ELECTRONIC SUPPLEMENTARY MATERIAL

How migratory populations become resident

Tiago de Zoeten and Francisco Pulido

Department of Biodiversity, Ecology and Evolution, Complutense University of Madrid, 28040 Madrid, Spain

Corresponding author: fpulido@ucm.es

Contents:

DETAILED DESCRIPTION OF SIMULATION MODEL

Initialisation

Sampling of initial allelic values

The initial number of alleles per locus was set to be slightly higher than observed at equilibrium under stabilising selection. Therefore, for simulations with 200 loci, 3 possible allelic values were sampled per locus, and for 10 loci, 10 potential allelic values were sampled. For each locus, possible allelic values were sampled from a normal distribution with mean $\frac{\mu_P}{2n}$ and variance 2.5 σ_α (see table S1 for a summary of the key parameters in the simulation model). These parameters, which determine the initial distribution of allele effects, were chosen to speed up the time to reach equilibrium under stabilising selection (see below) by ensuring that the mean migratory activity is close to the desired one and that there is an excess of genetic variance. Each of the *n* loci of each of the *N* individuals was then randomly assigned one of these possible allelic values. This generated an initial excess of genetic variation which then evolved to an equilibrium under stabilizing selection. Note that the characteristics of the population at equilibrium were independent of the starting genetic variance and number of alleles per locus and that these values were simply chosen to reduce the time to equilibrium.

Stabilizing selection

After sampling allelic values, populations were allowed to evolve under stabilizing selection for a total of 15000 generations. Stabilizing selection was implemented through the standard Gaussian model of phenotypic stabilizing selection [53,54].

$$w = f(l) = exp\left[\frac{-(l - \theta)^2}{2\omega^2}\right]$$

, where w is the fitness, l a given liability value, θ the optimal liability value. The strength of stabilizing selection of genotypic values, standardised by the environmental variance is given by:

$$Vs = \frac{{\sigma^2}_E + \omega^2}{{\sigma^2}_E}$$

Where smaller values of Vs are associated with stronger stabilizing selection. Since the environmental variance ${\sigma^2}_E$ is kept constant throughout the simulations, the strength of stabilizing selection is manipulated by adjusting the value of ω^2. The strength of stabilizing selection required to obtain an equilibrium heritability of 0.432 was determined to be approximately $Vs = 150$ with 10 loci and $Vs = 75$ with 200 loci.

For either genetic architecture (10 or 200 loci) a single population was created and allowed to evolve to equilibrium for 13000 generations (see, figure S4). The resulting populations were used as the starting point of 20 independent simulations of 2000 generations to obtain a sample of populations that is representative of equilibrium conditions. The result of this process were 20 populations for each of the 2 genetic architectures with the desired mean (μ_P), standard deviation ($\boldsymbol{\sigma_P}$) and heritability ($h_{\mathbf{0}}^2$) for migratory activity and with the allele frequencies that emerge as a result of selection under stabilizing selection.

Simulation model

Selection

After the initialisation, selection starts taking place through the chosen fitness function that relates an individual's liability with its survival probability. Individuals with a liability below the threshold all have the same fitness, while those with a liability greater than the threshold have their fitness determined by one of the following equations that relate their liability values with fitness.

Linear:

$$w = f(l) = -b\, l + c \tag{6}$$

Exponential:

$$w = f(l) = \alpha e^{-\lambda\, l} \tag{7}$$

Logistic:

$$w = f(l) = q - \frac{q}{1 + e^{-\boldsymbol{k}(l-\boldsymbol{\beta})}} \quad (8)$$

, where w is the fitness, l the liability value and the remaining are parameters which are manipulated to define the relationship between fitness and liability values. Fitness functions return individual fitness values which are converted into relative fitness by dividing them by the maximum individual fitness value observed in the population at the current generation. The fraction of the population that survives to breed, of size $\frac{2N}{f}$, is sampled from the entire population with sampling probabilities given by these relative fitness values.

Random mating

To simulate random mating, the $\frac{2N}{f}$ selected individuals are paired at random and each pair produces f offspring, such that the population size N remains constant. Genotypes are obtained by randomly sampling alleles from either parent. Therefore, to obtain the alleles of a new offspring, individual i, at locus v:

$Z_{i,v,1,g}$ is randomly sampled from discrete uniform distribution: $\{ Z_{p_1,v,1,g-1} , Z_{p_1,v,2,g-1} \}$

$Z_{i,v,2,g}$ is randomly sampled from discrete uniform distribution: $\{ Z_{p_2,v,1,g-1} , Z_{p_2,v,2,g-1} \}$ (9)

, where p_1 and p_2 correspond to the indices of the parents of individual i in generation $g - 1$.

