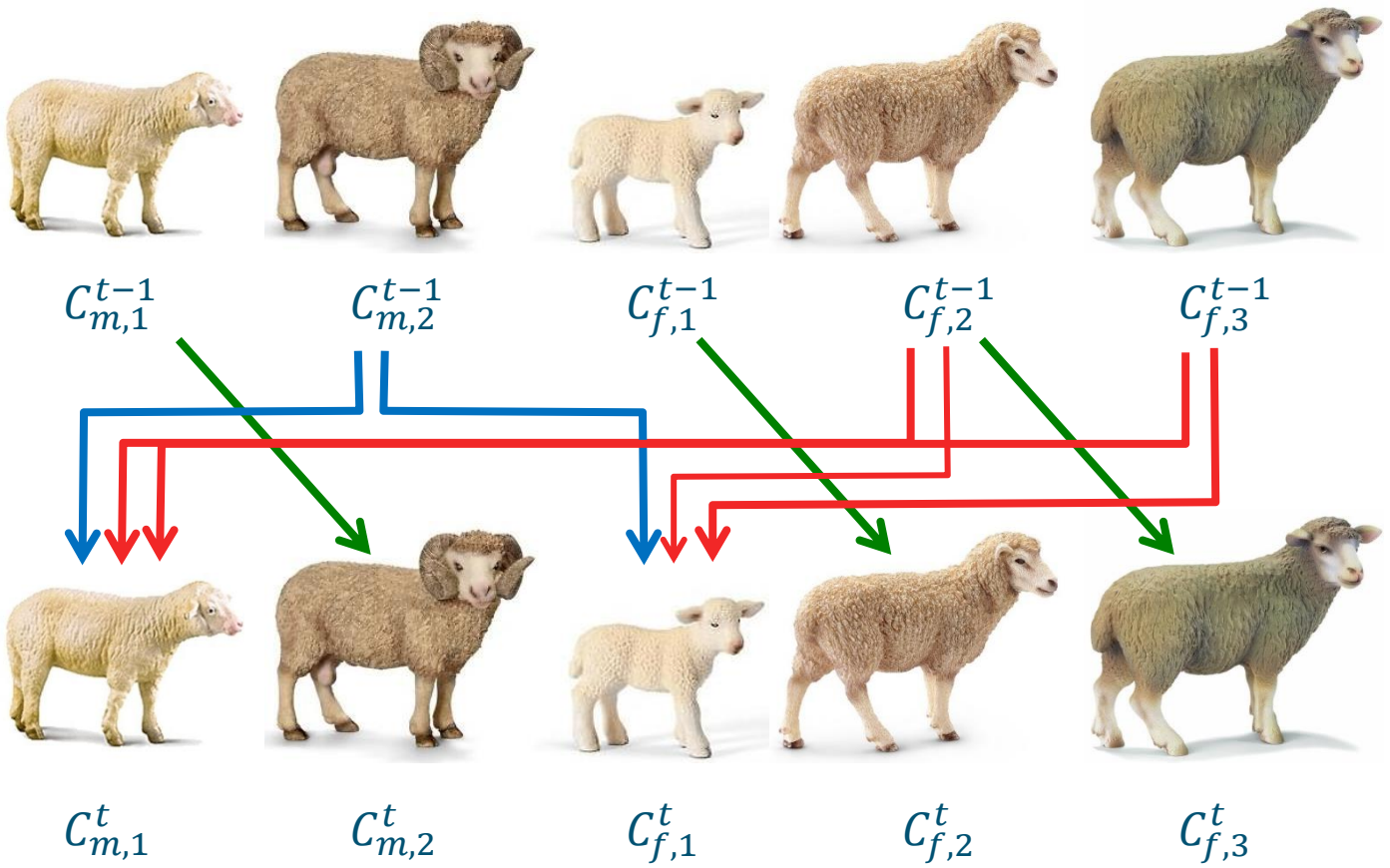
***Attention, there are 2 warnings! R Simulation most likely cannot be run unless they are fixed.***

**Warnings**

Warning 0 : Heritability for phenotype1 must be between 0 and 1

Warning 1 : Different sex between nodes SelectedCows and Bulls


$C_{m,1}^{t-1}$
$C_{m,2}^{t-1}$
$C_{f,1}^{t-1}$
$C_{f,2}^{t-1}$
$C_{f,3}^{t-1}$
$C_{m,1}^{t}$
$C_{m,2}^{t}$
$C_{f,1}^{t}$
$C_{f,2}^{t}$
$C_{f,3}^{t}$


$C_{m,1}^{t-1}$
$C_{m,2}^{t-1}$
$C_{f,1}^{t-1}$
$C_{f,2}^{t-1}$
$C_{f,3}^{t-1}$
$C_{m,1}^{t}$
$C_{m,2}^{t}$
$C_{f,1}^{t}$
$C_{f,2}^{t}$
$C_{f,3}^{t}$

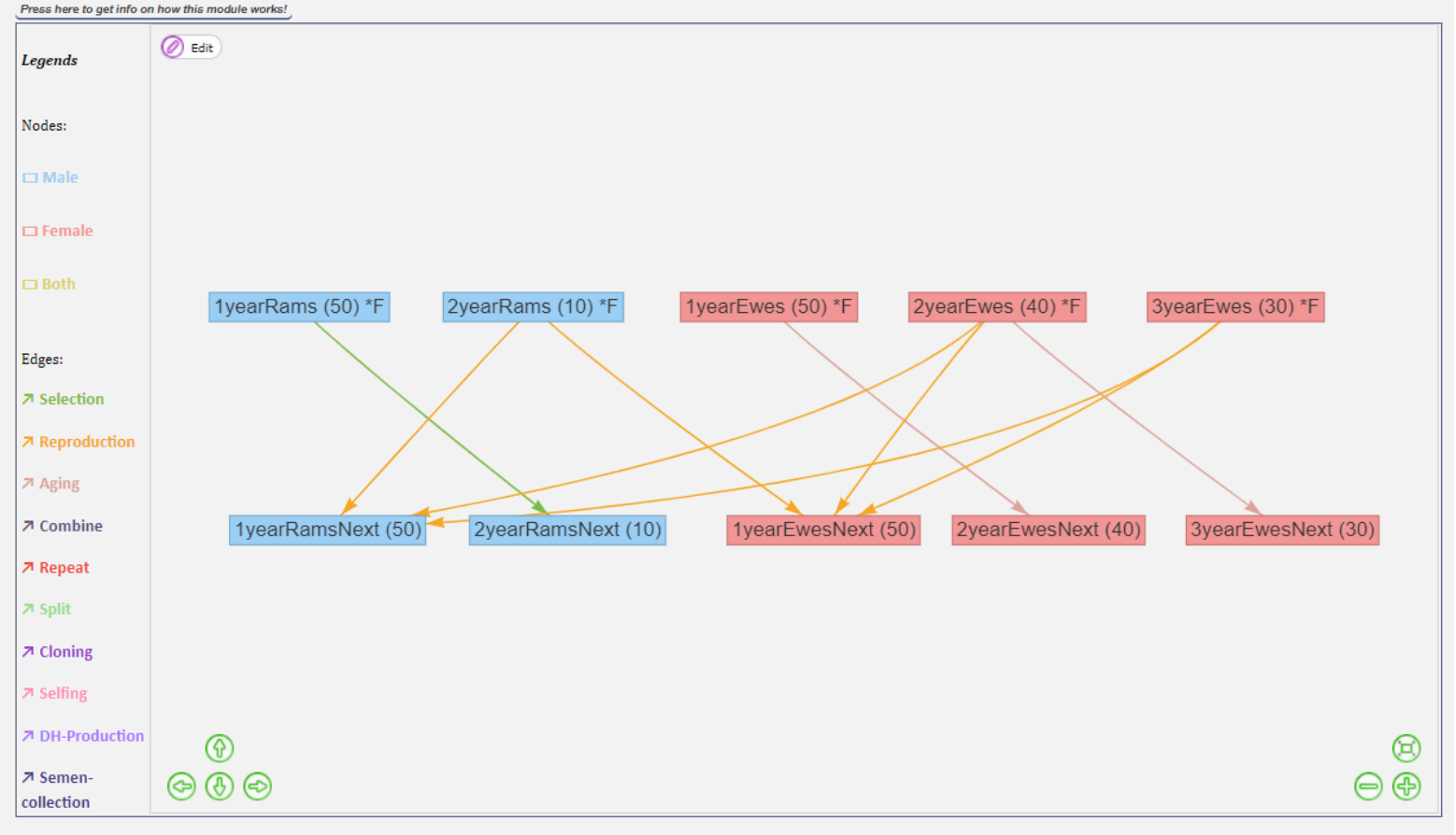
# Task 1: Sheep Breeding Program

- Simulate one cycle of the sheep breeding program with:
  - 50 1-year rams & 10 2-year rams
  - 50 1-year ewes & 40 2-year ewes & 30 3-year ewes
- One trait (Meat)
  - Phenotypic mean: 100
  - Heritability $h^2 = 0.3$, 1000 QTLs
- Selection on the male side is based on phenotypes
- Selection on the female side is done at random
- 2-year old rams and 2 & 3 –year old ewes are used for reproduction
- Simulate a genome with 5 chromosomes with 1000 SNPs / 1 Morgan

