The Logistic Genetic Project

Ivan Scotti, ivan.scotti@inrae.fr; François Lefèvre, francois.lefevre.2@inrae.fr

2023-07-25

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

The Evolutionary Resilience Project

This code is the workhorse of the Evolutionary Resilience Project, an open-ended research programme aiming at the study of the **population-genetic bases of resilience**.

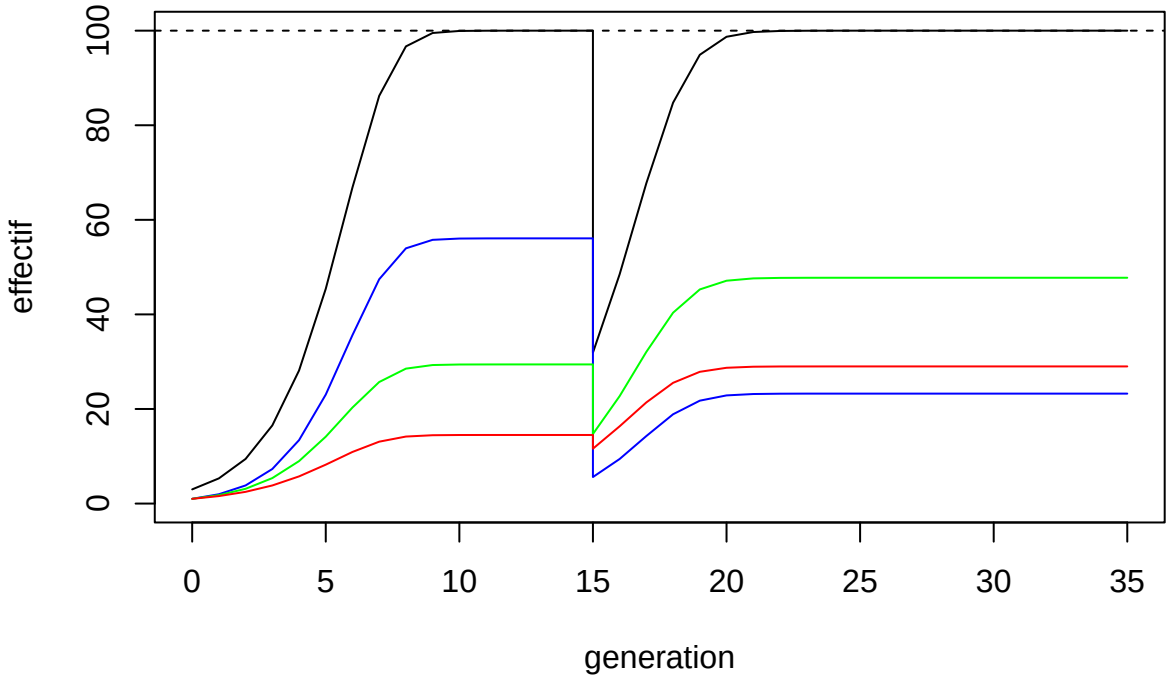
The sections of the document follow the development of the simulation mechanism, from the simplest model to increasingly complex ones. Each model is comprised within one chunk code, usually followed by plotting chunks.

So, when you scroll down you follow, somehow, the history of model development, and the document is both a code source and a log-book.

The following chunk contains the original script from *multilogistic.R*

```
#setwd("D:/homeD/flefevre/TEXTE/equipe/BioPopEvol/resilience_Ivan")

# matrice des effectifs, initialisation
N<-data.frame("generation"=0, "N1"=1, "N2"=1, "N3"=1, "Ntot"=3)

# vecteur des parametres (ri: repro; si: survie)
P<-c("K"=100, "r1"=1, "r2"=0.8, "r3"=0.6, "s1"=0.1, "s2"=0.5, "s3"=0.8)

# date de la perturbation
t0<-15

# debut incrementation
# pre-perturbation
for(t in 1:t0) {
  Nt<-N[dim(N)[1],2:4]*(1+(P[2:4]*(P[1]-N[dim(N)[1],5])/P[1]))
  N<-rbind(N,NA)
  N[dim(N)[1],]<-c(t,Nt,sum(Nt))
}
# perturbation
Nt0<-N[dim(N)[1],2:4]*P[5:7]
```

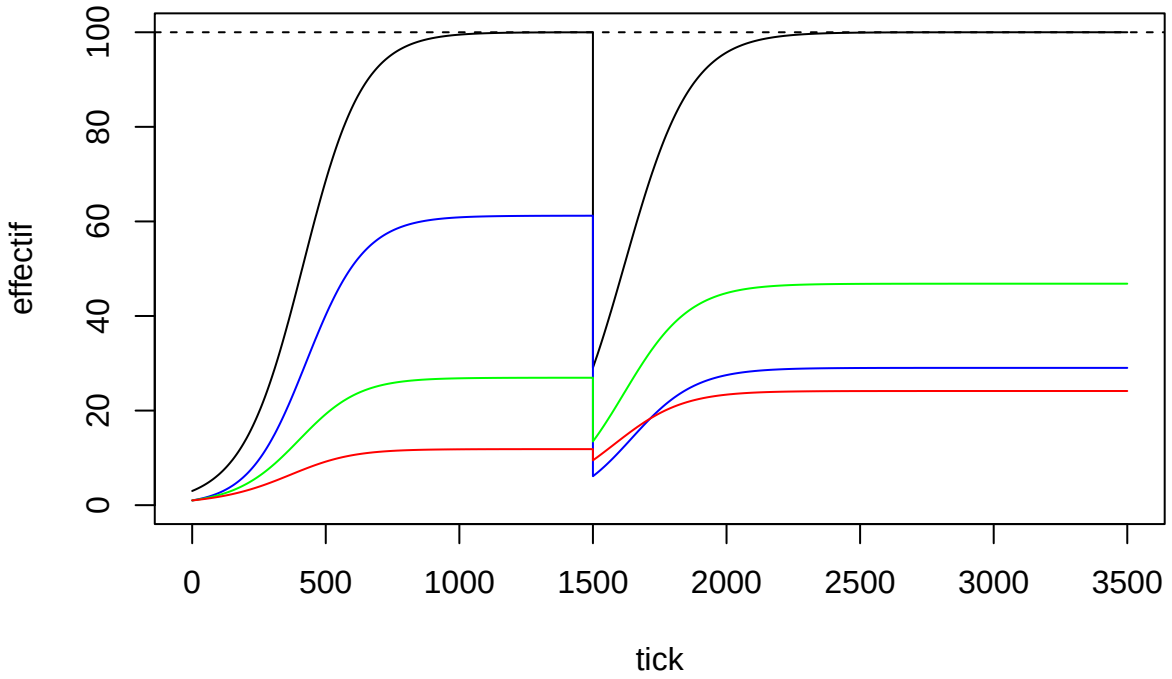
```
N<-rbind(N,NA)
N[dim(N)[1],]<-c(t0,Nt0,sum(Nt0))
# post-perturbation
for(t in (t0+1):(t0+20)) {
  Nt<-N[dim(N)[1],2:4]*(1+(P[2:4]*(P[1]-N[dim(N)[1],5])/P[1]))
  N<-rbind(N,NA)
  N[dim(N)[1],]<-c(t,Nt,sum(Nt))
}
# fin incrementation


# plot
plot(Ntot~generation,data=N,type="l", ylab="effectif", ylim=c(0,P[1]))
lines(N$generation,N$N1,col="blue")
lines(N$generation,N$N2,col="green")
lines(N$generation,N$N3,col="red")
abline(h=P[1],lty=2)
```



Next chunk: same as above, but with a new arbitrary "**tick**" (NB the word "tick" indicates an arbitrary step in time; the term is used for analogy with SLiM), which is a fraction of the "generation" above (say, **1/100 of a generation**). The mortality and reproduction rates are divided accordingly, and the result should be numerically the same, but over a longer "time frame":

```
# matrice des effectifs, initialisation
N<-data.frame("tick"=0, "N1"=1, "N2"=1, "N3"=1, "Ntot"=3)              # generation changed to tick

# vecteur des parametres (ri: repro; si: survie)
P<-c("K"=100, "r1"=0.01, "r2"=0.008, "r3"=0.006,                       # growth rates can be divided b
     "s1"=0.1, "s2"=0.5, "s3"=0.8)                                     # (see development in notebook)
# date de la perturbation
t0<-1500
```

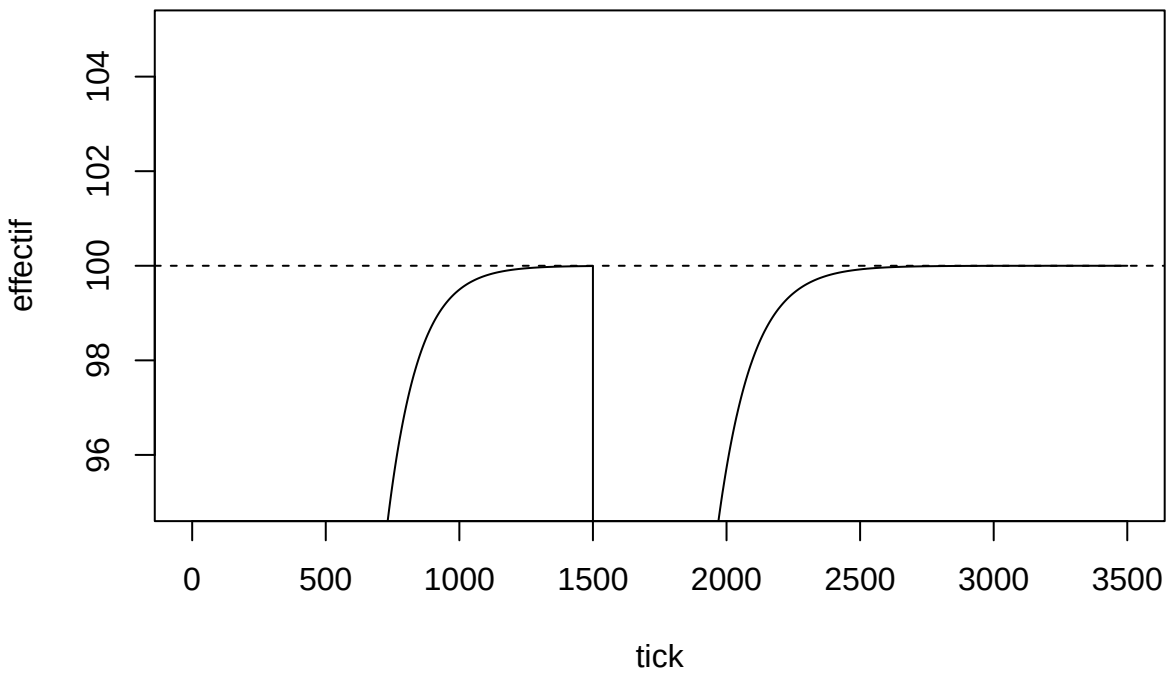
```r
# debut incrementation
# pre-perturbation
for(t in 1:t0) {
  Nt<-N[dim(N)[1],2:4]*(1+(P[2:4]*(P[1]-N[dim(N)[1],5])/P[1]))
  N<-rbind(N,NA)
  N[dim(N)[1],]<-c(t,Nt,sum(Nt))
}
# perturbation
Nt0<-N[dim(N)[1],2:4]*P[5:7]
N<-rbind(N,NA)
N[dim(N)[1],]<-c(t0,Nt0,sum(Nt0))
# post-perturbation
for(t in (t0+1):(t0+2000)) {
  Nt<-N[dim(N)[1],2:4]*(1+(P[2:4]*(P[1]-N[dim(N)[1],5])/P[1]))
  N<-rbind(N,NA)
  N[dim(N)[1],]<-c(t,Nt,sum(Nt))
}
# fin incrementation


# plot
plot(Ntot~tick,data=N,type="l", ylab="effectif", ylim=c(0,P[1]))
lines(N$tick,N$N1,col="blue")
lines(N$tick,N$N2,col="green")
lines(N$tick,N$N3,col="red")
abline(h=P[1],lty=2)
```



All right, this is fine, we confirm that moving from generations to *ticks* is numerically similar.