Assortative mating

To simulate assortative mating, we incorporate the folded normal distribution mating preference function proposed by Carvajal-Rodriguez and Rolán-Alvarez [24] which estimates the mating probability of two individuals as:

$$\frac{-C^2 D^2}{e^{s^2 D_{max}^2}} \quad (10)$$

, where D is the difference between the liability values of two individuals, D_{max} is the maximum value of D, and both C and s are parameters which are used to control the strength of assortative mating. This formulation is particularly useful for incorporating empirical data because it is scale-invariant, i.e. the strength of assortative mating does not depend on the

phenotypic scale being used. After selection and before reproduction, a matrix of pairwise differences in migratory activity between all selected individuals is computed (liability values below the threshold are assigned a value of 0). The mating preference function is then applied to each value of D to generate a matrix of pairwise mating probabilities. $\frac{N}{f}$ pairs of individuals are then sampled from all possible pairs with the sampling probabilities given by this mating probability matrix**.** Each mating pair produces f offspring whose genotype at any given locus is obtained by randomly sampling an allele from each of its parents at this locus.

Mutation

Mutation followed an incremental continuum-of-alleles model where the effect of an allele after mutation equals its effect before mutation plus a random deviate with mean zero:

$$Z^*_{ivjg} = \begin{cases} Z_{ivjg}, & if\ Binomial(1, \mu) = 0 \\ Z_{ivjg} + Normal(0, \sigma_\alpha), & if\ Binomial(1, \mu) = 1 \end{cases} \quad (11)$$

, where Z^*_{ivjg} is the allelic value after mutation, Z_{ivjg} is the allelic value before mutation and σ_α is the standard deviation of the distribution of effects of new mutations on the previous allelic value.

Computing genetic values and liabilities

An individual's genetic value for the liability corresponds to the sum of the effects of all its alleles.

$$A_{ig} = \sum_{v=1}^{n} \sum_{j=1}^{2} Z_{ivjg} \quad (12)$$

, where A_{ig} is the genetic value of individual i at generation g.

Individual environmental effects on the liability are sampled from a normal distribution:

$$E \sim Normal(0, \sigma_E) \quad (13)$$

Individual liability values are the sum of the genetic values and the environmental deviates.

$$l_{ig} = A_{ig} + E_{ig} \quad (14)$$

Threshold variance

To explore the effect of variance in threshold values, we introduce a step in the simulation process where individuals are randomly assigned a threshold value which is sampled from a normal distribution:

$$T \sim Normal(0, \sigma_T) \quad (15)$$

, where T is the threshold value and σ_T the standard deviation in threshold values. Note that we did not model a genetic contribution to threshold values, and that this algorithm assumes threshold values are environmentally determined. When the threshold is variable, the selection algorithm is modified to incorporate the individual threshold values and these values are stored as output so that individual phenotypes can be determined after running the simulations.

INPUT PARAMETER CHOICE

Genetic and environmental variance for migratory activity

From the phenotypic variance reported for the blackcap population in 1988 (VP = 387^2) and the heritability for migratory activity in this population ($h^2 = 0.432$) we estimated the genetic variance as

$$V_A = h^2 V_P \quad (1)$$

, where V_A is the additive genetic variance h^2 the heritability and V_P the phenotypic variance for the liability. We then estimated the standard deviation in environmental effects as:

$$\sigma_E = \sqrt{\left(\frac{V_A}{h^2}\right) - V_A} \quad (2)$$

While environmental variance remains constant over the course of a simulation, genetic variance is initially set to match the estimated value but then evolves as a stochastic property of the population.

Strength of assortative mating

To test the effect of assortative mating, we simulated scenarios where the correlation coefficient for the migratory activity of parents was 0.25 or 0.70. To find the parameters of the mating preference function which would lead to these correlation coefficient values, we ran simulations with the same initialisation and fixed parameters as for other simulations (see table S2) but fixed the value of s of the mating preference function at 0.1 and varied the value of C within a range of 0.001 and 0.9. For each value of C we ran 100 simulations and recorded the correlation coefficient at the second generation of every simulation. Figure S3 shows the resulting curve which relates de value of C with the correlation coefficient generated by simulation. A loess lines of best fit for this relationship was computed using the *loess* function in R [23], and the resulting model was used to obtain the values of C that would lead to the desired values of r. Thus, C should equal 0.194 to get a correlation coefficient of r = 0.25 and 0.433 to get r = 0.70.

With this approach, we manipulated the initial correlation coefficient but within a simulation the actual correlation coefficient could fluctuate. Importantly, as more individuals become resident there is less variation in migratory activity such that the value of r is reduced (see, figure S3). This is to be expected since assortative mating based on the amount of migratory activity is expected to be weaker when there is limited phenotypic variation for this trait.