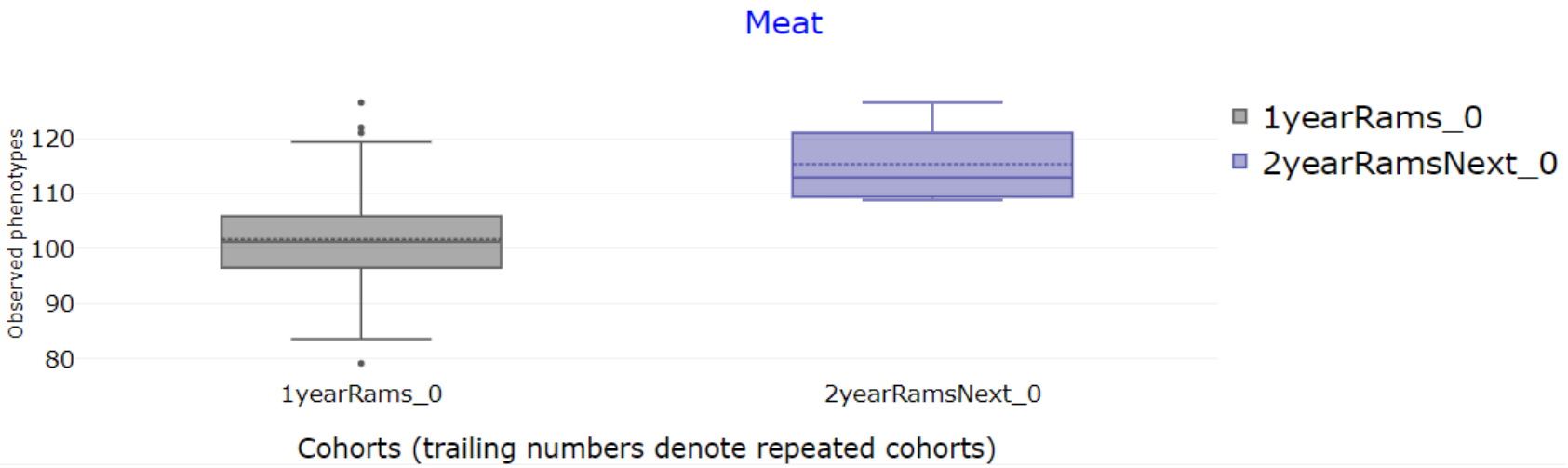
WAGENINGEN
UNIVERSITY & RESEARCH

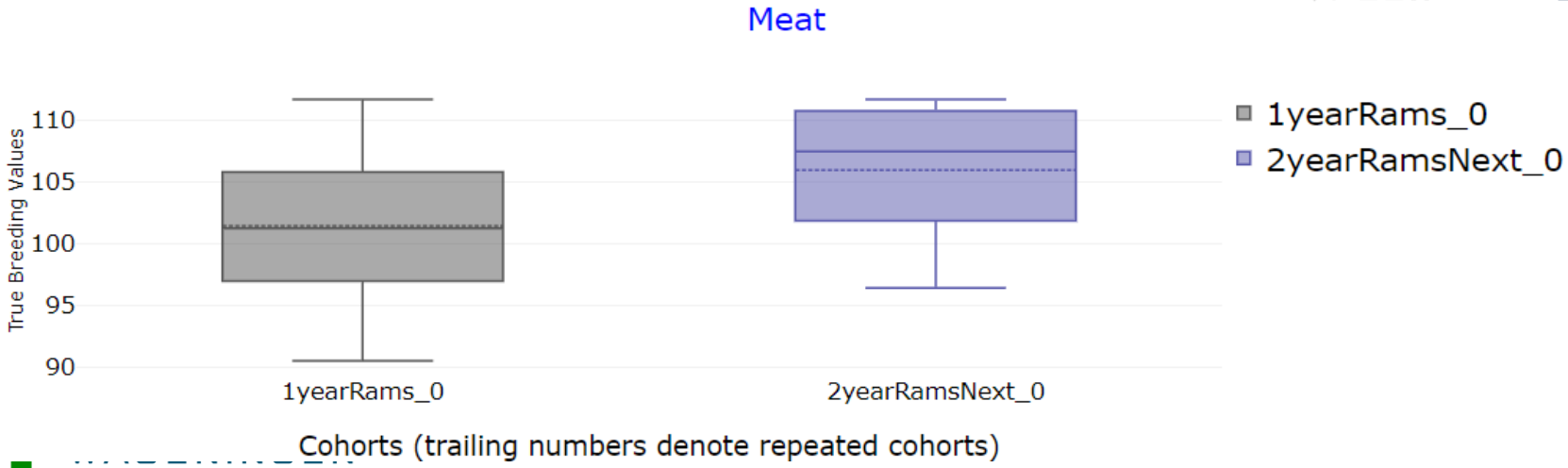
# Solution Task 1

- All details are given in the template: „Simple Sheep“



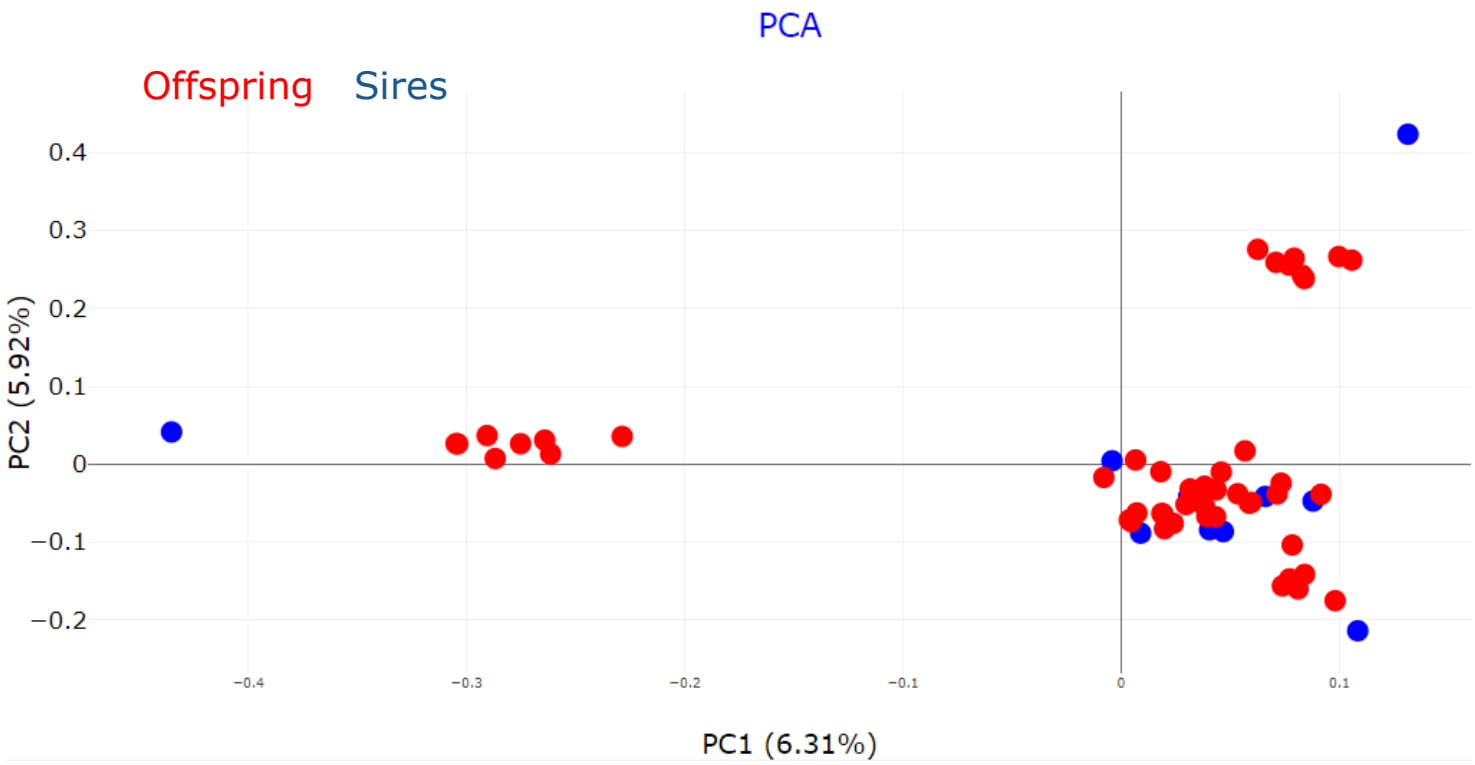
# Solution Task 1



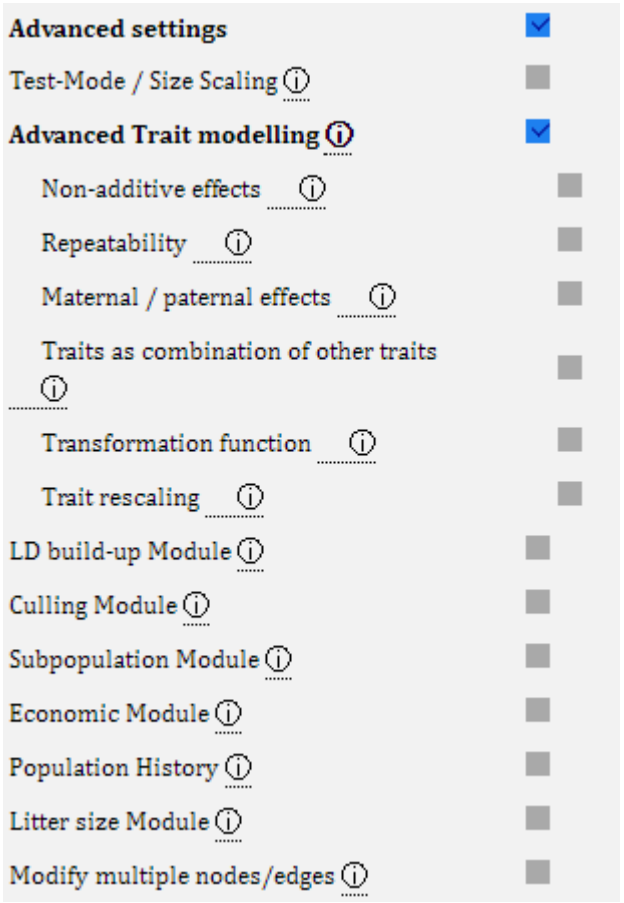



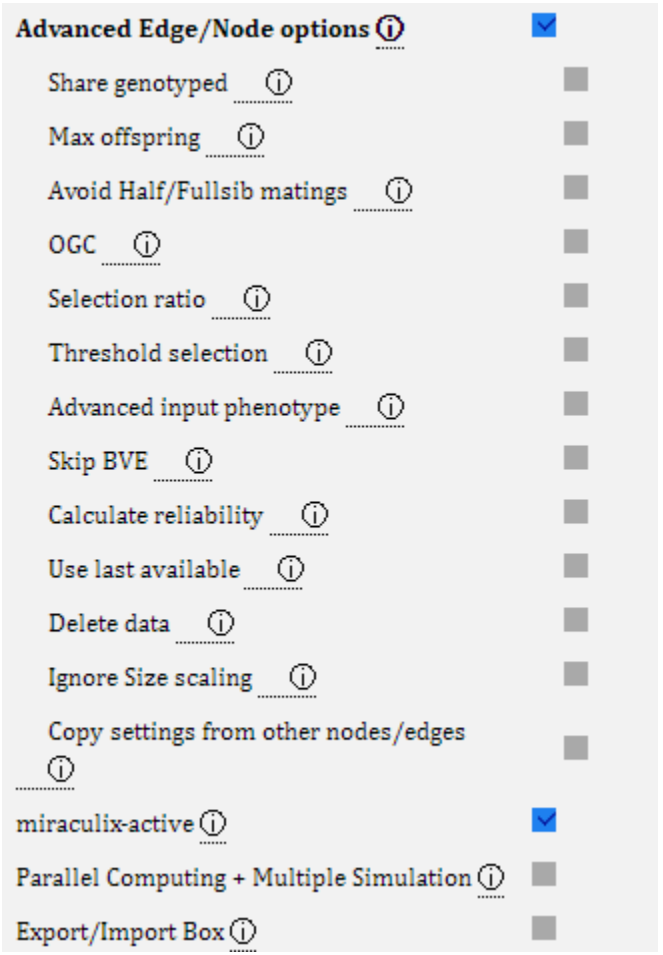

UNIVERSITY & RESEARCH

# Solution Task 1



# Advanced options in MoBPSweb

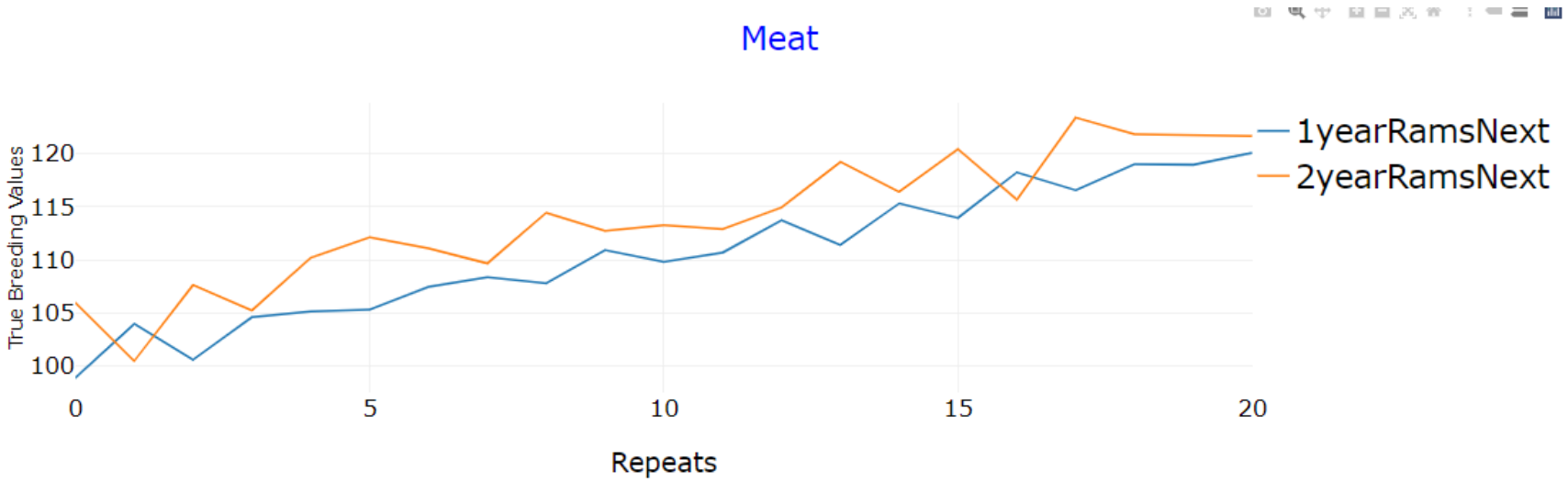


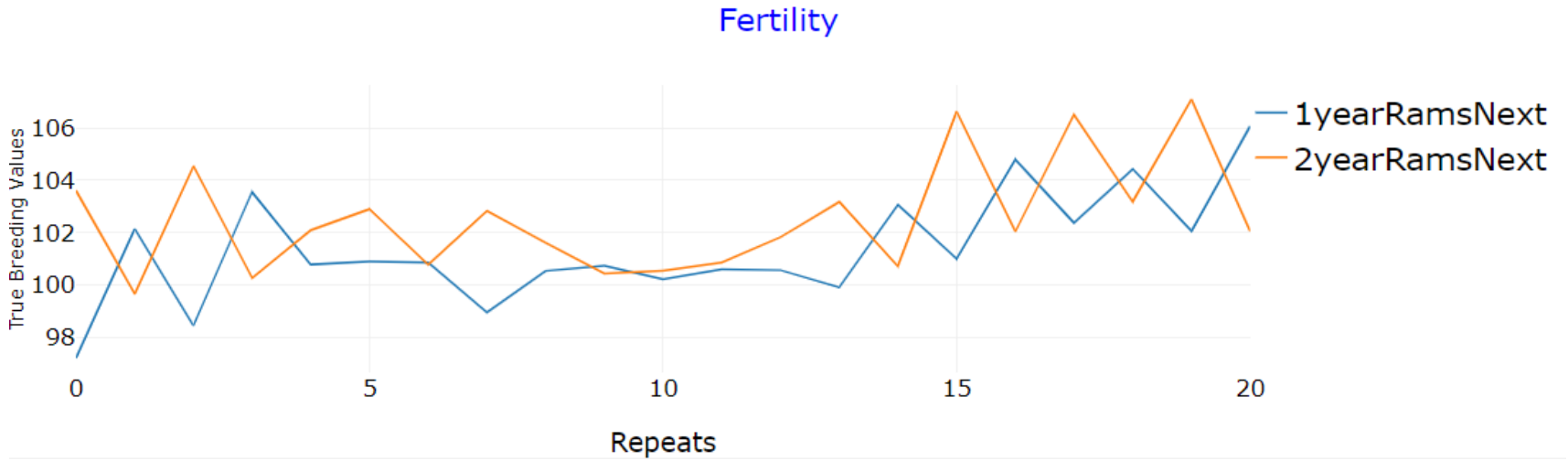



# Task 2: Make it more realistic

- Perform an LD build up to ensure that founders are related
  - Use 5 generations with 100 individuals
- Add a second trait (Fertility)
  - Phenotypic Mean: 100 & Heritability $h^2 = 0.2$
  - Only two and three-year-old animals should be phenotyped
  - Traits have a genetic correlation of 0.2
  - Residual effects are not correlated
- Simulate 20 cycles of the breeding program
- The number of offspring from a given mating (litter) should be 2 with a probability of 50%, 3 with a probability of 30% and 4 with a probability of 20%
- Use genomic selection for the selection of rams
  - Use all animals from the current cycle in the breeding value estimation
  - Put equal weight on both traits (use the scaling: „Per Genomic Value SD“)
  - Assume all individuals to be genotyped
- Use the IlluminaOvineSNP50 array as an underlying genomic map