We might see, though, the population oscilalte around K. Do we?

```r
# plot
plot(Ntot~tick,data=N,type="l", ylab="effectif", ylim = c(95, 105))
lines(N$tick,N$N1,col="blue")
lines(N$tick,N$N2,col="green")
lines(N$tick,N$N3,col="red")
abline(h=P[1],lty=2)
```

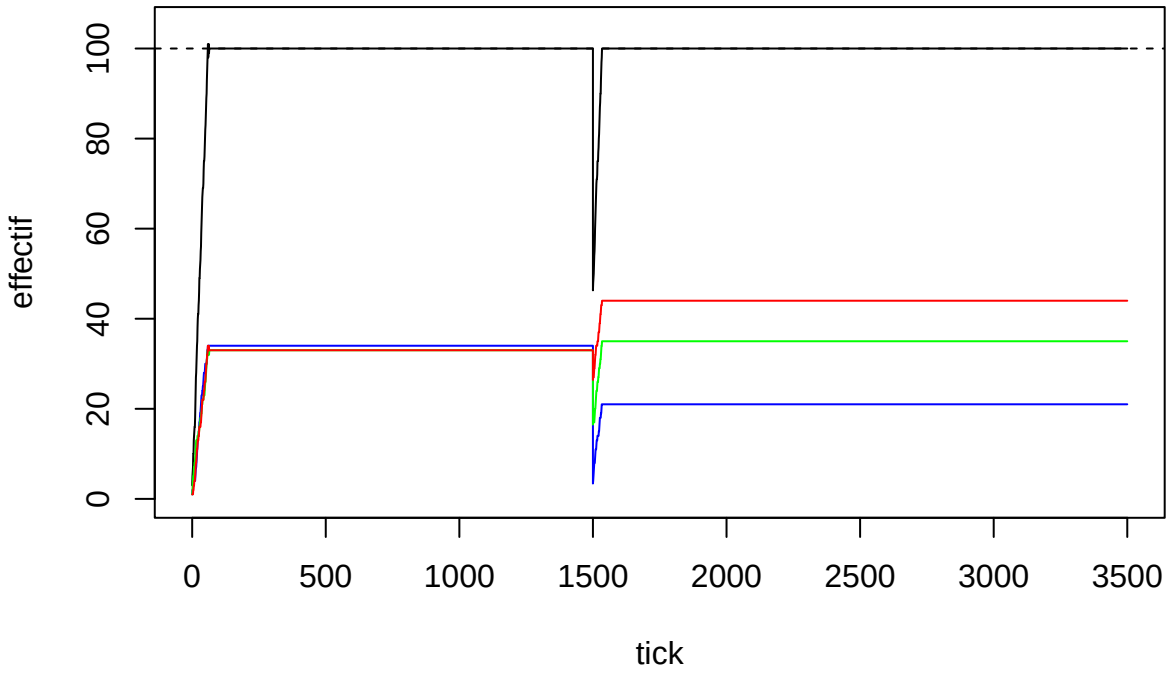


No, we do not because there is no approximation to integer numbers.

Let us try to see whether we obtain oscillations (which would include *negative* growth rates for N around K) if we introduce rounding. Notice that to do this, some stochasticity must be introduced, or else the growth process will be impossible due to rounding that prevents all change.

```r
# matrice des effectifs, initialisation
N<-data.frame("tick"=0, "N1"=1, "N2"=1, "N3"=1, "Ntot"=3)              # generation changed to tick

# vecteur des parametres (ri: repro; si: survie)
P<-c("K"=100, "r1"=0.01, "r2"=0.008, "r3"=0.006,                       # growth rates can be divided b
     "s1"=0.1, "s2"=0.5, "s3"=0.8)                                     # (see development in notebook)
# date de la perturbation
t0<-1500

# debut incrementation
# pre-perturbation
for(t in 1:t0) {
  Nt<-c(
        do.call(what = sample(x = c("ceiling","floor"), size = 1),                  # numbers are r
                args = list(N[dim(N)[1],2]*(1+(P[2]*(P[1]-N[dim(N)[1],5])/P[1])))),     # for individua
        do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(N[dim(N)[1],3]*(1+(P[3]*(P[1]-N[dim(N)[1],5])/P[1])))),
        do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(N[dim(N)[1],4]*(1+(P[4]*(P[1]-N[dim(N)[1],5])/P[1]))))
```

```
      )
  N<-rbind(N,NA)
  N[dim(N)[1],]<-c(t,Nt,sum(Nt))
}
# perturbation
Nt0<-N[dim(N)[1],2:4]*P[5:7]
N<-rbind(N,NA)
N[dim(N)[1],]<-c(t0,Nt0,sum(Nt0))
# post-perturbation
for(t in (t0+1):(t0+2000)) {
  Nt<-c(
        do.call(what = sample(x = c("ceiling","floor"), size = 1),                    # numbers are r
                args = list(N[dim(N)[1],2]*(1+(P[2]*(P[1]-N[dim(N)[1],5])/P[1])))),   # for individua
        do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(N[dim(N)[1],3]*(1+(P[3]*(P[1]-N[dim(N)[1],5])/P[1])))),
        do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(N[dim(N)[1],4]*(1+(P[4]*(P[1]-N[dim(N)[1],5])/P[1]))))
      )
  N<-rbind(N,NA)
  N[dim(N)[1],]<-c(t,Nt,sum(Nt))
}
# fin incrementation


# plot
plot(Ntot~tick,data=N,type="l", ylab="effectif", ylim=c(0,1.05*P[1]))
lines(N$tick,N$N1,col="blue")
lines(N$tick,N$N2,col="green")
lines(N$tick,N$N3,col="red")
abline(h=P[1],lty=2)
```



This is *basically* working, even though, as the population reaches K, there is no way of having any further

change, as before.

What we have to do now, to conform to population-genetic expectations, is to introduce the survival constraint for every tick, and then let the population grow again over the following tick.

Introducing mortality

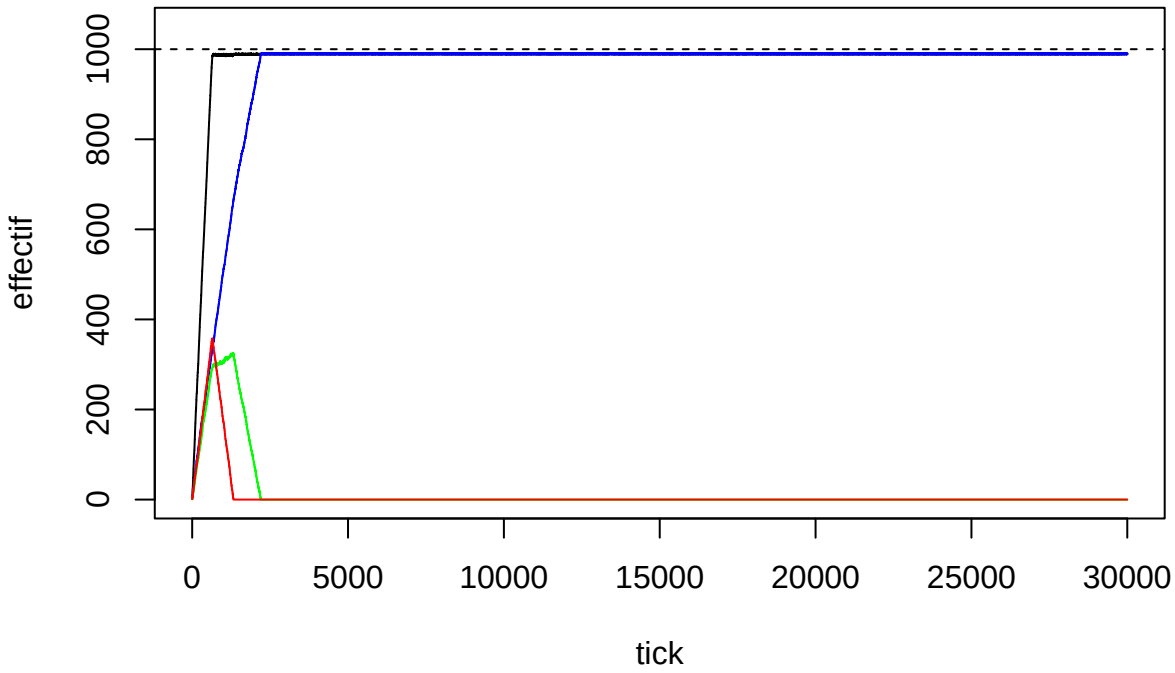
A first crude attempt: mortaility soon after growth

For the first attempt, few changes are needed:

- increasing K too obtain a higher resolution
- setting aside the disturbance (we focus on what happens at "equilibrium")
- running the process for a longer number of ticks (to follow the equilibrium simulation over a longer time)

Let us set a flat mortality rate at 0.0001 of the number for each genotype

```
survival <- 0.9999

N<-data.frame("tick"=0, "N1"=1, "N2"=1, "N3"=1, "Ntot"=3)

# vecteur des parametres (ri: repro; si: survie)
P<-c("K"=1000, "r1"=0.0100, "r2"=0.0080, "r3"=0.0060,                    # new K = 1000
     "s1"=0.1, "s2"=0.5, "s3"=0.8)                                       # (sX not used but kept for t
# date de la perturbation
t0<-30000

# debut incrementation
# pre-perturbation
for(t in 1:t0) {
  Nt<-c(
        do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(survival * N[dim(N)[1],2]*(1+(P[2]*(P[1]-N[dim(N)[1],5])/P[1])))),     # su
        do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(survival * N[dim(N)[1],3]*(1+(P[3]*(P[1]-N[dim(N)[1],5])/P[1])))),     # su
        do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(survival * N[dim(N)[1],4]*(1+(P[4]*(P[1]-N[dim(N)[1],5])/P[1]))))      # su
       )
  N<-rbind(N,NA)
  N[dim(N)[1],]<-c(t,Nt,sum(Nt))
}
# plot
plot(Ntot~tick,data=N,type="l", ylab="effectif", ylim=c(0,1.05*P[1]))
lines(N$tick,N$N1,col="blue")
lines(N$tick,N$N2,col="green")
lines(N$tick,N$N3,col="red")
abline(h=P[1],lty=2)
```



Cool! The population-genetic expectations are back. The genotype with the highest fitness gets fixed (forms of equilibrium can also be generated, though, by fiddling with the parameters, which is also quite good).

Now, the cool thing will be to introduce some form of mating...

A slightly more refined iteration: introducing random mating

This should be easy, **assuming that genotypes 1, 2 and 3 are respectively AA, Aa, and aa**. To obtain this, instead of assigning all "progeny" of a genotype to itself, we can use standard Hardy-Weinberg equilibrium (HWE) formulae for the proportion of offspring of each genotype. *This is going to be super-cool!*

So, the increase of the "population" of a genotype (its fecundity, or the number of offspring it produces) is not arking back to the same genotype, but it is attributed to each genotype according to H-W proportions.

