Introduction to Genetic Simulation

Philipp Heilmann

1 Crossing experiments **2**

1.1 Form of seeds: Mendel's experiments, one trait 2
1.2 Color and form of seeds: Mendel's experiments, two raits . . . 3
1.3 Backcrossing and recombination 4

2 Random Mating Populations **5**

2.1 F2 Population 5
2.2 Population of homozygous lines 6
2.3 Arbitrary allele frequencies 7

3 Populations under selection **8**

3.1 Selection against a recessive allele 8
3.2 Selection for fitness 11

4 Appendix: multiple traits **13**

1 Crossing experiments

1.1 Form of seeds: Mendel's experiments, one trait

is to define the genome and the positions of the loci to be simulated. There are two ways to do this. The simpler approach is to define a data frame containing one locus per row. Each row specifies the chromosome, the position on that chromosome, and a locus name. An optional fourth column specifies the locus class.

The locus name can be chosen freely. The class can be used to group loci, for example as markers or traits, and is used later when defining effects for selection. If the class column is omitted, define.genome() generates class names automatically. An arbitrary number of loci can be defined, each in a separate row.

```
> map <- data.frame(chrom = 1,
+                   pos = 0.0,
+                   name = "form")
> define.genome(map)
```

The number of chromosomes and their respective lengths are inferred from the data we provided.

Alternatively, the genome and the linkage map can be defined explicitly and separately. The definition of the linkage map consists of a sequence of information for each locus: locus-name class-name chromosome/position. An arbitrary number of loci can be defined, separated by commas. However: All loci must be defined on the same line of R input code!

```
> set.genome.par ( no.chrom = 1,       # Only one chromosome
+                  no.hom= 2,          # Diploid species
+                  chrom.len = 0.01 )  # Chromosome length 1cM
> define.map("form trait1 1/0")        # Chromosome1 / Position 0
```

Now we define the parental lines of the cross. They have the names P1 and P2 and are homozygous for the two alleles 1 and 2, respectively. Alleles must be coded with numbers, not with characters. The 1 stands for round seeds and the 2 for wrinkled. We know from previous experiments that round is dominant over wrinkled.

```
> init.population("P1", homozygote(1))
> init.population("P2", homozygote(2))
>
```

Then we cross the parental lines, the first argument of the cross command is the name of the newly generated population followed by the name of the two parental populations. The number at the end is the number of progenies.

```
> cross("F1","P1", "P2", 563)
```

We now look at our populations. The first argument of the `evaluate.genotype` command is the name of the population, the second is the locus name that should be evaluated. P1 consists of one individual that carries at both homologous chromosomes at the locus form the allele 1.

```
> evaluate.genotype("P1","form")
```

```
   form-1  form-2  count frequency
1       1       1     1         1
```

P2 consists of one plant that carries two times the allele 2

```
> evaluate.genotype("P2","form")
```

```
   form-1  form-2  count frequency
1       2       2     1         1
```

and F1 consists of 563 plants that carry one copy of allele 1 and one copy of allele 2. Because round is dominant, they are all round.

```
> evaluate.genotype("F1","form")
```

```
   form-1  form-2  count frequency
1       1       2   563         1
```

Mendel's 1 law: *The progenies from crosses of homozygous lines are uniform and heterozygous.*

We take a sample of 252 out of the 563 F1 plants

```
sample.population ("SampleFromF1","F1",252)
```

and self these. From the selfing we get 7223 seeds.

```
cross("F2","SampleFromF1","SampleFromF1",7323,self=1)
```

We count the wrinkled seeds and find 1853 of them.

```
evaluate.genotype("F2","form")
```

```
> evaluate.genotype("F2","form")
```

```
   form-1  form-2  count frequency
1       1       1  1855   0.25331
2       1       2  3662   0.50007
3       2       2  1806   0.24662
```

Mendel's 2 law: *After selfing the F1s from crosses of homozygous lines, the progenies in the F2 are segregating.*

On the genotypic level (which we simulate) the segregation ratio is 1:2:1. Depending on dominant/recessive or co-dominant inheritance this is a segregation of either 3:1 or 1:2:1 for the phenotypes.