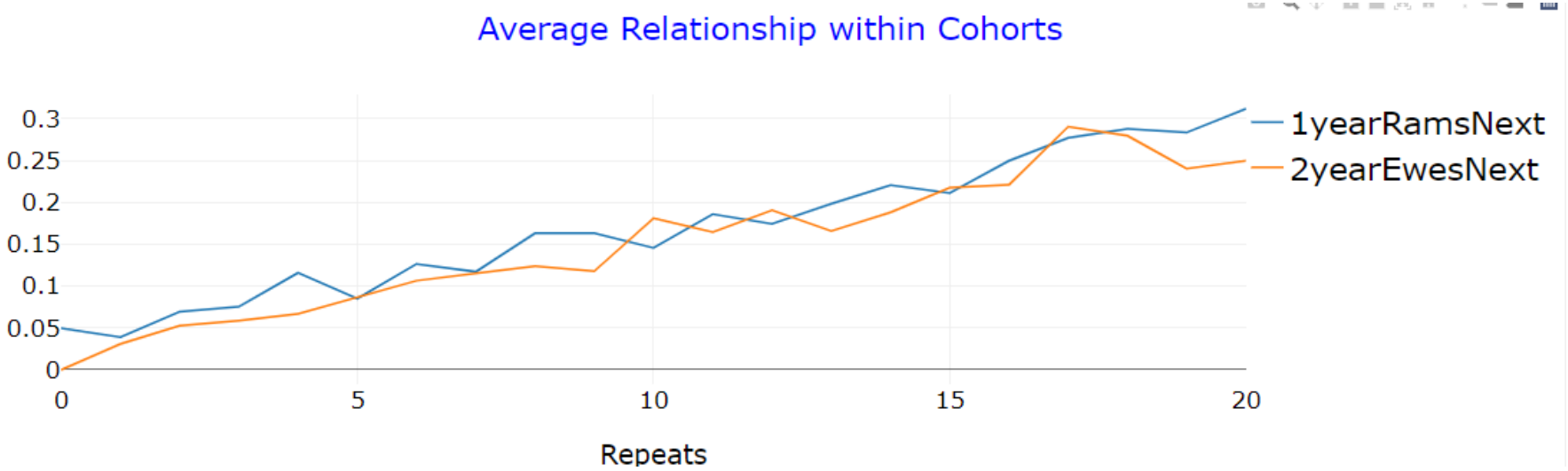
# Solution Task 2

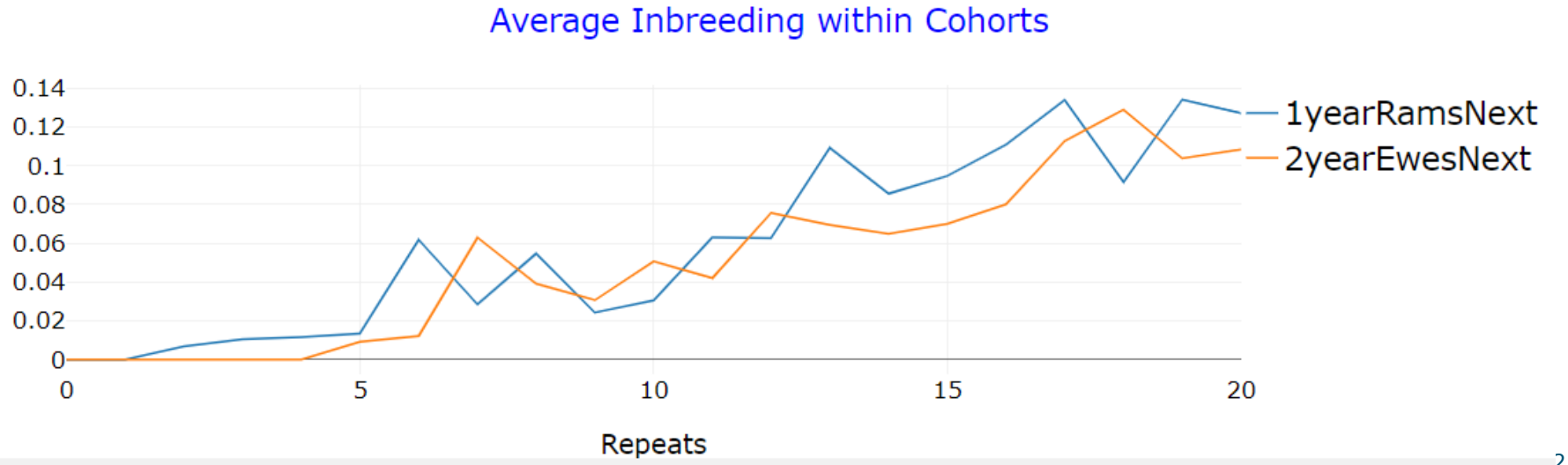



WAGENINGEN
UNIVERSITY & RESEARCH

# Solution Task 2

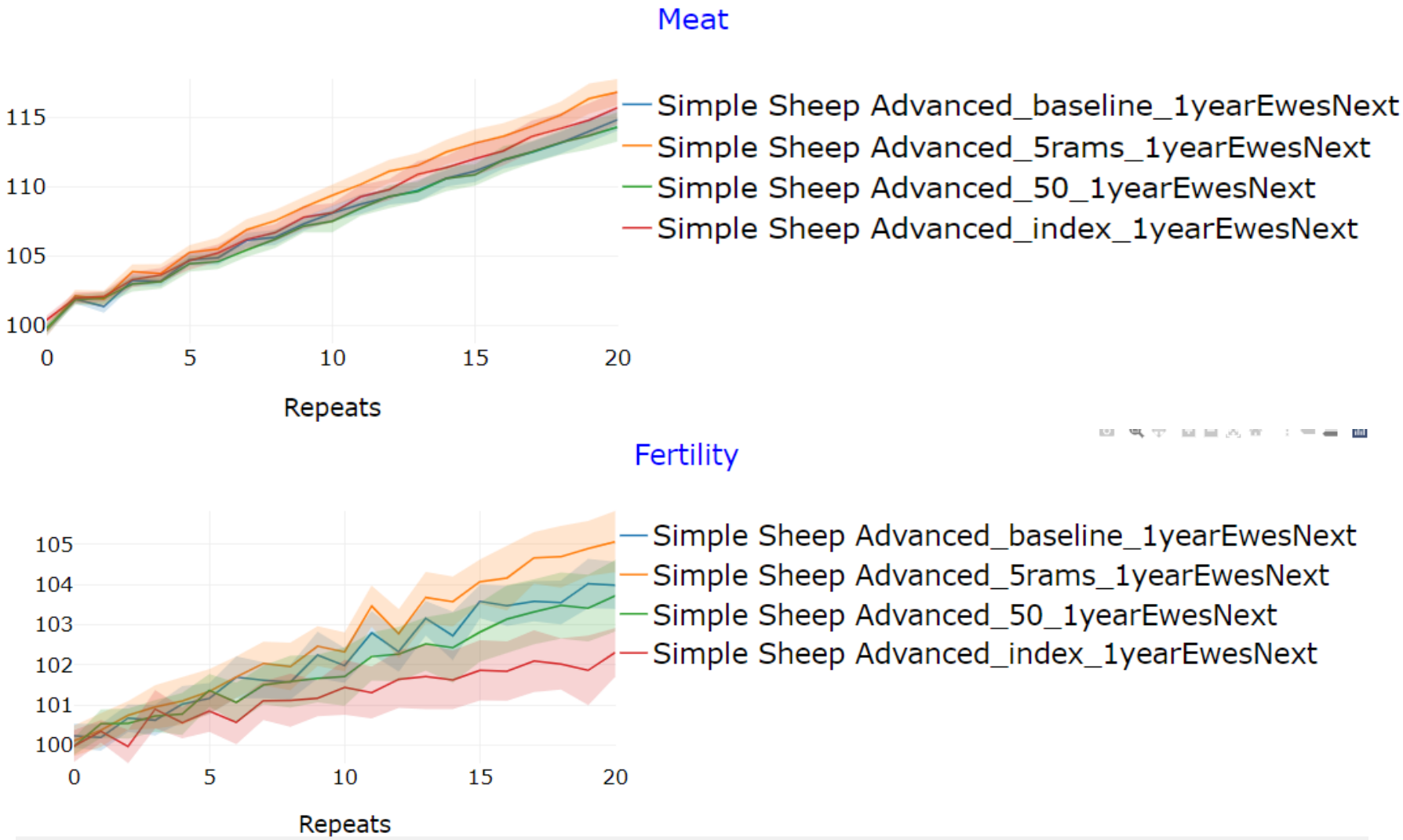


# Task 3: Comparison of scenarios

- Consider the following different variations of the breeding scheme as separate projects:
  1. Only 5 rams are selected for reproduction
  2. Only 50% of all ewes are genotyped
  3. Use a selection index with three times as much weight on the meat trait
- Simulate each breeding scheme at least 5 times
- Use the „Compare Project“ module to compare the different scenarios in regard to genetic gain and inbreeding
- Hint: Confidence intervals could be benefical to decide which differences are significant and which are not!

# Solution Task 3



# Solution Task 3

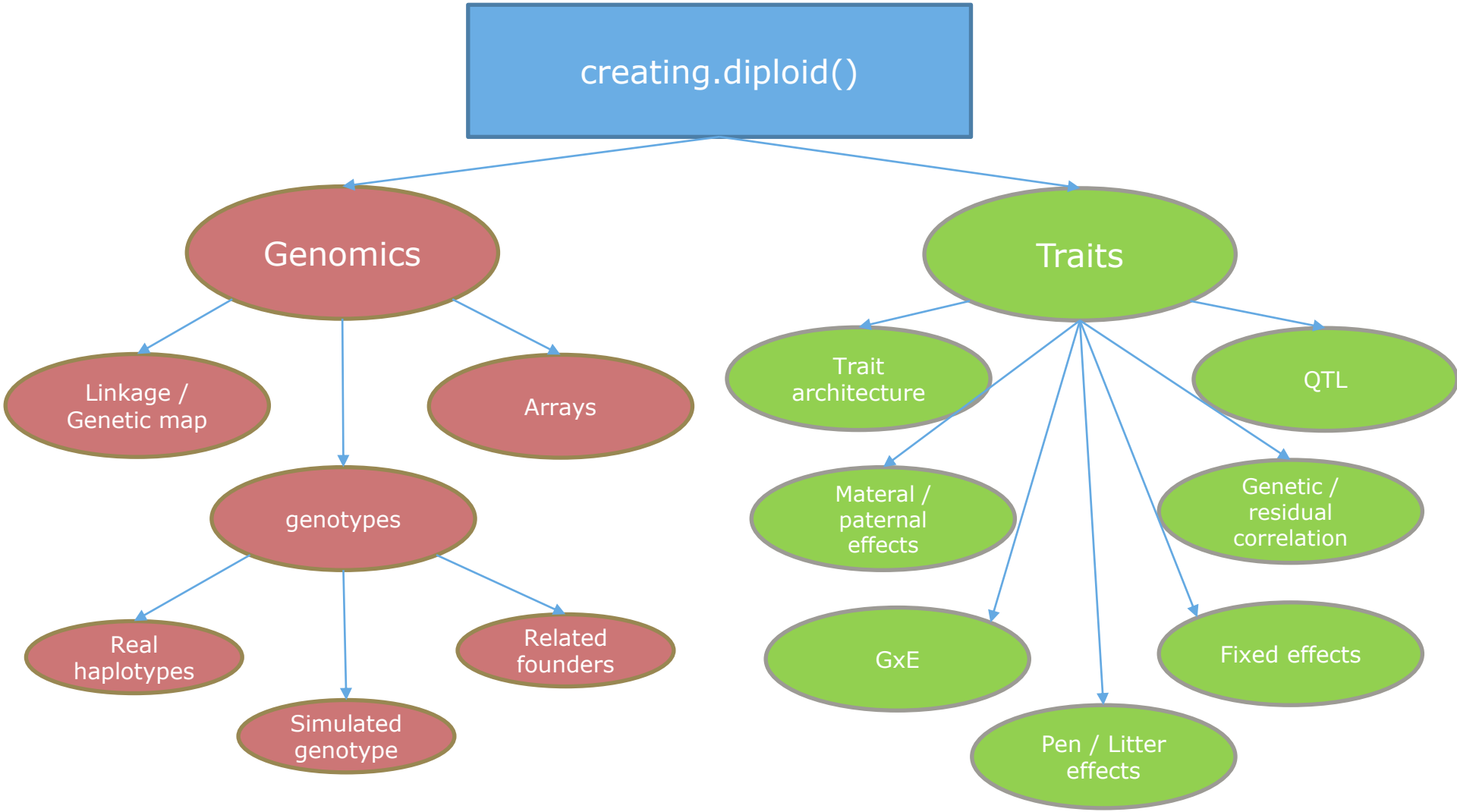
- Even 25 simulations over 20 generations are barely enough to distinguish between scenarios
  - High variance in the outcome of simulations
  - Small differences between scenarios
- Breeding in practice also has random factors!

# Task 4

- Install our packages
  - MoBPS (V1.10.48)
  - For Windows this requires Rtools (https://cran.r-project.org/bin/windows/Rtools/rtools40.html) on some systems
- In case you want some common genetic maps:
  - MoBPSmaps V0.1.13
- For computational efficiency:
  - RandomFieldsUtils: V1.0.6 for Linux / V0.6.6 for Windows
  - Miraculix: V1.0.5 for Linux / V1.0.0.1 for Windows
  - Examples in this workshop will be so small that this is not needed!

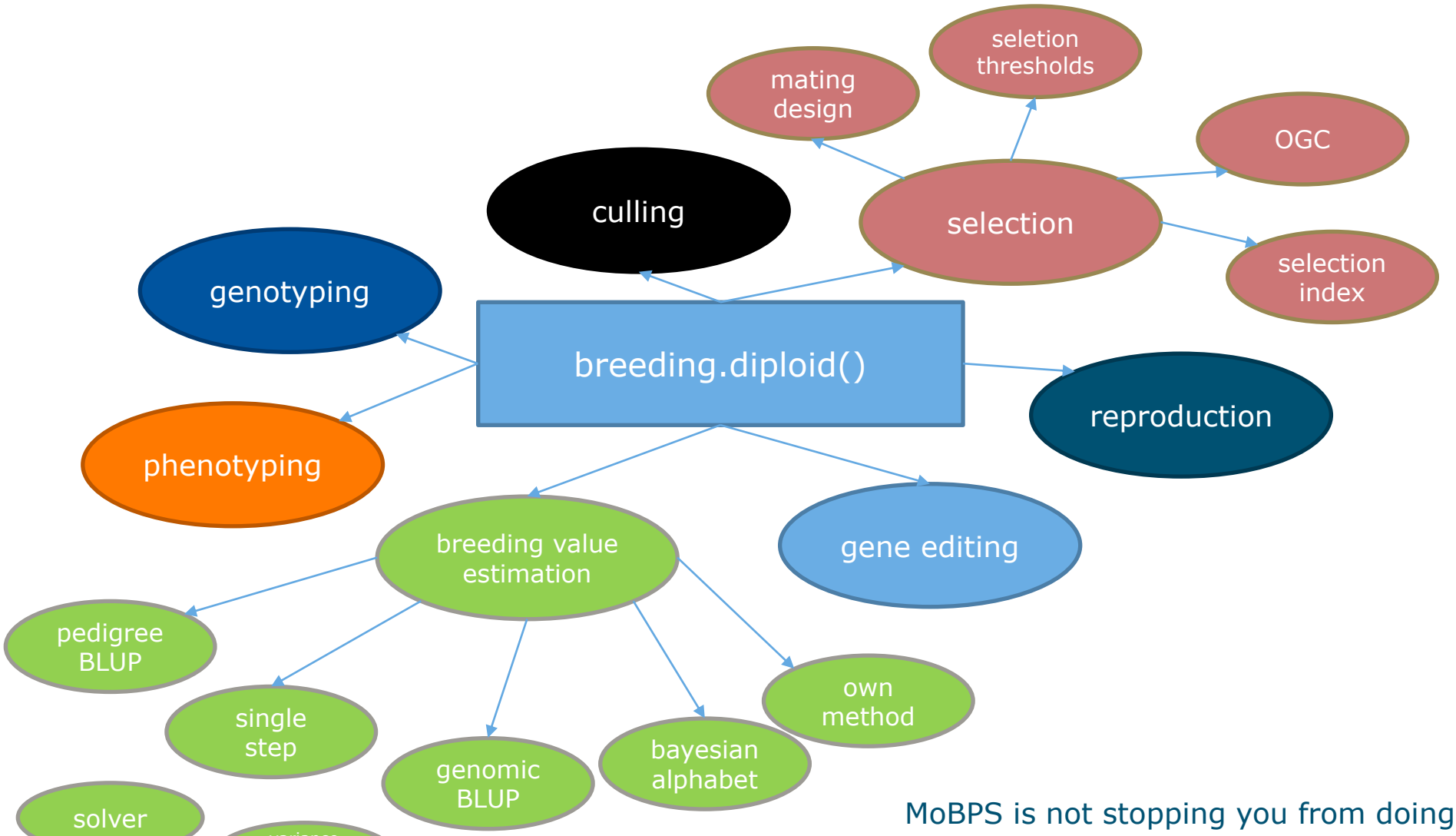
# Steps of your simulation

# Initialization of the founder population




→ The closer your design is to reality the more reliable your later results will be

# Simulation of breeding actions

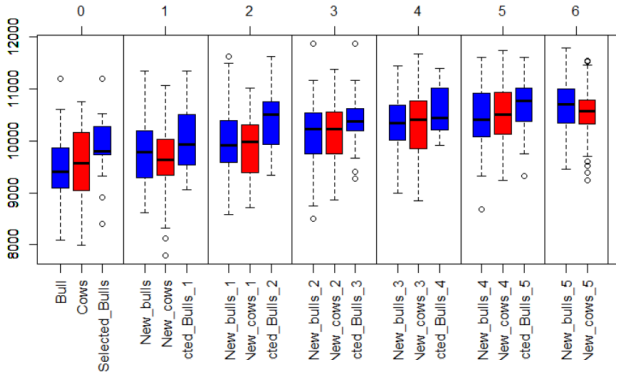


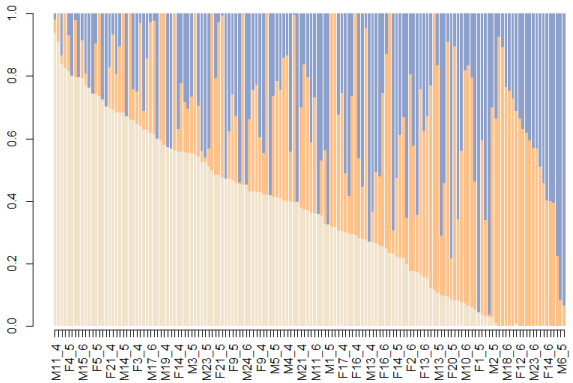

MoBPS is not stopping you from doing practically impossible things!