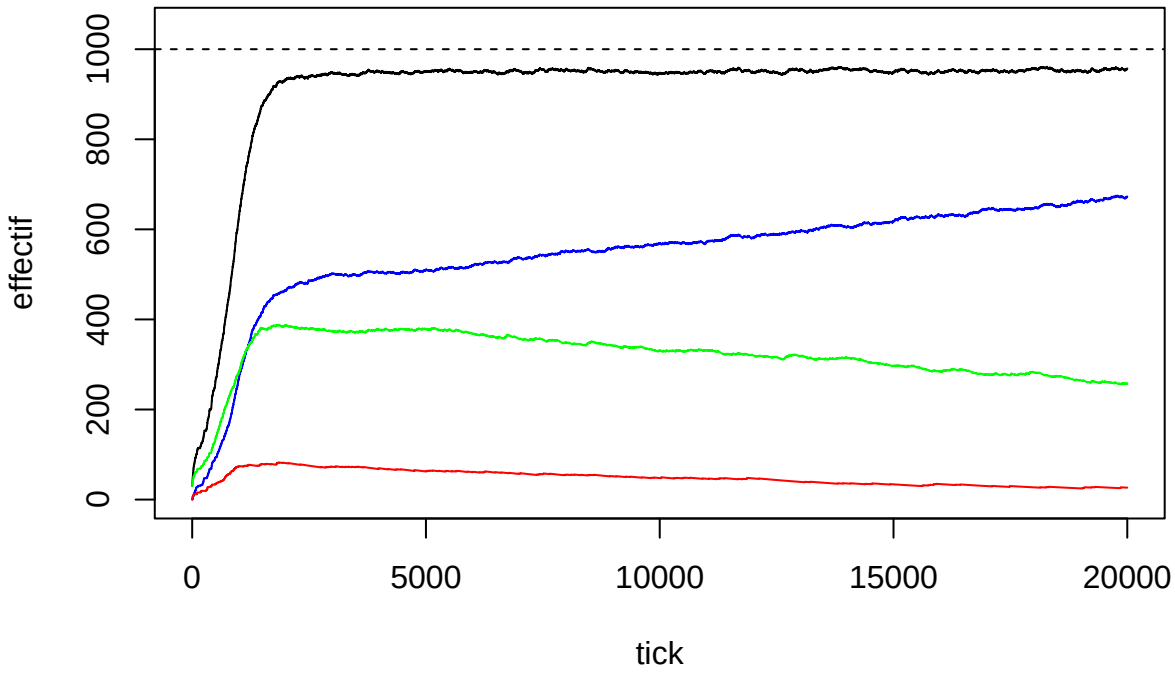
This requires tweaking the code a little further, but it should not be super-complicated.

The process should have three steps:

1. computing the deltaN for each genotype
2. assigning the genotypes to the deltaN individuals based on HWE
3. computing the final N for each genotype
4. applying mortality (the separation of mortality from other steps will be useful for subsequent, more refined iterations)

```
survival <- 0.9998

N<-data.frame("tick"=0, "N1"=0, "N2"=30, "N3"=0, "Ntot"=30)

# vecteur des parametres (ri: repro; si: survie)
P<-c("K"=1000, "r1"=0.0100, "r2"=0.0060, "r3"=0.0020,                    # new K = 1000
     "s1"=0.1, "s2"=0.5, "s3"=0.8)                                       # (sX not used but kept for t
# date de la perturbation
t0<-20000
```

```
# debut incrementation
# pre-perturbation
for(t in 1:t0) {
# computing increments for each genotype (= their "progeny")
deltaN1 <- do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(N[dim(N)[1],2]*(1+(P[2]*(P[1]-N[dim(N)[1],5])/P[1])))) - N[dim(N)[1],2]
deltaN2 <- do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(N[dim(N)[1],3]*(1+(P[3]*(P[1]-N[dim(N)[1],5])/P[1])))) - N[dim(N)[1],3]
deltaN3 <- do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(N[dim(N)[1],4]*(1+(P[4]*(P[1]-N[dim(N)[1],5])/P[1])))) - N[dim(N)[1],4]
# attributing progeny to each genotype (through a random process, based on H-W expectations: usage of a
progenyFrom1 <- rmultinom(n = 1, size = deltaN1, prob = c(N[dim(N)[1],2] + 0.5*N[dim(N)[1],3],    # con
                                                          0.5*N[dim(N)[1],3] + N[dim(N)[1],4],    # gen
                                                          0))                                     # fro
progenyFrom2 <- rmultinom(n = 1, size = deltaN2, prob = c(0.5*N[dim(N)[1],2] + 0.25*N[dim(N)[1],3],
                                                          0.5*N[dim(N)[1],2] + 0.5*N[dim(N)[1],3] + 0.5
                                                          0.25*N[dim(N)[1],3] + 0.5*N[dim(N)[1],4]))
progenyFrom3 <- rmultinom(n = 1, size = deltaN3, prob = c(0,
                                                          N[dim(N)[1],2] + 0.5*N[dim(N)[1],3],
                                                          0.5*N[dim(N)[1],3] + N[dim(N)[1],4]))
progenyTot <- cbind(progenyFrom1, progenyFrom2, progenyFrom3)
N1postRepro <- N[dim(N)[1],2] + rowSums(progenyTot)[1]
N2postRepro <- N[dim(N)[1],3] + rowSums(progenyTot)[2]
N3postRepro <- N[dim(N)[1],4] + rowSums(progenyTot)[3]
  Nt<-c(
        survival * N1postRepro,
        survival * N2postRepro,
        survival * N3postRepro
       )
  N<-rbind(N,NA)
  N[dim(N)[1],]<-c(t,Nt,sum(Nt))
}
# plot
plot(Ntot~tick,data=N,type="l", ylab="effectif", ylim=c(0,1.05*P[1]))
lines(N$tick,N$N1,col="blue")
lines(N$tick,N$N2,col="green")
lines(N$tick,N$N3,col="red")
abline(h=P[1],lty=2)
```



The addition of random mating confers realism to the whole thing and allows to fulfill all the fundamental population genetic expectations.

Back to resilience: including the effect of a disturbance

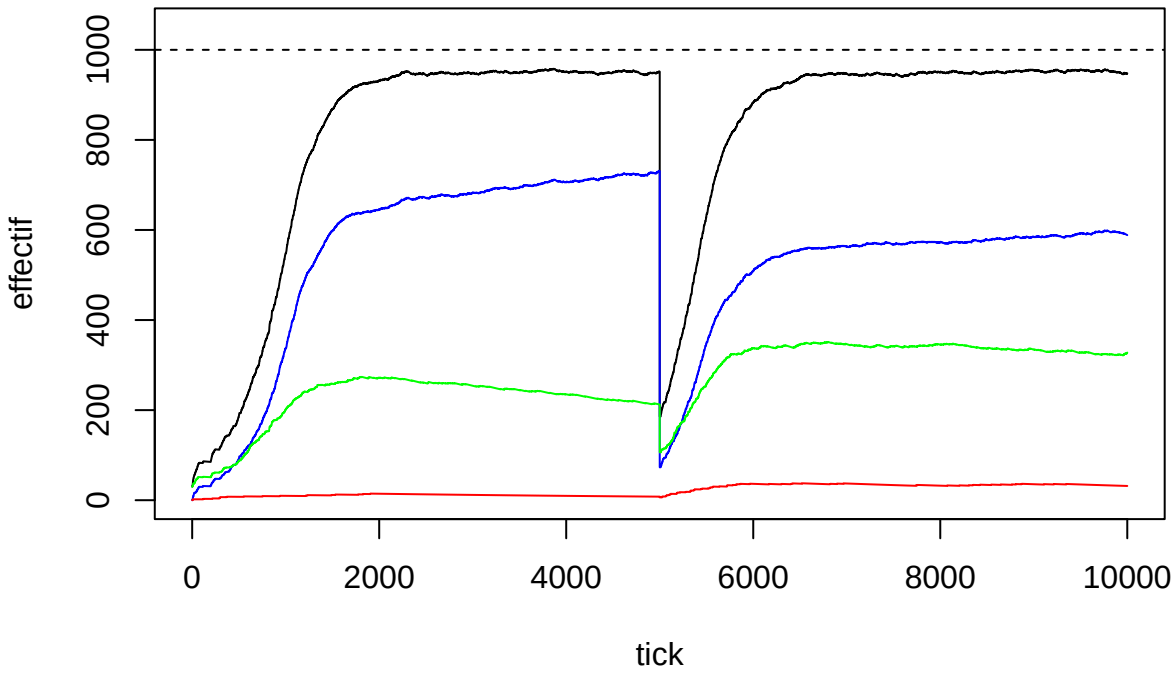
The disturbance and post-disturbance kynetics are re-instated, using the new algorithm including mortality and random mating. Further variables added:

- survival is subdivided into three values, one for each genotype
- survivalPost is the survival rate *after disturbance*, also with one value for each genotype
- Kpost is the carrying capacity after disturbance (corresponding to **P**[**8**])
- r1post, r2post, r3post are the reproductive rates after disturbance (stored in **P**[**9:11**])
- tEnd is the end of the simulation

```
survival <- c("surv1" = 0.9998, "surv2" = 0.9998, "surv3" = 0.9998)          # allowing for variable s
                                                                             # genotype
survivalPost <- c("surv1post" = 0.9998,                                      # allowing for variable s
                  "surv2post" = 0.9998, "surv3post" = 0.9998)                # genotype after disturba
N<-data.frame("tick"=0, "N1"=0, "N2"=30, "N3"=0, "Ntot"=30)

# vecteur des parametres (ri: repro; si: survie)
P<-c("K"=1000, "r1"=0.0100, "r2"=0.0020, "r3"=0.0020,                        # new K = 1000
     "s1"=0.1, "s2"=0.5, "s3"=0.8, "Kpost"=1000,                             # Kpost = carrying capaci
     "r1post"=0.0100, "r2post"=0.0060,"r3post"=0.0020)
# date de la perturbation
t0<-5000
# end of simulation
tEnd <- 10000
# debut incrementation
# pre-perturbation
for(t in 1:t0) {
```

```
# computing increments for each genotype (= their "progeny")
deltaN1 <- do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(N[dim(N)[1],2]*(1+(P[2]*(P[1]-N[dim(N)[1],5])/P[1])))) - N[dim(N)[1],2]
deltaN2 <- do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(N[dim(N)[1],3]*(1+(P[3]*(P[1]-N[dim(N)[1],5])/P[1])))) - N[dim(N)[1],3]
deltaN3 <- do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(N[dim(N)[1],4]*(1+(P[4]*(P[1]-N[dim(N)[1],5])/P[1])))) - N[dim(N)[1],4]
# attributing progeny to each genotype (through a random process, based on H-W expectations: usage of a
progenyFrom1 <- rmultinom(n = 1, size = deltaN1, prob = c(N[dim(N)[1],2] + 0.5*N[dim(N)[1],3],    # con
                                                          0.5*N[dim(N)[1],3] + N[dim(N)[1],4],    # gen
                                                          0))                                     # fro
progenyFrom2 <- rmultinom(n = 1, size = deltaN2, prob = c(0.5*N[dim(N)[1],2] + 0.25*N[dim(N)[1],3],
                                                          0.5*N[dim(N)[1],2] + 0.5*N[dim(N)[1],3] + 0.5
                                                          0.25*N[dim(N)[1],3] + 0.5*N[dim(N)[1],4]))
progenyFrom3 <- rmultinom(n = 1, size = deltaN3, prob = c(0,
                                                          N[dim(N)[1],2] + 0.5*N[dim(N)[1],3],
                                                          0.5*N[dim(N)[1],3] + N[dim(N)[1],4]))
progenyTot <- cbind(progenyFrom1, progenyFrom2, progenyFrom3)
N1postRepro <- N[dim(N)[1],2] + rowSums(progenyTot)[1]
N2postRepro <- N[dim(N)[1],3] + rowSums(progenyTot)[2]
N3postRepro <- N[dim(N)[1],4] + rowSums(progenyTot)[3]
  Nt<-c(
        survival[1] * N1postRepro,
        survival[2] * N2postRepro,
        survival[3] * N3postRepro
       )
  N<-rbind(N,NA)
  N[dim(N)[1],]<-c(t,Nt,sum(Nt))
}
#
# disturbance step
Nt0<-N[dim(N)[1],2:4]*P[5:7]
N<-rbind(N,NA)
N[dim(N)[1],]<-c(t0,Nt0,sum(Nt0))
#
# post-disturbance phase (same code as pre-disturbance)
for(t in (t0+1):tEnd) {
  # computing increments for each genotype (= their "progeny")
deltaN1 <- do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(N[dim(N)[1],2]*(1+(P[9]*(P[8]-N[dim(N)[1],5])/P[1])))) - N[dim(N)[1],2]
deltaN2 <- do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(N[dim(N)[1],3]*(1+(P[10]*(P[8]-N[dim(N)[1],5])/P[1])))) - N[dim(N)[1],3]
deltaN3 <- do.call(what = sample(x = c("ceiling","floor"), size = 1),
                args = list(N[dim(N)[1],4]*(1+(P[11]*(P[8]-N[dim(N)[1],5])/P[1])))) - N[dim(N)[1],4]
# attributing progeny to each genotype (through a random process, based on H-W expectations: usage of a
progenyFrom1 <- rmultinom(n = 1, size = deltaN1, prob = c(N[dim(N)[1],2] + 0.5*N[dim(N)[1],3],    # con
                                                          0.5*N[dim(N)[1],3] + N[dim(N)[1],4],    # gen
                                                          0))                                     # fro
progenyFrom2 <- rmultinom(n = 1, size = deltaN2, prob = c(0.5*N[dim(N)[1],2] + 0.25*N[dim(N)[1],3],
                                                          0.5*N[dim(N)[1],2] + 0.5*N[dim(N)[1],3] + 0.5
                                                          0.25*N[dim(N)[1],3] + 0.5*N[dim(N)[1],4]))
progenyFrom3 <- rmultinom(n = 1, size = deltaN3, prob = c(0,
                                                          N[dim(N)[1],2] + 0.5*N[dim(N)[1],3],
```

```
                                                    0.5*N[dim(N)[1],3] + N[dim(N)[1],4]))
progenyTot <- cbind(progenyFrom1, progenyFrom2, progenyFrom3)
N1postRepro <- N[dim(N)[1],2] + rowSums(progenyTot)[1]
N2postRepro <- N[dim(N)[1],3] + rowSums(progenyTot)[2]
N3postRepro <- N[dim(N)[1],4] + rowSums(progenyTot)[3]
  Nt<-c(
        survivalPost[1] * N1postRepro,
        survivalPost[2] * N2postRepro,
        survivalPost[3] * N3postRepro
       )
  N<-rbind(N,NA)
  N[dim(N)[1],]<-c(t,Nt,sum(Nt))
    }
# plot
plot(Ntot~tick,data=N,type="l", ylab="effectif", ylim=c(0,1.05*P[1]))
lines(N$tick,N$N1,col="blue")
lines(N$tick,N$N2,col="green")
lines(N$tick,N$N3,col="red")
abline(h=P[1],lty=2)
```




```
save.image()
```