ADDITIONAL REFERENCES

53. Haldane, J. B. S. 1954 The measurement of natural selection. *Proc. IX Internal. Cong. Genet.* ***1****,* 480–487.
54. Weldon,W. F. R. 1895. An attempt to measure the death rate due to the selective destruction of *Carcinus moenas* with respect to a particular dimension. *Proc. Roy. Soc. Lond*. **57**, 360–379.

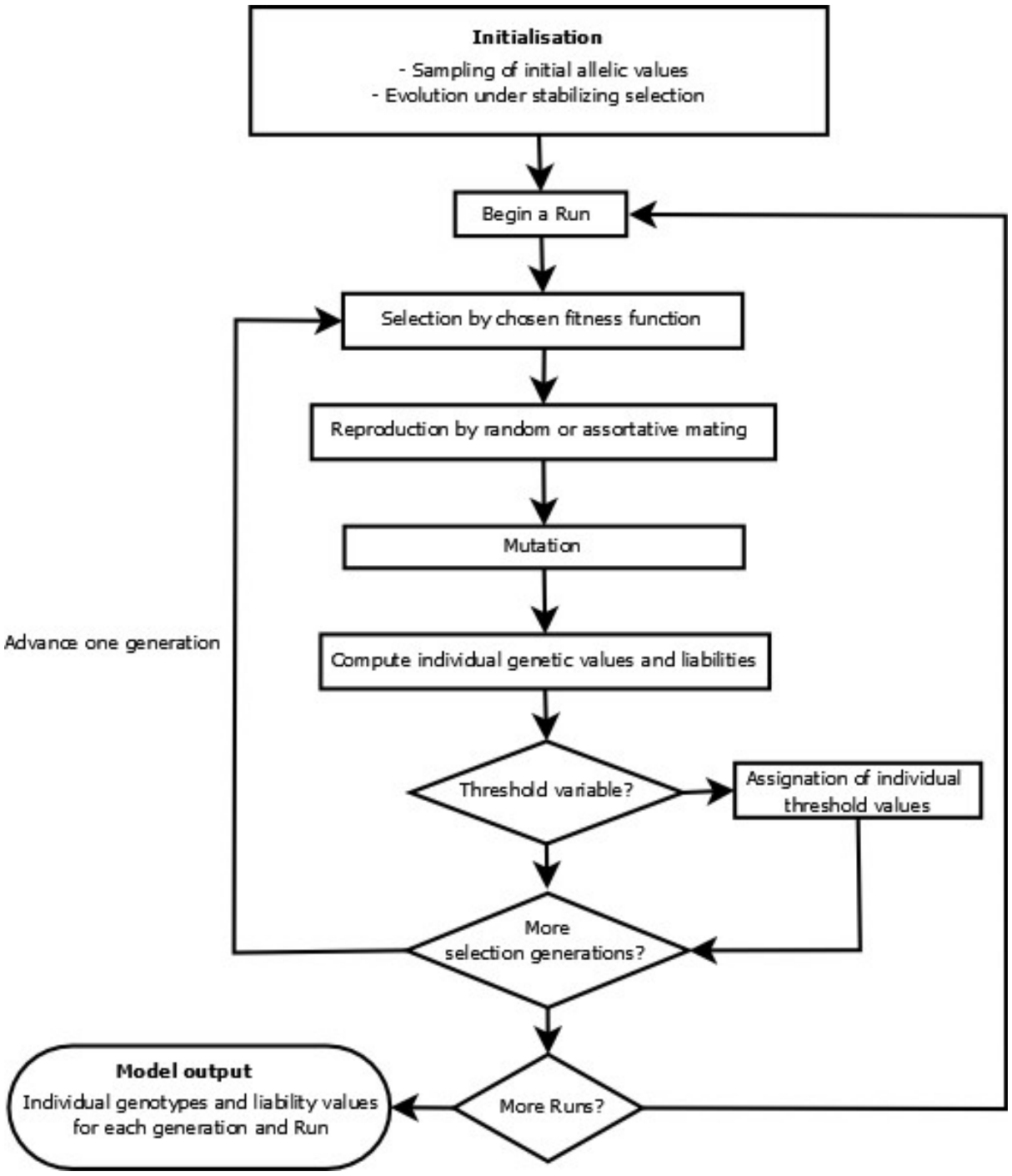

Figure S1: Overview of the process schedule used in the simulation model.

