Workshop GfT & KWS

Modular Breeding Program Simulator

Torsten Pook, Johannes Geibel, Azadeh Hassanpour




- You do not learn how to use a software by listening
  - Most of this course will be practical sessions

- Form small groups (~1 – 4)
- Azadeh, Johannes and I will be switching between breakout rooms

- Joint discussion & sample solution

- All sample solutions and slides will be shared

Agenda

- Time-table:
  - 9:00 – 17:00
  - 1.5 hour lunch break (target: 12:30 – 14:00)
  - No fixed coffee breaks
    - Ask when you need one // take them during programming sessions
  - Agenda:
    - General Introduction of the MoBPS framework
    - Basic functionality of the web-interface
    - More advanced features of the web-interface
    - Scenario comparision in the web-interface
    - Basic functionality of the R-package
    - Trait generation in the R-package
    - 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

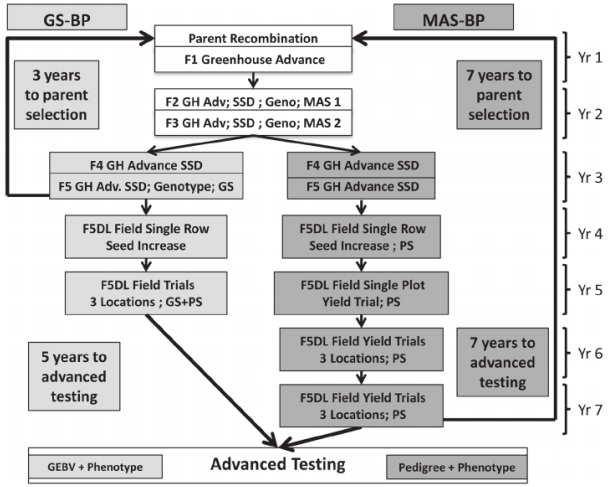
Thursday

Friday

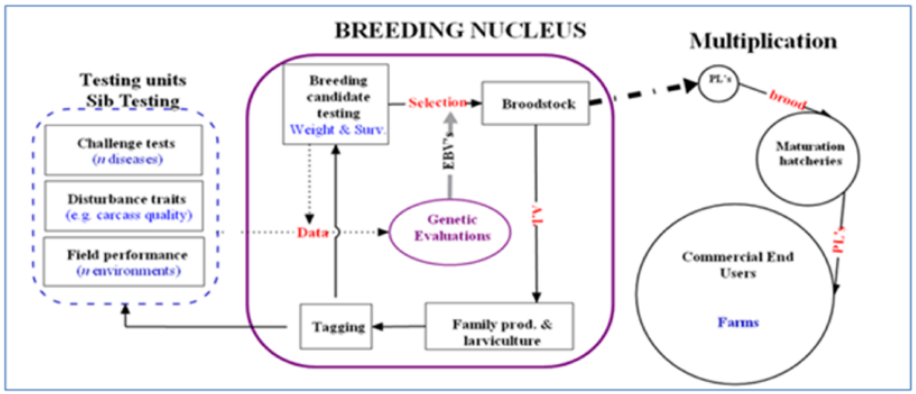
General introduction

What is a breeding program

- In the closed sense:
  - Planned breeding of a group of animals or plants over several generations
  - Goal: Change characteristics of animals/plant through careful selection of breeding partners



(Wheat breeding: Heffner et al., 2010)



(Fish breeding: Rye 2012)

What are we interested in?

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

- How to control it?
  - How many animals / plants to use
  - Generate genotype / phenotype data of all animals / plants
  - 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)
  - Reliant on underlying theory
  - E.g. Breeders equation:

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

R: response to selection

i: selection intensity

h^2 : heritability

σ_a^2: additive genetic variance

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

→ Stochastic simulation

- Stochastic simulations does not have to stop at a breeding program
- In principle any mating and mating behaviour can be modelled in this way
  - Population genetic analysis
  - Historic populations
  - How can be use gene banks / which material to preserve (IMAGE)

What is MoBPS?

- Modular Breeding Program Simulator
- Environment for the simulation
- The software takes care of backend-“stuff“ that you have to account for in a simulation study 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
- ~ monthly updates
- 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 1.8.01 (01.03.22)

Rework of sigma.e / heritability estimation.

sigma.e now correctly means residual standard deviations - not variance...

Updated documentation

Version 1.7.13 (24.01.22)

Added fixed effects

get.database() now accepts individual IDs as input

Added kinship.emp.fast.between() for calculation of kinship between gen/database/cohorts of individuals

Added get.pheno.single() to allow for the export of individual phenotypes instead of only the mean phenotype for an individual

Version 1.6.63 (25.10.21)

Default mutation rate has been changed to 10^-8 (mutation.rate in breeding.diploid())

Added option to use a PCG solver to solve the mixed model in the breeding value estimation ((bve.solve = "pcg"))

Added option to selected based on avg. offspring phenotype (selection.criteria = "offpheno")

Version 1.6.54 (24.08.21)

Breeding value estimation now supports difference residual variance when a sample was phenotyped multiple times

kinship.exp.store() has been renamed to kinship.exp()

Individual names in get.X() functions can now be the unique MoBPS IDs instead names according to generation, sex, and animal nr.

founder IDs from pedigree-file can be kept when using pedigree.simulation()

- The parameters in MoBPS are used to take 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
  - E.g.:
    - 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

On the efficiency of MoBPS

- When R is storing a numeric value it looks like this:
  - 8 bytes (64 bits):

    1101010100101000101001010100110110101010110101001010101010101000
- Slightly better: Integer (4 bytes)
- Genotypes are just 0/1/2 values
  - A genotype for one allele should only take 2 bits
- This is implemented in the R-package miraculix
  - ~ 30 times less memory than use of numerics
  - Machine specific matrix operations
    - internal screening if things like avx2, sse4 or a graphics card are available
    - Highly efficient computations
    - Between 14 – 34 times faster than regular R // when using GPU a lot more!

BLAS / LAPACK

- Basic Linear Algebra Subprograms are internally used operations from linear algebra (e.g. matrix multiplications)
- We did not optimize regular matrix operations but recommend the user to do so himself!

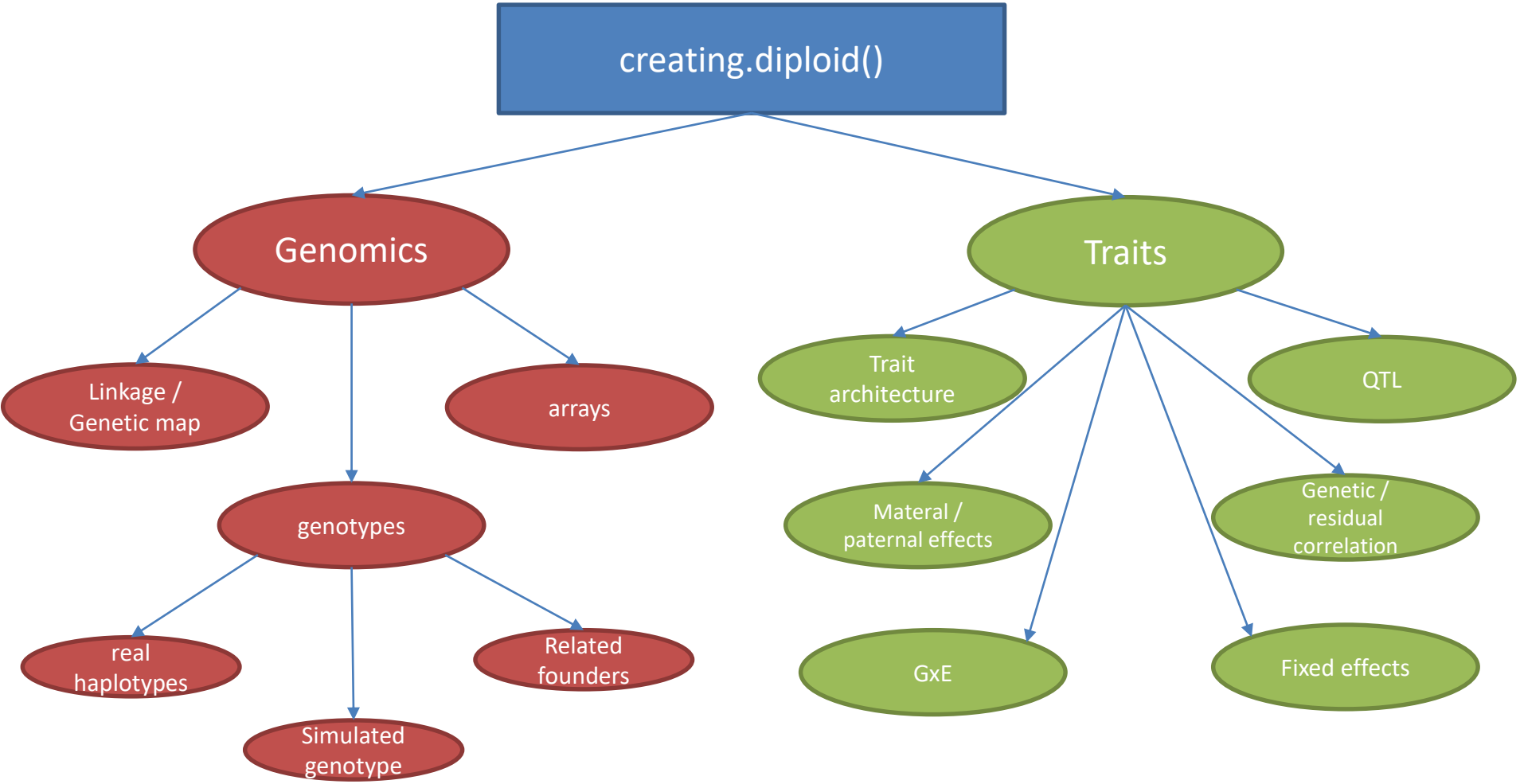
- To check if your system is good, running the following R code:

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

- This matrix multiplication on a very good system takes ~ 2 seconds
- If your system takes substantially lower, you will run into problems earlier when simulations get large!
- We are developing MoBPS using OpenBlas // Intel MKL

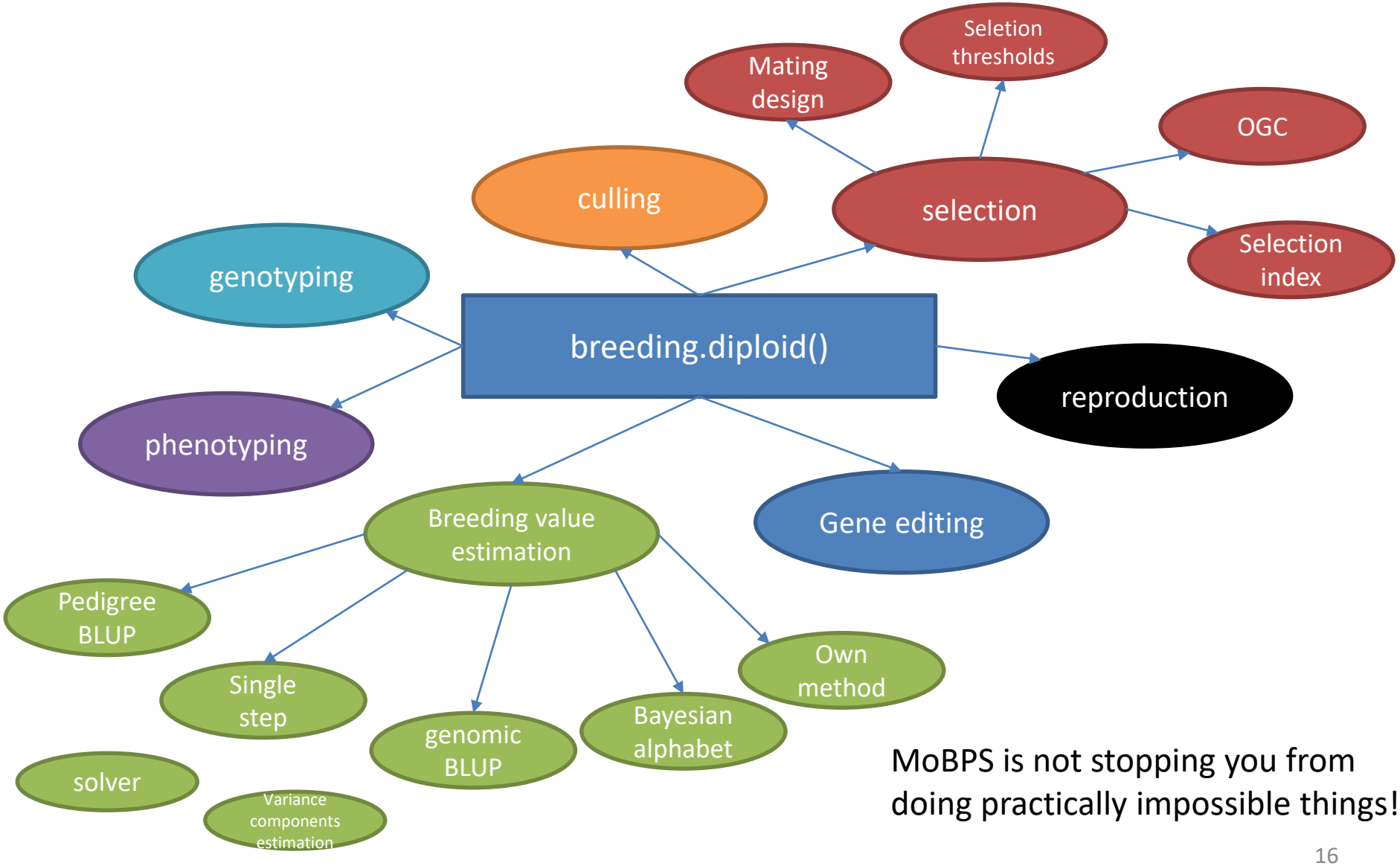
Steps of your simulation

Initialization of the founder population



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

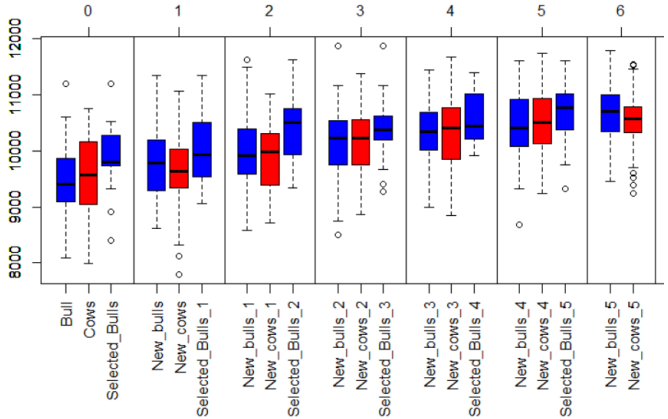
Simulation of breeding actions

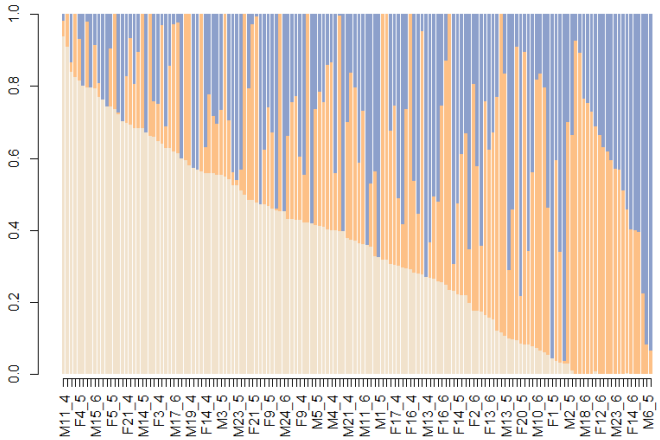


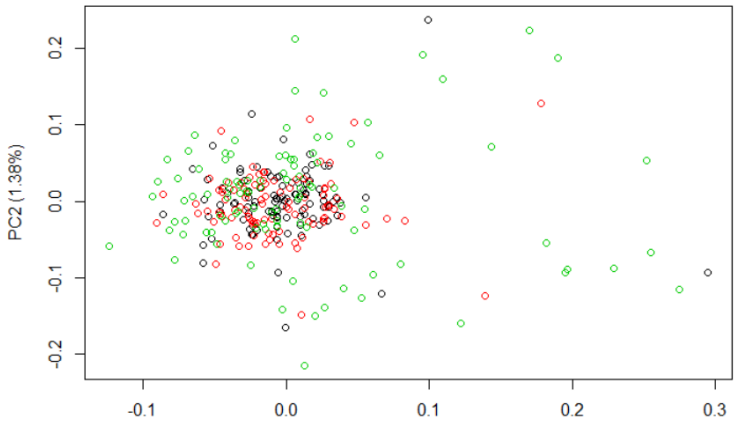
MoBPS is not stopping you from doing practically impossible things!