We're now set to play with the parameters!

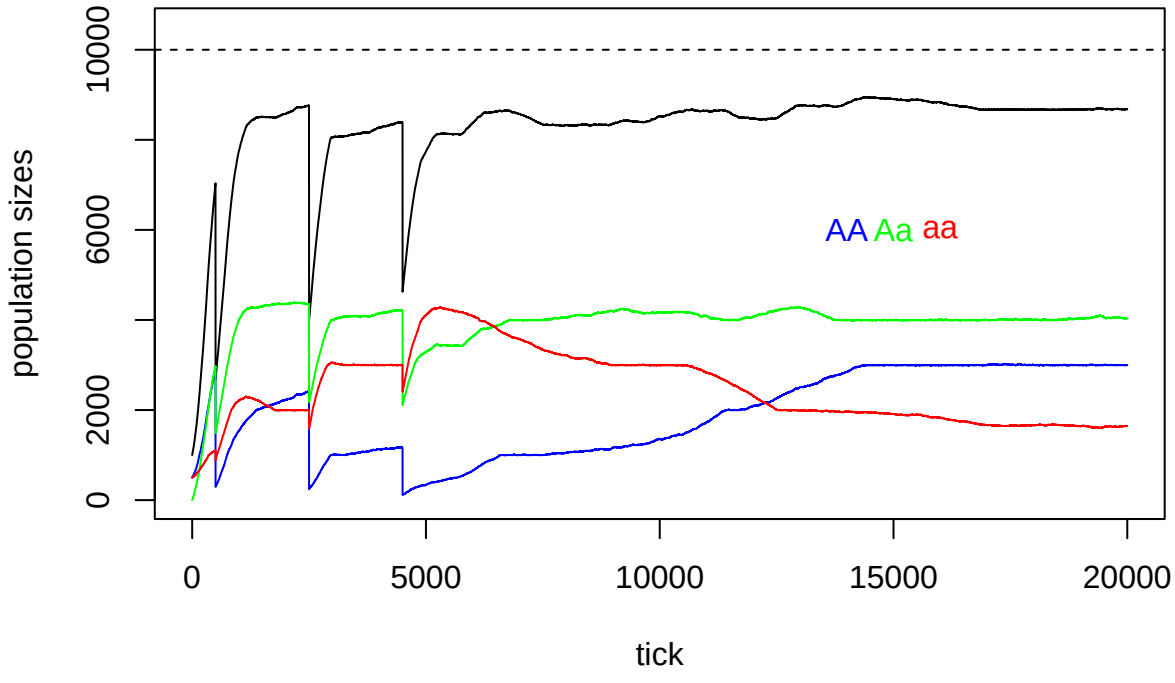
A new generalisation: multiple disturbances

This requires a rather fundamental change in the way the code is written: there is a single loop section, and depending on whether the tick is a 'normal' or a 'disturbance' one, one or the other of the functions for growth / population decrease is used. In a way, it is just a generalisation of the previous version, but it makes the whole code much easier to read.

Also, some mistakes (such as not rounding after mortality) have been fixed.

```
survival <- c("surv1" = 0.9990, "surv2" = 0.9990, "surv3" = 0.9990)             # allowing for variable
                                                                                # genotype
survivalPost <- c("surv1post" = 0.9990,                                         # allowing for variable
                  "surv2post" = 0.9990, "surv3post" = 0.9990)                   # genotype after distur
N<-data.frame("tick"=0, "N1"=500, "N2"=0, "N3"=500, "Ntot"=1000)                # setting initial count

# parameter vectors (r(i)post: repro; s(i): surv at disturbance)
P<-c("K"=10000, "r1"=0.0100, "r2"=0.0075, "r3"=0.0050,                          # new K = 10000 (carryi
     "s1"=0.1, "s2"=0.5, "s3"=0.8, "Kpost"=10000,                               # Kpost = carrying capa
     "r1post"=0.0100, "r2post"=0.0060,"r3post"=0.0020)
# disturbance dates
t0<-c(500,2500,4500)
# end of simulation
tEnd <- 20000
# start increment
# pre-perturbation
for(t in 1:tEnd) {
# computing increments for each genotype (= their "progeny")
                    if (!is.element(el = t, set = t0)) {                        # checks whether we are
                                                                                # in the following line
                                                                                # the do.call() calls c
                                                                                # based on GROWTH rates
                                                                                # and then the differen
                                                                                # the list() calls prov
                                                                                # application of the lo
                                                                                # N(1,s) = N(0,s) * (1
                                                                                # where:
                                                                                # N(0,s) is provided by
                                                                                # r(s) is provided by P
                                                                                # N(max) is provided by
                                                                                # N(0,tot) is provided
                                                        deltaN1 <- do.call(what = sample(x = c("ceiling
                                                                           args = list(N[dim(N)[1],2]*(
                                                        deltaN2 <- do.call(what = sample(x = c("ceiling
                                                                           args = list(N[dim(N)[1],3]*(
                                                        deltaN3 <- do.call(what = sample(x = c("ceiling
                                                                           args = list(N[dim(N)[1],4]*(
# attributing progeny to each genotype (through a random process, based on H-W expectations: usage of a
                                                        progenyFrom1 <- rmultinom(n = 1, size = deltaN1



                                                        progenyFrom2 <- rmultinom(n = 1, size = deltaN2


                                                        progenyFrom3 <- rmultinom(n = 1, size = deltaN3


                                                        progenyTot <- cbind(progenyFrom1, progenyFrom2,
                                                        N1postRepro <- N[dim(N)[1],2] + rowSums(progeny
                                                        N2postRepro <- N[dim(N)[1],3] + rowSums(progeny
                                                        N3postRepro <- N[dim(N)[1],4] + rowSums(progeny
```

```
                                                    Nt<-c(
                                                          do.call(what = sample(x = c("ceiling","flo
                                                          do.call(what = sample(x = c("ceiling","flo
                                                          do.call(what = sample(x = c("ceiling","flo
                                                         )
                                                    N<-rbind(N,NA)
                                                    N[dim(N)[1],]<-c(t,Nt,sum(Nt)) # updating the v
                                                   }
#
# disturbance step
# at disturbance, only mortality occurs, with mortality rates that apply specifically to disturbance ev
                      else {
                            Nt0<-c(
                                   do.call(what = sample(x = c("ceiling","floor"), size = 1), args = list
                                   do.call(what = sample(x = c("ceiling","floor"), size = 1), args = list
                                   do.call(what = sample(x = c("ceiling","floor"), size = 1), args = list
                                  )
                            N<-rbind(N,NA)
                            N[dim(N)[1],]<-c(t,Nt0,sum(Nt0))                    # updating the vector o
                           }
                  }
# plot
plot(Ntot~tick,data=N,type="l", ylab="population sizes", ylim=c(0,1.05*P[1]))
lines(N$tick,N$N1,col="blue")
lines(N$tick,N$N2,col="green")
lines(N$tick,N$N3,col="red")
abline(h=P[1],lty=2)
text(x = c(14000,15000,16000), y = rep(6000,3), labels = c("AA","Aa","aa"), col = c("blue","green","red
```




```
save.image()
```

A new step: modelling two independent loci driving survival and fecundity

The general equations stem from the developments in Chunk C8.

Yet the generation of two-locus genotypes cannot be performed at the population level, because this would prevent the development of any two-locus pattern.

Therefore, now **genotypes, their reproduction and mortality events must be modelled individually** (while summaries such as plotting will still be analysed at the summary, (sub)population level).

This requires that each individual "genotype" undergoes reproduction and survival events with probabilities provided by its genotype at the respective r and s locus, respectively.

This is a rather big change in the coding, but equations stay the same and the global (sub)populations' net growth(reproduction) and decrease (mortality) will be the outcome of the sum of individual events.

To make calculations simple, at each locus the genotype will be coded by fecundity and survival rates, respectively (as opposed to being coded as qualitative genotypes, e.g. RR, Rr, rr and SS, Ss, ss).