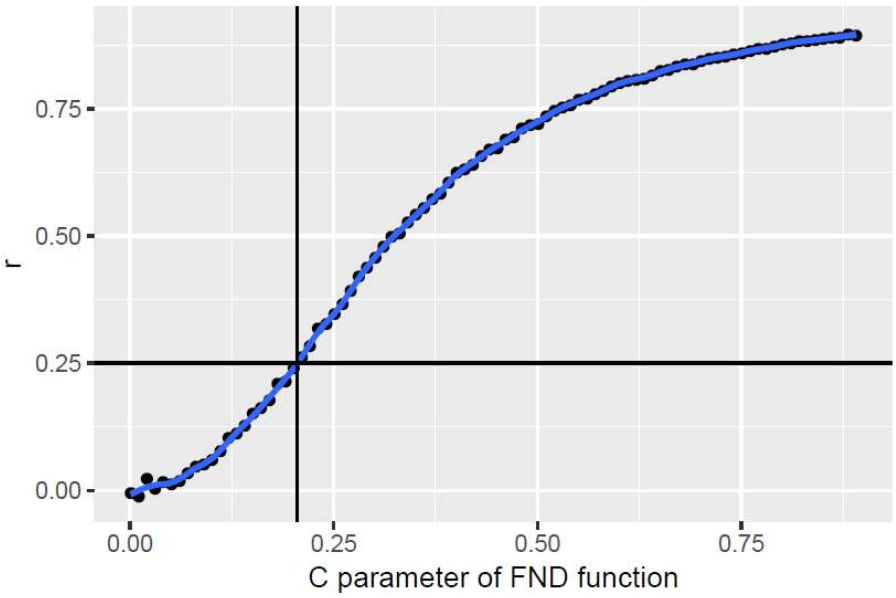

Figure S2: Relationship between the C parameter of the folded normal distribution mating-preference-function (see equation 10 in the assortative mating section above) and the correlation coefficient for the migratory activity of parents (r). Each point represents the average value of r of 100 simulations. The blue line shows the line of best fit that we used to calculate the value of C.

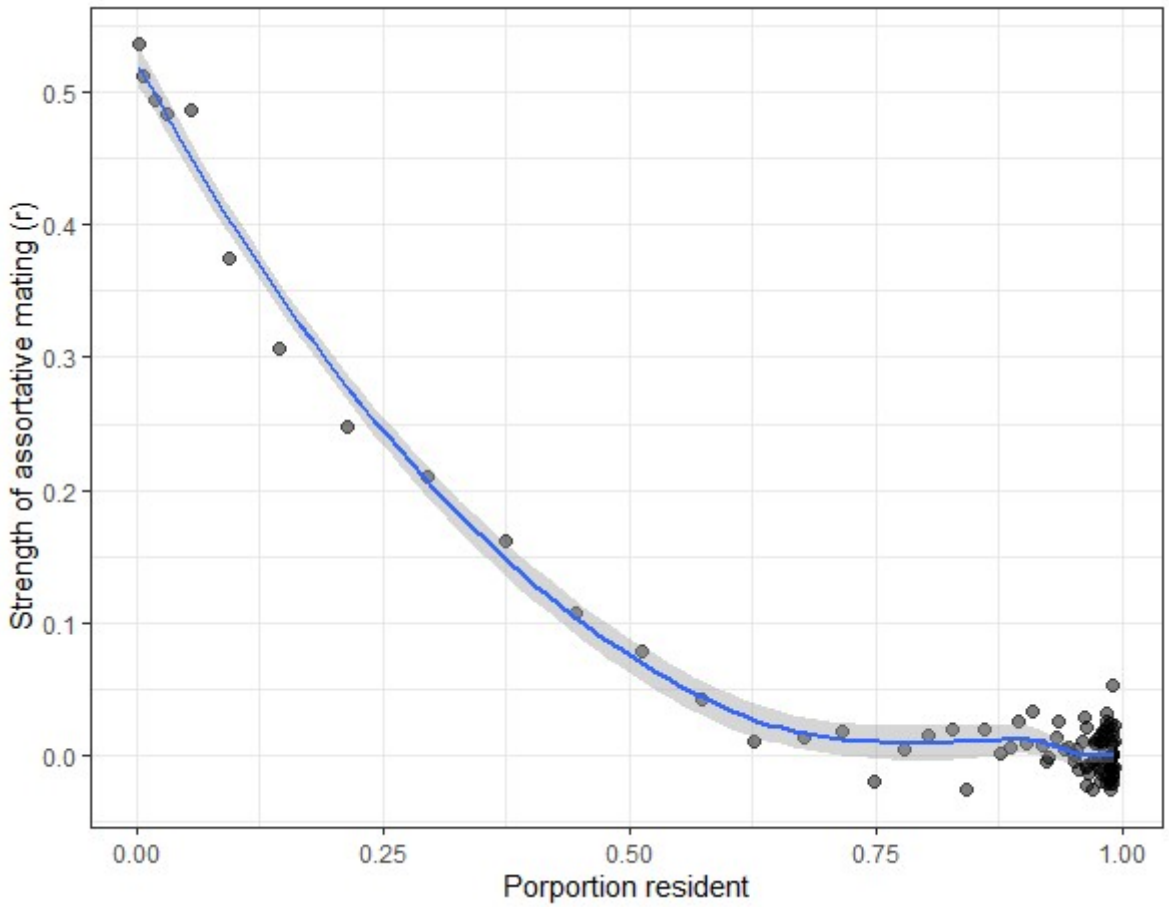

Figure S3: Relationship between the strength of assortative mating and the proportion of resident individuals in a population.