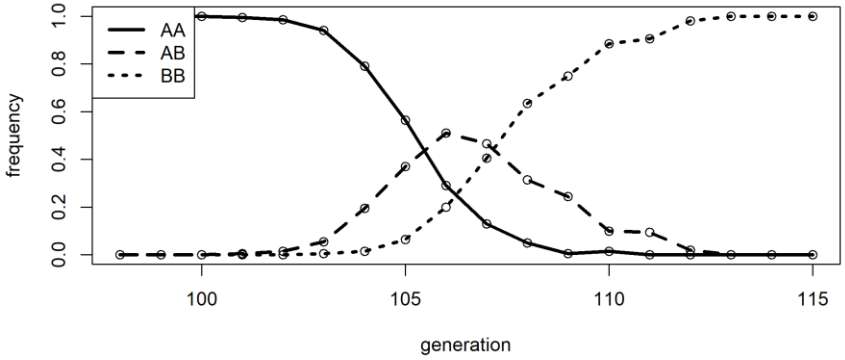
- Output of our simulation is a highly compressed object (population list)
- This contains information on all individuals ever generated in our simulation and is an expected input for breeding.diploid()
- We know the exact truth!
  - Genotypes / haplotypes
  - Where are the QTLs in the genome
  - What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

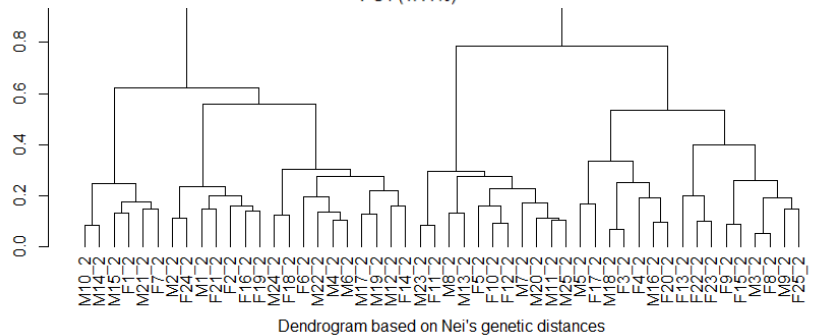
Analysis functions

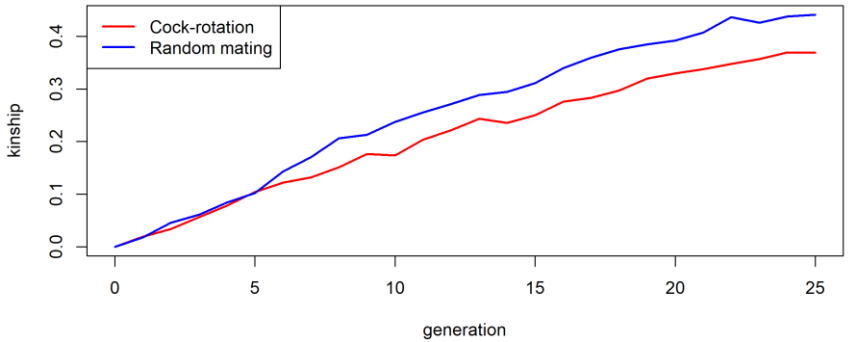






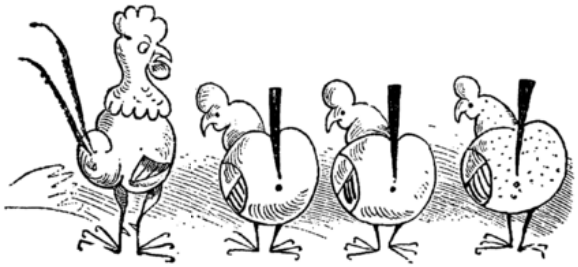


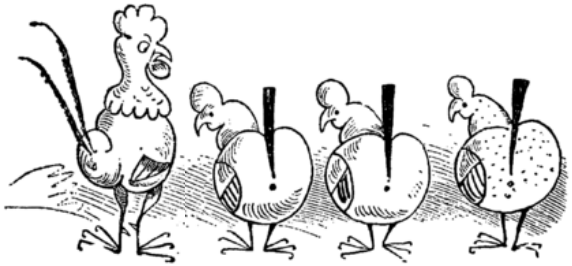


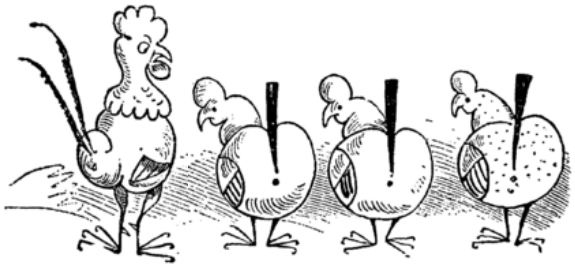


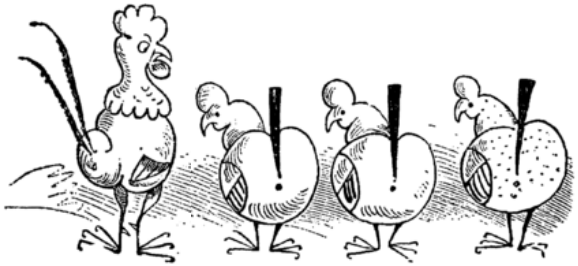
Some simulations to inspire

Cock rotation

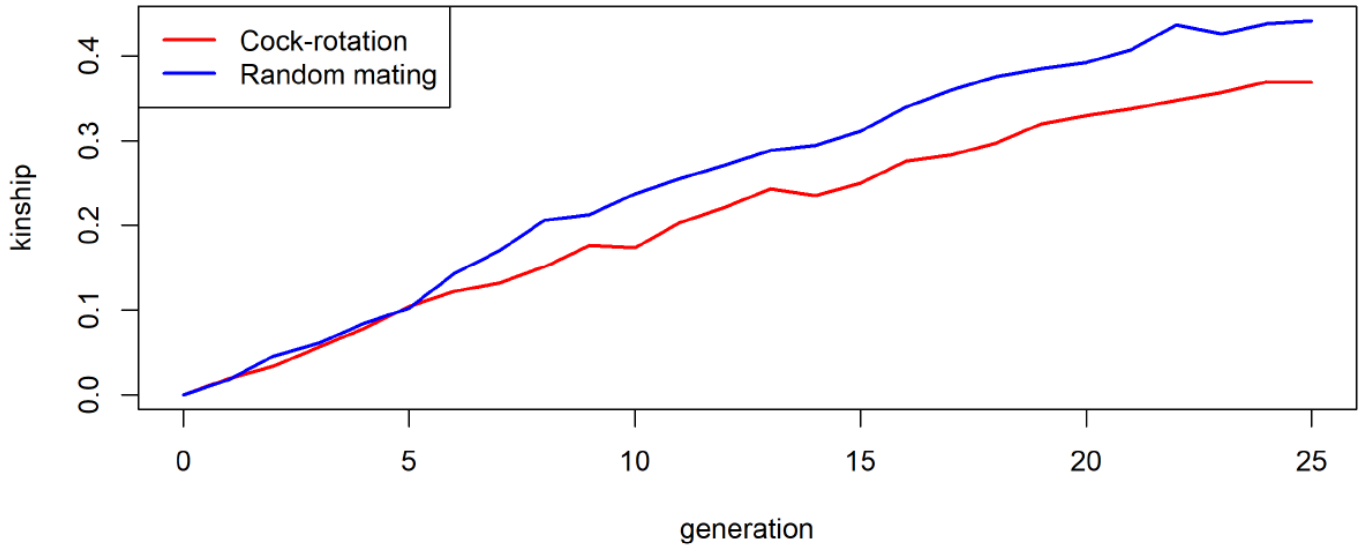




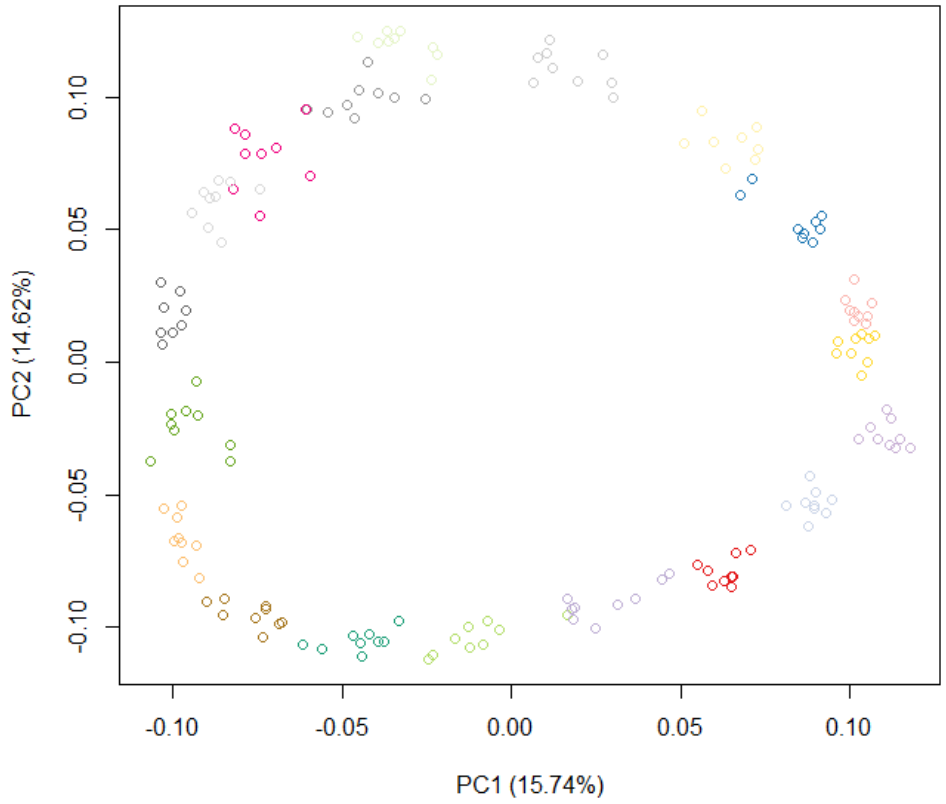




- Breeding scheme to reduce loss of genetic diversity

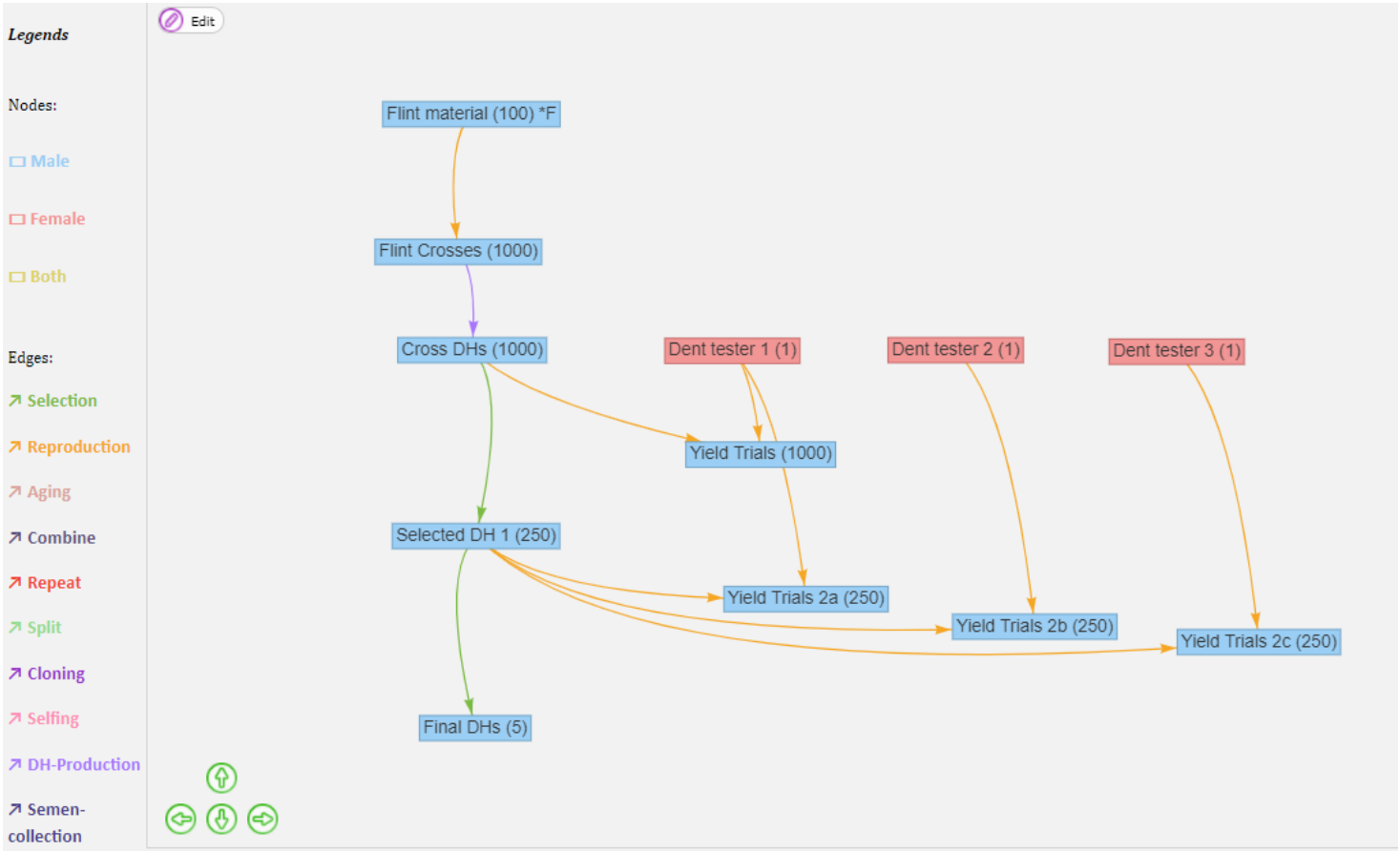


- Genetic distances between animals according to a principle components analysis



Each color represents one box

Test crossing scheme in maize



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

Genetic Correlation

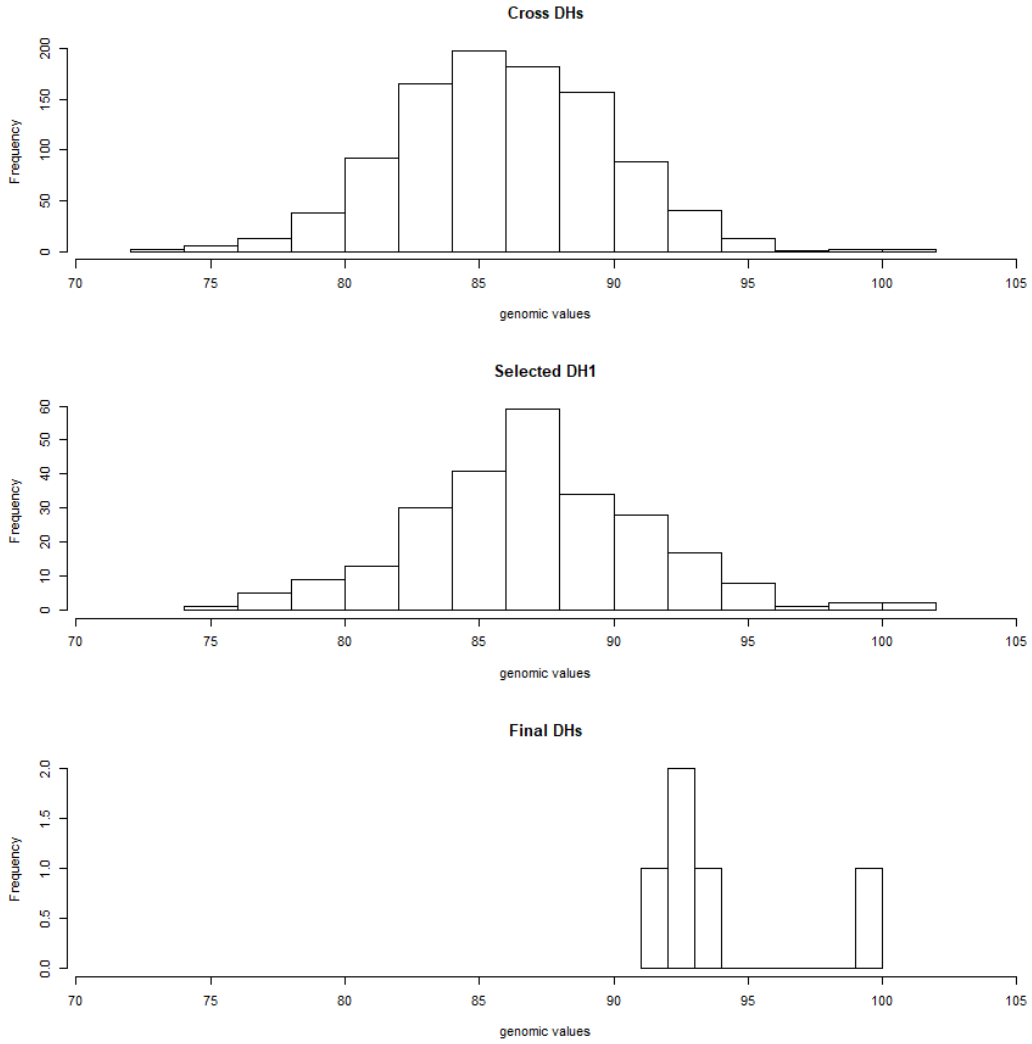
	Grain Yield Loc1	Grain Yield Loc2	Grain Yield Loc3
Grain Yield Loc1	1	0.8	0.8
Grain Yield Loc2	0.8	1	0.8
Grain Yield Loc3	0.8	0.8	1