1.2 Color and form of seeds: Mendel's experiments, two raits

We now consider two traits. We first assume that the two loci underlying the traits form and color are located on two different chromosomes and generate an F2 population:

```
> reset.all()
> map <- data.frame(chrom = 1:2,              # Two loci placed
+                   pos = c(0.0, 0.0),        # at the beginning of
+                   name = c("form","color")) # chromosome 1 and 2
> define.genome(map)
> init.population("P1",homozygote(1))
> init.population("P2",homozygote(2))
> cross("F1","P1","P2",10)
> cross("F2","F1","F1",1000)
> evaluate.genotype("F2", map$name)           # for all loci
```

	form-1	form-2	color-1	color-2	count	frequency
1	1	1	1	1	64	0.064
2	1	1	1	2	129	0.129
3	1	1	2	2	49	0.049
4	1	2	1	1	120	0.120
5	1	2	1	2	253	0.253
6	1	2	2	2	130	0.130
7	2	2	1	1	63	0.063
8	2	2	1	2	138	0.138
9	2	2	2	2	54	0.054

```
> evaluate.genotype("F2", map$name, mode=0)
```

	form-1	form-2	color-1	color-2	count	frequency
1	1	1	1	1	64	0.064
2	1	1	1	2	66	0.066
3	1	1	2	1	63	0.063
4	1	1	2	2	49	0.049
5	1	2	1	1	57	0.057
6	1	2	1	2	75	0.075
7	1	2	2	1	57	0.057
8	1	2	2	2	68	0.068
9	2	1	1	1	63	0.063
10	2	1	1	2	66	0.066
11	2	1	2	1	55	0.055
12	2	1	2	2	62	0.062
13	2	2	1	1	63	0.063
14	2	2	1	2	66	0.066
15	2	2	2	1	72	0.072
16	2	2	2	2	54	0.054

***Mendel's 3 law**: Genes are inherited independently.*

1.3 Backcrossing and recombination

Backrossing to the recessive parent is much easier to interpret:

```
> reset.all()
> map <- data.frame(chrom = 1:2,
+                   pos = c(0.0, 0.0),
+                   name = c("form","color"))
> define.genome(map)
> init.population("P1",homozygote(1))
> init.population("P2",homozygote(2))
> cross ("F1","P1","P2",563)
> cross ("BC1","F1","P2",1000)            # Backcrossing to parent 2
> evaluate.genotype("BC1", map$name)
```

In our simulated BC1 population, we observe the four phenotypes with approximately the same frequencies:

```
  form-1 form-2 color-1 color-2 count frequency
1      1      2       1       2   247     0.247
2      1      2       2       2   266     0.266
3      2      2       1       2   246     0.246
4      2      2       2       2   241     0.241
```

Now the two loci are on the same chromosome, the first at the telomere, and the second in a map distance of 0.1 M:

```
map <- data.frame(chrom = c(1,1),
        pos = c(0.0, 0.1),
        name = c("form","color"),
        class = c("trait1","trait2"))
define.genome(map)
...
```

```
> reset.all()
> map <- data.frame(chrom = c(1,1),
+                   pos = c(0.0, 0.1),
+                   name = c("form","color"))
> define.genome(map)
> init.population("P1",homozygote(1))
> init.population("P2",homozygote(2))
> cross ("F1","P1","P2",563)
> cross ("BC1","F1","P2",1000)
> evaluate.genotype("BC1", map$name)
```

```
  form-1 form-2 color-1 color-2 count frequency
1      1      2       1       2   477     0.477
2      1      2       2       2    52     0.052
3      2      2       1       2    45     0.045
4      2      2       2       2   426     0.426
```

Recombination frequencies *can be estimated in backcross populations by summing up the frequency of recombinant gametes.*