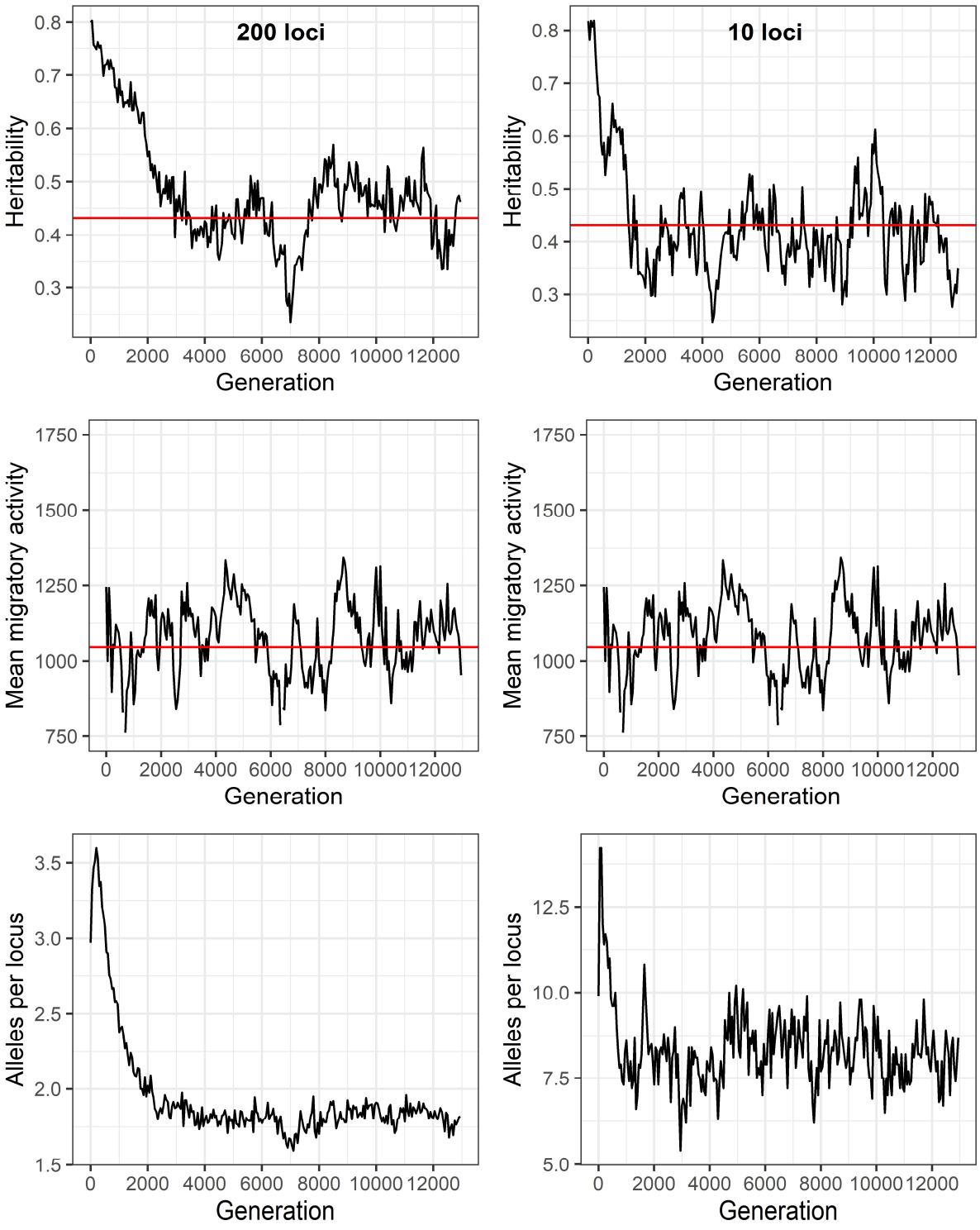

Figure S4 : Evolution of the heritability, mean migratory activity and number of alleles per locus under stabilising selection during the initialisation process for simulations with 10 or 200 loci.