Phenotype Information

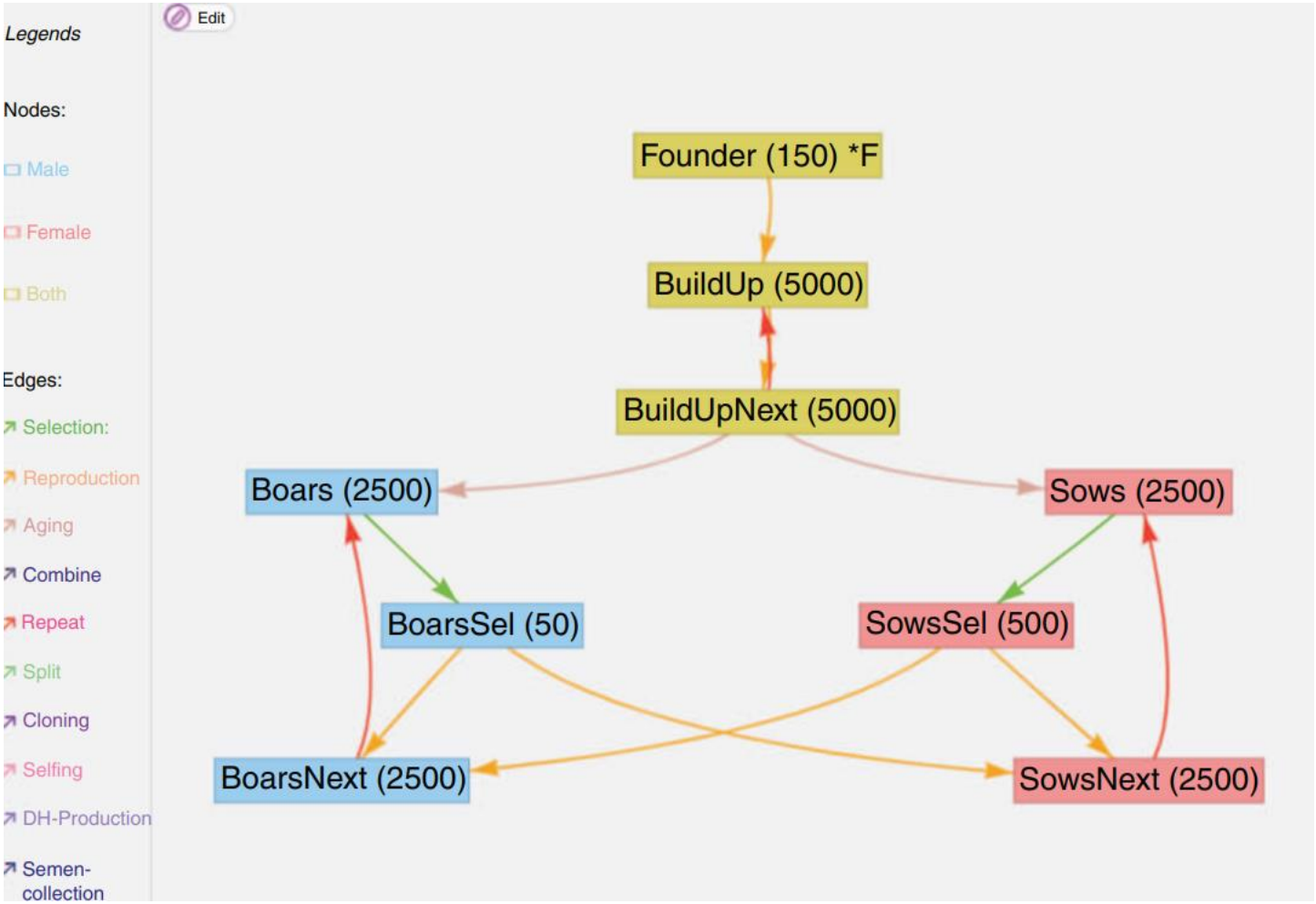
Press here to get info on how this module works!

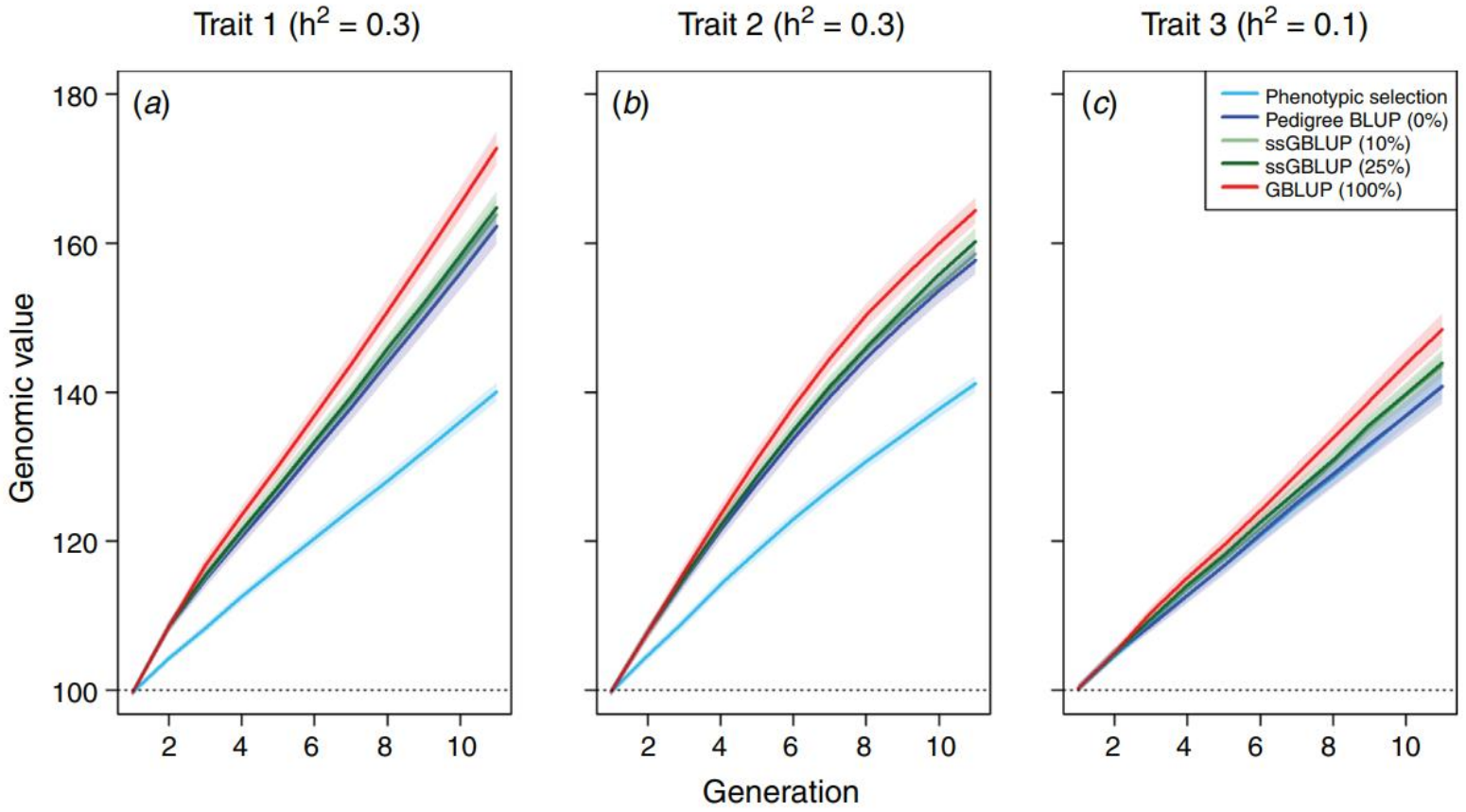
Add new phenotype | Show/Hide 3 phenotypes | Show/Hide QTLs | Show/Hide residual correlation | Show/Hide genetic correlation

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	0	☑	X
Grain Yield Lc		100	10	0.3	0.5	500	500	0	0	0	0	0	☑	X
Grain Yield Lc		100	10	0.3	0.5	500	500	0	0	0	0	0	☑	X


Cross DHs
Frequency
0
50
100
150
200
70
75
80
85
90
95
100
105
genomic values
Selected DH1
Frequency
0
10
20
30
40
50
60
70
75
80
85
90
95
100
105
genomic values
Final DHs
Frequency
0.0
0.5
1.0
1.5
2.0
70
75
80
85
90
95
100
105
genomic values

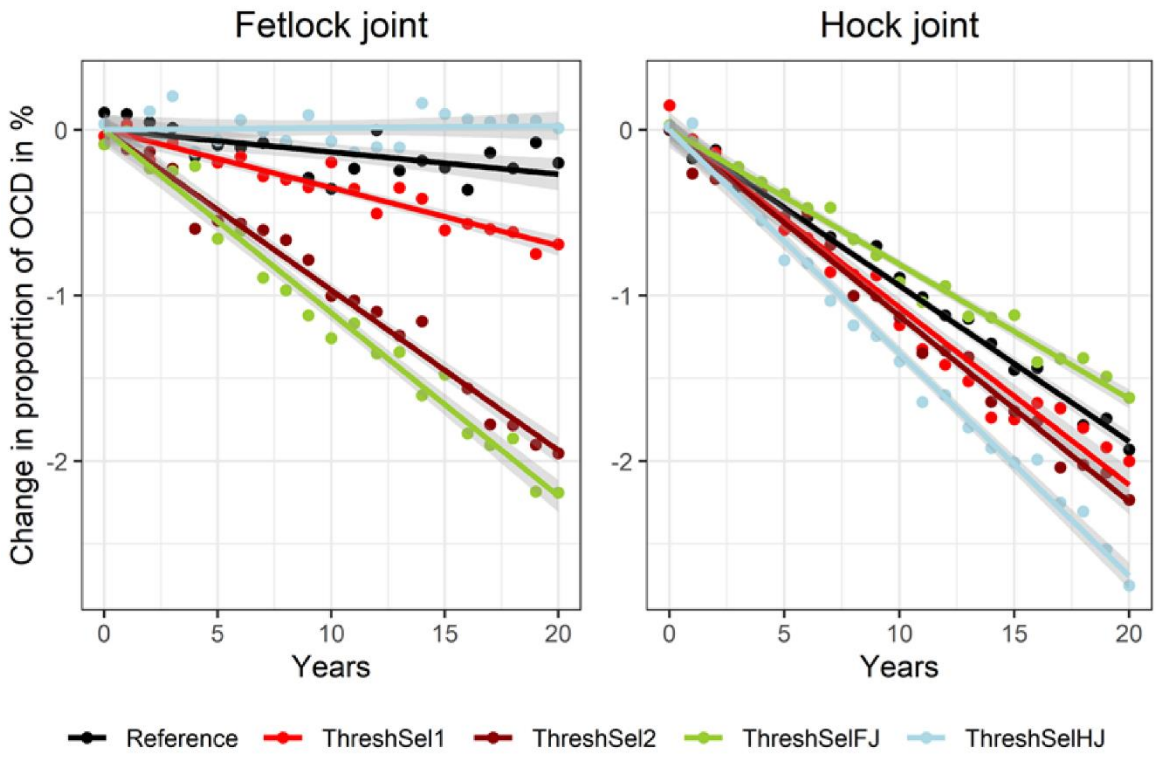
Pig breeding (Pook et al. 2021)




Trait 1 (h^2 = 0.3)
Trait 2 (h^2 = 0.3)
Trait 3 (h^2 = 0.1)
(a)
(b)
(c)
Genomic value
180
160
140
120
100
2
4
6
8
10
Generation
Phenotypic selection
Pedigree BLUP (0%)
ssGBLUP (10%)
ssGBLUP (25%)
GBLUP (100%)

Horse breeding scheme (Buettgen et al. 2020)

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



- Some simulations from the group:
  - Exemplary scripts in the Guidelines (Section 6)
  - Exemplary script on github https://github.com/tpook92/MoBPS/tree/master/Exemplary_scripts
- Tobias Niehoff (WUR)
  - https://github.com/tobiasniehoff/exploring_MoBPS

Questions?

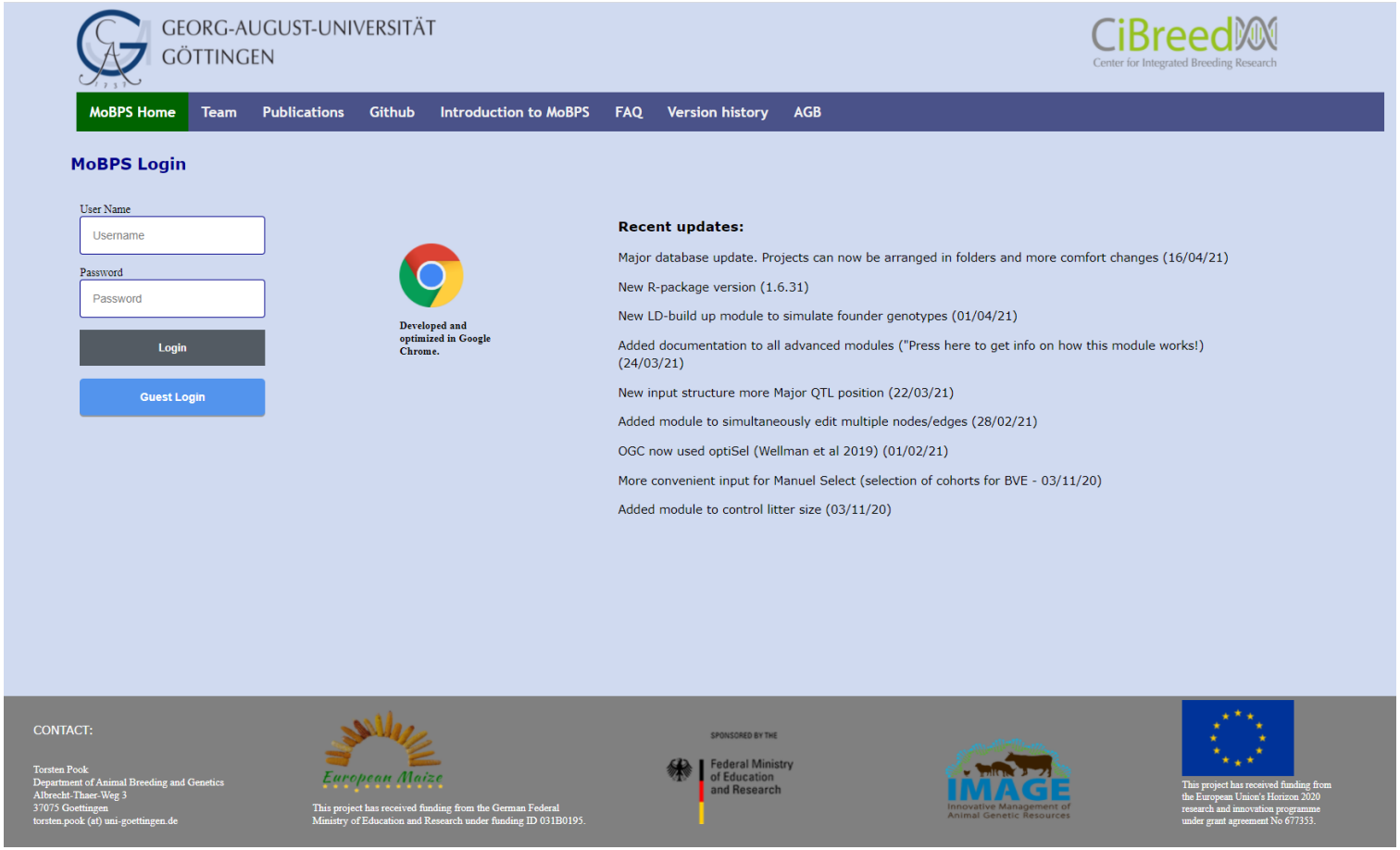
- MoBPS is mostly used as an R-package
- Most flexible & efficient
- Not the easiest to get started!
- User-Manual: https://github.com/tpook92/MoBPS/blob/master/Guidelines%20to%20MoBPS.pdf

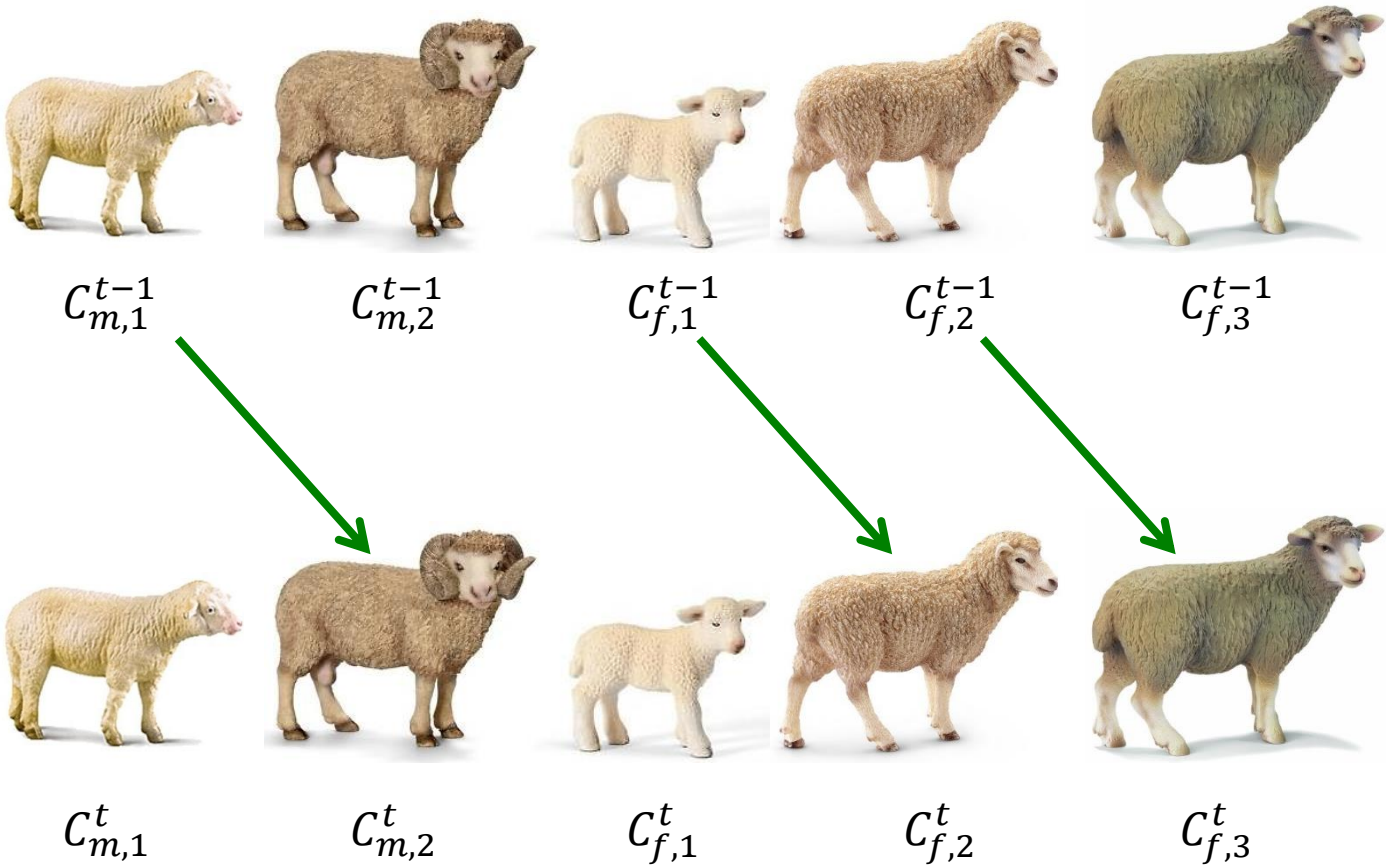
```
breeding.diploid <- function(population, mutation.rate = 10^-5, remutation.rate = 10^-5, recombination.rate = 1,
          selection.m = "random", selection.f = NULL, new.selection.calculation = TRUE, selection.function.matrix = NULL,
          selection.size = 0, ignore.best = 0, breeding.size = 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, db.mating = FALSE, db.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){
```