2 Random Mating Populations

2.1 F2 Population

We start with an F2 population of peas and look at the genotype frequencies and allele frequencies of the seed form.

```
> map <- data.frame(1,0.01,"form")
> define.genome(map)
> init.population("P1",homozygote(1))
> init.population("P2",homozygote(2))
> cross("F1","P1","P2",1)
> cross("F2","F1","F1",1000)
> evaluate.genotype ("F2","form")
```

```
  form-1 form-2 count frequency
1      1      1   238     0.238
```

```
2        1       2   509     0.509
3        2       2   253     0.253
```

```
> evaluate.allele.freq("F2","form")
```

```
   form count   freq
1     1   985 0.4925
2     2  1015 0.5075
```

We carry out random mating and investigate genotype and allele frequencies:

```
> cross("SYN1","F2","F2",1000)
> cross("SYN2","SYN1","SYN1",1000)
> cross("SYN3","SYN2","SYN2",1000)
> evaluate.genotype("SYN3","form")
```

```
   form-1  form-2  count frequency
1        1       1   237     0.237
2        1       2   509     0.509
3        2       2   254     0.254
```

```
> evaluate.allele.freq("SYN3","form")
```

```
   form count   freq
1     1   983 0.4915
2     2  1017 0.5085
```

Hardy-Weinberg principle:: *In random mating populations, allele and genotype frequencies are not changing.*

2.2 Population of homozygous lines

We start with a population of homozygous lines and look at the genotype and allele frequencies after one generation of random mating:

```
> remove.all.populations()
> init.population("P11",homozygote(1,500))
> init.population("P22",homozygote(2,500))
> append.population("ADM","P11")
> append.population("ADM","P22")
> evaluate.genotype ("ADM","form")
```

```
   form-1  form-2  count frequency
1        1       1   500       0.5
2        2       2   500       0.5
```

```
> evaluate.allele.freq2("ADM","form")
```

```
   form count freq
1     1  1000  0.5
2     2  1000  0.5
```

```
> cross("SYN1","ADM","ADM",1000)
> evaluate.genotype("SYN1","form")
```

```
   form-1  form-2  count frequency
1       1       1   268     0.268
2       1       2   467     0.467
3       2       2   265     0.265
```

```
> evaluate.allele.freq("SYN1","form")
```

```
   form count   freq
1     1  1003 0.5015
2     2   997 0.4985
```

***Hardy-Weinberg proportions**: $p^2 : 2pq : q^2$ are reached in one generation of random mating, irrespective of the gentype frequencies in the base population.*

2.3 Arbitrary allele frequencies

```
> remove.all.populations()
> init.population("P11",homozygote(1,100))
> init.population("P22",homozygote(2,900))
> append.population("ADM","P11")
> append.population("ADM","P22")
> evaluate.genotype ("ADM","form")
```

```
   form-1  form-2  count frequency
1       1       1   100       0.1
2       2       2   900       0.9
```

```
> evaluate.allele.freq2("ADM","form")
```

```
   form count freq
1     1   200  0.1
2     2  1800  0.9
```

```
> cross("SYN1","ADM","ADM",1000)
> evaluate.genotype("SYN1","form")
```

```
   form-1  form-2  count frequency
1       1       1    10     0.010
2       1       2   179     0.179
3       2       2   811     0.811
```

```
> evaluate.allele.freq("SYN1","form")
```

```
   form count   freq
1     1   199 0.0995
2     2  1801 0.9005
```

```
> cross("SYN1","ADM","ADM",1000)
> cross("SYN2","SYN1","SYN1",1000)
> cross("SYN3","SYN2","SYN2",1000)
> evaluate.genotype("SYN3","form")
```

```
   form-1  form-2  count frequency
1       1       1    14     0.014
2       1       2   187     0.187
3       2       2   799     0.799
```

```
> evaluate.allele.freq("SYN3","form")
```

```
   form count   freq
1     1   215 0.1075
2     2  1785 0.8925
```