I expect that this will eventually lead to the fixation of a genotype with the best survival and fecundity rates, as no real trade-off (e.g., energy invested in each life-history trait) is involved. Yet for the sake of manipulating multi-locus, multi-trait individual genotypes, this will be fair enough.

```
# ********************************************************************************
# setting the parameters:
#`same as in C8 for the fecundity and survival rates,
# but the numbers of genotypes are now defined
# separately for two loci
#
survival <- c("surv1" = 0.9990, "surv2" = 0.9990, "surv3" = 0.9990)             # allowing for variable
                                                                                # genotype (no longer i
survivalPost <- c("surv1post" = 0.9990,                                         # allowing for variable
                  "surv2post" = 0.9990, "surv3post" = 0.9990)                   # genotype after distur
N<-data.frame("tick"=0,                                                         # define starting time
              "F1"=500, "F2"=0, "F3"=500,                                       # define initial genoty
              "S1"=500, "S2"=0, "S3"=500,                                       # define initial genoty
              "Ntot"=1000)                                                      # define initial total
#
NtwoLocus <- data.frame(matrix(nrow = 0, ncol = 10))                            # table for two-locus g
names(NtwoLocus) <-  c("tick","FFSS","FFSs","FFss","FfSS","FfSs","Ffss","ffSS","ffSs","ffss")
# parameter vectors (r(i)post: repro; s(i): surv at disturbance)
P<-c("K"=5000, "r1"=0.0100, "r2"=0.0075, "r3"=0.0050,                          # new K = 10000 (carryin
     "s1"=0.1, "s2"=0.5, "s3"=0.8, "Kpost"=5000,                               # Kpost = carrying capac
                                                                                # (optional; not implem
     "r1post"=0.0100, "r2post"=0.0060,"r3post"=0.0020)                          # r(i)post: fecundity a
# disturbance dates : either fixed or random
# fixed dates:
# t0 <- c(500,2500,4500)
# random dates:
library(KScorrect)
t0 <- sort(round(rlunif(5,200,19000)))
# end of simulation
tEnd <- 20000
#
# pace of genotype recording
recordSpacing <- 200
#
```

```
# ****************************************************************************
#
# Generating genotypes
genoF <- sample(x = P[2:4], size = N[dim(N)[1], dim(N)[2]], replace = T,        # I keep using dim() bu
                prob = N[dim(N)[1], 2:4])                                       # This is a random samp
                                                                                # proportions of the tw
genoS <- sample(x = P[5:7], size = N[dim(N)[1], dim(N)[2]], replace = T,        # I keep using dim() bu
                prob = N[dim(N)[1], 5:7])                                       # This is a random samp
                                                                                # proportions of the tw
                                                                                #   REMINDER: the "S" g
                                                                                #   flat survival rates
                                                                                #   otherwise in furthe
# sqnity checks
table(genoF); table(genoS)
```

```
## genoF
## 0.005  0.01
##   477   523
```

```
## genoS
## 0.1 0.8
## 491 509
```

```
# Creating the two-locus table
genoAll <- data.frame(genoF, genoS)
# sanity check
table(genoAll)
```

```
##        genoS
## genoF   0.1 0.8
##   0.005 233 244
##   0.01  258 265
```

```
#
# filling starting two-locus genotype table
NtwoLocusInit <- data.frame(matrix(data = c(0,
                     sum(genoAll[1] == P[2] & genoAll[2] == P[7]),              # number of RRSS
                     sum(genoAll[1] == P[2] & genoAll[2] == P[6]),              # number of RRSs
                     sum(genoAll[1] == P[2] & genoAll[2] == P[5]),              # number of RRss
                     sum(genoAll[1] == P[3] & genoAll[2] == P[7]),              # number of RrSS
                     sum(genoAll[1] == P[3] & genoAll[2] == P[6]),              # number of RrSs
                     sum(genoAll[1] == P[3] & genoAll[2] == P[5]),              # number of Rrss
                     sum(genoAll[1] == P[4] & genoAll[2] == P[7]),              # number of rrSS
                     sum(genoAll[1] == P[4] & genoAll[2] == P[6]),              # number of rrSs
                     sum(genoAll[1] == P[4] & genoAll[2] == P[5])),             # number of rrss
                          nrow = 1, ncol = 10))
names(NtwoLocusInit) <- names(NtwoLocus)
NtwoLocus <- rbind(NtwoLocus, NtwoLocusInit)
# End of genotype generation
#
# ****************************************************************************
# For the purposes of the reproduction phase, a "genotype transition matrix"
```

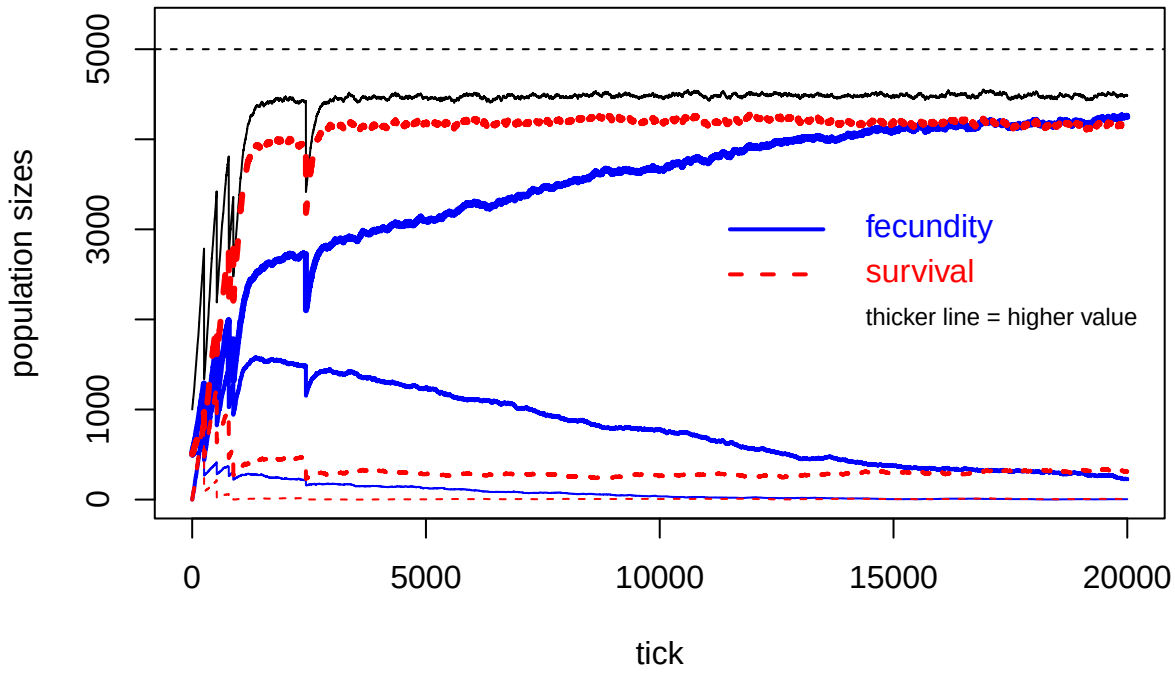
```
# giving the probability of obtaining a given genotype from a given cross
# must be provided (for each locus)
genoTransitionR <- data.frame(rep(P[2:4], 3),
                              rep(P[2:4], each = 3),
                              c(1,0.5,0,0.5,0.25,0,0,0,0),
                              c(0,0.5,1,0.5,0.5,0.5,1,0.5,0),
                              c(0,0,0,0,0.25,0.5,0,0.5,1))
names(genoTransitionR) <- c("geno1","geno2","p1","p2","p3")
genoTransitionS <- data.frame(rep(P[5:7], 3),
                              rep(P[5:7], each = 3),
                              c(1,0.5,0,0.5,0.25,0,0,0,0),
                              c(0,0.5,1,0.5,0.5,0.5,1,0.5,0),
                              c(0,0,0,0,0.25,0.5,0,0.5,1))
names(genoTransitionS) <- c("geno1","geno2","p1","p2","p3")
#
# ***************************************************************************
# Iterations: inheriting from Chunk C8 with (major!) modifications
#
for(t in 1:tEnd) {
# computing increments for each genotype (= their "progeny")
                    if (!is.element(el = t, set = t0)) {                        # checks whether we are
                                                                                # in the following line
                                                                                # the do.call() calls c
                                                                                # based on GROWTH rates
                                                                                # and then the differen
                                                                                # the list() calls prov
                                                                                # application of the lo
                                                                                # N(1,s) = N(0,s) * (1
                                                                                # where:
                                                                                # N(0,s) is provided by
                                                                                # r(s) is provided by i
                                                                                # N(max) is provided by
                                                                                # N(0,tot) is provided
# ***************************************************************************
# FECUNDITY step:
#       NEW in C9: each individual reproduces with a probability driven by its "F" genotype (and the po
#       and if it does reproduce, a new individual is drawn following H-W and added to the population
#
#
#   Function to determine success and assign genotype to progeny
                     repro <- function(y)
                     {
                       success <- sample(x = 1:0, size = 1,                     # here, the
                                         prob = c(y[1]*(P[1]-N[dim(N)[1],8])/P[1],      # reproduces
                                                  1 - y[1]*(P[1]-N[dim(N)[1],8])/P[1])   # and the ov
                                         )
                      if (success == 1)                                         # checks whe
                      {                                                         #
                        mate <- unlist(genoAll[sample(x = 1:nrow(genoAll), size = 1),1:2])# if so, ONE
                        progenyProbsR <- genoTransitionR[genoTransitionR[,1] == y[1] & genoTransitionR
                        progenyProbsS <- genoTransitionS[genoTransitionS[,1] == y[2] & genoTransitionS
                        progenyGeno <- c(sample(x = P[2:4], size = 1, prob = progenyProbsR), # and fin
                                         sample(x = P[5:7], size = 1, prob = progenyProbsS)) # for eac
```

```
                        return(progenyGeno)                                    # the function finally
                      }
                     }
# the repro() function needs to be applied to all genotypes, and the resulting progeny genotypes
# stored in a table
                     newGenotypes <- apply(X = genoAll, MARGIN = 1, FUN = repro)
                     if (!is.null(newGenotypes))                               # check that at least o
                     {
                       newGenotypes.cleaned <- newGenotypes[-which(sapply(newGenotypes, is.null))]
                       newGenotypes.df <- data.frame(do.call(what = rbind, args = newGenotypes.cleaned))
                     } else {                                                  # if no offspring produ
                            newGenotypes.df <- data.frame(matrix(nrow = 0, ncol = 2))
                            }
                     names(newGenotypes.df) <- names(genoAll)
# ***************************************************************************
# Mortality step:
# this is flat-rate outside the disturbance ticks
# individuals are randomly drawn to be kept
                    survivors <- sample(x = c(T,F), size = N[dim(N)[1], dim(N)[2]], replace = T,     # r
                                        prob = c(survival[1], 1- survival[1]))                       # f
                    genoAll.surviving <- genoAll[survivors,]                                         # a
# updating the genotype table for the new loop:
                    genoAll <- rbind(genoAll.surviving, newGenotypes.df)
# recomputing genotype counts and total numbers
                    Nt <- c(sum(genoAll[1] == P[2]),                          # Notice that counting
                            sum(genoAll[1] == P[3]),                          # table() command
                            sum(genoAll[1] == P[4]),                          # because it would misb
                            sum(genoAll[2] == P[5]),                          # one or more genotypes
                            sum(genoAll[2] == P[6]),                          #
                            sum(genoAll[2] == P[7]))                          #
                    N<-rbind(N,NA)
                    N[dim(N)[1],]<-c(t,Nt,sum(Nt[1:3])) # updating the vector of sub-population and pop
                                                      }
#
# ***************************************************************************
# disturbance step
# at disturbance, only mortality occurs, with mortality rates that apply specifically to disturbance ev
# and depending on genotype at the "S" locus
                    else {
#       Function to determine survival success
                      surv <- function(y)
                      {
                        S <- sample(x = c(T,F), size = 1, prob = c(y[2], 1-y[2]))
                        return(S)
                      }
# Application of the function to the "S" locus "genotypes"
                          survivorship <- unlist(apply(X = genoAll, MARGIN = 1, FUN = surv))
                          # updating genoAll (keeping successfull survivors)
                          genoAll <- genoAll[survivorship,]
# recomputing genotype counts and total numbers
                    Nt <- c(sum(genoAll[1] == P[2]),                          # Notice that counting
                            sum(genoAll[1] == P[3]),                          # table() command
                            sum(genoAll[1] == P[4]),                          # because it would misb
```

```
                        sum(genoAll[2] == P[5]),                          # one or more genotypes
                        sum(genoAll[2] == P[6]),                          #
                        sum(genoAll[2] == P[7]))                          #
                  N<-rbind(N,NA)
                  N[dim(N)[1],]<-c(t,Nt,sum(Nt[1:3])) # updating the vector of sub-population and pop
                  }
# computing ans storing TWO-LOCUS genotype frequencies
NtwoLocusLast <- data.frame(matrix(data = c(t,
                   sum(genoAll[1] == P[2] & genoAll[2] == P[7]),          # number of RRSS
                   sum(genoAll[1] == P[2] & genoAll[2] == P[6]),          # number of RRSs
                   sum(genoAll[1] == P[2] & genoAll[2] == P[5]),          # number of RRss
                   sum(genoAll[1] == P[3] & genoAll[2] == P[7]),          # number of RrSS
                   sum(genoAll[1] == P[3] & genoAll[2] == P[6]),          # number of RrSs
                   sum(genoAll[1] == P[3] & genoAll[2] == P[5]),          # number of Rrss
                   sum(genoAll[1] == P[4] & genoAll[2] == P[7]),          # number of rrSS
                   sum(genoAll[1] == P[4] & genoAll[2] == P[6]),          # number of rrSs
                   sum(genoAll[1] == P[4] & genoAll[2] == P[5])),         # number of rrss
                        nrow = 1, ncol = 10))
names(NtwoLocusLast) <- names(NtwoLocus)
NtwoLocus <- rbind(NtwoLocus, NtwoLocusLast)
  # storing genotypes
  if (t / recordSpacing == round(t / recordSpacing)) assign(x = paste0("genoAll_",t), value = genoAll)
                  }
# Saving
save.image()
```