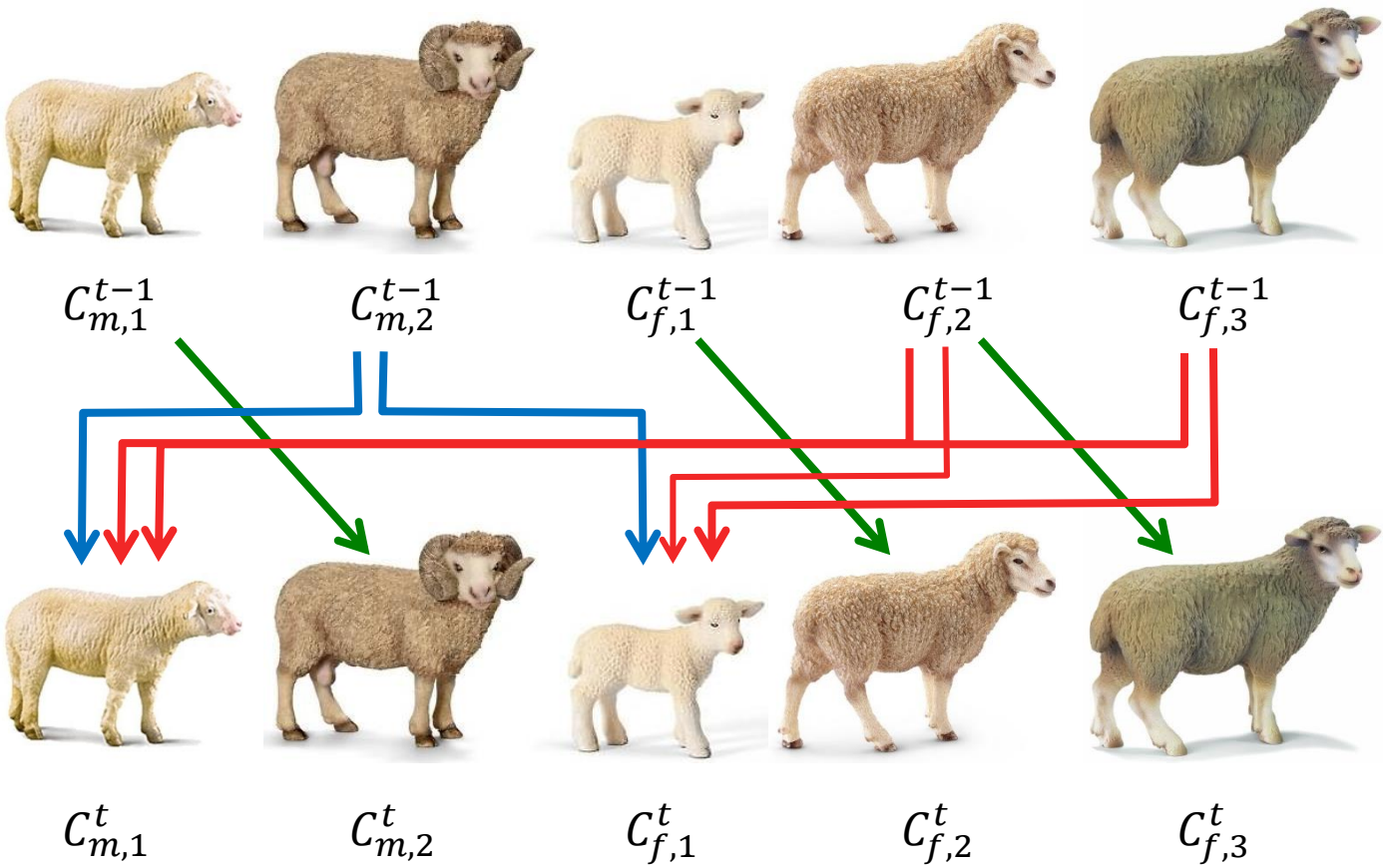
Table of Contents

MoBPSweb

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



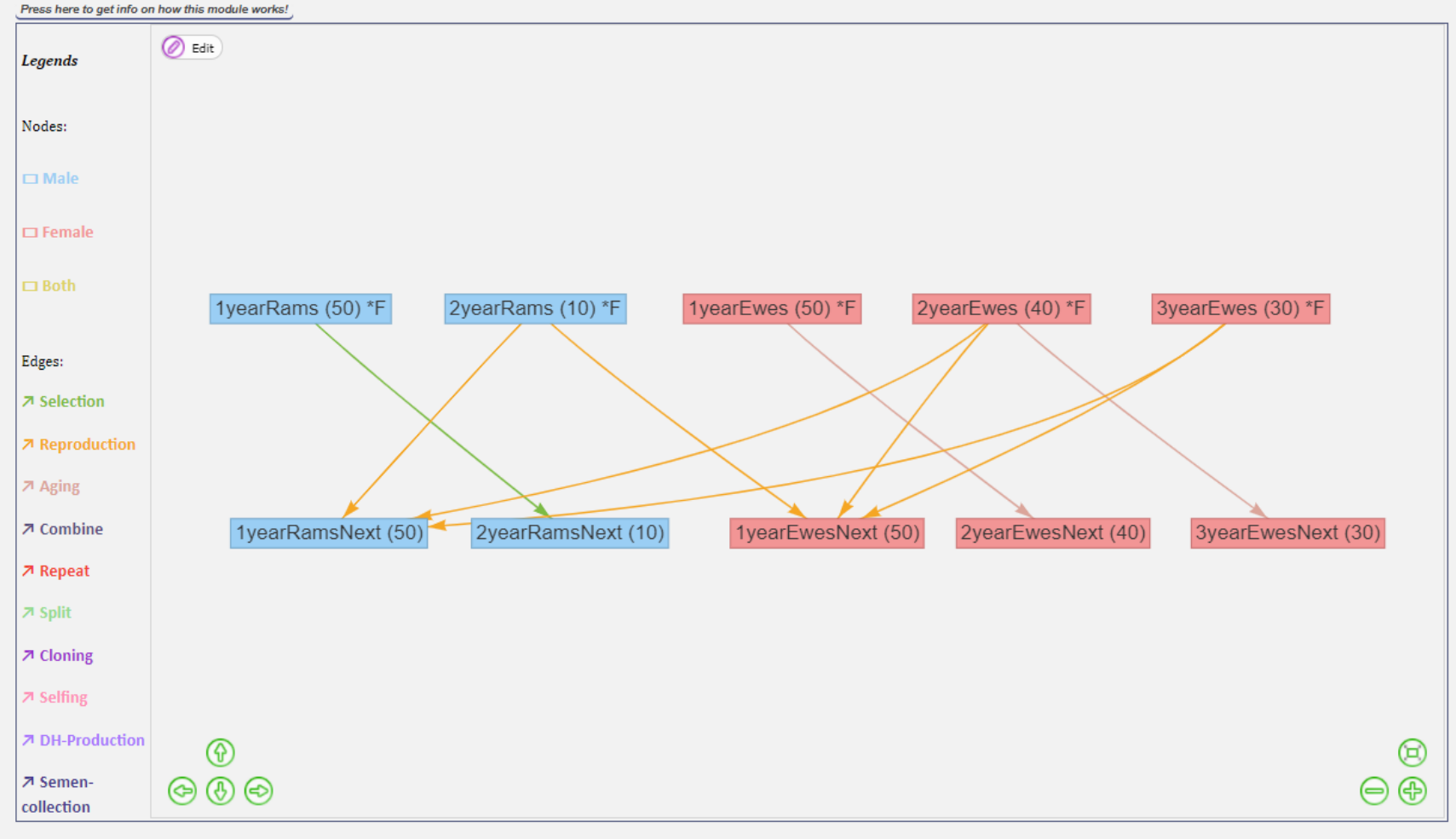

$C_{m,1}^{t-1}$
$C_{m,2}^{t-1}$
$C_{f,1}^{t-1}$
$C_{f,2}^{t-1}$
$C_{f,3}^{t-1}$
$C_{m,1}^{t}$
$C_{m,2}^{t}$
$C_{f,1}^{t}$
$C_{f,2}^{t}$
$C_{f,3}^{t}$


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

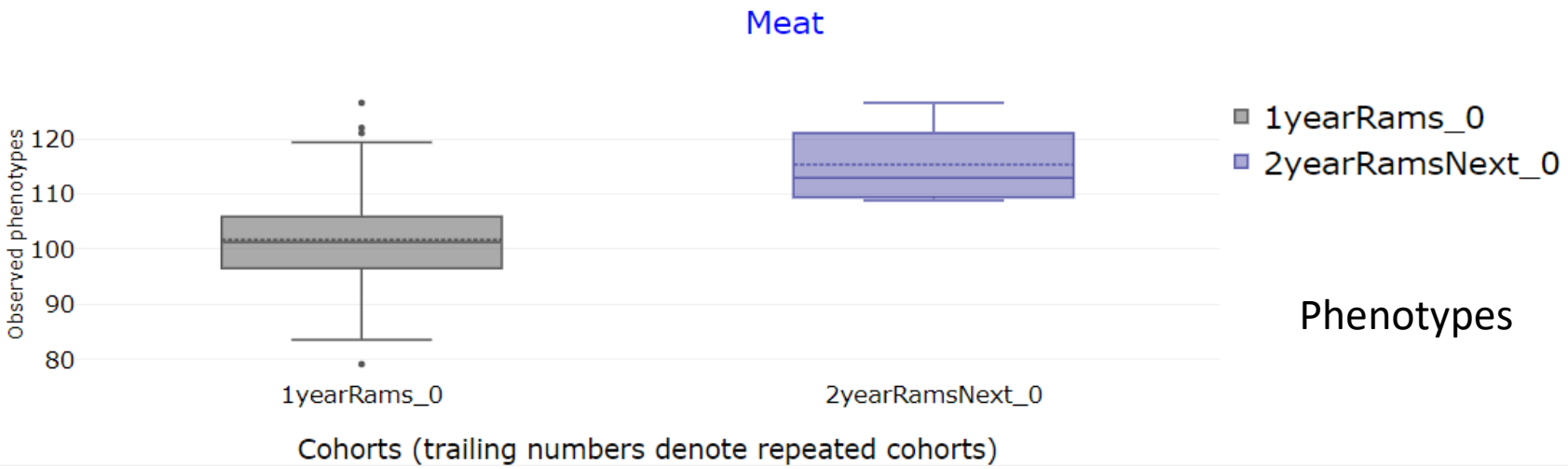
Task 1: Sheep Breeding Program

- 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 $h^2 = 0.3$
- Selection on the male side is according to 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

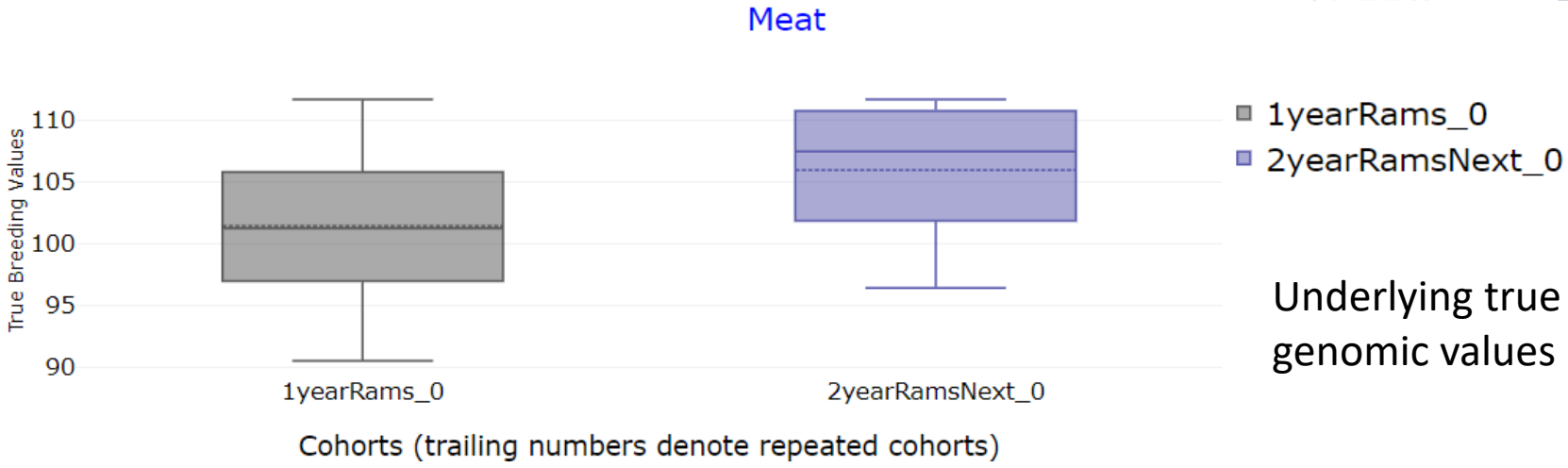
- All details are given in the template: „Simple Sheep“