Hardy-Weinberg law *is valid for arbitrary allele frequencies p and q.*

3 Populations under selection

3.1 Selection against a recessive allele

We start with a population of peas in HWE in which the recessive allele ‘wrinkeld’, which is coded as allele ‘2’ has the allele frequency $q \approx 0.1$

```
> reset.all()
> map <- data.frame(1,0.0,"form")
> define.genome(map)
> init.population("P11",homozygote(1,900))
> init.population("P22",homozygote(2,100))
> append.population("ADM","P11")
> append.population("ADM","P22")
> cross("SYN1","ADM","ADM",1000)
> evaluate.genotype("SYN1","form")
```

```
   form-1  form-2  count frequency
1       1       1   799     0.799
2       1       2   195     0.195
3       2       2     6     0.006
```

```
> evaluate.allele.freq2("SYN1","form")
```

```
   form count   freq
1     1  1793 0.8965
2     2   207 0.1035
```

We now want to select all round seeds. This means, we select against the recessive allele. For simulating selection, effects are required. An effect assigns values to alleles, for this assignment the class names of the loci are used. Effects can be defined explicitly

```
define.effects("effect1", "0, form 1 uniform 1")
```

If an individual has at a locus that belongs to the class `form` the allele 1 then it gets a uniform value of 1 added to its genotypic value. This effect definition is stored under name `effect1`. Often effects are defined for quantitative characters, there a population mean is needed. In our example we don't need a population mean, this is the first 0 in the effect definition.

With this effect definition, a genotypic value for each individual of the population can be calculated. Those individuals that carry two times the allele 1 at the locus form, get a 2, those having one copy of the allele get a 1, and those that don't carry the allele 1 get a zero. We generate a new population called `allrounds` that consists of individuals selected from `SYN1`. Defining an effect is in this case not necessary, as the function `select.genotypes` creates an effect for selection if none is defined prior.

We are interested in all genotypes that have the round phenotype. This is the case if the genotype is carrying allele 1 at least once. We pass the trait and the allele combinations that we want to select as a data.frame to the function. Here, 1|1 and 1|2 can result in round seeds. Effects are estimated automatically and then selection is carried out. The selected genotypes end up in a new population called "allrounds".

```
> sel.crit.form <- data.frame(
+   locus    =  c("form", "form") ,
+   allele1  = c(1,1),
+   allele2  = c(1,2)
+ )
> sel.crit.form
```

```
  locus allele1 allele2
1  form       1       1
2  form       1       2
```

```
> select.genotypes("allrounds","SYN1",
+                  sel.crit.form)
```

We look at the genotype of the selected plants:

```
> evaluate.genotype("allrounds","form")
```

```
   form-1  form-2  count frequency
1       1       1   799   0.80382
2       1       2   195   0.19618
```

We now grow the seeds and random mate the plants.

```
> cross ("SYNA","allrounds","allrounds",1000)
> evaluate.genotype ("SYNA","form")
```

```
   form-1  form-2  count frequency
1       1       1   833     0.833
2       1       2   151     0.151
3       2       2    16     0.016
```

```
> evaluate.allele.freq("SYNA","form")
```

```
   form count   freq
1     1  1817 0.9085
2     2   183 0.0915
```

The selection against wrinkled seeds was not working well! However, it is difficult to draw conclusions just from one simulation. We repeat our simulation 100 times:

```
> reset.all()
> map <- data.frame(1,0.0,"form")
> define.genome(map)
> for (i in 1:100){
+   init.population("P11",homozygote(1,900))
+   init.population("P22",homozygote(2,100))
+   append.population("ADM","P11")
+   append.population("ADM","P22")
+   cross("SYN1","ADM","ADM",1000)
+   select.genotypes("allrounds","SYN1",
+                    sel.crit.form)
+   cross ("SYNA","allrounds","allrounds",1000)
+   append.population("STORE","SYNA")
+ }
>
```