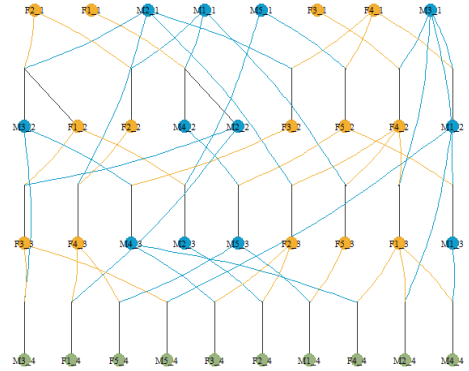
# Output of a simulation

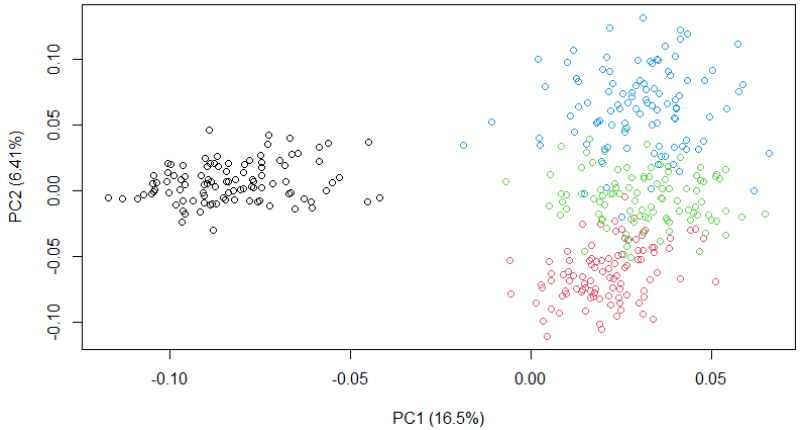
- Highly compressed object (population list)
- This contains information on all individuals ever generated in our simulation and is an expected input for breeding.diploid()
- We know the exact truth!
  - Genotypes / haplotypes
  - Where are the QTLs in the genome
  - What are the underlying genomic values of individuals
  - When/Where were recombination events happen
  - Pedigree without errors

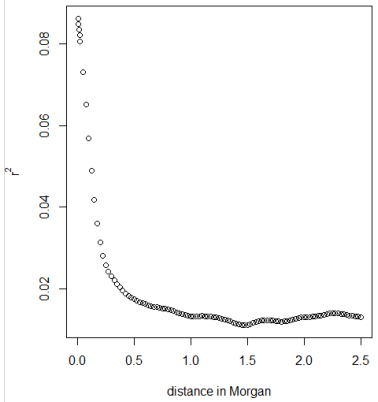
# Analysis functions

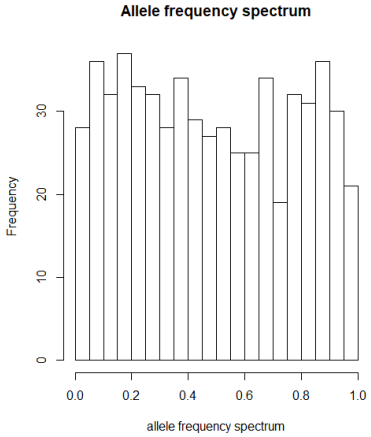


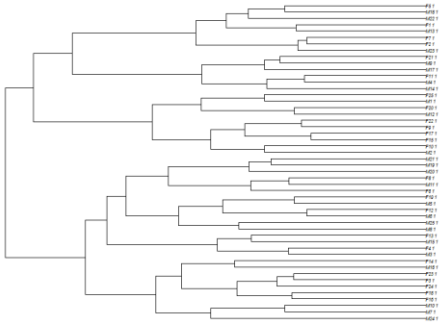



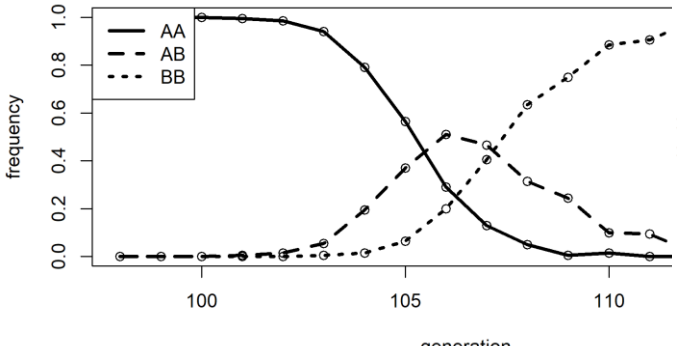



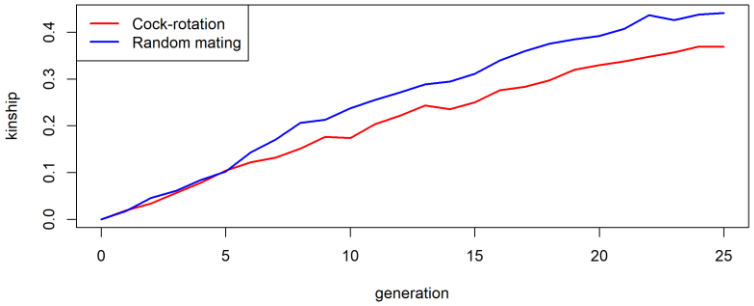




# Basic example

```
# Generation of a founder population
# with 100 individuals and 10,000 SNPs
# The genome contains 5 chromosomes with a length of 2 Morgan each
# Generation of one trait with 60 underlying QTL
# of which 50 are purely additive and 10 have a dominate effect
# The genomic variance of the trait simulated to be 1
# The generated cohort is named "Founder"

pop <- creating.diploid(nsnp=10000, nindi=100,
                        chr.nr=5, chromosome.length=2,
                        n.additive=50, n.dominant=10,
                        name.cohort="Founder",
                        var.target = 1)

# Generate phenotypic observations for all individuals
# Residual variance is set to result in a heritablity of 0.5
pop <- breeding.diploid(pop, heritability=0.5,
                        phenotyping="all")
```

# Basic example

```
# Generate 100 offspring
# Use the top 20 male and top 20 female based on their BVE
# All male individuals from the "Founder" cohort are used
# as potential sires.
# All female individuals from the "Founder cohort are used
# as potential dams.
# The resulting cohort in named "Offspring".
pop1 <- breeding.diploid(pop, breeding.size=100,
                         selection.size=c(20,20),
                         selection.criteria = "bve",
                         selection.m.cohorts="Founder_M",
                         selection.f.cohorts="Founder_F",
                         name.cohort="Offspring")

# Same procedure, just with a higher selection intensity
# on the male side.
pop2 <- breeding.diploid(pop, breeding.size=100,
                         selection.size=c(5,20),
                         selection.criteria = "bve",
                         selection.m.cohorts="Founder_M",
                         selection.f.cohorts="Founder_F",
                         name.cohort="Offspring")
```

# How to figure out what does what?

- Enter ?breeding.diploid & ?creating.diploid in your R console
- Guidelines_to_MoBPS.pdf
  - This is a 120 page Manual!
  - Do not read it from front to back in one piece
    - Ctrl + F / Search
  - Depending on how you learn / what you need different sections might be more helpful

# Prints & warnings are your friend!

```
> population = breeding.diploid(population, selection.size = c(10,10))
No individuals for selection provided (male side). Use last available.
No individuals for selection provided (female side). Use last available.
Start selection procedure.
Selection male size:
Select 10 individuals out of  2.
Selection female size:
Select 10 individuals out of  3.
Warning messages:
1: In breeding.diploid(population, selection.size = c(10, 10)) :
  Less individuals available for selection than given in selection.size.
 Automatically reduce the number of selected individuals to 2
2: In breeding.diploid(population, selection.size = c(10, 10)) :
  Less individuals available for selection than given in selection.size.
 Automatically reduce the number of selected individuals to 3
```

# Position of individuals

- „gen“: Each new group of individuals is assigned to a generation
- „database“: Specification within a generation
  - database = cbind(5,1,20,30)
  - generation: 5
  - sex: 1 (Male)
  - individual number: 20 to 30
- „cohorts“: The names you assigned them to have

```
Start generation of new individuals (cohort: Offspring).
  |============================================================| 100%Successfully
 generated 50 individuals.

Added _M, _F to cohort names!
Successfully generated cohort: Offspring_M
Database position: 3 (gen), 1 (sex), 1 (first), 20 (last).
Successfully generated cohort: Offspring_F
Database position: 3 (gen), 2 (sex), 1 (first), 30 (last).
Generation of new individuals took 0.103 seconds
```

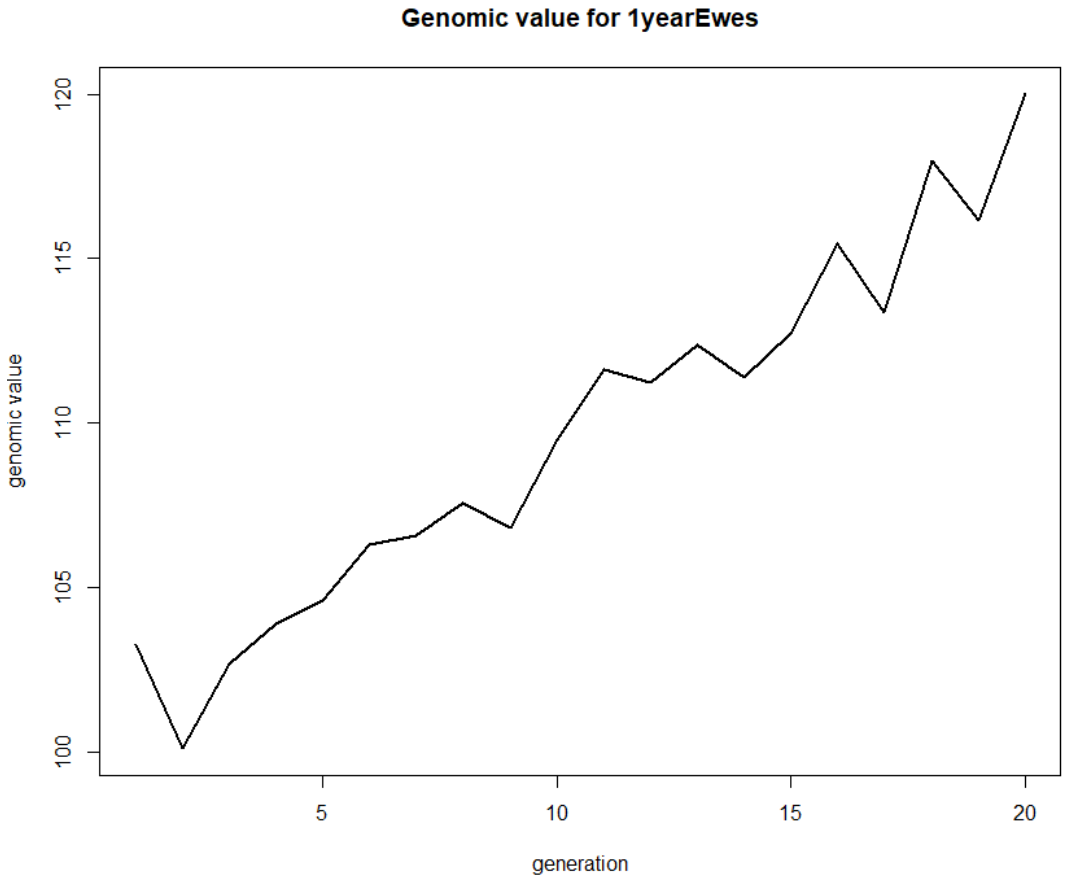
# Task 5: Switching to R

- Export the „Simple Sheep Advanced“ breeding scheme from the interface
- Simulate the scheme by use of *json.simulation()*
- Extract the average genomic value for the 1 year old ewes for the different cycles
    - Hint: get.cohorts & paste0() might be a helpful functions
- Perform the simulation from „Simple Sheep“ directly in R

# Solution Task 5

```
> summary(population)
Population size:
Total: 6060 Individuals
Of which 2370 are male and 3690 are female.
There are 22 generations
and 152 unique cohorts.
3780 individuals are copies of previously generated individuals.

Genome Info:
There are 26 unique chromosomes.
In total there are 5000 SNPs.
The genome has a total length of 24.30970424 Morgan.
The genome has a physical size of about: 2.431 GB

Trait Info:
There are 2 modelled traits.
Of which 2 have underlying QTL.
Trait names are:Meat Fertility
Highest correlation between genetics of traits is 0.2 (absolut value).
There are no interactions between residual effects.
Total time spent for generation: 16 seconds.

Time spent per step:
0.8 seconds for creation of founder population.
0.3 seconds for calculation of true genomic values.
0.1 seconds for phenotyping.
0.7 second for breeding value estimation.
0.8 seconds for selection.
13.3 seconds for generation of new individuals.
```

# Solution Task 5



# Solution Task 5

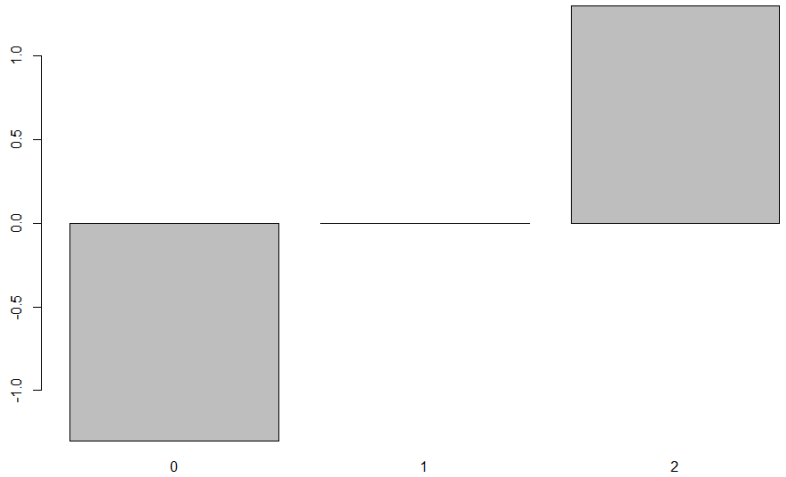
```
> get.cohorts(population, extended = TRUE)[,1:4]
              name             generation male individuals female individuals
1yearRams     "1yearRams"      "1"        "50"              "0"
2yearRams     "2yearRams"      "1"        "10"              "0"
1yearEwes     "1yearEwes"      "1"        "0"               "50"
2yearEwes     "2yearEwes"      "1"        "0"               "40"
3yearEwes     "3yearEwes"      "1"        "0"               "30"
1yearRamsNext "1yearRamsNext"  "2"        "50"              "0"
1yearEwesNext "1yearEwesNext"  "2"        "0"               "50"
2yearRamsNext "2yearRamsNext"  "2"        "10"              "0"
2yearEwesNext "2yearEwesNext"  "2"        "0"               "40"
3yearEwesNext "3yearEwesNext"  "2"        "0"               "30"
```

```
> summary(population)
Population size:
Total: 360 Individuals
Of which 120 are male and 240 are female.
There are 2 generations
and 10 unique cohorts.
80 individuals are copies of previously generated individuals.

Genome Info:
There are 5 unique chromosomes.
In total there are 5000 SNPs.
The genome has a total length of 5 Morgan.
The genome has a physical size of about: 0.4998 GB

Trait Info:
There is 1 modelled trait.
The trait has underlying QTL
The trait is named: Meat
Total time spent for generation: 0.7 seconds.

Time spent per step:
0.3 seconds for creation of founder population.
0.1 seconds for phenotyping.
0.3 seconds for generation of new individuals.
```