MoBPSweb analysis

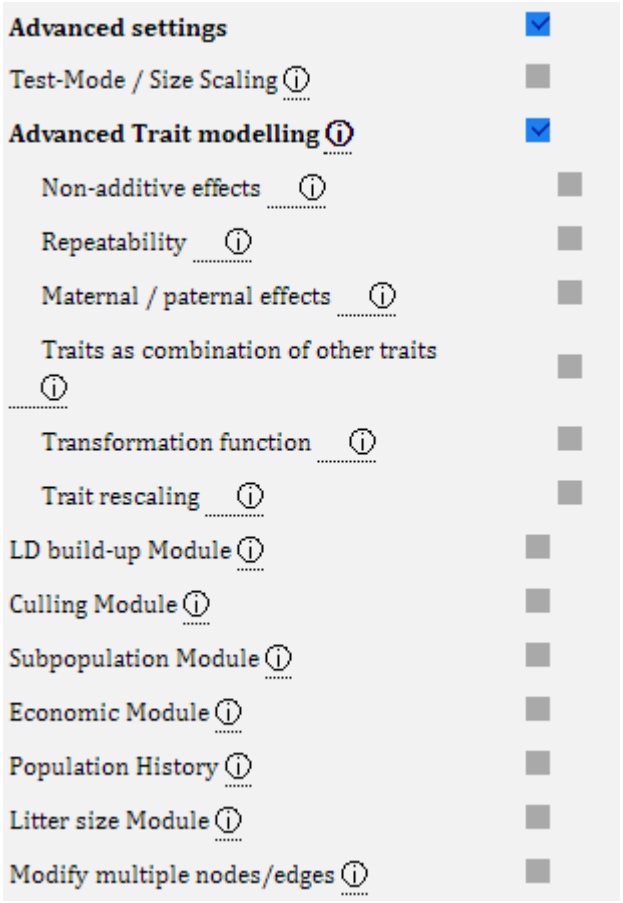


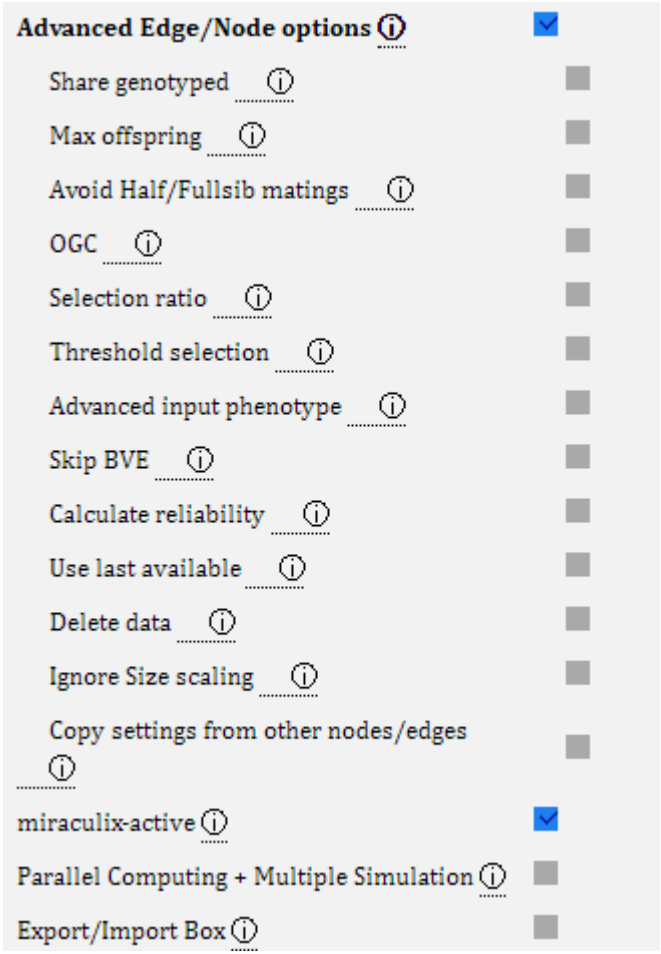
Phenotypes



Underlying true genomic values

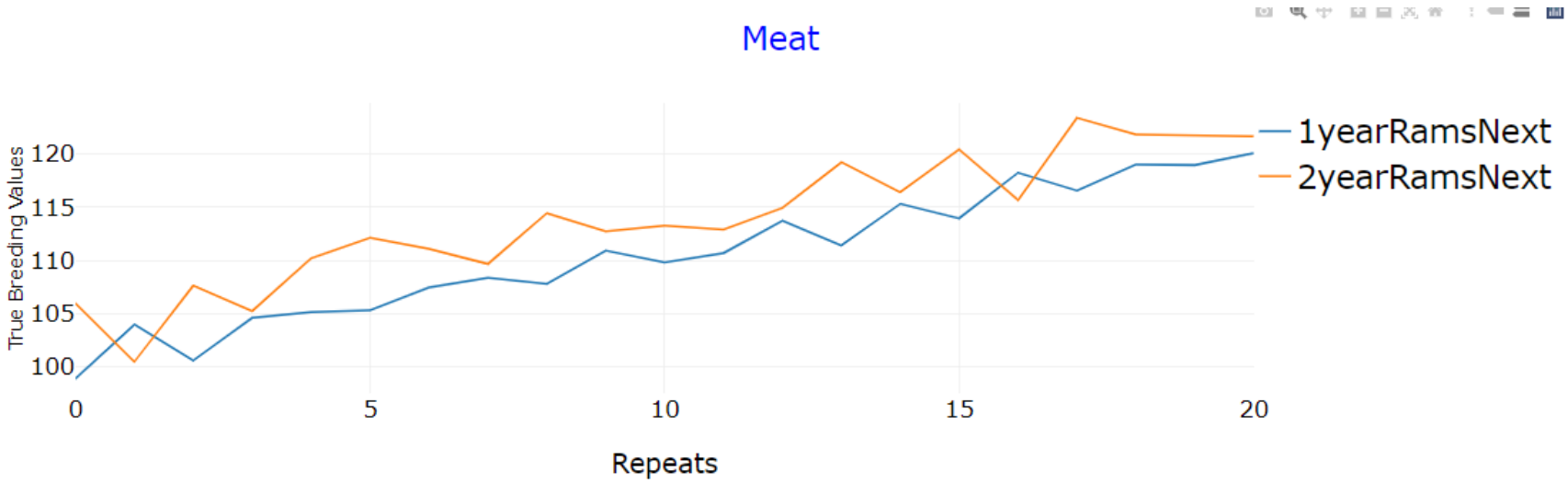
Advanced options in MoBPSweb

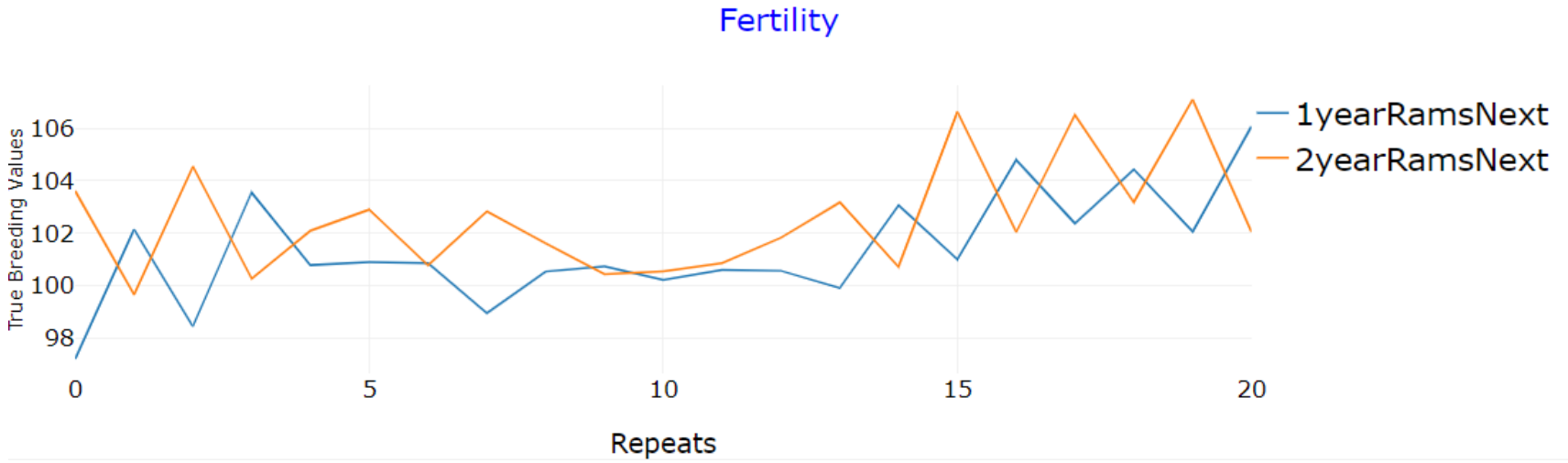


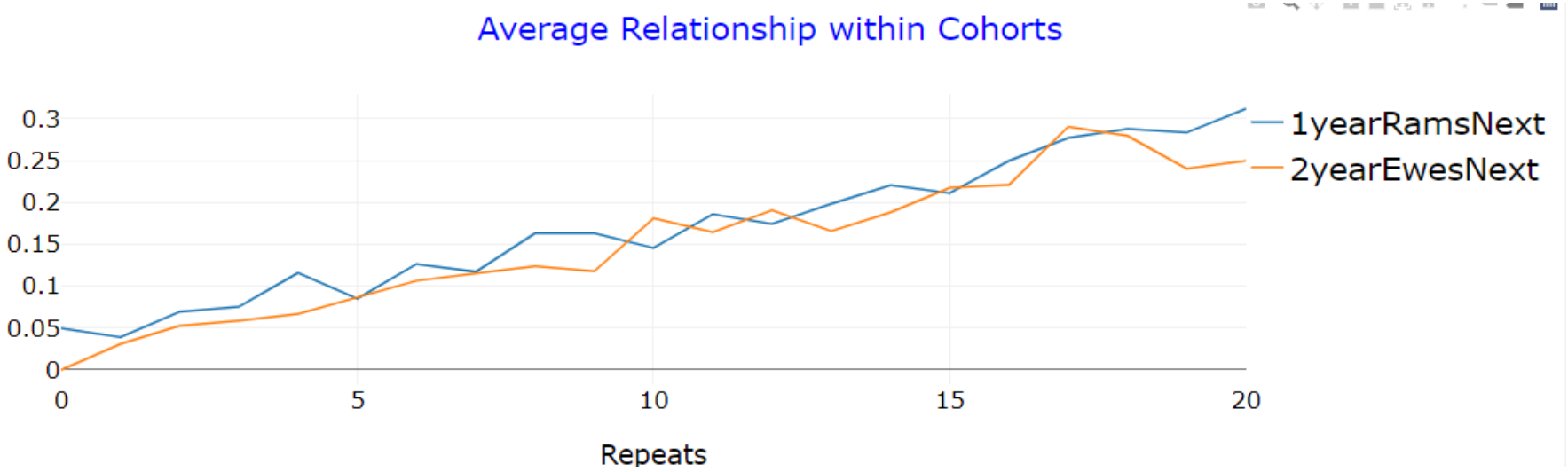


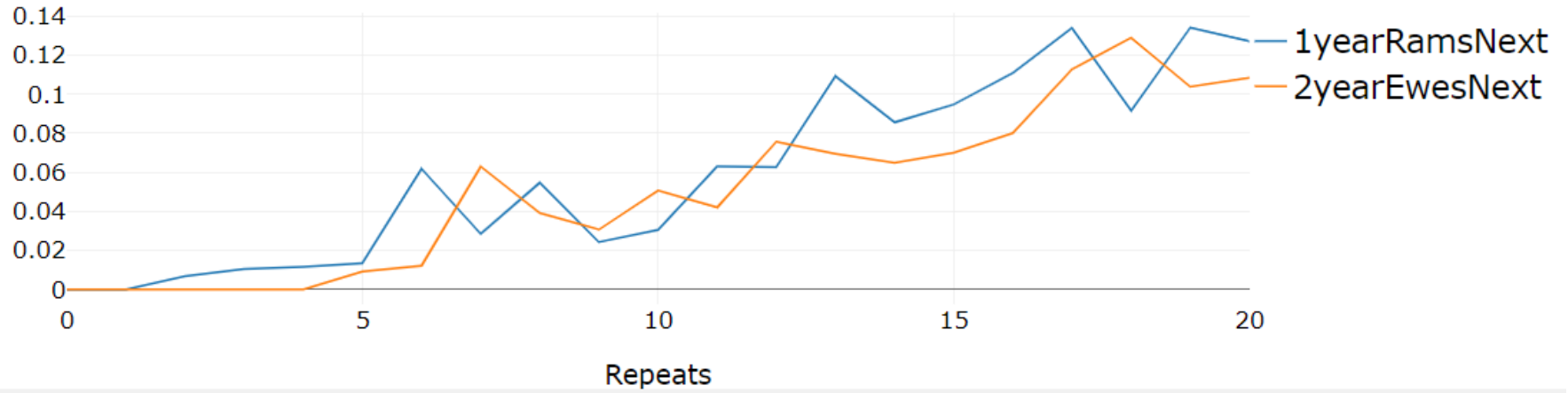
Task 2: Make it more realistic

- On default founders are unrelated
  - 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 (which is the default settings!)
- Use the IlluminaOvineSNP50 array as an underlying genomic map


Meat
True Breeding Values
120
115
110
105
100
0
5
10
15
20
Repeats
1yearRamsNext
2yearRamsNext


Fertility
True Breeding Values
106
104
102
100
98
0
5
10
15
20
Repeats
1yearRamsNext
2yearRamsNext

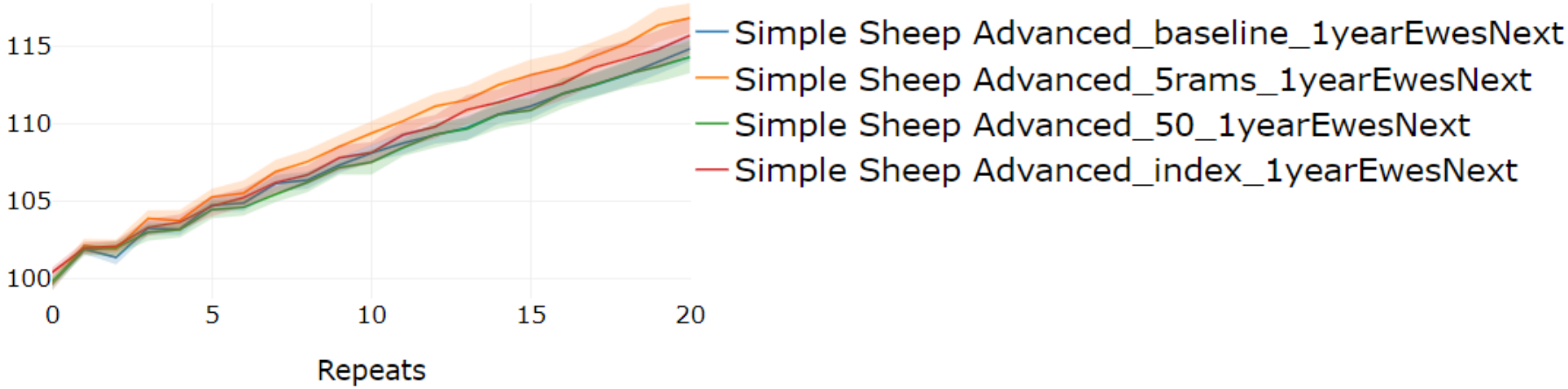

Average Relationship within Cohorts
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
Repeats
1yearRamsNext
2yearEwesNext


Average Inbreeding within Cohorts
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
5
10
15
20
Repeats
1yearRamsNext
2yearEwesNext

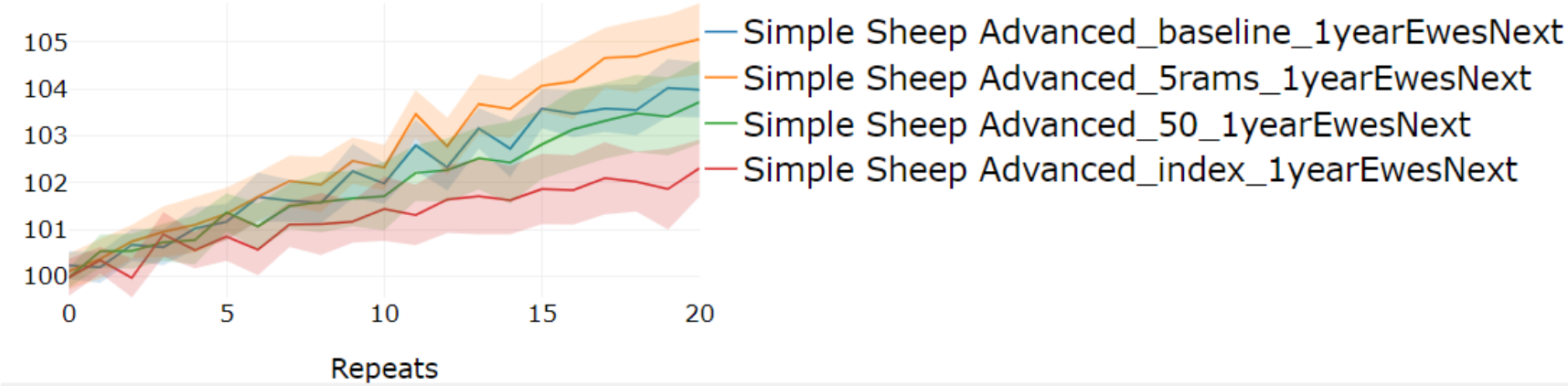
Task 3: Comparision of scenarios

- 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 10 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!

Meat



Fertility



- Even 25 simulations over 20 generations are not enough to distinguish performance levels for fertility
- Potential reasons:
  - Low accuracy in breeding value esimation
  - Small population size
  - Small differences between scenarios
  - High variance in the outcome of simulations
- Breeding in practise also has random factors!

Task 4: Install R-package

- Install our packages
  - MoBPS (1.8.04)
  - 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 miraculix
  - Examples in this workshop will be so small that this is not needed!

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")
```

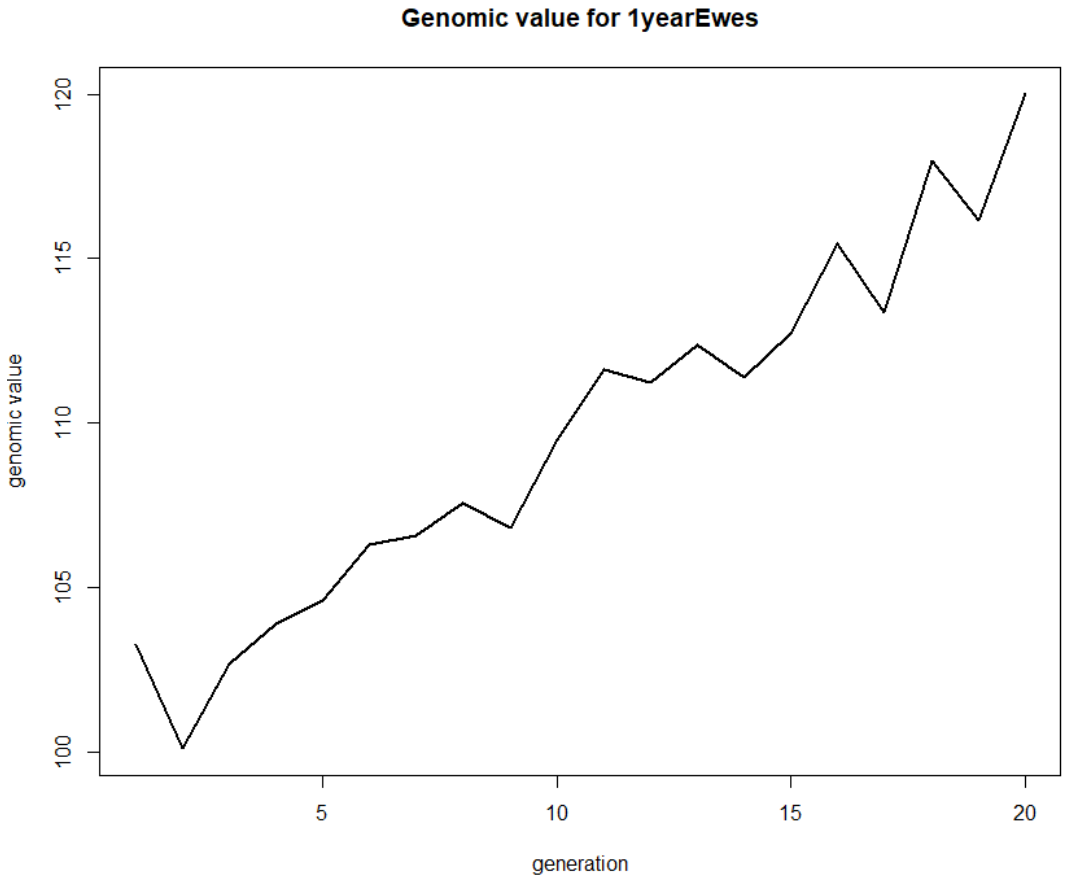
```r
# Perform a breeding value estimation using all individuals
# of generation 1 (which are all).
pop <- breeding.diploid(pop, bve=TRUE,
                        bve.gen = 1)