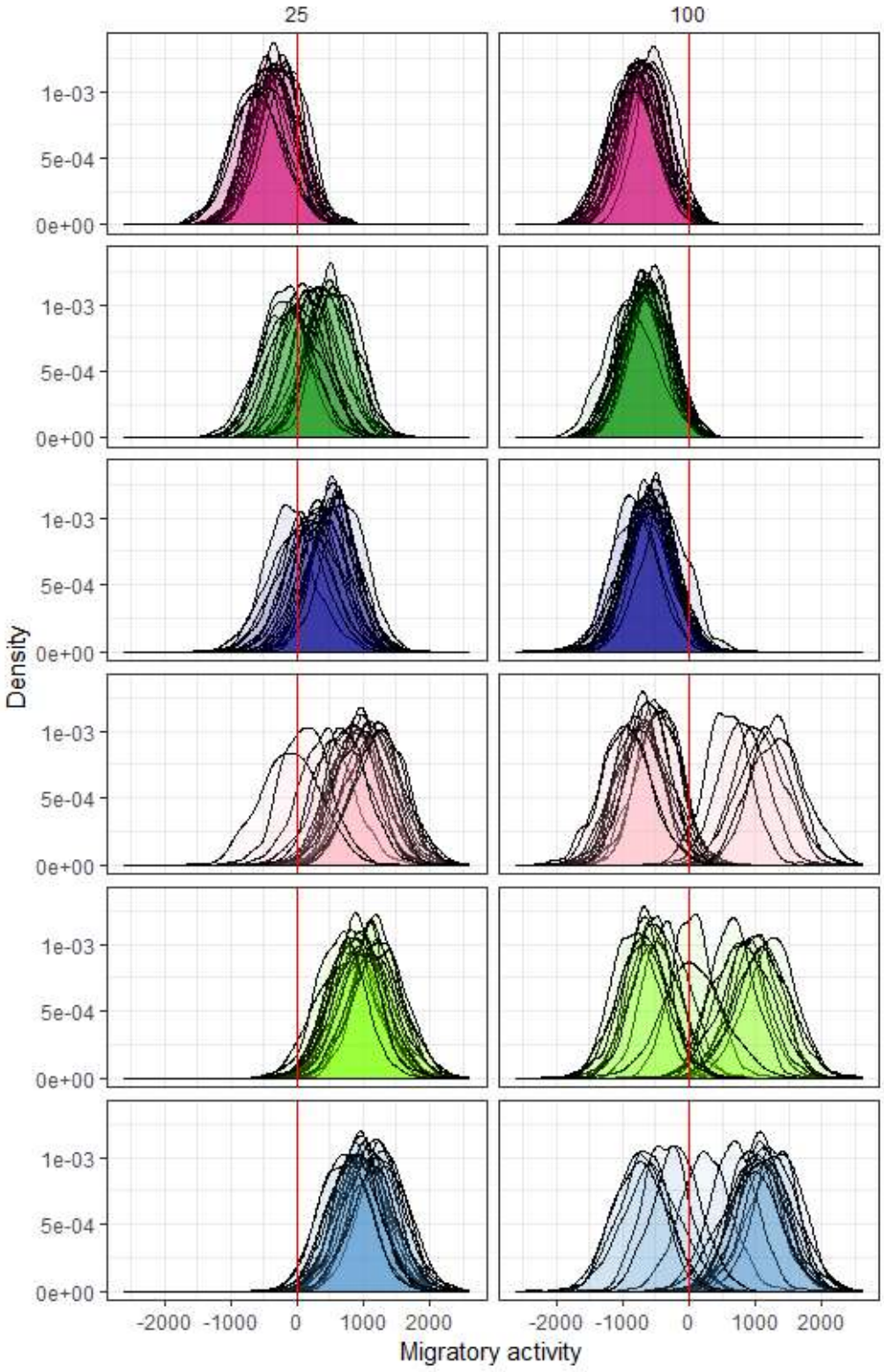

Figure S5: Smoothed density distribution for migratory activity under the 5 fitness functions after 25 and 100 generations of selection. Each curve represents the smoothed density of migratory activity in the population within a single simulation. The results from different fitness functions are distinguished by colour as shown in figure 1 of the main text.

Table S1: Input parameters used by the simulation model with the values used here to apply it to migratory behaviour in a blackcap population.

Parameter	Value
(i) Derived from previous studies	
Initial mean liability (μ_P)	1045*
Initial standard deviation in liability (σ_P)	376*
Initial heritability (h_0^2)	0.432
Number of offspring per mating (f)	5
Population size (N)	1000
Mutational heritability (h_m^2)	0.00184
(ii) Influence on simulation outcomes tested	
Genetic architecture	10 loci: σα = 86* and μ = 10^{-3} or 200 loci: σα = 61* and μ = 10^{-4}
Fitness function	Exponential: $0.5e^{-0.003\,l}$ Linear: $2e^{-4}l + 0.5$ Logistic: $0.5 - \frac{0.5}{1+e^{-0.01(l-1300)}}$ Discrete1: 0.160 Discrete2: 0.375 Discrete3: 0.465
Environmental effect on threshold positions	0, 100 or 300 *

*given in number of half-hour intervals with activity

Table S2: Mean fitness of migrants obtained with the exponential, linear and logistic fitness functions with either 10 or 200 loci compared to the values expected from their corresponding discrete functions. The observed values are summarised by the median and interquartile range based on the 20 simulations.