# On the generation of traits

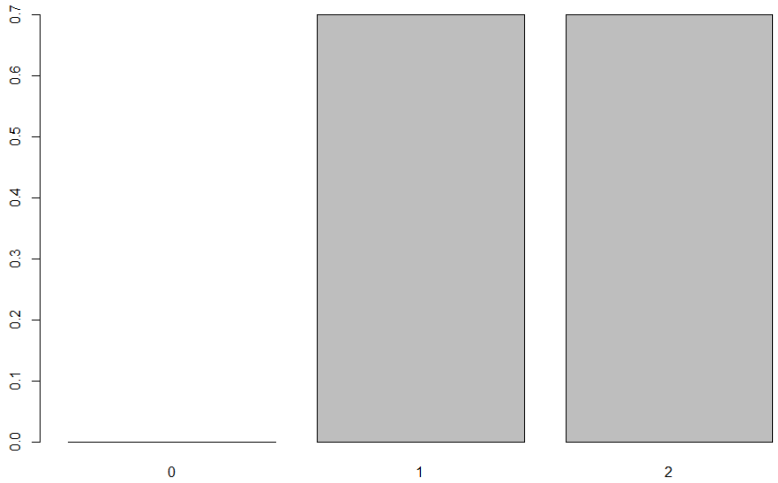
# On the generation of traits

- Predefined architectures:
  - Purely additive QTLs with effect size drawn from $N(0,1)$ (n.additive)
  - Purely additive QTLs with equal effect size (n.equal.additive)



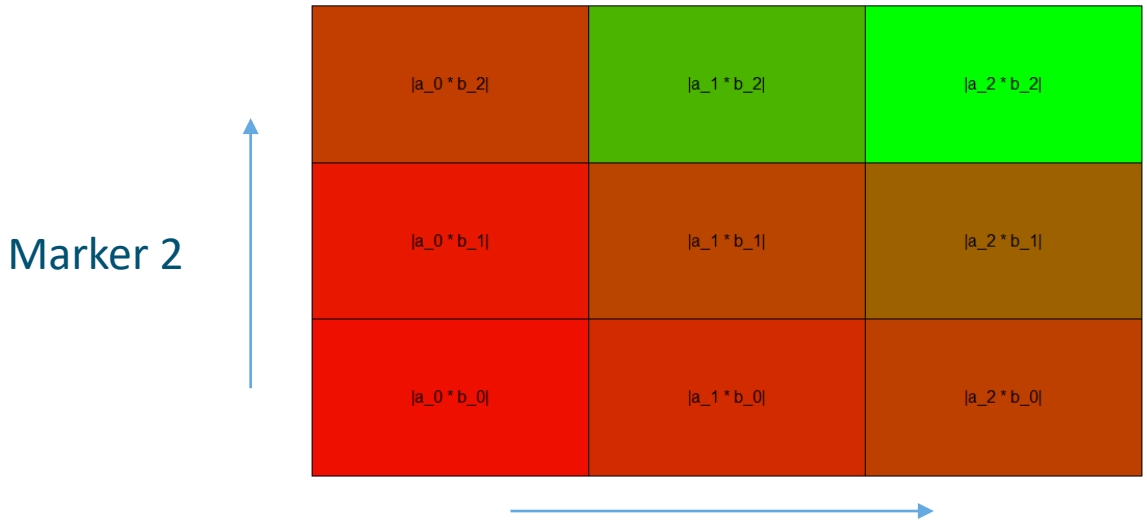
# On the generation of traits

- Predefined architectures:
  - Dominant QTLs with effect size drawn from $N(0,1)$ (n.dominant)
  - Dominant QTLs with equal effect size (n.equal.dominant)



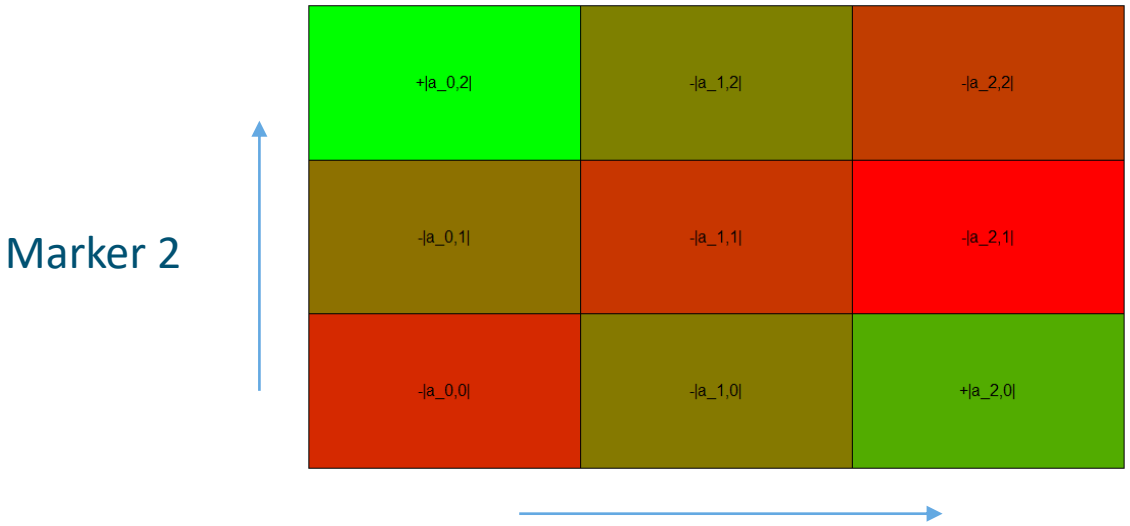
# On the generation of traits

- Quantitative epistatic effects (n.quantitative)
  - Generate three $N(0,1)$ random variables for each of the two markers and sort them according to absolute size $(a_0, a_1, a_2)\ and\ (b_0, b_1, b_2)$.
  - Effect of (X,Y) is $|a_X \cdot b_Y|$



# On the generation of traits

- Qualitative epistatic effects (n.qualitative)
  - Different effects for each marker combination ($N(0,1)$)
  - Effects for (0,2) and (2,0) are positive, rest negative



# What if this is not enough?

- Manual general of QTLs via:
  - Single marker QTL: real.bv.add

| | SNP | Chromo | Effect AA | Effect AB | Effect BB |
|---|---|---|---|---|---|
| [1,] | 1 | 1 | 0.4294301864 | 0 | -0.4294301864 |
| [2,] | 2 | 1 | -0.0756397721 | 0 | 0.0756397721 |
| [3,] | 8 | 1 | 0.2381887064 | 0 | -0.2381887064 |

- There is a QTL on marker 8 on chromosome 1
  - Homozygosity in the first allele has an effect of -0.238...
  - Heterozygosity has an effect of 0
  - Homozygosity in the second allele has an effect of 0.238...

# More advanced features

| | SNP | Chromo | Effect AA | Effect AB | Effect BB | Overall position | Pool | Pool-based Effect |
|---|---|---|---|---|---|---|---|---|
| [1,] | 1 | 1 | 0.4294301864 | 0 | -0.4294301864 | 1 | 0 | 0 |
| [2,] | 2 | 1 | -0.0756397721 | 0 | 0.0756397721 | 2 | 0 | 0 |
| [3,] | 8 | 1 | 0.2381887064 | 0 | -0.2381887064 | 8 | 0 | 0 |

- Traits can only be evaluated based on:
  - Maternal / paternal genotype
  - Different effects for alleles from different founder pools
  - Not based on realized allele but originating founder pool

# Epistatic effects

```
> real.bv.mult
     First SNP First chromosome Second SNP Second chromosome effect 00 effect 01 effect 02 effect 10
[1,]       144                1        145                 1      1.00      0.00      0.00      0.00
[2,]         6                3        188                 5      0.37      0.16      1.33      1.49
[3,]         5               17          1                10      1.18      2.60      0.18      1.74
     effect 11 effect 12 effect 20 effect 21 effect 22
[1,]      0.00      0.00      0.00      0.00      0.00
[2,]      1.58      2.51      0.38      2.12      0.98
[3,]      0.69      1.39     -1.21      0.96      1.94
```

```
> real.bv.dice
$location
$location[[1]]
     SNP chromosome
[1,]  11          1
[2,]  12          1
[3,]  16          4

$location[[2]]
     SNP chromosome
[1,]  14          2
[2,]  77          6
[3,]  15          9


$effects
$effects[[1]]
 [1]  1.8212212  1.5939013  1.9189774  1.7821363  1.0745650 -0.9893517  1.6198257  0.9438713  0.8442045 -0.4707524  0.5218499  1.4179416  2.3586796
[14]  0.8972123  1.3876716  0.9461950 -0.3770596  0.5850054  0.6057100  0.9406866  2.1000254  1.7631757  0.8354764  0.7466383  1.6969634  1.5566632
[27]  0.3112443

$effects[[2]]
 [1]  0.29250484  1.36458196  1.76853292  0.88765379  1.88110773  1.39810588  0.38797361  1.34111969 -0.12936310  2.43302370  2.98039990  0.63277852
[13] -0.04413463  1.56971963  0.86494540  3.40161776  0.96076000  1.68973936  1.02800216  0.25672679  1.18879230 -0.80495863  2.46555486  1.15325334
[25]  3.17261167  1.47550953  0.29005357
```

- In case SNP/chromosome positions set to NA, they are automatically filled with reasonable values
- If effects are placed on SNPs that are not there, effects will be removed

# Simulation of correlated traits

- Correlated traits are a combination of original traits
- Cholesky decomposition to calculate weights

Target correlation: $\begin{pmatrix} 1 & 0.2 \\ 0.2 & 1 \end{pmatrix}$

Cholesky decomposition: $\begin{pmatrix} 1 & 0.2 \\ 0 & 0.9797 \end{pmatrix}$

- Trait 1 QTL effects: old trait 1 / trait1_sd
- Trait 2 QTL effects:
  old trait 1* 0.2 / trait1_sd
  + old trait 2 * 0.9797 / trait2_sd
- Number of QTLs will be higher, but trait correlation will stay consistent over time

WAGENINGEN
UNIVERSITY & RESEARCH

# Definition of heritability

- In line with definitions from animal breeding
  - Heritability for a single observation / plot
  - Higher number of observations / plots will reduce residual variance
  - The residual variance is split into a permanent effect made for all observations and a temporary part for a single observation

$$h^2 = \frac{\sigma_a^2}{\sigma_a^2 + \sigma_e^2}$$

$$\sigma_e^2 = \sigma_{PU}^2 + \sigma_{TU}^2$$

$$w^2 = \frac{\sigma_a^2 + \sigma_{PU}^2}{\sigma_a^2 + \sigma_{PU}^2 + \sigma_{TU}^2}$$

- The repeatability gives information on the share of the permanent effect on the total residual effect

# Use of genomic data

- For all individuals in the population, underlying haplotypes stored
- This is also the case when they are not used directly
- You can control if individuals are genotyped / partially genotyped (low-density array) and the tool will automatically take care of the use in a breeding value estimation
- You can also just manually selected which markers to include in the breeding value estimation directly

# Founder genotypes

- Default: randomly generated
  - Very fast but no linkage structure
- Simulated LD build-up
  - External tools (MaCS, Chen et al. 2009)
  - founder.simulation() within MoBPS
  - Manual simulation of the population history in MoBPS
  - Burn-in cycles (e.g. start analysis in cycle 10)
- Import of real data
  - Haplotype matrix (nSNP x nHaplotypes – matrix) + map in creating.diploid() in dataset / map parameter
  - VCF / PedMap format

Make sure data is phased! (e.g. BEAGLE, Browning et al. 2018)

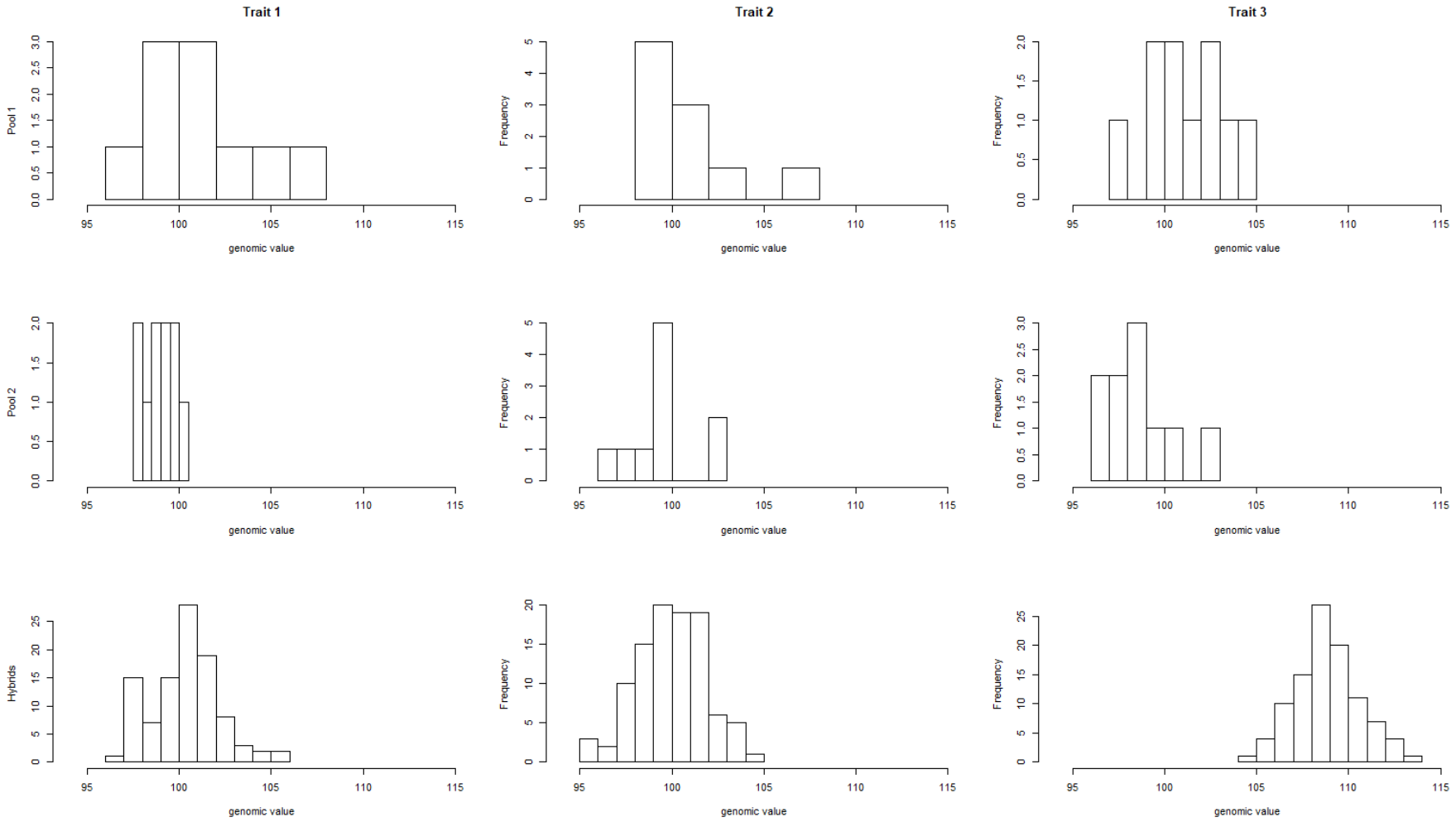
# Task 6: Import of real data and generation of traits

- Generate a founder population with 10 individuals from two different gene pools
- Genotypic data for the individuals is given via Pool1.vcf and Pool2.vcf
- Add three traits:
  - One with 10 purely additive QTLs
  - One with 1000 purely additive QTLs
  - One with 500 purely additive QTLs and 500 dominant QTLs of equal size
  - Traits should be uncorrelated
  - For all traits phenotypic mean for the founders should be 100 with a genetic variance of 5
- Generate 100 offspring by random mating between individuals from the two gene pools
- Compare the genomic values of the parents and offspring
- How often was each individual used for reproduction?
- Generate a PCA for all simulated individuals