# 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.m="function",
                         selection.m.cohorts="Founder_M",
                         selection.f.cohorts="Founder_F",
                         name.cohort="Offspring")
```

- Position of individuals:
  - „gen“: The generation each new group 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

- Sex
  - This is mostly intended to help you structure your simulation
  - For plant breeding you can oftentimes ignore sex
  - Sex is not binding for downstream operations
    - You can mate females with females, phenotype males for milk traits, etc.
    - selection.m.cohorts in a plant breeding program should be seen as the first parent used in reproduction and can also use individuals stored as females

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
- Perform the simulation from „Simple Sheep“ directly in R


Genomic value for 1yearEwes
genomic value
100
105
110
115
120
5
10
15
20
generation

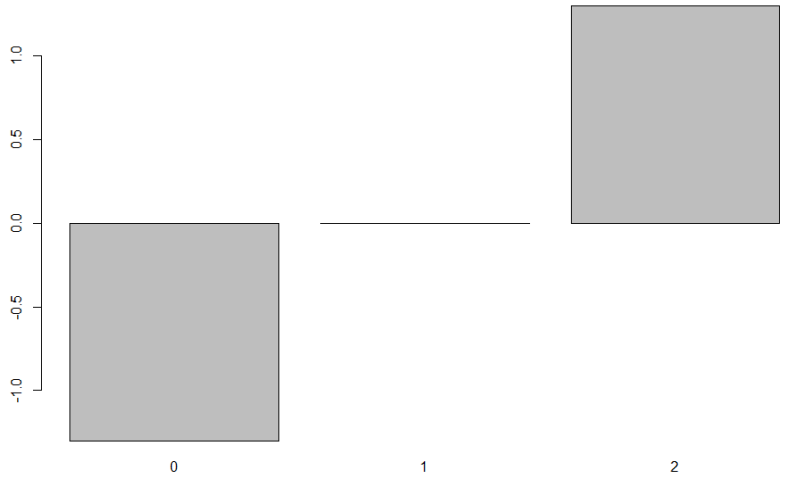
```
> get.cohorts(population, extended = TRUE)
              name              generation male individuals female individuals class position first male
1yearRams     "1yearRams"       "1"        "50"             "0"                "0"   "1"
2yearRams     "2yearRams"       "1"        "10"             "0"                "0"   "51"
1yearEwes     "1yearEwes"       "1"        "0"              "50"               "0"   "61"
2yearEwes     "2yearEwes"       "1"        "0"              "40"               "0"   "61"
3yearEwes     "3yearEwes"       "1"        "0"              "30"               "0"   "61"
1yearRamsNext "1yearRamsNext"   "2"        "50"             "0"                "0"   "1"
1yearEwesNext "1yearEwesNext"   "2"        "0"              "50"               "0"   "51"
2yearRamsNext "2yearRamsNext"   "2"        "10"             "0"                "0"   "51"
2yearEwesNext "2yearEwesNext"   "2"        "0"              "40"               "0"   "61"
3yearEwesNext "3yearEwesNext"   "2"        "0"              "30"               "0"   "61"
              position first female time point creating.type
1yearRams     "1"                     "0"        "0"
2yearRams     "1"                     "0"        "0"
1yearEwes     "1"                     "0"        "0"
2yearEwes     "51"                    "0"        "0"
3yearEwes     "91"                    "0"        "0"
1yearRamsNext "1"                     "0"        "0"
1yearEwesNext "1"                     "0"        "0"
2yearRamsNext "51"                    "0"        "0"
2yearEwesNext "51"                    "0"        "0"
3yearEwesNext "91"                    "0"        "0"
```

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

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
```

On the generation of traits

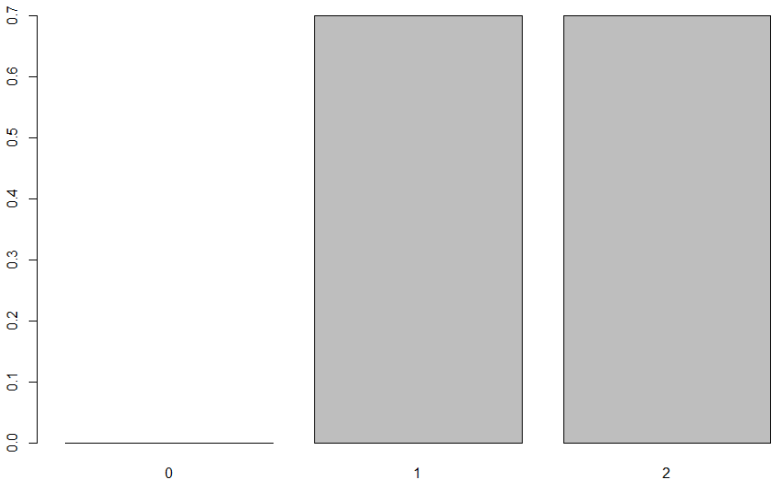
On the generation of traits

- 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)



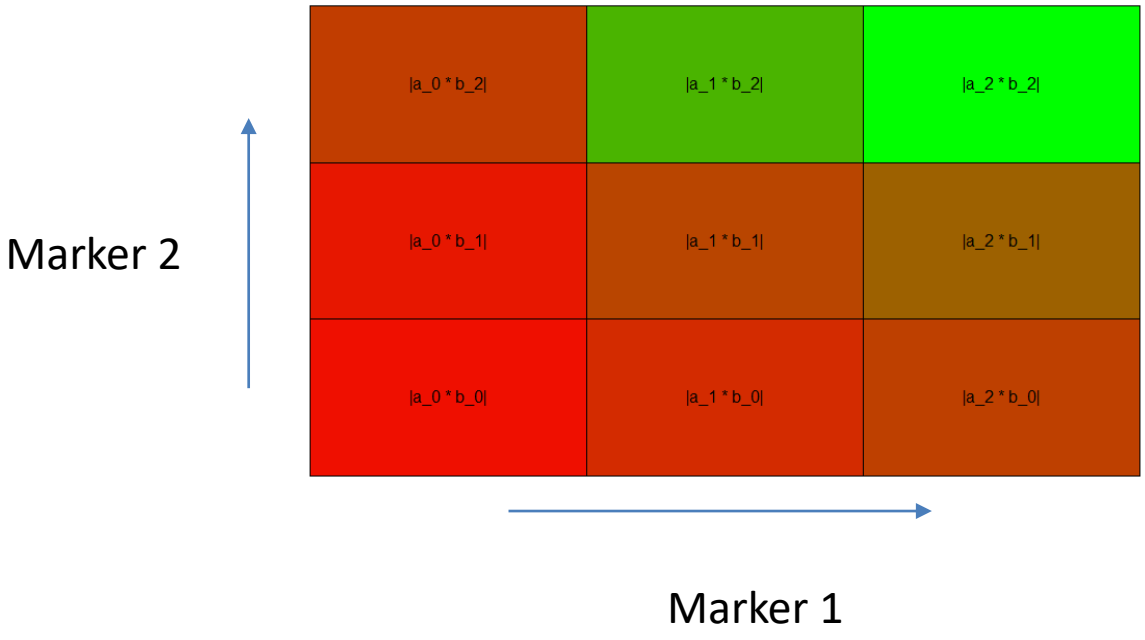
On the generation of traits

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



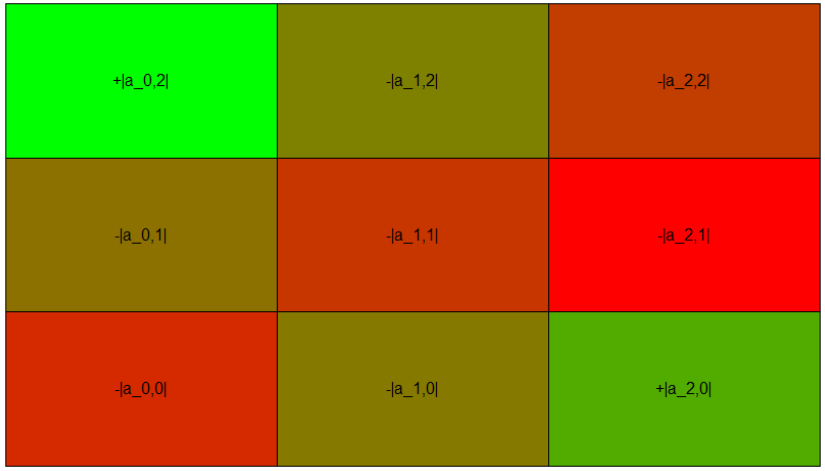
On the generation of traits

- 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
  - Two marker interaction: real.bv.mult
  - More than two marker interaction: real.bv.dice

```
> real.bv.add
     SNP chromosome Effect 0 effect 1 effect 2
[1,] 120          1     -1.0      0.0        1
[2,]  42          5      0.0      0.0        2
[3,]  17         22      0.1      0.1        0
```

- There is a QTL on marker 120 on chromosome 1
  - Homozygosity in the first allele has an effect of -1
  - Heterozygosity has an effect of 0
  - Homozygosity in the second allele has an effect of 1

- An so on…

```
> real.bv.mult
     First SNP First chromosome Second SNP Second chromosome effect 00 effect 01 effect 02 effect 10 effect 11 effect 12 effect 20 effect 21 effect 22
[1,]       144                1        145                 1      1.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
[2,]         6                3        188                 5      0.37      0.16      1.33      1.49      1.58      2.51      0.38      2.12      0.98
[3,]         5               17          1                10      1.18      2.60      0.18      1.74      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 more, but trait correlation will stay consistent over time

- The definition of heritability is in line with definitions from animal breeding
  - Heritability for a single observation / plot
  - With 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

Genomic data

- For all individuals in the population, underlying haplotypes are 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 with markers to include in the breeding value estimation directly

Founder genotypes

- Options for generating founder haplotypes:
  - Default: Randomly generated
    - Pro: fast
    - Con: 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

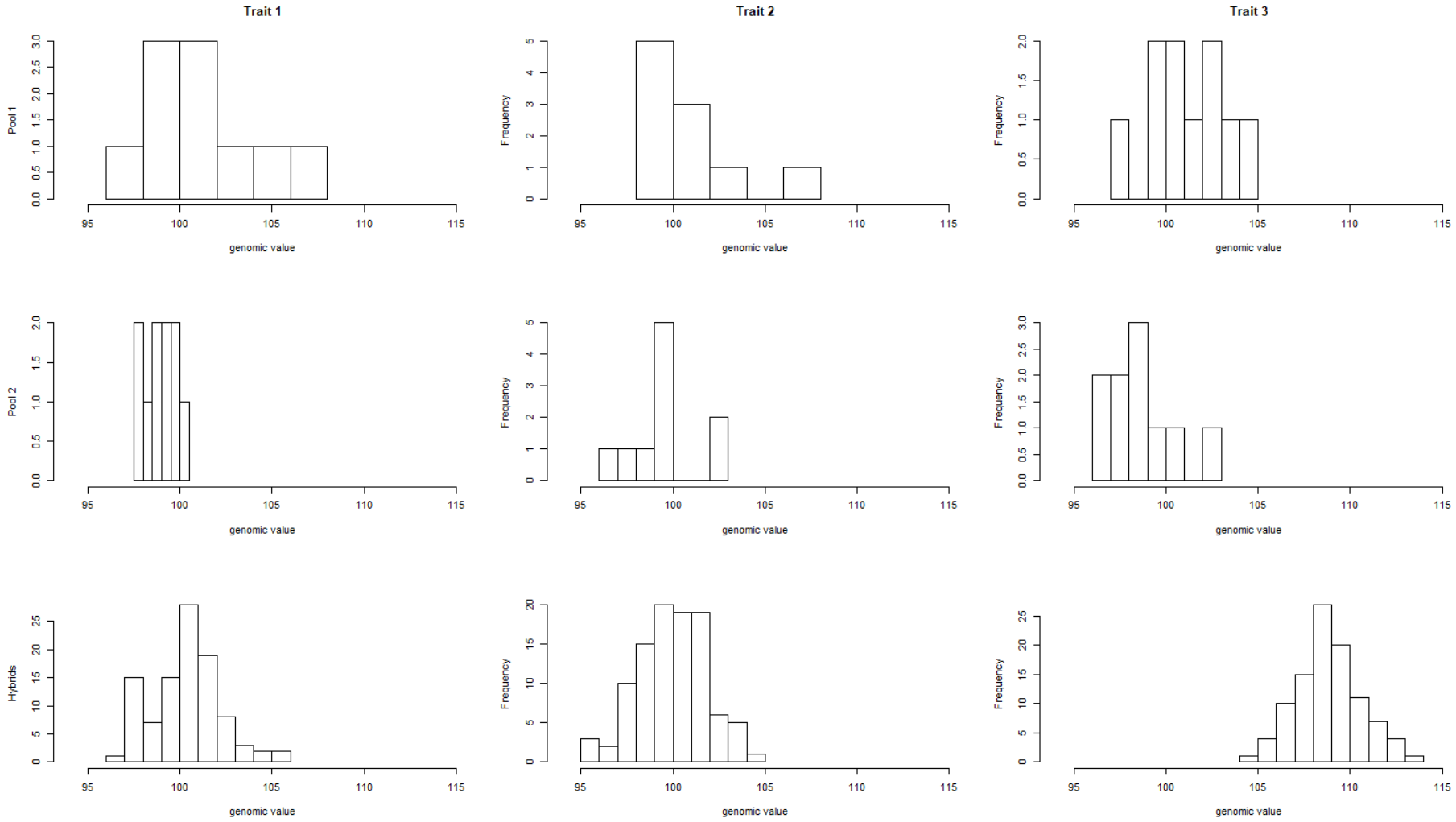
- Add three traits:
  - One with 10 purely additive QTLs
  - One with 1000 purely additive QTLs
  - One with 500 purely additive QTLs and 500 dominant QTLs of equal size
  - Traits should be uncorrelated
  - For all traits phenotypic mean for the founders should be 100 with a genetic variance of 5

- Generate 100 offspring by random mating between individuals from the two gene pools

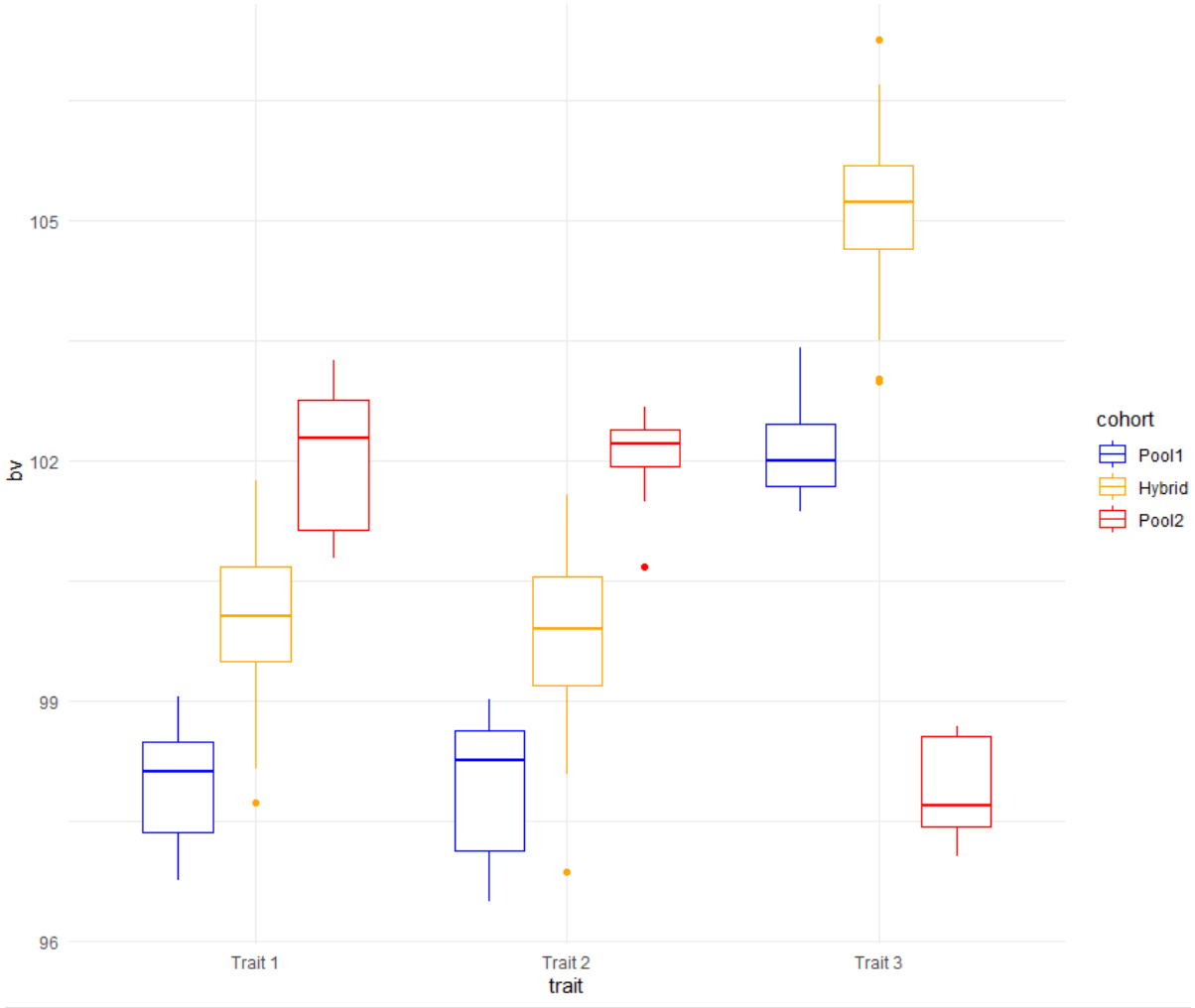
- Compare the genomic values of the parents and offspring

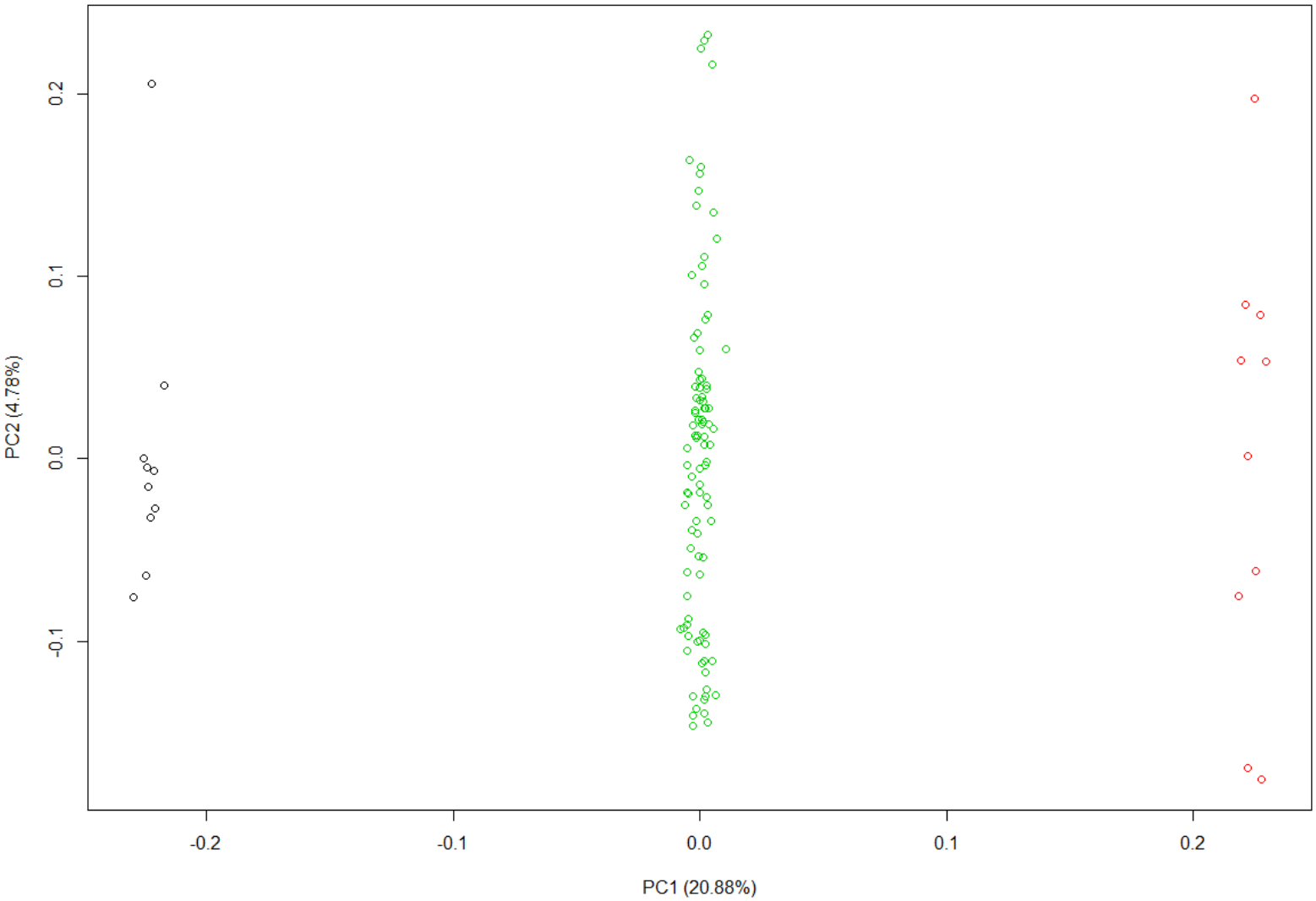
- How often was each individual used for reproduction?

- Generate a PCA for all simulated individuals


Trait 1
Trait 2
Trait 3
Pool 1
Pool 2
Hybrids
Frequency
genomic value

Or if you prefer ggplot




PC2 (4.78%)
0.2
0.1
0.0
-0.1
-0.2
-0.1
0.0
0.1
0.2
PC1 (20.88%)

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“)
- Selection can be based on all 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 breeding value estimations for individuals in generation 10, 11, 12 or combinations of the cohorts

    Use:
    - Genomic breeding value estimation
    - Pedigree-based breeding value estimation
    - Single-step
    - Assume individuals are only genotyped for 10‘000 / 100 randomly selected markers
  - Generate a plot to showcase real genomic values and breeding values for generation 10.

- 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, singlestep.active = TRUE)
Start genomic BVE.
Start derive Single-step relationship matrix
Construct pedigree matrix for 100 individuals.
Derive pedigree-matrix based for  100  individuals based on  797  individuals.
Derived pedigree matrix in  0.06 seconds.
Start deriving of H matrix for 18 genotyped and 82 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 = 254.0462; sigma_e^2 = 1111.8964 ; h^2 = 0.186
0 seconds for BVE.
Correlation between genetic values and BVE:
0.4442224
```

```
> 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

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