Fitness function	Expected	Observed with 200 loci	Observed with 10 loci
Exponential	0.160	0.161 (0.158 - 0.163)	0.166 (0.162 - 0.170)
Linear	0.375	0.375 (0.374 - 0.378)	0.377 (0.373 - 0.380)
Logistic	0.465	0.465 (0.463 - 0.467)	0.467 (0.464 - 0.468)

Table S3: Mean absolute error (MAE) and variance explained by cross validation (VECV) for the comparison of mean migratory activity predicted by simulations using different combinations of parameters and the data observed between 1988 and 2002 for the German blackcap population.

Fitness function	**Number of loci**	**MAE**	**VECV**
Exponential	10	65	80
Exponential	200	69	74
Linear	10	269	-151
Linear	200	293	-200
Logistic	10	242	-107
Logistic	200	262	-141
Discrete1	10	380	-423
Discrete1	200	397	-471
Discrete2	10	384	-434
Discrete2	200	400	-479
Discrete3	10	388	-446
Discrete3	200	401	-482

Table S4: Number of generations until populations become more than 5 or 95% resident and proportion of initial genetic variance maintained after 100 generations of selection for different fitness functions and number of loci. The values represent the median of 20 simulations and within brackets are the upper and lower 95% confidence intervals calculated through bootstrapping for 20000 iterations.

Fitness function	**Number of loci**	**Generations until > 5% resident**	**Generations until > 95% resident**	**Proportion of initial genetic variance**
Exponential	10	6 (5-8)	42 (36-46)	0.57 (0.35 – 0.73)
Exponential	200	8 (5-7)	37 (32-37)	1.00 (0.85 – 1.16)
Linear	10	16 (11-19)	73 (70-81)	0.58 (0.48 – 0.68)
Linear	200	23 (12-18)	68 (54-72)	0.99 (0.83 – 1.26)
Logistic	10	18 (12-22)	84 (76-94)	0.71 (0.53 – 0.80)
Logistic	200	24 (12 – 20)	79 (63-83)	1.09 (0.90 – 1.19)

Table S5: Description of the R objects used in the code that runs the simulation model.

Object name in R code	Description
start.a	Matrix with half of the allele effects for every individual in starting population
start.b	Matrix with other half of the allele effects for every individual in starting population
start.ma	Matrix with the migratory activity of every individual in starting population
mu.eff	Standard deviation of the distribution of effects of new mutations on the previous allelic value.
C	C parameter of folded normal distribution mating preference function
sd.ma	Standard deviation of genetic values for initial population
env.deviates	Environmental deviates per individual
fitness	Form of the fitness function i.e: linear, exponential or logistic)
gens	Number of generations to simulate
h2	Initial heritability
infl	Fitness function parameter
k	Fitness function parameter
mE	Mean of the distribution of environmental effects
mu	Mutation rate
mut.a	Mutation matrix for one gene copy per individual
mut.b	Mutation matrix for the other gene copy per individual
N	Population size
n.loci	Number of loci governing liability trait
offspring*	Matrix with genetic values per individual
offspring*	Matrix with liability values per individual
op	Number of offspring per pair
P.a	Matrix with a gene copy per individual
P.b	Matrix with the other gene copy per individual
s	s parameter of folded normal distribution mating preference function
sdE	Standard deviation of the distribution of allele effects
slope	Fitness function parameter
w.mean.m	Fitness function parameter
t	Fitness function parameter
Tr	Position of the threshold
Tr.mean	Mean position of the threshold

`Tr.sd`	Standard deviation in position of the threshold
VA	Initial additive genetic variance
VP	Initial phenotypic variance
`w`	Relative fitness

*In the R code, `offspring` refers to the genetic values *and then* to the phenotypic values after adding the environmental deviates.