Now the numbers are easier to interpret:

```
> evaluate.genotype("STORE","form")
```

```
   form-1  form-2  count frequency
1       1       1 82658   0.82658
2       1       2 16539   0.16539
3       2       2   803   0.00803
```

```
> evaluate.allele.freq("STORE","form")
```

```
   form  count     freq
1     1 181855 0.909275
2     2  18145 0.090725
```

Selection against recessive alleles: *Even if strong selection is carried out against those individuals of a population, that show the phenotype of a recessive allele, this phenotype will reoccur in subsequent generations.*

3.2 Selection for fitness

A human population lives in a Malaria region. Assume the allele frequency of the sickle cell anemia gene is $f(S) \approx 0.1$.

```
> reset.all()
> map <- data.frame(1,0.0,"SZA","trait1")
> define.genome(map)
> init.population("P11",homozygote(1,90))
> init.population("P22",homozygote(2,10))
> append.population("ADM","P11")
> append.population("ADM","P22")
> cross("SYN1","ADM","ADM",1000)
> evaluate.genotype("SYN1","SZA")
```

```
  SZA-1 SZA-2 count frequency
1     1     1   815     0.815
2     1     2   180     0.180
3     2     2     5     0.005
```

```
> evaluate.allele.freq("SYN1","SZA")
```

```
  SZA count  freq
1   1  1810 0.905
2   2   190 0.095
```

We first split the population in three subpopulations, corresponding to the genotype. As selected genotypes are removed from the populations, we first remove both homozygous genotypes. What remains in the original dataset are the heterozygous genotypes.

```
> select.genotypes("R2","SYN1",data.frame("SZA",1,1))
> evaluate.genotype("R2","SZA")
```

```
  SZA-1 SZA-2 count frequency
1     1     1   815         1
```

```
> select.genotypes("R0","SYN1",data.frame("SZA",2,2))
> evaluate.genotype("R0","SZA")
```

```
  SZA-1 SZA-2 count frequency
1     2     2     5         1
```

```
> rename.population ("SYN1","R1")
> evaluate.genotype("R0","SZA")
```

```
  SZA-1 SZA-2 count frequency
1     2     2     5         1
```

Under strong malaria pressure the probability that an individual contributes to the next generation are

```
> w2 <- 0.5 # Homozygous normal, allele 1
> w1 <- 1.0 # Heterozygous
> w0 <- 0.2 # Homozygous sickle cell anemia, allele 2
```

Using random numbers for the binomial distribution generate the fraction of the population that contributes to the next generation

```
> n.inds <- population.list()
> ( nR2 <- n.inds$count[n.inds$PopName=="R2"])
```

```
[1] 815
```

```
> ( nR1 <- n.inds$count[n.inds$PopName=="R1"])
```

```
[1] 180
```

```
> ( nR0 <- n.inds$count[n.inds$PopName=="R0"])
```

```
[1] 5
```

```
> ( sR2 <- rbinom(n=1,size=nR2,prob=w2) )
```

```
[1] 415
```

```
> ( sR1 <- rbinom(n=1,size=nR1,prob=w1) )
```

```
[1] 180
```

```
> ( sR0 <- rbinom(n=1,size=nR0,prob=w0) )
```

```
[1] 1
```

```
> sample.population("S2","R2",sR2)
> sample.population("S1","R1",sR1)
> sample.population("S0","R0",sR0)
> remove.population("Sx")
> append.population("Sx","S2")
> append.population("Sx","S1")
> append.population("Sx","S0")
> evaluate.genotype ("Sx","SZA")
```

```
   SZA-1  SZA-2  count frequency
1      1      1   415 0.6963087
2      1      2   180 0.3020134
3      2      2     1 0.0016779
```

```
> evaluate.allele.freq("Sx","SZA")
```

```
   SZA count    freq
1    1  1010 0.84732
2    2   182 0.15268
```

The next generation:

```
> cross ("SYNA","Sx","Sx",1000)
> evaluate.genotype ("SYNA","SZA")
```

```
   SZA-1  SZA-2  count frequency
1      1      1   736     0.736
2      1      2   245     0.245
3      2      2    19     0.019
```

```
> evaluate.allele.freq2("SYNA","SZA")
```

```
   SZA count   freq
1    1  1717 0.8585
2    2   283 0.1415
```

4 Appendix: multiple traits

We have a large population of peas that show all different kinds of phenotypes. These peas can be round or wrinkled (`form`) and they can either be green or yellow (`col`). Additionally, their flowers can either be white, red or pink (when heterozygous) (`flow`).