```
> analyze.bv(population, database = cbind(10,1))
[[1]]
               Trait 1
BV / BVE     0.6149589
BV / Pheno   0.5781879
BVE / Pheno  0.9635030

[[2]]
Trait 1
476.527

> analyze.bv(population, database = cbind(10,2))
[[1]]
                  Trait 1
BV / BVE     -0.004783966
BV / Pheno             NA
BVE / Pheno            NA

[[2]]
Trait 1
230.535
```

- MoBPS „only“ provides standard approaches for selection & breeding value estimation
- On-going project MoBPSplant
- We are happy to add new functionality

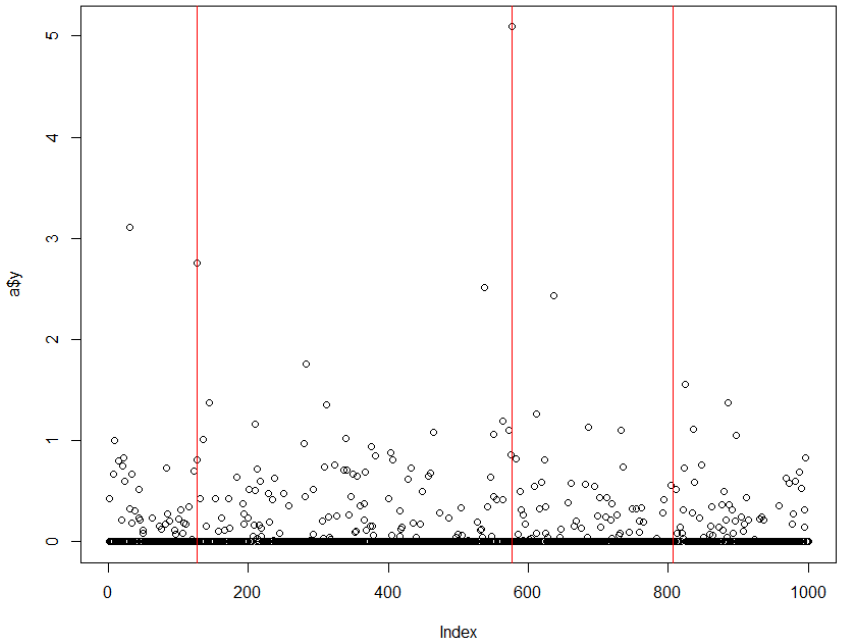
- Use of own methodology is also possible
- MoBPS takes care of phenotype simulation, meiosis, computational efficiency etc.

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)
```

- What if the phenotypes of related individuals are of interest?
  - Cattle breeding: Bulls do not give milk
  - Maize: Performance of a line crossed to a tester from a different gene pool

Task 8: Offspring phenotypes

- Simulate a population with 10 male individuals and 90 female individuals
- Generate a single trait with 1000 purely additive QTL

- Generate 45 male and 45 female offspring
- Generate phenotypes for all female offspring
- Calculate the average phenotype of the offspring for each respective founder

- Perform a breeding value estimation. Is more reliable than average offspring phenotype for selection?

- Males on average have substantially more offspring in this breeding scheme
- Some females do not have any phenotyped offspring

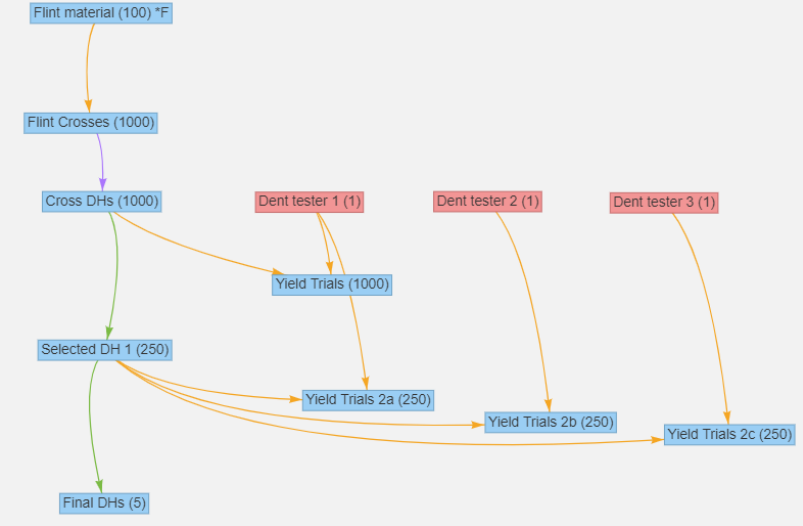
```
> get.pheno.off(population, gen=1)
          M1_1     M2_1     M3_1     M4_1     M5_1     M6_1     M7_1     M8_1     M9_1    M10_1 F1_1 F2_1     F3_1 F4_1
Trait 1 104.2653 99.27285 97.90456 97.04388 104.3544 94.86267 99.63414 97.01326 97.75193 100.9571   NA   NA 100.6894   NA
        F5_1 F6_1     F7_1 F8_1     F9_1    F10_1 F11_1    F12_1 F13_1 F14_1 F15_1 F16_1    F17_1 F18_1 F19_1 F20_1 F21_1
Trait 1   NA   NA 109.6192   NA 96.80363 95.84827    NA 94.81592    NA    NA    NA    NA 100.0315    NA    NA    NA    NA
        F22_1 F23_1    F24_1    F25_1    F26_1 F27_1    F28_1    F29_1 F30_1 F31_1 F32_1 F33_1    F34_1   F35_1 F36_1
Trait 1    NA    NA 95.73383 97.99519 99.61739    NA 97.13684 96.43145    NA    NA    NA    NA 97.23148 104.321    NA
        F37_1    F38_1 F39_1 F40_1    F41_1 F42_1 F43_1    F44_1 F45_1   F46_1 F47_1 F48_1 F49_1    F50_1 F51_1    F52_1
Trait 1    NA 103.0486    NA    NA 105.8705    NA    NA 88.39208    NA 94.5238    NA    NA    NA 108.9811    NA 93.76115
        F53_1    F54_1 F55_1 F56_1 F57_1 F58_1 F59_1    F60_1    F61_1 F62_1   F63_1 F64_1 F65_1    F66_1 F67_1 F68_1
Trait 1    NA 95.48397    NA    NA    NA    NA    NA 102.9172 101.8986    NA 101.127    NA    NA 93.08994    NA    NA
        F69_1    F70_1 F71_1 F72_1 F73_1    F74_1 F75_1    F76_1 F77_1    F78_1    F79_1 F80_1 F81_1 F82_1    F83_1
Trait 1 103.4797 104.8924    NA    NA    NA 100.6212    NA 96.30255    NA 96.66566 94.82729    NA    NA    NA 99.25576
          F84_1 F85_1 F86_1    F87_1 F88_1    F89_1 F90_1
Trait 1 108.4208    NA    NA 94.71207    NA 90.73539    NA
> get.pheno.off.count(population, gen=1)
        M1_1 M2_1 M3_1 M4_1 M5_1 M6_1 M7_1 M8_1 M9_1 M10_1 F1_1 F2_1 F3_1 F4_1 F5_1 F6_1 F7_1 F8_1 F9_1 F10_1 F11_1 F12_1
Trait 1    3    4    8    6    3    3    4    6    5     3    0    0    1    0    0    0    1    0    1     1     0     1
        F13_1 F14_1 F15_1 F16_1 F17_1 F18_1 F19_1 F20_1 F21_1 F22_1 F23_1 F24_1 F25_1 F26_1 F27_1 F28_1 F29_1 F30_1 F31_1
Trait 1     0     0     0     0     1     0     0     0     0     0     0     1     1     1     0     3     1     0     0
        F32_1 F33_1 F34_1 F35_1 F36_1 F37_1 F38_1 F39_1 F40_1 F41_1 F42_1 F43_1 F44_1 F45_1 F46_1 F47_1 F48_1 F49_1 F50_1
Trait 1     0     0     1     2     0     0     1     0     0     1     0     0     1     0     2     0     0     0     1
        F51_1 F52_1 F53_1 F54_1 F55_1 F56_1 F57_1 F58_1 F59_1 F60_1 F61_1 F62_1 F63_1 F64_1 F65_1 F66_1 F67_1 F68_1 F69_1
Trait 1     0     1     0     1     0     0     0     0     0     1     2     0     1     0     0     2     0     0     2
        F70_1 F71_1 F72_1 F73_1 F74_1 F75_1 F76_1 F77_1 F78_1 F79_1 F80_1 F81_1 F82_1 F83_1 F84_1 F85_1 F86_1 F87_1 F88_1
Trait 1     1     0     0     0     2     0     2     0     2     2     0     0     0     1     1     0     0     1     0
        F89_1 F90_1
Trait 1     1     0
```

Task 9: 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
    - Hint: there are various ways to do this! – You can use a PCA to confirm
  - Simulate a single trait with 1‘000 QTLs
  - Use 10‘000 SNPs. You can use a subset of the maize 600k array from MoBPSmaps

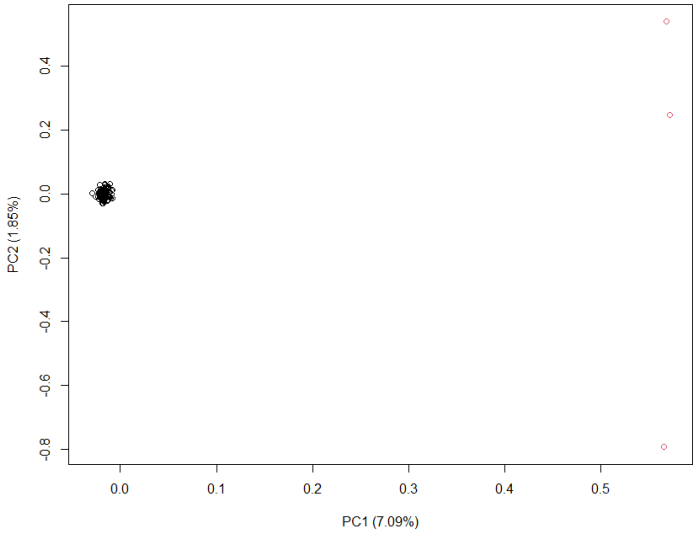
- Generate 1000 crosses within the Flint gene pool
- Generate 1000 DH lines from the crosses

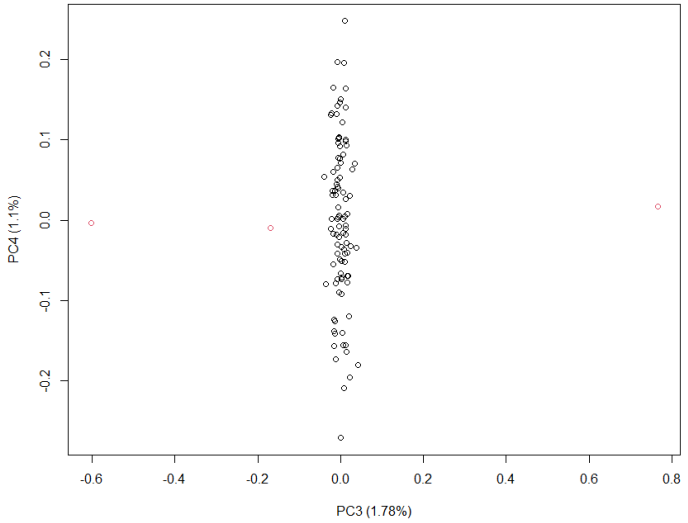
- Mate all DHs to one of the three dent lines
- Phenotype the offspring (h^2 = 0.3)
- Select the top 250 DHs lines based on the performance in the yield trial
  - Hint: make sure that each DH is cross to the tester exactly once!