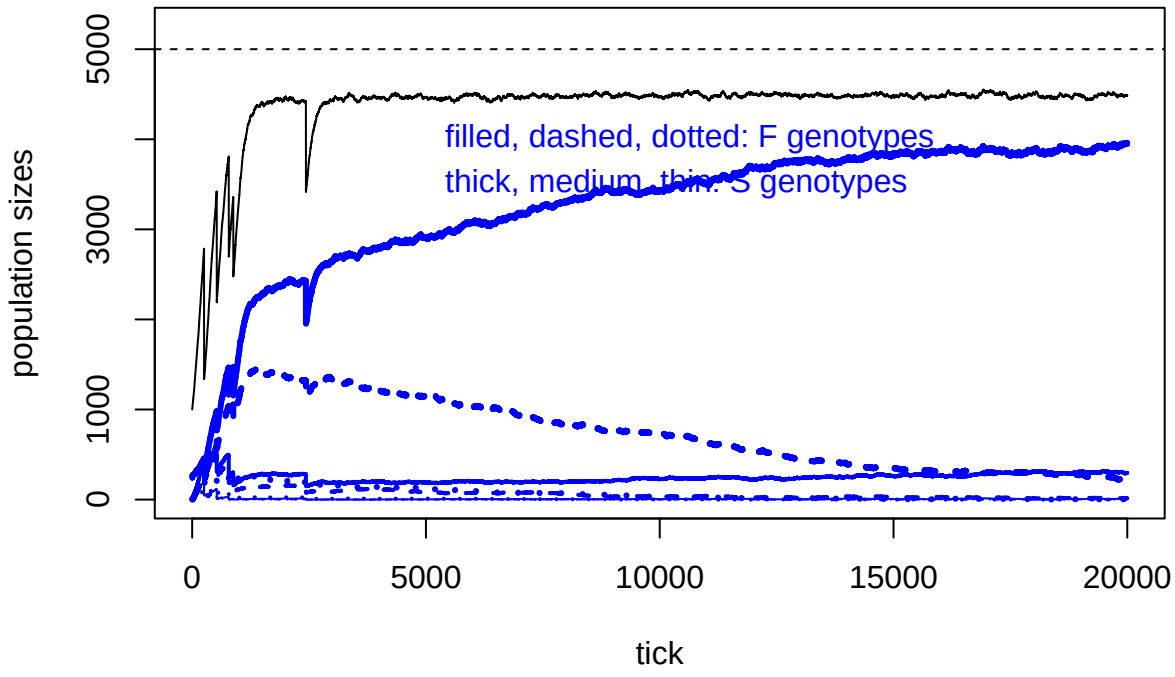
Plotting of single-locus genotype frequencies

```
# plot
plot(Ntot~tick,data=N,type="l", ylab="population sizes", ylim=c(0,1.05*P[1]))
lines(N$tick,N$F1,col="blue",lwd=3)
lines(N$tick,N$F2,col="blue",lwd=2)
lines(N$tick,N$F3,col="blue",lwd=1)
lines(N$tick,N$S1,col="red",lwd=1, lty = "dashed")
lines(N$tick,N$S2,col="red",lwd=2, lty = "dashed")
lines(N$tick,N$S3,col="red",lwd=3, lty = "dashed")
abline(h=P[1],lty=2)
text(x = c(14000,14000), y = c(0.6*P[1],0.5*P[1]), labels = c("fecundity","survival"),
     col = c("blue","red"), pos = 4)
text(x = 14000, y = 0.4*P[1], labels = "thicker line = higher value",
     pos = 4, cex = 0.75)
lines(x = c(11500,13500), y = c(0.6*P[1],0.6*P[1]), col="blue", lwd = 2)
lines(x = c(11500,13500), y = c(0.5*P[1],0.5*P[1]), col="red", lty = "dashed", lwd = 2)
```



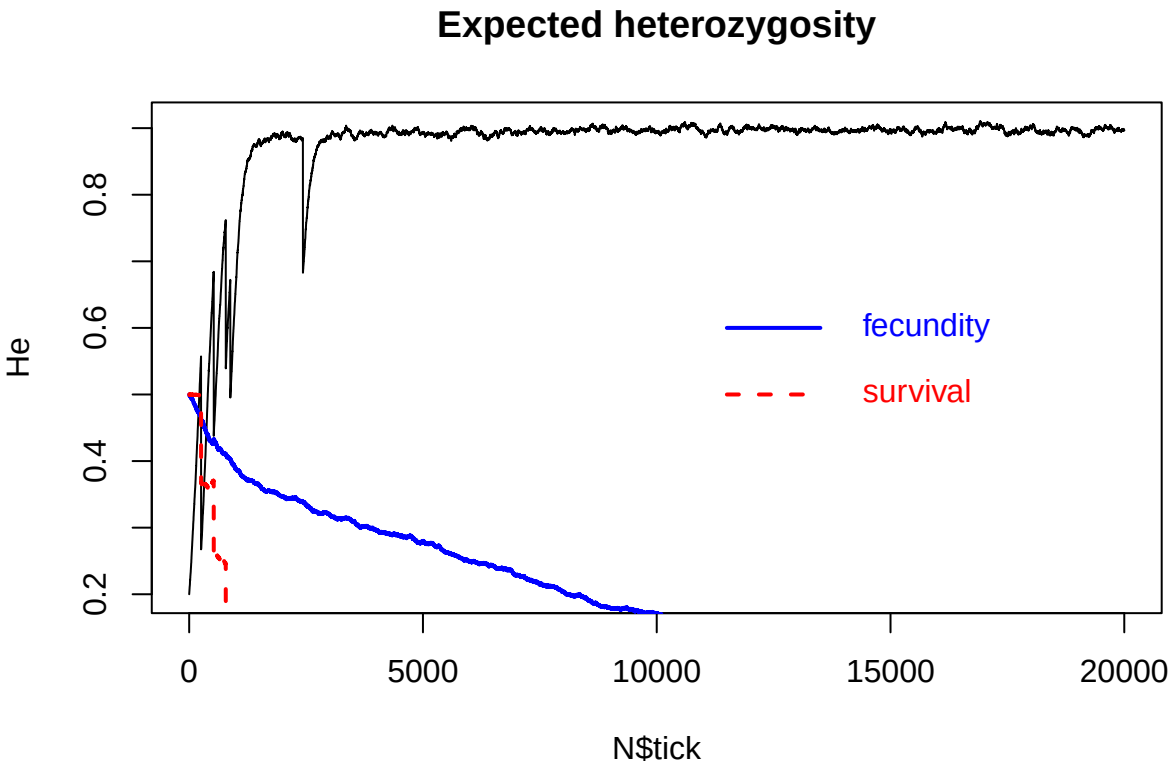
Plotting of two-locus genotype frequencies

```
# plot
plot(Ntot~tick,data=N,type="l", ylab="population sizes", ylim=c(0,1.05*P[1]))
lines(NtwoLocus$tick,NtwoLocus$FFSS,col="blue",lwd=3)
lines(NtwoLocus$tick,NtwoLocus$FFSs,col="blue",lwd=2)
lines(NtwoLocus$tick,NtwoLocus$FFss,col="blue",lwd=1)
lines(NtwoLocus$tick,NtwoLocus$FfSS,col="blue",lwd=3, lty = "dashed")
lines(NtwoLocus$tick,NtwoLocus$FfSs,col="blue",lwd=2, lty = "dashed")
lines(NtwoLocus$tick,NtwoLocus$Ffss,col="blue",lwd=1, lty = "dashed")
lines(NtwoLocus$tick,NtwoLocus$ffSS,col="blue",lwd=3, lty = "dotted")
lines(NtwoLocus$tick,NtwoLocus$ffSs,col="blue",lwd=2, lty = "dotted")
lines(NtwoLocus$tick,NtwoLocus$ffss,col="blue",lwd=1, lty = "dotted")
abline(h=P[1],lty=2)
text(x = c(5000,5000), y = c(0.8*P[1],0.7*P[1]), labels = c("filled, dashed, dotted: F genotypes","thick
     col = "blue", pos = 4)
```



Computing and plotting of genetic diversity

```
# Calculation of expected heterozygosity from genotype frequencies at the two loci
# function for calculations
expHet <- function(x)
{
  HeF <- 2* ((2*x[2] + x[3]) / (2 * sum(x[2:4]))) * (1 - ((2*x[2] + x[3]) / (2 * sum(x[2:4]))))
  HeS <- 2* ((2*x[5] + x[6]) / (2 * sum(x[5:7]))) * (1 - ((2*x[5] + x[6]) / (2 * sum(x[5:7]))))
  return(c(HeF,HeS))
}
# application of the function
He.df <- data.frame(t(apply(X = N, MARGIN = 1, FUN = expHet)))
names(He.df) <- c("HeF","HeS")
# plotting
plot((N$Ntot/P[1]) ~ N$tick,type="l", ylab="He", main = "Expected heterozygosity")
lines(N$tick, He.df$HeF, col="blue",lwd=2)
lines(N$tick, He.df$HeS, col="red",lwd=2, lty = "dashed")
text(x = c(14000,14000), y = c(0.6,0.5), labels = c("fecundity","survival"),
     col = c("blue","red"), pos = 4)
lines(x = c(11500,13500), y = c(0.6,0.6), col="blue", lwd = 2)
lines(x = c(11500,13500), y = c(0.5,0.5), col="red", lwd = 2, lty = "dashed")
```



A new step: trait-trait trade-off through locus-locus epistasis

The general equations stem from the developments in Chunk C9.

As expected, complete independence of effects (no trade-off) led to the increase in frequency of a genotype with the best survival and fecundity rates.

Now, a trade-off is introduced through a form of dominant epistasis, whereby the high-value homozygote genotype at one locus reduces the value at the other locus (and a genotype with two "plus" genotypes has depressed values at both traits).

Schematically, this will look like this (in the table, the genotypes are represented as 0, 1, 2, where 0 is the lowest breeding value, and the phenotypes are defined relative than the breeding values, e.g., "0-" means that the phenotype is lower than the breeding value):

F genotype	S genotype	F phenotype	S phenotype
0	0	0	0
0	1	0	1
0	2	0-	2+
1	0	1	0
1	1	1	1
1	2	1-	2
2	0	2+	0-
2	1	2	1-
2	2	1-	1-