# Solution Task 6

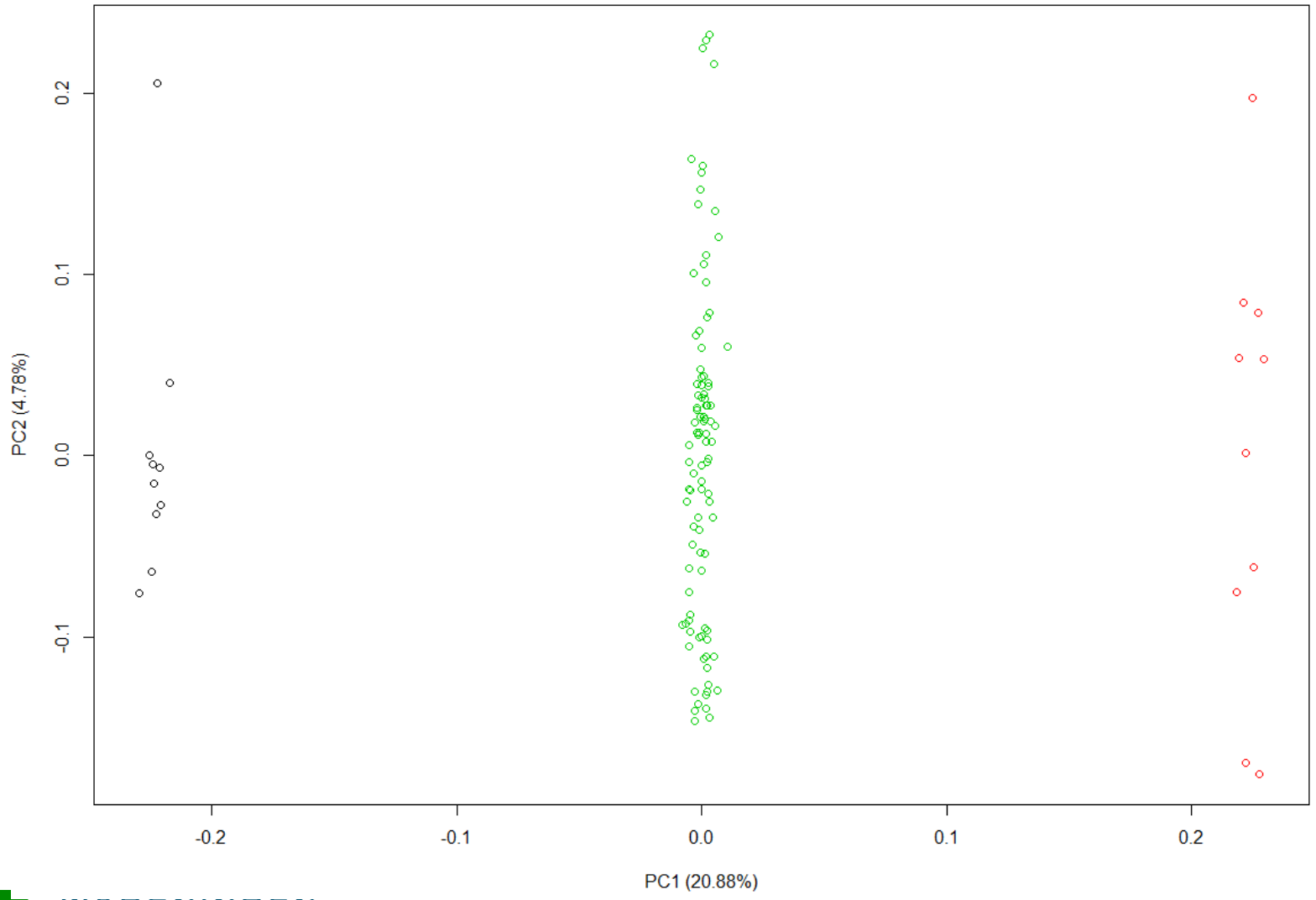
```
> summary(population)
Population size:
Total: 120 Individuals
Of which 110 are male and 10 are female.
There are 2 generations
and 3 unique cohorts.

Genome Info:
There are 5 unique chromosomes.
In total there are 5000 SNPs.
The genome has a total length of 25 Morgan.
The genome has a physical size of about: 0.5 GB

Trait Info:
There are 3 modelled traits.
Of which 3 have underlying QTL.
Trait names are:Trait 1 Trait 2 Trait 3
Genetics of traits are uncorrelated.
There are no interactions between residual effects.
Total time spent for generation: 0.8 seconds.

Time spent per step:
0.4 seconds for creation of founder population.
0.4 seconds for generation of new individuals.
```

# Solution Task 6




WAGENINGEN
UNIVERSITY & RESEARCH

# Solution Task 6




WAGENINGEN
UNIVERSITY & RESEARCH

# Agenda - Friday

- Breeding value estimation in the R-package
- Implementing own methodology within R
- Use of offspring phenotypes in the R-package
- Phenotypes with discrete realizations & litter sizes in the R-package
- Setting up a simulation with multiple generation cycles in the R-package
- Simulating multiple scenarios in the R-package
- Evaluating different simulation scenarios in R

# Handling of phenotypes, breeding values and genomic values

- For each individuals and each trait, we are storing:
  - The observed phenotype („pheno“)
  - The estimated breeding value („bve“)
  - The underlying true genomic value („bv“)
  - MoBPS V1.11 will allow “ebv” instead of “bve” and “gv” for “bv”
- Selection can be based on these criteria
- The underlying true genomic value can usually not be observed in practice
  - Powerful tool to evaluate methods!

# Task 7: Breeding value estimation

- This task is split into two subparts – if you are struggling with the first part you can load in the .Rdata object with an already generated population
- Generate a population list with 12 generations:
    - Each generation contains 50 males, 50 female with parents of the previous generation
    - Use a genetic map with 25.000 SNPs, 5 chromosomes with a length of 3 Morgan each
    - Generate a trait with heritability of 0.3 and 1‘000 purely additive QTLs
    - Make sure that:
        - In generation 10 only males are phenotyped
        - In generation 11 & 12 all individuals are phenotyped
        - In generation 10 & 11 all individuals are genotyped
        - In generation 12 only 20% of all individuals are genotyped
- Perform a breeding value estimations for individuals in generation 10, 11, 12 or combinations of the cohorts

    Use:

        - Genomic breeding value estimation (GBLUP)
        - Pedigree-based breeding value estimation (PBLUP)
        - Single-step breeding value estimation (ssGBLUP)
        - Assume individuals are only genotyped for 10‘000 / 2‘000 / 100 randomly selected markers

    - Generate a plot to showcase real genomic values and breeding values for generation 10.

# Solution Task 7

- Having a look at prints / logs can be very helpful to check if the tool is doing what you expect it to do

```
> population <- breeding.diploid(population, bve=TRUE, bve.gen=12)
Start genomic BVE.
Use Single Step GBLUP
Start derive Single-step relationship matrix
Construct pedigree matrix for 100 individuals.
Derive pedigree-matrix based for  100  individuals based on  793  individuals.
Derived pedigree matrix in  0.09 seconds.
Start deriving of H matrix for 16 genotyped and 84 non-genotyped individuals.
Derived H matrix in  0 seconds.
100 phenotyped individuals in BVE (Trait: Trait 1).
100 individuals considered in BVE.
Variance components in BVE: sigma_g^2 = 456.1813; sigma_e^2 = 738.2976 ; h^2 = 0.382
chol (own), 5 cores
0 seconds for BVE.
Correlation between genetic values and BVE:
0.6376494
```

# Solution Task 7

```
> population <- breeding.diploid(population, bve=TRUE, bve.gen=11)
Start genomic BVE.
100 phenotyped individuals in BVE (Trait: Trait 1).
100 individuals considered in BVE.
Variance components in BVE: sigma_g^2 = 337.1231; sigma_e^2 = 1111.8964 ; h^2 = 0.233
0 seconds for BVE.
Correlation between genetic values and BVE:
0.6032743
> # BVE with various different arrays
> population <- breeding.diploid(population, bve=TRUE, bve.gen=11, bve.array = 2)
Start genomic BVE.
10000 markers survived filtering for BVE.
100 phenotyped individuals in BVE (Trait: Trait 1).
100 individuals considered in BVE.
Variance components in BVE: sigma_g^2 = 337.1231; sigma_e^2 = 1111.8964 ; h^2 = 0.233
0 seconds for BVE.
Correlation between genetic values and BVE:
0.5975752
> population <- breeding.diploid(population, bve=TRUE, bve.gen=11, bve.array = 3)
Start genomic BVE.
2000 markers survived filtering for BVE.
100 phenotyped individuals in BVE (Trait: Trait 1).
100 individuals considered in BVE.
Variance components in BVE: sigma_g^2 = 337.1231; sigma_e^2 = 1111.8964 ; h^2 = 0.233
0 seconds for BVE.
Correlation between genetic values and BVE:
0.6137002
> population <- breeding.diploid(population, bve=TRUE, bve.gen=11, bve.array = 4)
Start genomic BVE.
100 markers survived filtering for BVE.
100 phenotyped individuals in BVE (Trait: Trait 1).
100 individuals considered in BVE.
Variance components in BVE: sigma_g^2 = 337.1231; sigma_e^2 = 1111.8964 ; h^2 = 0.233
0 seconds for BVE.
Correlation between genetic values and BVE:
0.4980535
```

25k SNPs

10k SNPs

2k SNPs

0.1k SNPs

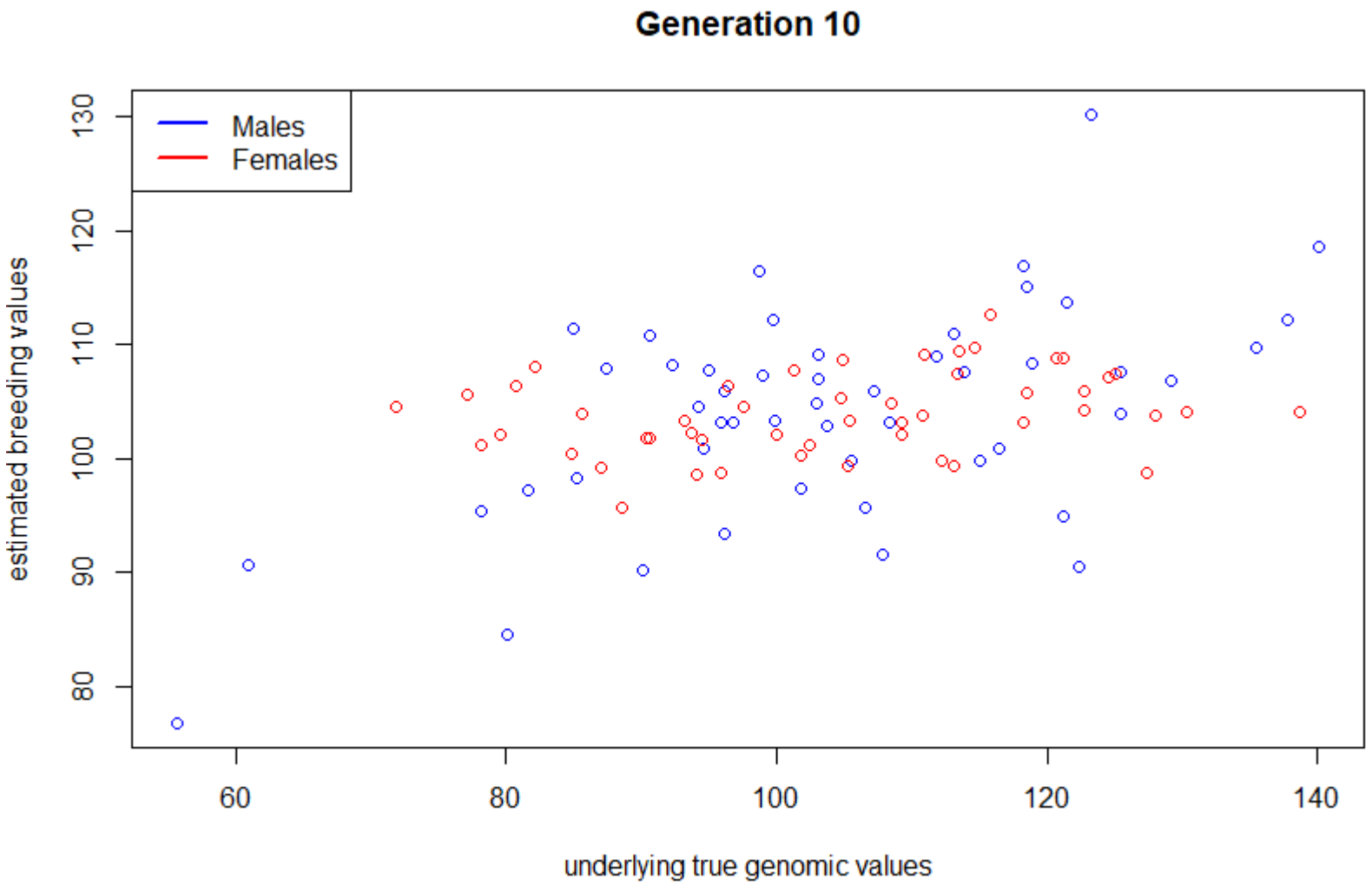
With 100 markers prediction accuracies are going down substantially

# Solution Task 7

- Prediction accuracies for males are much higher than for females
- Only males are phenotyped in generation 10!

```
> analyze.bv(population, database = cbind(10,1))
[[1]]
              Trait 1
BV / BVE    0.5439709
BV / Pheno  0.5372799
BVE / Pheno 0.9727460

[[2]]
 Trait 1
316.4133

> analyze.bv(population, database = cbind(10,2))
[[1]]
              Trait 1
BV / BVE    0.2907731
BV / Pheno         NA
BVE / Pheno        NA

[[2]]
 Trait 1
255.9466
```

# Solution Task 7



# Solution Task 7

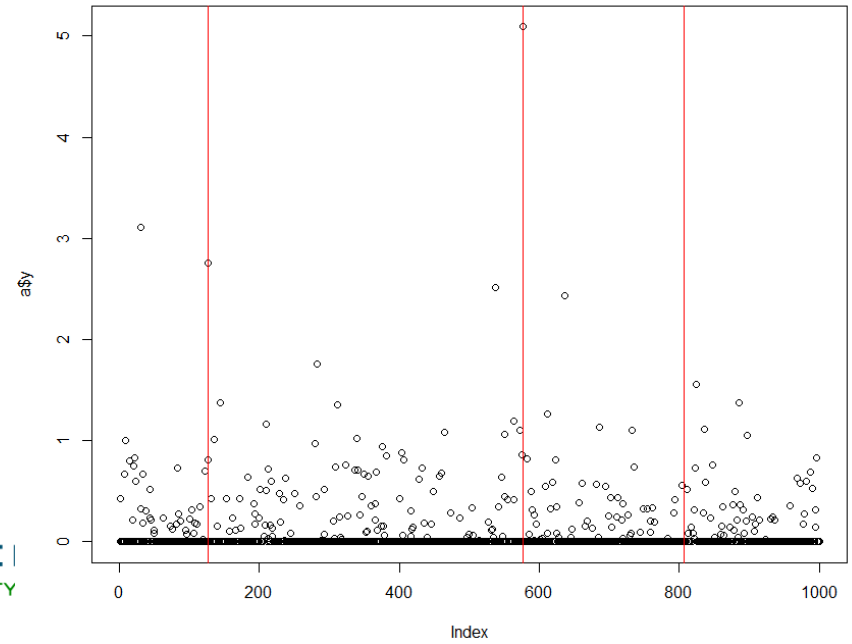
- MoBPS „only“ provides standard approaches for selection & breeding value estimation
- On-going project MiXBLUP integration
  - mixblup.bve = TRUE
  - mixblup.path="/cm/shared/apps/mixblup/current/MiXBLUP.exe"
- Use of own methodology is also possible
- MoBPS takes care of phenotype simulation, meiosis, computational efficiency etc.