We create a population with three loci on three chromosomes:

```
> reset.all()
> map <- data.frame(chrom = c(1,2,3),
+                   pos = c(0.01,0.05,0.1),
+                   name = c("form","col","flow"))
> define.genome(map)
```

Assume for `form` that round is dominant and for colour (`col`), green is dominant. We want to select: wrinkled, green peas with pink flowers.

```
> sel.crit.wrinkeled <- data.frame(
+   locus    = "form" ,
+   allele1  = 2       ,
+   allele2  = 2
+ )
> sel.crit.green <- data.frame(
```

```
+   locus    = c ("col" , "col"),
+   allele1  = c (1       ,  1     ),
+   allele2  = c( 1       ,  2     )
+ )
> sel.crit.pink <- data.frame(
+   locus    = "flow" ,
+   allele1  = 1       ,
+   allele2  = 2
+ )
> criteria <- rbind (sel.crit.green, sel.crit.wrinkeled, sel.crit.pink)
```

Some cycles of random mating:

```
> init.population("P1",homozygote(1))
> init.population("P2",homozygote(2))
> cross("F1","P1","P2",1)
> cross("F2","F1","F1",1000)
> cross("F3","F2","F2",1000)
```

And now our selection:

```
> select.genotypes("Sel","F3", criteria)
```

```
> evaluate.genotype("F3", map$name)
```

	form-1	form-2	col-1	col-2	flow-1	flow-2	count	frequency
1	1	1	1	1	1	1	18	0.019715
2	1	1	1	1	1	2	30	0.032859
3	1	1	1	1	2	2	19	0.020811
4	1	1	1	2	1	1	35	0.038335
5	1	1	1	2	1	2	59	0.064622
6	1	1	1	2	2	2	34	0.037240
7	1	1	2	2	1	1	10	0.010953
8	1	1	2	2	1	2	42	0.046002
9	1	1	2	2	2	2	18	0.019715
10	1	2	1	1	1	1	22	0.024096
11	1	2	1	1	1	2	61	0.066813
12	1	2	1	1	2	2	28	0.030668
13	1	2	1	2	1	1	67	0.073384
14	1	2	1	2	1	2	139	0.152245
15	1	2	1	2	2	2	64	0.070099
16	1	2	2	2	1	1	29	0.031763
17	1	2	2	2	1	2	66	0.072289
18	1	2	2	2	2	2	29	0.031763
19	2	2	1	1	1	1	17	0.018620
20	2	2	1	1	2	2	11	0.012048
21	2	2	1	2	1	1	35	0.038335
22	2	2	1	2	2	2	32	0.035049
23	2	2	2	2	1	1	12	0.013143
24	2	2	2	2	1	2	29	0.031763
25	2	2	2	2	2	2	7	0.007667

```
> evaluate.genotype("Sel", map$name)
```

	form-1	form-2	col-1	col-2	flow-1	flow-2	count	frequency
1	2	2	1	1	1	2	28	0.32184
2	2	2	1	2	1	2	59	0.67816

Allele frequencies for multiple loci using a vector can be done using `evaluate.allele.freq2`.

```
> evaluate.allele.freq2("Sel", c("form", "col", "flow"))
```

```
   form col flow count    freq
1     0   0    1    87 0.16667
2     0   0    2    87 0.16667
3     0   1    0   115 0.22031
4     0   2    0    59 0.11303
5     2   0    0   174 0.33333
```