The fecundity and survival probabilities will be based on *phenotype* as determined from the *genotype* after taking into account the epistatic interactions above.

```r
# ***************************************************************************
# setting the parameters:
#`same as in C8 for the fecundity and survival rates,
# but the numbers of genotypes are now defined
# separately for two loci
#
survival <- c("surv1" = 0.9990, "surv2" = 0.9990, "surv3" = 0.9990)             # allowing for variable
                                                                                # genotype (no longer i
survivalPost <- c("surv1post" = 0.9990,                                         # allowing for variable
                  "surv2post" = 0.9990, "surv3post" = 0.9990)                   # genotype after distur
N<-data.frame("tick"=0,                                                         # define starting time
              "F1"=500, "F2"=0, "F3"=500,                                       # define initial genoty
              "S1"=500, "S2"=0, "S3"=500,                                       # define initial genoty
              "Ntot"=1000)                                                      # define initial total
#
NtwoLocus <- data.frame(matrix(nrow = 0, ncol = 10))                            # table for two-locus g
names(NtwoLocus) <-  c("tick","FFSS","FFSs","FFss","FfSS","FfSs","Ffss","ffSS","ffSs","ffss")
# parameter vectors (r(i)post: repro; s(i): surv at disturbance)
P<-c("K"=5000, "r1"=0.0100, "r2"=0.0075, "r3"=0.0050,                          # new K = 10000 (carryin
     "s1"=0.1, "s2"=0.5, "s3"=0.8, "Kpost"=5000,                               # Kpost = carrying capac
                                                                                # (optional; not implem
     "r1post"=0.0100, "r2post"=0.0060,"r3post"=0.0020)                          # r(i)post: fecundity a
# disturbance dates : either fixed or random
# fixed dates:
# t0 <- c(500,2500,4500)
# random dates:
library(KScorrect)
t0 <- sort(round(rlunif(5,200,19000)))
# end of simulation
tEnd <- 20000
#
# pace of genotype recording
recordSpacing <- 200
#
# ***************************************************************************
#
# Generating genotypes
genoF <- sample(x = P[2:4], size = N[dim(N)[1], dim(N)[2]], replace = T,        # I keep using dim() bu
                prob = N[dim(N)[1], 2:4])                                       # This is a random samp
                                                                                # proportions of the tw
genoS <- sample(x = P[5:7], size = N[dim(N)[1], dim(N)[2]], replace = T,        # I keep using dim() bu
                prob = N[dim(N)[1], 5:7])                                       # This is a random samp
                                                                                # proportions of the tw
                                                                                #   REMINDER: the "S" g
                                                                                #   flat survival rates
                                                                                #   otherwise in furthe
# sqnity checks
table(genoF); table(genoS)
```

```
## genoF
## 0.005  0.01
##   475   525

## genoS
```

```
## 0.1 0.8
## 498 502
```

```
# Creating the two-locus table
genoAll <- data.frame(genoF, genoS)
row.names(genoAll) <-  NULL
# sanity check
table(genoAll)
```

```
##        genoS
## genoF   0.1 0.8
##   0.005 236 239
##   0.01  262 263
```

```
#
# filling starting two-locus genotype table
NtwoLocusInit <- data.frame(matrix(data = c(0,
                     sum(genoAll[1] == P[2] & genoAll[2] == P[7]),              # number of RRSS
                     sum(genoAll[1] == P[2] & genoAll[2] == P[6]),              # number of RRSs
                     sum(genoAll[1] == P[2] & genoAll[2] == P[5]),              # number of RRss
                     sum(genoAll[1] == P[3] & genoAll[2] == P[7]),              # number of RrSS
                     sum(genoAll[1] == P[3] & genoAll[2] == P[6]),              # number of RrSs
                     sum(genoAll[1] == P[3] & genoAll[2] == P[5]),              # number of Rrss
                     sum(genoAll[1] == P[4] & genoAll[2] == P[7]),              # number of rrSS
                     sum(genoAll[1] == P[4] & genoAll[2] == P[6]),              # number of rrSs
                     sum(genoAll[1] == P[4] & genoAll[2] == P[5])),             # number of rrss
                          nrow = 1, ncol = 10))
names(NtwoLocusInit) <- names(NtwoLocus)
NtwoLocus <- rbind(NtwoLocus, NtwoLocusInit)
# End of genotype generation
#
# ***************************************************************************
# For the purposes of the reproduction phase, a "genotype transition matrix"
# giving the probability of obtaining a given genotype from a given cross
# must be provided (for each locus)
genoTransitionR <- data.frame(rep(P[2:4], 3),
                              rep(P[2:4], each = 3),
                              c(1,0.5,0,0.5,0.25,0,0,0,0),
                              c(0,0.5,1,0.5,0.5,0.5,1,0.5,0),
                              c(0,0,0,0,0.25,0.5,0,0.5,1))
names(genoTransitionR) <- c("geno1","geno2","p1","p2","p3")
genoTransitionS <- data.frame(rep(P[5:7], 3),
                              rep(P[5:7], each = 3),
                              c(1,0.5,0,0.5,0.25,0,0,0,0),
                              c(0,0.5,1,0.5,0.5,0.5,1,0.5,0),
                              c(0,0,0,0,0.25,0.5,0,0.5,1))
names(genoTransitionS) <- c("geno1","geno2","p1","p2","p3")
#
# ***************************************************************************
# For introducing epistatic interactions, a matrix of transition of two-locus
# genotypes into two-character phenotypes is established, along with a function
# that allows to map phenotype onto genotype.
# Transition matrix:
```

```
# modification factors are set to 0.8 and 1.2 for negative and positive epistasis
epiNeg <- 0.8
epiPos <- 1.2
GenoToPheno <- data.frame(
  rep(P[4:2], each = 3),
  rep(P[5:7], times = 3),
  c(P[4],P[4],epiNeg*P[4], P[3],P[3],epiNeg*P[3], epiPos*P[2],P[2],epiNeg*P[3]),
  c(P[5],P[6],epiPos*P[7], P[5],P[6],P[7], epiNeg*P[5],epiNeg*P[6],epiNeg*P[6])
)
names(GenoToPheno) <- c("Fgeno", "Sgeno","Fpheno","Spheno")
# Transition function:
GtoPfunction <- function(x)
{
  pheno <- GenoToPheno[GenoToPheno[1] == x[1] & GenoToPheno[2] == x[2] ,3:4]
  return(pheno)
}
# Generating starting phenotypes:
phenoAll <- do.call(what = rbind, args = apply(X = genoAll, MARGIN = 1, FUN = GtoPfunction))
row.names(phenoAll) <-  NULL
#
# Merging genotype and phenotype
genoPhenoAll <- cbind(genoAll, phenoAll)
# ****************************************************************************
# Iterations: inheriting from Chunk C8 with (major!) modifications
#
for(t in 1:tEnd) {
# computing increments for each genotype (= their "progeny")
                    if (!is.element(el = t, set = t0)) {                        # checks whether we are
                                                                                # in the following line
                                                                                # the do.call() calls c
                                                                                # based on GROWTH rates
                                                                                # and then the differen
                                                                                # the list() calls prov
                                                                                # application of the lo
                                                                                # N(1,s) = N(0,s) * (1
                                                                                # where:
                                                                                # N(0,s) is provided by
                                                                                # r(s) is provided by i
                                                                                # N(max) is provided by
                                                                                # N(0,tot) is provided
# ****************************************************************************
# FECUNDITY step:
#       NEW in C9: each individual reproduces with a probability driven by its "F" genotype (and the po
#       and if it does reproduce, a new individual is drawn following H-W and added to the population
#
#
#   Function to determine success and assign genotype to progeny
                      repro <- function(y)
                      {
                        success <- sample(x = 1:0, size = 1,                                    # here, the
                                          prob = c(y[3]*(P[1]-N[dim(N)[1],8])/P[1],             # reproduces
                                                   1 - y[3]*(P[1]-N[dim(N)[1],8])/P[1])         # and the ov
                                          )
```

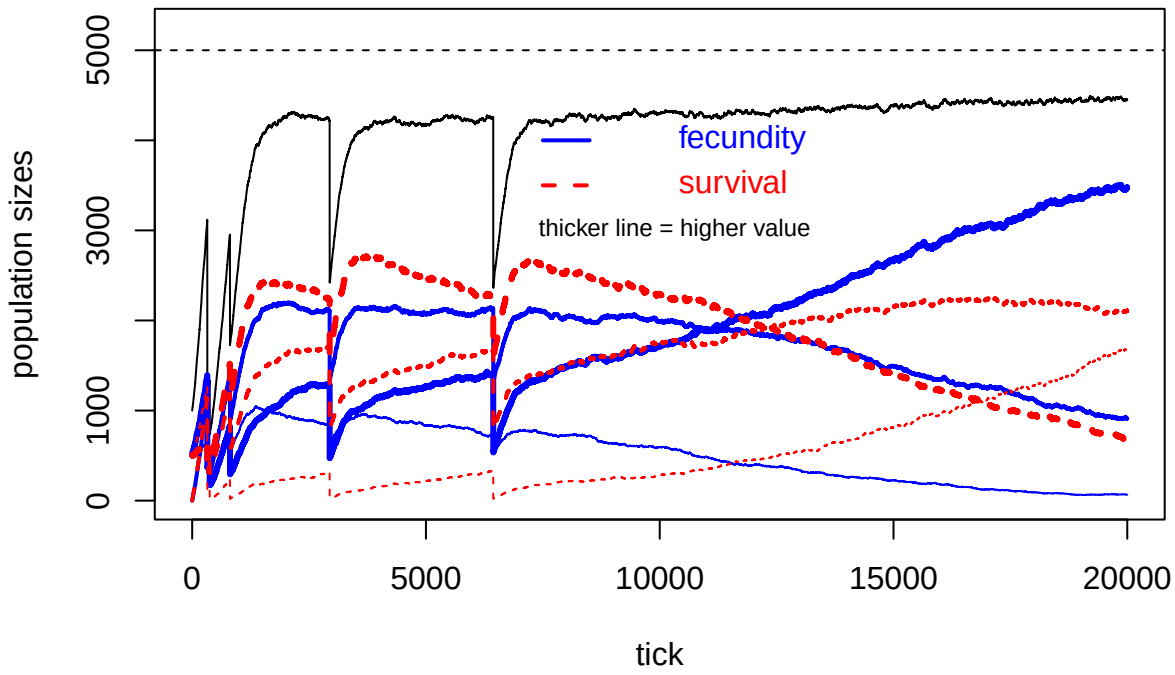
```
                    if (success == 1)                                                    # checks whe
                    {                                                                    #
                      mate <- unlist(genoAll[sample(x = 1:nrow(genoAll), size = 1),1:2])# if so, ONE
                      progenyProbsR <- genoTransitionR[genoTransitionR[,1] == y[1] & genoTransitionR
                      progenyProbsS <- genoTransitionS[genoTransitionS[,1] == y[2] & genoTransitionS
                      progenyGeno <- c(sample(x = P[2:4], size = 1, prob = progenyProbsR), # and fin
                                       sample(x = P[5:7], size = 1, prob = progenyProbsS)) # for eac
                      return(progenyGeno)                                                # the function finally
                    }
                   }
# the repro() function needs to be applied to all genotypes, and the resulting progeny genotypes
# stored in a table
                   newGenotypes <- apply(X = genoPhenoAll, MARGIN = 1, FUN = repro)
                   if (!is.null(newGenotypes))                                         # check that at least o
                   {
                     newGenotypes.cleaned <- newGenotypes[-which(sapply(newGenotypes, is.null))]
                     newGenotypes.df <- data.frame(do.call(what = rbind, args = newGenotypes.cleaned))
                   } else {                                                            # if no offspring produ
                          newGenotypes.df <- data.frame(matrix(nrow = 0, ncol = 2))
                          }
                   names(newGenotypes.df) <- names(genoAll)
# ***************************************************************************
# Mortality step:
# this is flat-rate outside the disturbance ticks
# individuals are randomly drawn to be kept
                   survivors <- sample(x = c(T,F), size = N[dim(N)[1], dim(N)[2]], replace = T,     # r
                                       prob = c(survival[1], 1- survival[1]))                       # f
                   genoAll.surviving <- genoAll[survivors,]                                         # a
# updating the genotype table for the new loop:
                   genoAll <- rbind(genoAll.surviving, newGenotypes.df)
# generating the phenotype table for the new loop:
                   phenoAll <- do.call(what = rbind, args = apply(X = genoAll, MARGIN = 1, FUN = GtoPf
                   row.names(phenoAll) <-  NULL
                   genoPhenoAll <- cbind(genoAll,phenoAll)
# recomputing genotype counts and total numbers
                   Nt <- c(sum(genoAll[1] == P[2]),                                    # Notice that counting
                           sum(genoAll[1] == P[3]),                                    # table() command
                           sum(genoAll[1] == P[4]),                                    # because it would misb
                           sum(genoAll[2] == P[5]),                                    # one or more genotypes
                           sum(genoAll[2] == P[6]),                                    #
                           sum(genoAll[2] == P[7]))                                    #
                   N<-rbind(N,NA)
                   N[dim(N)[1],]<-c(t,Nt,sum(Nt[1:3])) # updating the vector of sub-population and pop
                                                 }
#
# ***************************************************************************
# disturbance step
# at disturbance, only mortality occurs, with mortality rates that apply specifically to disturbance ev
# and depending on genotype at the "S" locus
                   else {
#       Function to determine survival success
                     surv <- function(y)
                     {
```

```
                        S <- sample(x = c(T,F), size = 1, prob = c(y[4], 1-y[4]))
                        return(S)
                      }
# Application of the function to the "S" locus "genotypes"
                        survivorship <- unlist(apply(X = genoPhenoAll, MARGIN = 1, FUN = surv))
                        # updating genoAll (keeping successfull survivors)
                        genoAll <- genoAll[survivorship,]
                        # generating the phenotype table for the new loop:
                        phenoAll <- do.call(what = rbind,
                                            args = apply(X = genoAll, MARGIN = 1, FUN = GtoPfunction))
                        row.names(phenoAll) <-  NULL
                        genoPhenoAll <- cbind(genoAll,phenoAll)
# recomputing genotype counts and total numbers
                    Nt <- c(sum(genoAll[1] == P[2]),                              # Notice that counting
                            sum(genoAll[1] == P[3]),                              # table() command
                            sum(genoAll[1] == P[4]),                              # because it would misb
                            sum(genoAll[2] == P[5]),                              # one or more genotypes
                            sum(genoAll[2] == P[6]),                              #
                            sum(genoAll[2] == P[7]))                              #
                    N<-rbind(N,NA)
                    N[dim(N)[1],]<-c(t,Nt,sum(Nt[1:3])) # updating the vector of sub-population and pop
                    }
# computing ans storing TWO-LOCUS genotype frequencies
NtwoLocusLast <- data.frame(matrix(data = c(t,
                     sum(genoAll[1] == P[2] & genoAll[2] == P[7]),                # number of RRSS
                     sum(genoAll[1] == P[2] & genoAll[2] == P[6]),                # number of RRSs
                     sum(genoAll[1] == P[2] & genoAll[2] == P[5]),                # number of RRss
                     sum(genoAll[1] == P[3] & genoAll[2] == P[7]),                # number of RrSS
                     sum(genoAll[1] == P[3] & genoAll[2] == P[6]),                # number of RrSs
                     sum(genoAll[1] == P[3] & genoAll[2] == P[5]),                # number of Rrss
                     sum(genoAll[1] == P[4] & genoAll[2] == P[7]),                # number of rrSS
                     sum(genoAll[1] == P[4] & genoAll[2] == P[6]),                # number of rrSs
                     sum(genoAll[1] == P[4] & genoAll[2] == P[5])),               # number of rrss
                          nrow = 1, ncol = 10))
names(NtwoLocusLast) <- names(NtwoLocus)
NtwoLocus <- rbind(NtwoLocus, NtwoLocusLast)
  # storing genotypes
  if (t / recordSpacing == round(t / recordSpacing)) assign(x = paste0("genoAll_",t), value = genoAll)
                  }
# Saving
save.image()
```