# Use of own methods for GWAS/BVE

```
library(MoBPS)

# Fixate random seed for a uniform result
set.seed(1)
dataset <- founder.simulation(nsnp=1000)
set.seed(2)

population <- creating.diploid(dataset = dataset)

geno <- get.geno(population, gen=1)

hist(rowMeans(geno))

# Place QTL effects on some markers with
QTLs <- matrix(c(126,1, 0,1,2,
                 577,1,0,1,2,
                 806,1,0,1,2), byrow=TRUE, ncol=5)

population <- creating.trait(population, real.bv.add = QTLs)


population <- breeding.diploid(population, heritability = 0.5, phenotyping = "all")

pheno <- get.pheno(population, gen=1)
```

# Genome-wide association study

```
# Use any software solution for GWAS analysis
ge <- data.frame(marker=1:1000, chrom=rep(1,1000), pos = 1:1000, geno, check.names = FALSE)
ph <- data.frame(line = colnames(geno), y = pheno[1,])

a <- rrBLUP::GWAS(pheno=ph, geno=ge, plot=FALSE)
plot(a$y)
abline(v=c(126,577,806), col="red")
```




The QTL at marker 806 is not found

Depending on the used p-value markers 126 & 577 will be found but there will also be false positives then!

# Marker assisted selection

```
## Marker assistent selection

# use 10 randomly selected markers

geno_mas <- geno[sample(1:1000,10),]

model <- lm(pheno[1,] ~t(geno_mas))

y_hat <- model$fitted.values

cor(y_hat, pheno[1,])

new.bve <- cbind(colnames(pheno), y_hat)
population <- insert.bve(population, bves = new.bve)

get.bve(population, gen=1)
```

# Computational efficiency of MoBPS

# On the computational efficiency of MoBPS

- Smallest Unit of saving things in R: Integer
- 4 Bytes (32 bits)
  - Can contain any integer number from -2^31 to 2^31
  - Adding 1 + 1:

```
        0000000000 0000000000 0000000000 01
+       0000000000 0000000000 0000000000 01
-------------------------------------------
=       0000000000 0000000000 0000000000 10
=                                         2
```

WAGENINGEN
UNIVERSITY & RESEARCH

# miraculix

- Genotypes only take values 0, 1, 2
- 30 of 32 bits are wasted
- This is implemented in the R-package miraculix
  - ~ 15 times less memory than use of integers
  - Matrix operations are performed on bit-wise
    - Between 14 – 34 times faster than regular R
    - internal screening if things like avx2, sse4 or a graphics card are available
  - Jeremie is currently working with Martin Schlather & Alexander Freudenberg to implement a version of this in MiXBLUP

# BLAS / LAPACK

- Basic Linear Algebra Subprograms are internally used operations from linear algebra (e.g. matrix multiplications on non- 0/1/2 data)
- To check if your system is good, running the following R code:

```
n <- 5000
T <- matrix(0, nrow=n, ncol=n)
A <- T %*% T
```

- This matrix multiplication on a very good system takes ~ 2 seconds
- I am using OpenBlas // Intel MKL from a conda container
- On anunna: LD_PRELOAD=/shared/apps/openblas/gcc-6.1.0/sandy-bridge/0.3.3/lib/libopenblas.so R --vanilla

# Planning of computing time and memory

- Identify computational heavy steps of your simulation:
  - Run simulation with lower individual numbers (size.scaling)
  - How much time does it take to generate one individual
  - Breeding value estimation using GBLUP
    - Inversion of a matrix: o(n^3)
    - Calculation of the G matrix: o(p*n^2)
    - The G matrix for 11.000 individuals requires about 1 GB of memory (scales quadratically)
  - get.comp.times() // sacct
  - There is usually tricks to reduce computing times

# Simplifications of reality

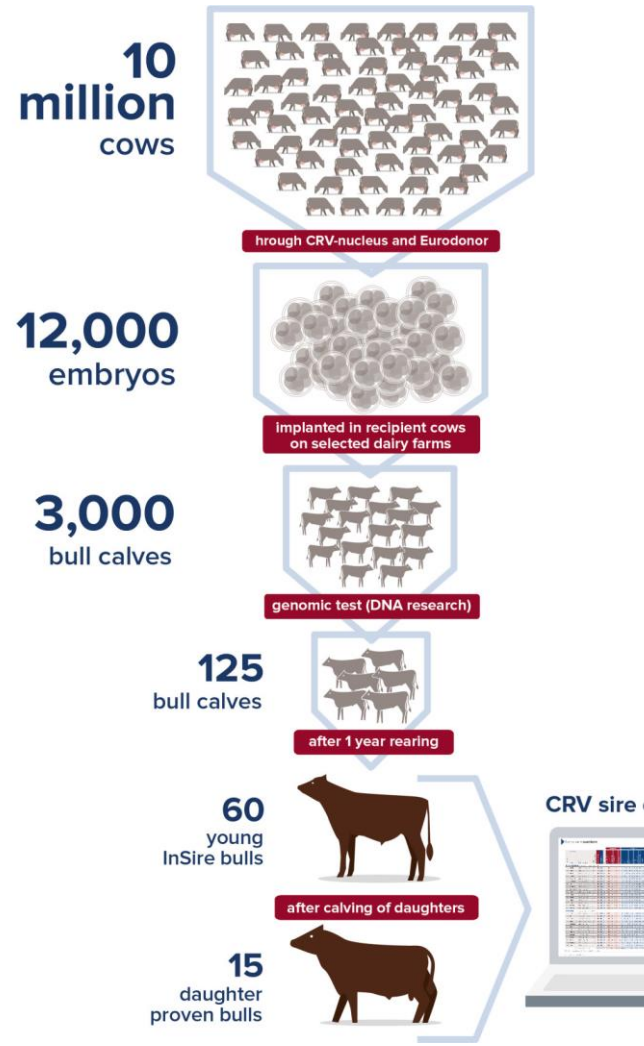
"All models are wrong, but some are useful." (George E.P. Box)

- Dairy cattle simulations:
  - We want to simulate multiple years of breeding
  - Different scenarios
  - Each scenario has to be simulated multiple times

  → Extremely high computational burden

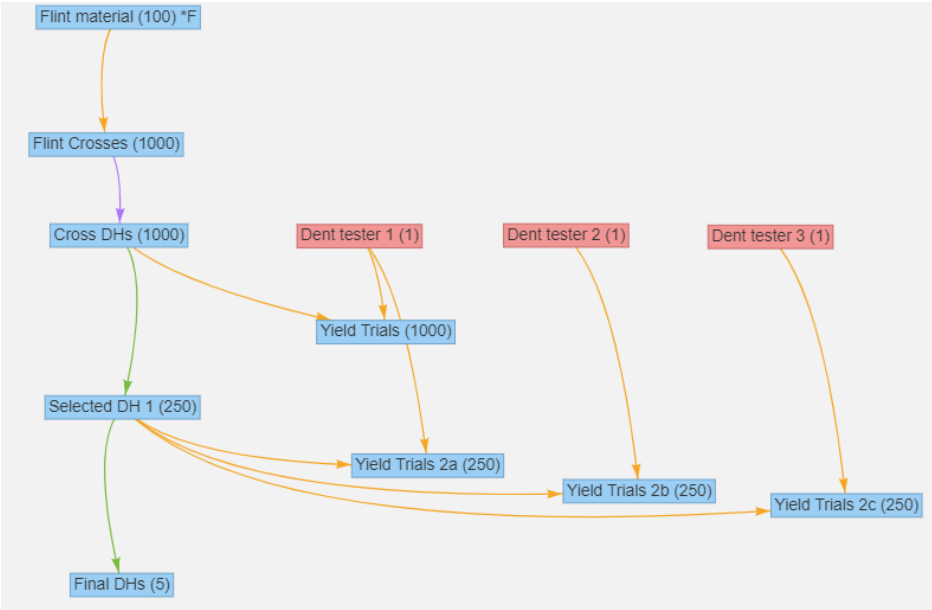
# Simplifications of reality

- Highest computational burden:
  - Breeding value estimation with millions of animals
- Reduce number of EuroDonor animals but be aware:
  - Genetic diversity might go down stronger
  - Accuracy of breeding values might be lower
  - Best EuroDonors are relatively worse to DeltaNucleus



# Task 8: Offspring phenotypes yield trials

- Simulate the following breeding scheme:
- Founders:
    - Simulate a founder population with 100 lines from one pool (flint lines) and 3 lines from a second pool (dent lines)
    - Make sure that allele frequencies in the two pools are different
    - Simulate a single trait with 1‘000 QTLs
    - Use 10‘000 SNPs. You can use a subset of the maize 600k array from MoBPSmaps
- Generate 1000 crosses within the Flint gene pool
- Generate 1000 DH lines from the crosses
- Mate the DHs to one of the three dent lines
- Phenotype the offspring (h^2 = 0.3)
- Select the top 250 DHs lines based on the performance in the yield trial
    - Hint: make sure that each DH is cross to the tester exactly once!
- Mate the selected DHs to all three dent lines
- Phenotype the offspring (h^2 = 0.3)
- Select the top 5 DH lines based on the performance in the yield trail
    - Hint: You can use breeding.all.combination or fixed.breeding (for a challenge) to generate your second yield trial

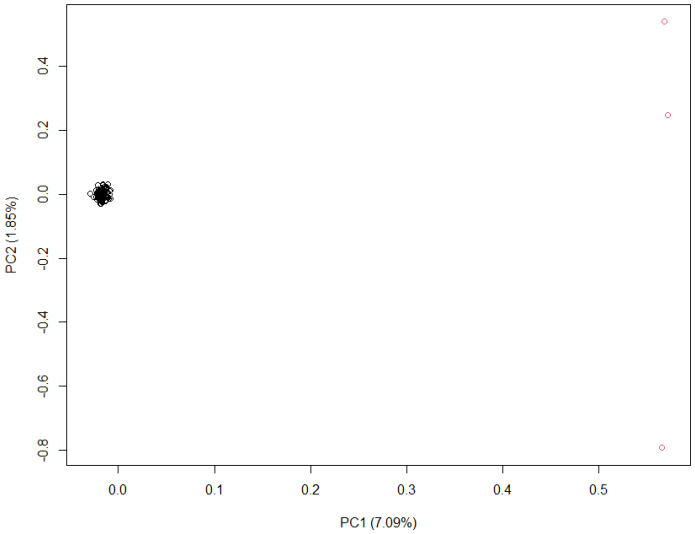


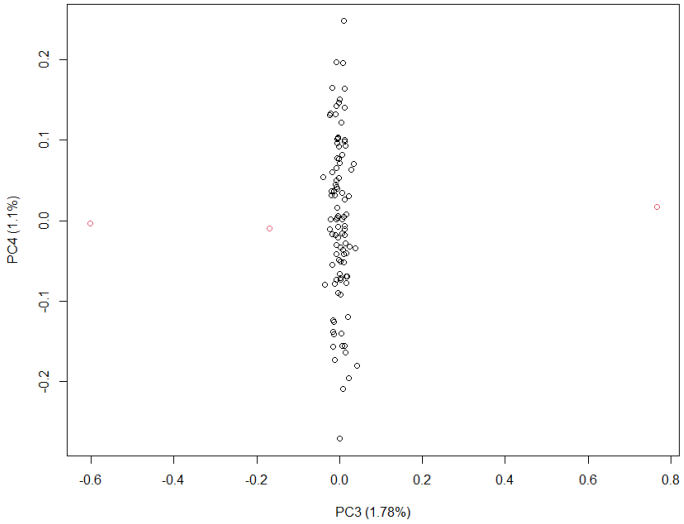
# Solution Task 8

```
> summary(population)
Population size:
Total: 4108 Individuals
Of which 4105 are male and 3 are female.
There are 7 generations
and 10 unique cohorts.
255 individuals are copies of previously generated individuals.

Genome Info:
There are 10 unique chromosomes.
In total there are 10000 SNPs.
The genome has a total length of 21.05182721 Morgan.
The genome has a physical size of about: 2.102 GB

Trait Info:
There is 1 modelled trait.
The trait has underlying QTL
The trait is named: Trait 1
Total time spent for generation: 18.3 seconds.

Time spent per step:
0.7 seconds for creation of founder population.
0.8 seconds for phenotyping.
7.3 second for breeding value estimation.
0.1 seconds for selection.
9.3 seconds for generation of new individuals.
```

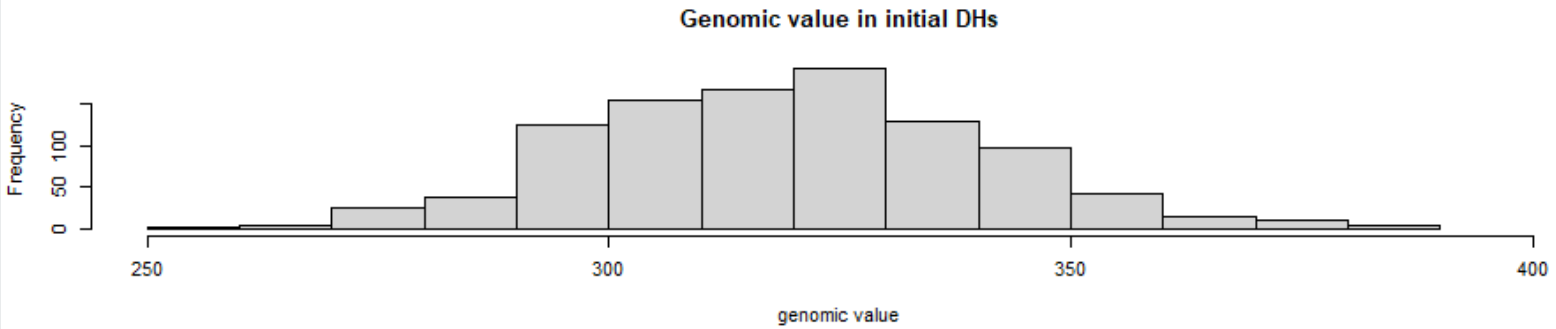
# Solution Task 8

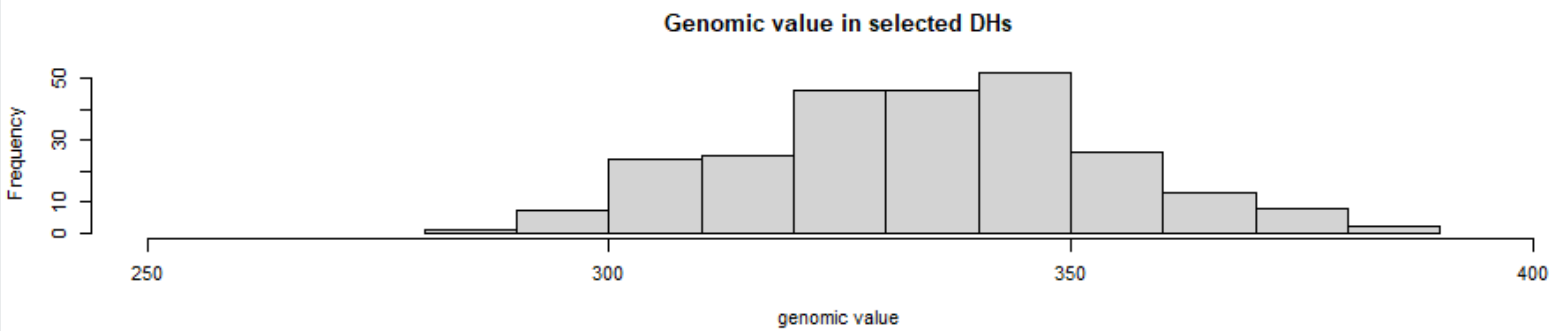
- PC 1 splits between the two pools
- PC 2,3 seem to mainly differentiates between the lines in pool 2
- PC 4 differentiates between lines in pool 1