- 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

- 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



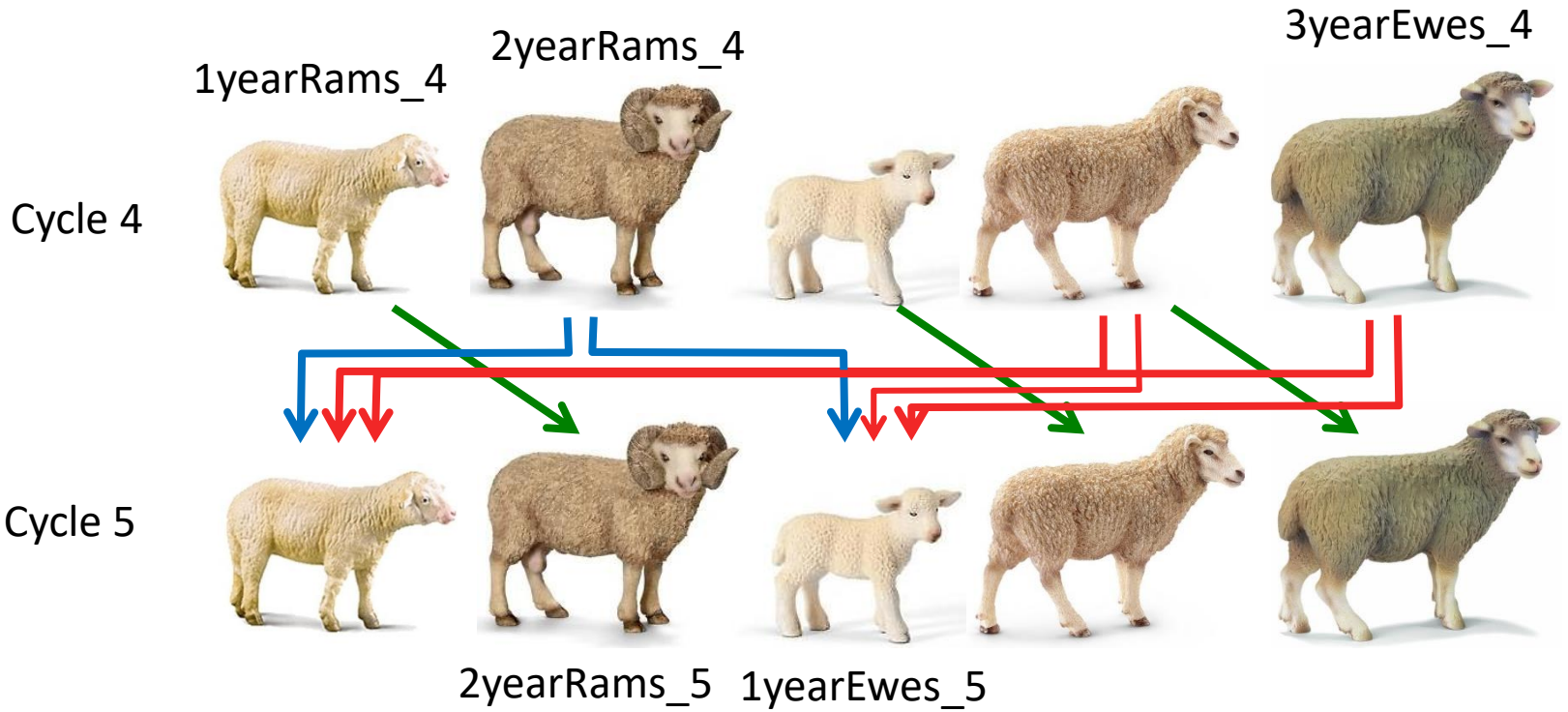


- Analysis of breeding program should usually consider multiple cycles of a breeding program
- The first few cycles of a breeding program are oftentimes not reliable:
  - E.g., a pedigree-based breeding value estimation in founder individuals will not work when no relations between individuals are known

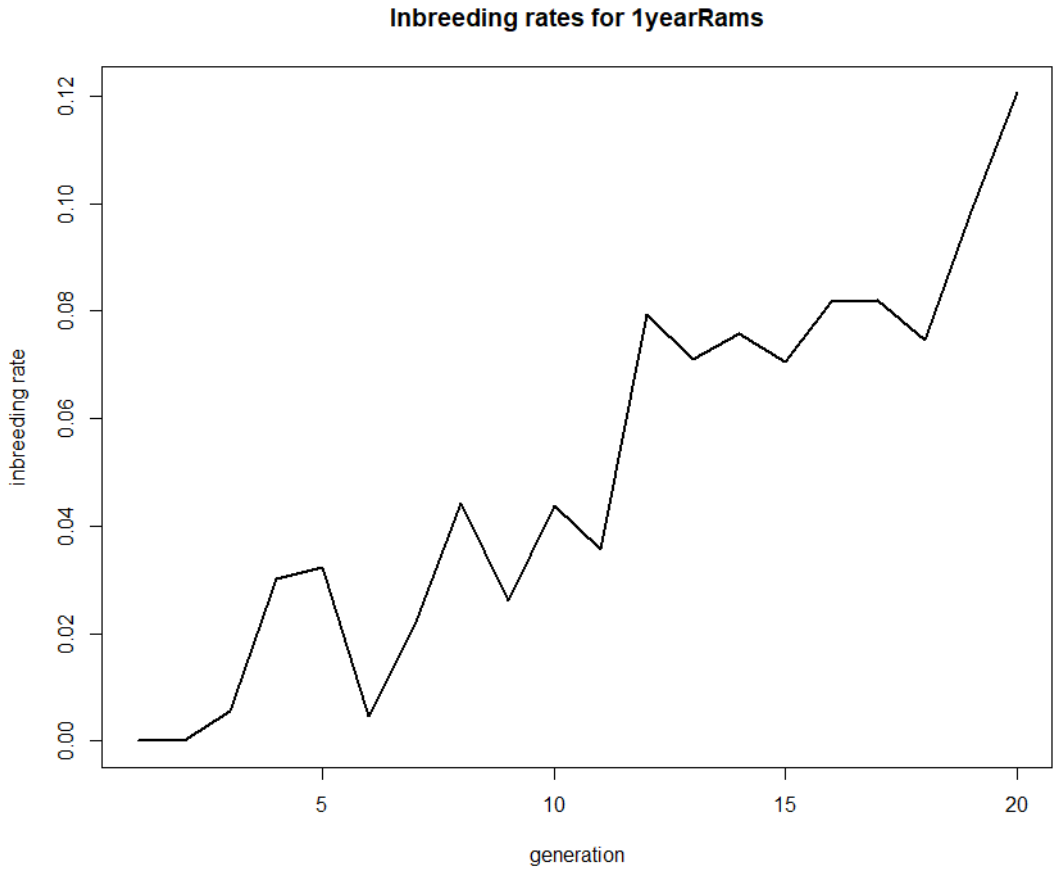
→ Necessarity to run multiple breeding cycles

Task 10

- 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!



```
> rowMeans(get.bv(population, cohorts="1yearRams_20"))
 Trait 1  Trait 2
118.2650 104.0191
> rowMeans(get.bv(population, cohorts="2yearRams_20"))
 Trait 1  Trait 2
118.5768 105.5124
```



Task 11

- Extend the script from Task 10 to allow for the simulation of multiple runs and multiple scenarios

- Simulate the four scenarios considered in Task 3 and analyze the outcome:

    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

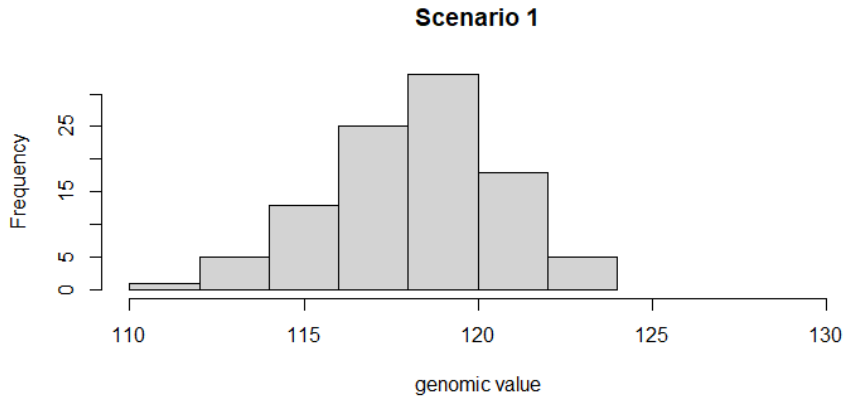
- Evaluate underlying true genomic values, kinships and prediction accuracies for all cohorts:

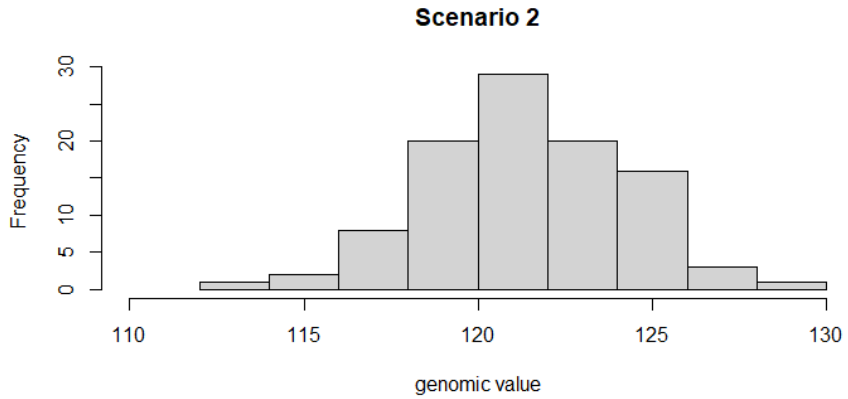
```
cohorts <- get.cohorts(population)

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"))
```

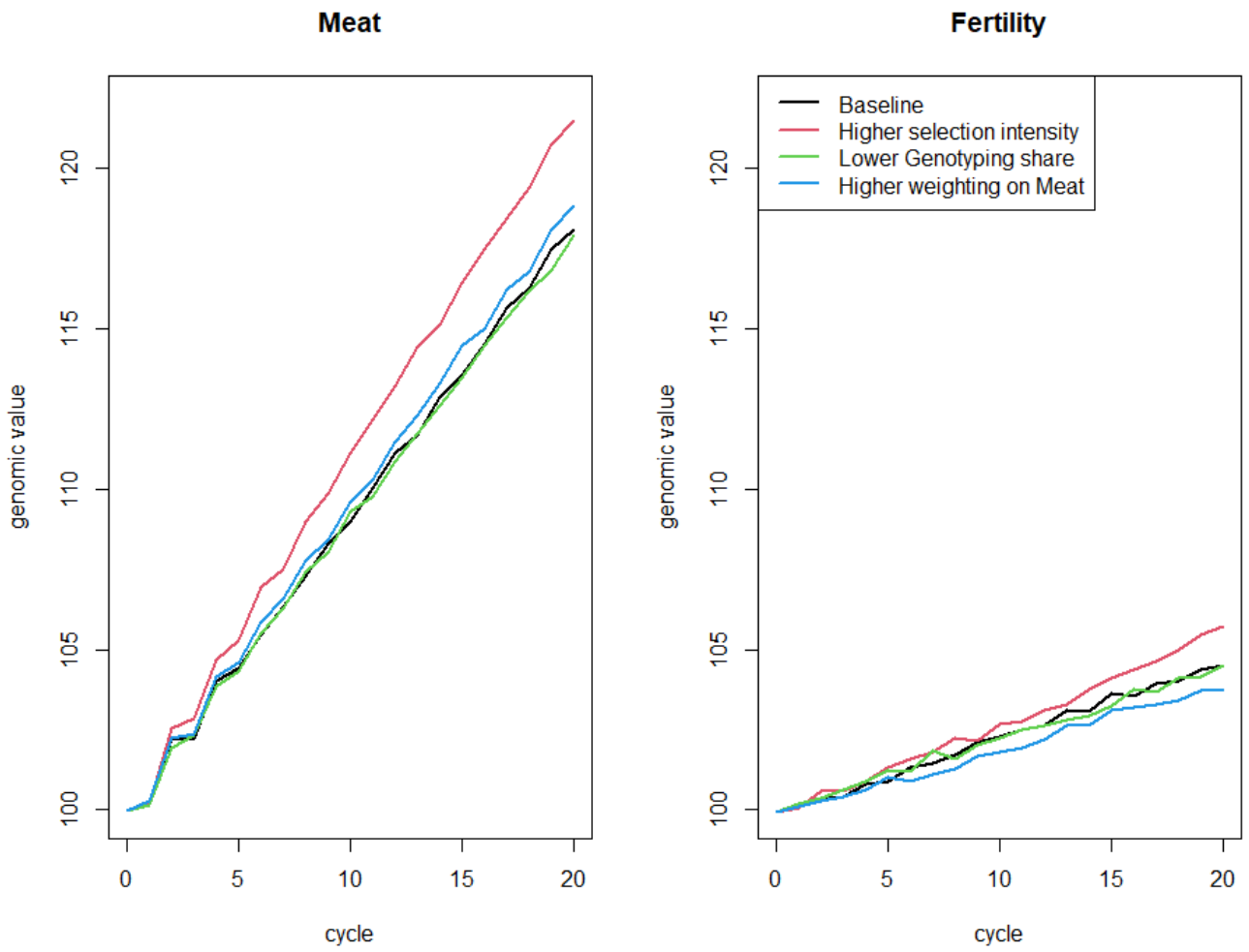
Task 12

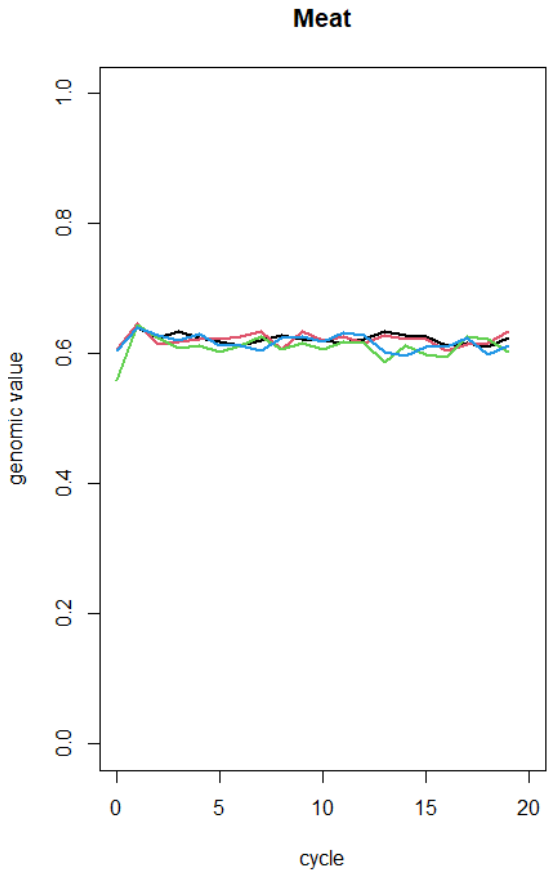
- Each of the four scenarios has been run 100 times by us
- Analyze the results of the simulation:
  - Are genetic gains for the meat trait different between scenario 1 & 2 after 20 cycles – use a t-test
  - Are the obtained prediction accuracies different in scenario 1 & 3
  - Generate a plot showing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

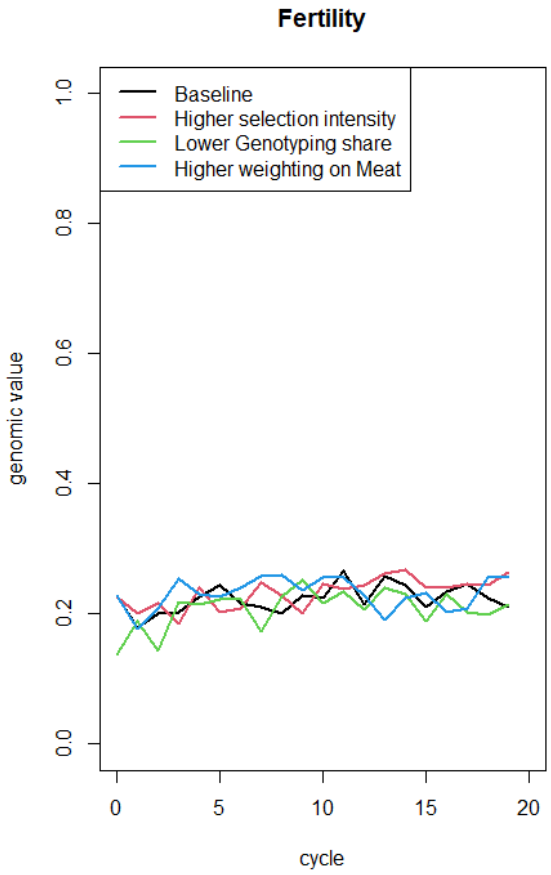
```
> t.test(bvs[[1]], bvs[[2]])

        Welch Two Sample t-test

data:  bvs[[1]] and bvs[[2]]
t = -8.67, df = 193.84, p-value = 1.718e-15
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -4.015541 -2.527181
sample estimates:
mean of x mean of y
 118.1367  121.4080
```






Meat
Fertility
Baseline
Higher selection intensity
Lower Genotyping share
Higher weighting on Meat
genomic value
cycle
100
105
110
115
120
0
5
10
15
20






```
> mean(ACC_avg[[1]][2,to_plot], na.rm=TRUE)
[1] 0.2233643
> mean(ACC_avg[[2]][2,to_plot], na.rm=TRUE)
[1] 0.2325648
> # less genotyping leads to lower accuracies
> mean(ACC_avg[[3]][2,to_plot], na.rm=TRUE)
[1] 0.2082105
> mean(ACC_avg[[4]][2,to_plot], na.rm=TRUE)
[1] 0.2315578
> mean(ACC_avg[[1]][1,to_plot], na.rm=TRUE)
[1] 0.6223516
> mean(ACC_avg[[2]][1,to_plot], na.rm=TRUE)
[1] 0.6220569
> # less genotyping leads to lower accuracies
> mean(ACC_avg[[3]][1,to_plot], na.rm=TRUE)
[1] 0.6099669
> mean(ACC_avg[[4]][1,to_plot], na.rm=TRUE)
[1] 0.6171243
```

Differences between approaches are minor!

Acknowledgments

- Gesellschaft für Tierzuchtwissenschaften e.V.
- KWS SAAT SE & Co. KGaA
- European Union‘s Horizon 2020 Research and Innovation Program
  - Grant agreement n 677353 IMAGE
- Project: MAZE - Accessing the genomic and functional diversity of maize to improve quantitative traits (031B0882E)
- Animal breeding and Genetics Group
  University of Goettingen