Plotting of single-locus genotype frequencies

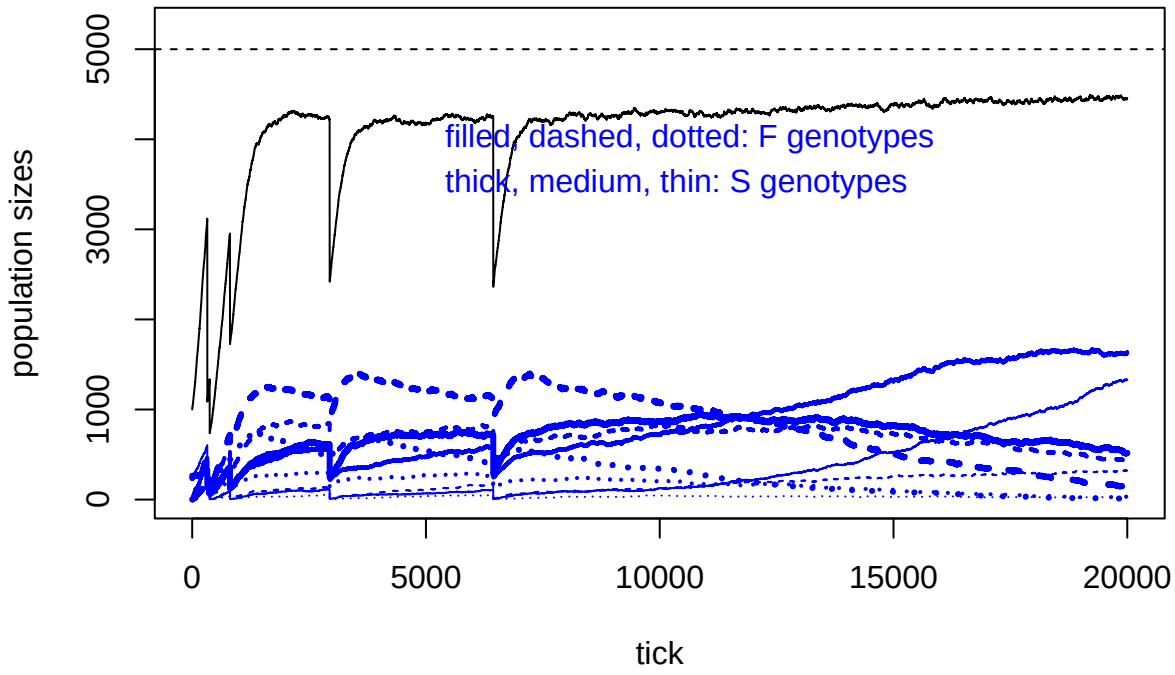
```
# plot
plot(Ntot~tick,data=N,type="l", ylab="population sizes", ylim=c(0,1.05*P[1]))
lines(N$tick,N$F1,col="blue",lwd=3)
lines(N$tick,N$F2,col="blue",lwd=2)
lines(N$tick,N$F3,col="blue",lwd=1)
lines(N$tick,N$S1,col="red",lwd=1, lty = "dashed")
lines(N$tick,N$S2,col="red",lwd=2, lty = "dashed")
lines(N$tick,N$S3,col="red",lwd=3, lty = "dashed")
```

```
abline(h=P[1],lty=2)
text(x = c(10000,10000), y = c(0.8*P[1],0.7*P[1]), labels = c("fecundity","survival"),
     col = c("blue","red"), pos = 4)
text(x = 7000, y = 0.6*P[1], labels = "thicker line = higher value",
     pos = 4, cex = 0.75)
lines(x = c(7500,8500), y = c(0.8*P[1],0.8*P[1]), col="blue", lwd = 2)
lines(x = c(7500,8500), y = c(0.7*P[1],0.7*P[1]), col="red", lty = "dashed", lwd = 2)
```



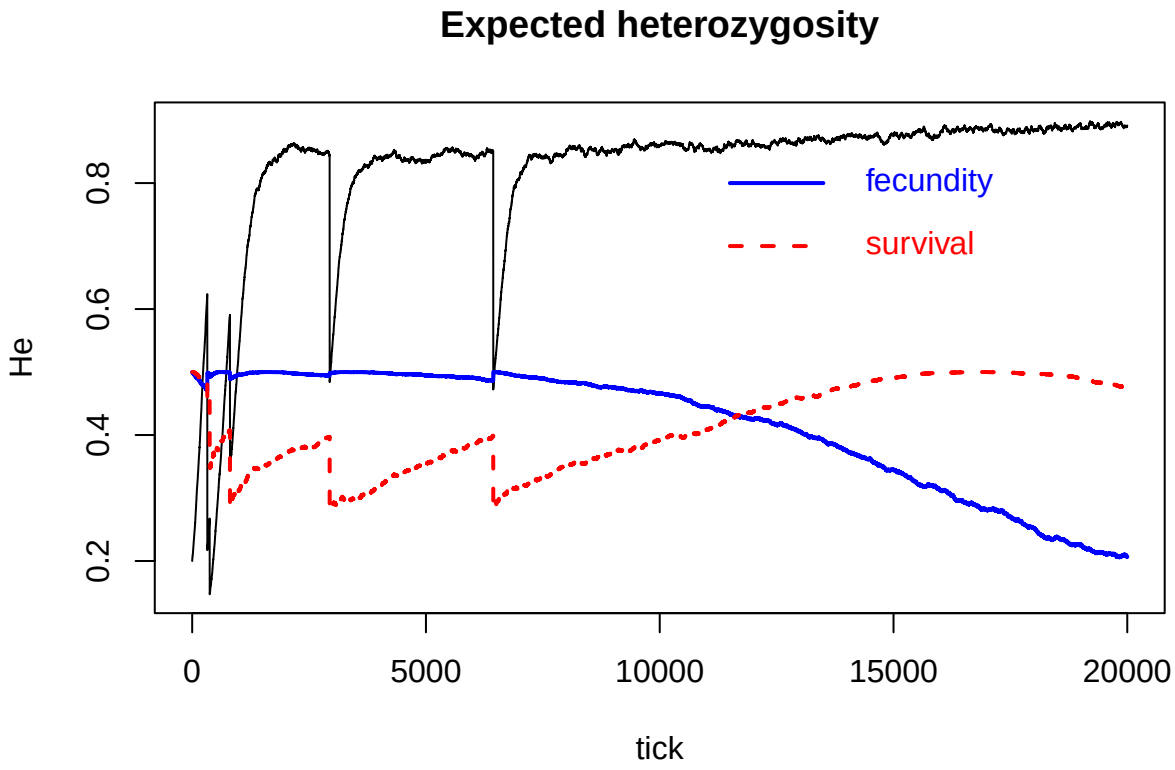
Plotting of two-locus genotype frequencies

```
# plot
plot(Ntot~tick,data=N,type="l", ylab="population sizes", ylim=c(0,1.05*P[1]))
lines(NtwoLocus$tick,NtwoLocus$FFSS,col="blue",lwd=3)
lines(NtwoLocus$tick,NtwoLocus$FFSs,col="blue",lwd=2)
lines(NtwoLocus$tick,NtwoLocus$FFss,col="blue",lwd=1)
lines(NtwoLocus$tick,NtwoLocus$FfSS,col="blue",lwd=3, lty = "dashed")
lines(NtwoLocus$tick,NtwoLocus$FfSs,col="blue",lwd=2, lty = "dashed")
lines(NtwoLocus$tick,NtwoLocus$Ffss,col="blue",lwd=1, lty = "dashed")
lines(NtwoLocus$tick,NtwoLocus$ffSS,col="blue",lwd=3, lty = "dotted")
lines(NtwoLocus$tick,NtwoLocus$ffSs,col="blue",lwd=2, lty = "dotted")
lines(NtwoLocus$tick,NtwoLocus$ffss,col="blue",lwd=1, lty = "dotted")
abline(h=P[1],lty=2)
text(x = c(5000,5000), y = c(0.8*P[1],0.7*P[1]), labels = c("filled, dashed, dotted: F genotypes","thick
     col = "blue", pos = 4)
```



Computing and plotting of genetic diversity

```
# Calculation of expected heterozygosity from genotype frequencies at the two loci
# function for calculations
expHet <- function(x)
{
  HeF <- 2* ((2*x[2] + x[3]) / (2 * sum(x[2:4]))) * (1 - ((2*x[2] + x[3]) / (2 * sum(x[2:4]))))
  HeS <- 2* ((2*x[5] + x[6]) / (2 * sum(x[5:7]))) * (1 - ((2*x[5] + x[6]) / (2 * sum(x[5:7]))))
  return(c(HeF,HeS))
}
# application of the function
He.df <- data.frame(t(apply(X = N, MARGIN = 1, FUN = expHet)))
names(He.df) <- c("HeF","HeS")
# plotting
plot((N$Ntot/P[1]) ~ N$tick,type="l", ylab="He",
     xlab = "tick", main = "Expected heterozygosity")
lines(N$tick, He.df$HeF, col="blue",lwd=2)
lines(N$tick, He.df$HeS, col="red",lwd=2, lty = "dashed")
text(x = c(14000,14000), y = c(0.8,0.7), labels = c("fecundity","survival"),
     col = c("blue","red"), pos = 4)
lines(x = c(11500,13500), y = c(0.8,0.8), col="blue", lwd = 2)
lines(x = c(11500,13500), y = c(0.7,0.7), col="red", lwd = 2, lty = "dashed")
```



To do list

- compute & display allele frequencies
- find a way to compute slopes analytically