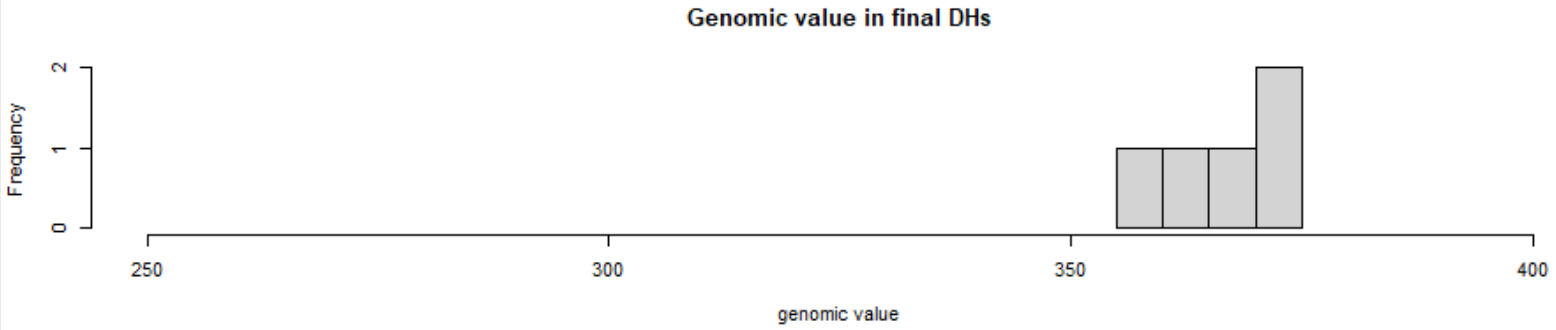



WAGENINGEN
UNIVERSITY & RESEARCH




UNIVERSITY & RESEARCH

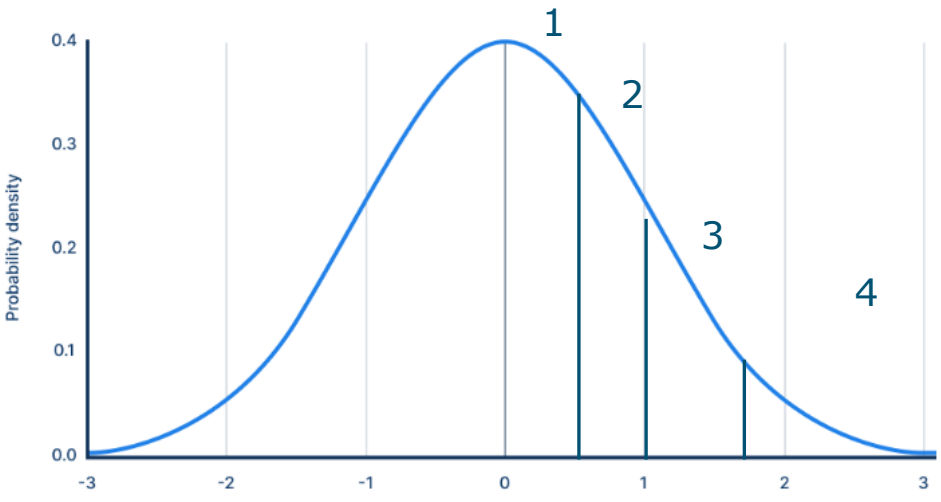
```r
# figure out position of the first parent
get.database(population, cohorts="Cross_DHs_sel")
#[1,]    5    1    1  250

# figure out position of the first parent
get.database(population, cohorts=c("Dent_tester_1", "Dent_tester_2", "Dent_tester_3"))
#[1,]    1    2    1    3

# Each row codes the mating between two individuals (column 1-3; column 4-6)
# This will mate the individual stored in generation 5, sex 1, nr 1 with individual generation 1, sex 2, nr 1
fixed.breeding <- cbind(5,1,1, 1,2,1)
population <- breeding.diploid(population, fixed.breeding = fixed.breeding, name.cohort = "Fixed_example")

# This is how you could set up your mating structure with fixed breeding

fixed.breeding <- matrix(0, nrow=750, ncol=6)
# Dent tester 1
fixed.breeding[1:250,] <- cbind(5,1,1:250,1,2,1)
# Dent tester 2
fixed.breeding[251:500,] <- cbind(5,1,1:250,1,2,2)
# Dent tester 3
fixed.breeding[501:750,] <- cbind(5,1,1:250,1,2,3)

population <- breeding.diploid(population, breeding.size = c(750,0),
                               fixed.breeding = fixed.breeding, name.cohort = "Yield_trial_2_alt")
```

# Discrete phenotypes & litter sizes
## Guidelines section 6.19

# Transformation from normal distribution to any distribution

- Target distribution:

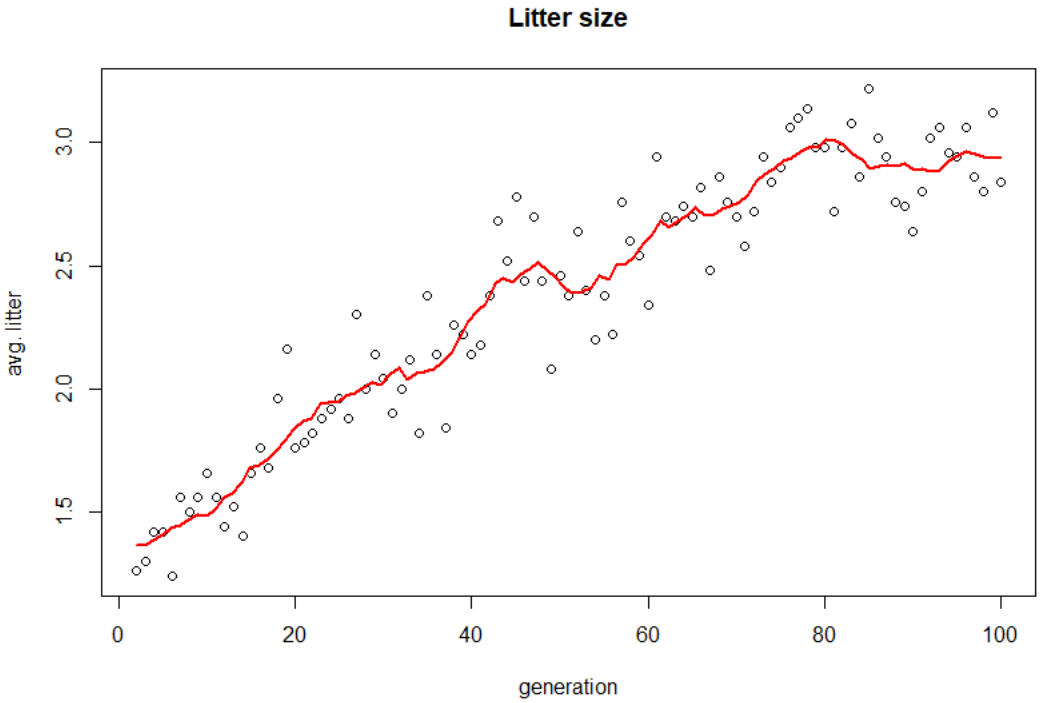
| Probability | Litter size |
|---|---|
| 0.70 | 1 |
| 0.15 | 2 |
| 0.10 | 3 |
| 0.05 | 4 |




70% / 85%/95% quantile

- Lowest values are projected to lowest discrete realizations
- Variance of the normal distribution is known (use mean/var.target)

WAGENINGEN
UNIVERSITY & RESEARCH


Litter size
avg. litter
1.5
2.0
2.5
3.0
0
20
40
60
80
100
generation

Warning:

The following tasks require substantially more knowledge of programming in R.

This is less about MoBPS itself but more about how to use R for data analysis / evaluation

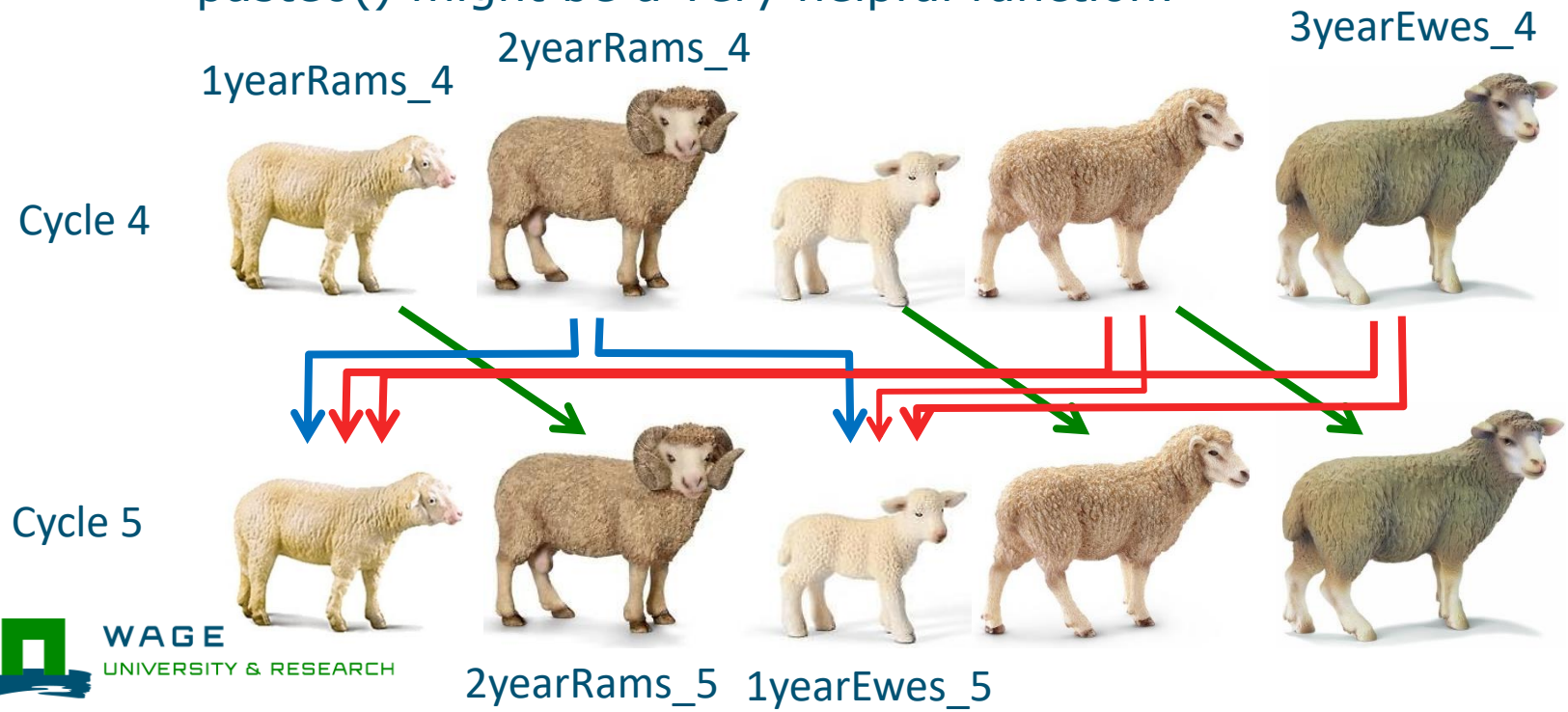
# Simulation of multiple breeding cycles

- Use of loops:

```
for(index in 1:10){
    print(index)
}

for(index in 1:10){
    .\ Code to perform all breeding actions in cycle index
}
```

# Task 9: Simulation of multiple breeding cycles

- Open the breeding scheme „Simple_Sheep_Advanced_single“
- Extend the code to simulate 20 generations of your breeding scheme
- Hints:
  - Try to use a loop instead of writting the same code multiple times
  - Use cohort names that include the cycle each cohort is in
  - paste0() might be a very helpful function!



# Task 9: Simulation of multiple breeding cycles

```
> summary(population)
Population size:
Total: 5780 Individuals
Of which 2260 are male and 3520 are female.
There are 21 generations
and 145 unique cohorts.
3600 individuals are copies of previously generated individuals.

Genome Info:
There are 26 unique chromosomes.
In total there are 46545 SNPs.
The genome has a total length of 24.57031306 Morgan.
The genome has a physical size of about: 2.4488 GB

Trait Info:
There are 2 modelled traits.
Of which 2 have underlying QTL.
Trait names are:Trait 1 Trait 2
Highest correlation between genetics of traits is 0.2 (absolut value).
There are no interactions between residual effects.
Total time spent for generation: 21.3 seconds.

Time spent per step:
1.1 seconds for creation of founder population.
0.2 seconds for calculation of true genomic values.
2.3 seconds for phenotyping.
4.4 second for breeding value estimation.
0.9 seconds for selection.
12.5 seconds for generation of new individuals.
```

# Task 10: Simulating different scenarios in R

- Extend the script from Task 9 to allow for the simulation of multiple runs and multiple scenarios
- Think of an interesting alternative scenario to compare the baseline against:
  1. Baseline
  2. Only 5 rams are selected for reproduction
  3. Only 50% of all ewes are genotyped
  4. Use a selection index with three times as much weight on the meat trait
- Use set.seed() to make sure that the genetic architecture of the trait is the same between the four simulations
- Store results of an individual simulation in an .RData object

# Solutions Task 10

- Evaluate underlying true genomic values, kinships and prediction accuracies for all cohorts:

```
cohorts <- get.cohorts(population)

genomic_values <- accuracies <- kinships <- matrix(0, nrow=2, ncol=length(cohorts))

for(index in 1:length(cohorts)){
  # This extracts the genomic values for animals from the cohort and calculates the mean
  genomic_values[,index] <- rowMeans(get.bv(population, cohorts=cohorts[[index]]))
  # This extracts accuracies of the breeding value estimation for selected cohorts
  accuracies[,index] <- analyze.bv(population, cohorts=cohorts[index])[[1]][1,]
  # This approximates avg. kinship between animals and within ((kinship -0.5) *2 is inbreeding)
  kinships[,index] <- kinship.emp.fast(population=population, cohorts = cohorts[index])

}

save(file=paste0("Sheep_simulation_scenario", scenario, "run", run,".RData"), list=c("cohorts", "genomic_values", "accuracies", "kinships"))
```

# Evaluating different scenarios in R

- Main recommendations:
  - Avoid using different scripts for different scenarios!

  →Error prone when you have to do changes in different script

  →Initialize all parameters subject to change in the beginning of your script

  - Documentation / comments on what you are doing

  →Good readability of code leads to less errors

  →Setting parameters not needed can help both you and other understand your code when returning to it

# Task 11: Evaluating different scenarios in R

- Each of the four scenarios has been simulated 100 times
- Analyze the results of the simulation:
  - Are genetic gains for the meat trait different between scenario 1 & 2 after 20 cycles – use a t-test
  - Are the obtained prediction accuracies different in scenario 1 & 3
  - Generate a plot showing the avg. genomic values for the 1-year rams for both traits in all scenarios in the different cycles

Questions?

Particular topics you want to discuss?

How to simulate XYZ?