R CODE FOR SIMULATION MODEL

The genetic simulation model is implemented as an R function [4] called threshold_model_endpoint_seq. This function takes data from a starting population and several model parameters, runs the desired number of simulations and saves the data for further analysis. To make the code more readable, Table S5 (above) gives a short description of the key R objects within it.

```
threshold_model_endpoint_seq<-function(
          name,  # name to give to simulation
          simulation, # number of simulations
          save.every, # number of generations after which data is saved be
fore continuing simulation
          N, # population size
         start.a, # allele effects matrix for intial population
         start.b, # other allele effects matrix for initial population
         start.ma, # matrix of individual migratory activity values of ini
tial population
         mE, # Mean of the distribution of environmental effects
         n.loci,  # number of loci
         w.mean.ma,  # fitness function parameter
         w2, # fitness function parameter
         sd.ma, # Standard deviation of genetic values for initial populat
ion
         Tr.mean, # Average threshold value
         Tr.sd,  # Standard deviation in threshold values
         gen,  # number of generations to run
         mu,   # mutation probability
         mu.eff, # Standard deviation of the distribution of effects of ne
w mutations
         h2, # inital heritability
         fitness, # fitness function parameter
         t,  # fitness function parameter
         slope, # fitness function parameter
         lambda,  # fitness function parameter
         k,  # fitness function parameter
         infl,  # fitness function parameter
         wmax,  # fitness function parameter
         wres, # fitness function parameter
         assort, # weather to simulate assortative mating
         C, #  C parameter of FND assortative mating function
         s, # s parameter of FND assortaative mating function
        op # number of offspring per pairing
){
  require(reshape2)
  require(plyr)

 ###### Calculate (initial) genetic and (constant) environmental variance

  VP<-sd.ma^2
```

```r
 VA<-h2*VP
 sdE<-sqrt((VA/h2)-VA)
 npairs<-N/op

###### Creating fitness function

 if(fitness=="linear"){
   wfunc<-function(ma,slope,t,Tr){
     if(ma<Tr){
       w<-wres
     }else{
       w<-t-slope*(ma-Tr)
       if(w<0){
         w<-0.0001
       }
     }
     return(w)
   }

 } else if(fitness=="ex"){
   wfunc<-function(ma,lambda,Tr,c){
     if(ma<Tr){
       w<-wres
     }else{
       w<-c*(lambda*exp(-lambda*(ma)))
       if(w<0){
         w<-0.0001
       }
     }
     return(w)
   }
   c<-t/lambda

 } else if(fitness=="logistic"){
   wfunc<-function(ma,k,infl,Tr,wmax,wres){
     if(ma<Tr){
       w<-wres
     }else{
       w<-wmax-(wmax/(1+exp(-k*(ma-infl))))
     }
     if(w<0){
       w<-0.0001
     }
     return(w)
   }
 }else if(fitness=="stabilizing"){
   wfunc<-function(ma,w2,w.mean.ma){
     w <- exp((-(ma-w.mean.ma)^2)/(2*w2))
     return(w)
   }
 }

###### Creating mating function
```

```
  mating<-function(parent,a,b,n){
    sub.a<-sample(x = c(T,F),size = n,replace = T)
    sub.a<-rbind(sub.a,!sub.a)

    sub.b<-sample(x = c(T,F),size = n,replace = T)
    sub.b<-rbind(sub.b,!sub.b)

    aa<-rbind(a[parent[1],],b[parent[1],])[sub.a]
    bb<-rbind(a[parent[2],],b[parent[2],])[sub.b]
    return(rbind(aa,bb))
  }

###### Running for multiple generations

  all.a<-all.b <-list(length=save.every)
  all.a[[1]]<-start.a
  all.b[[1]]<-start.b

  ma<- Trm<-matrix(nrow = N,ncol=save.every)

  ma[,1] <- start.ma
  Trm[,1] <- Tr.mean
  assort.plots <- NULL
  r <- NULL
  save.points<-seq(save.every,gen,save.every)
  n.save.points<-gen/save.every

# determine maximum difference in migratory activity between 2 individuals
in parental generation, this will be used as Dmax in the following generat
ions of assortative mating

  if(isTRUE(assort)){

# Folded Normal distribution function for the calculation of mating probab
ilities

    FND <- function(D,Dmax,C,s){
      p <- exp(( (-C^2) * (D^2) ) / ( (s^2) * (Dmax^2) ))
      return(p)
    }

 # Compute matrix of individual pairwise differences in migratory activity
(NB residents with -ma must be set to have ma = 0)
    ma.pos <- ma
    ma.pos[which(ma<0)] <- 0
    Pdist <- as.matrix(abs(dist(ma.pos[,1])))

# Dmax is the maxmimum possible absolute phneotypic diference between two
individuals.

    Dmax <- max(Pdist)
  }

# -------------- START OF LOOP FOR MULTIPLE GENERATIONS ------ #
```

```
  for(sv in 1:n.save.points){
  for(g in 2:(save.every+1)){

 # Determine individual fitness

        Tr <- round(rnorm(N,Tr.mean,Tr.sd))  # environemental variance in
threshold
        if(fitness=="linear"){
          w<-mapply(wfunc,ma[,g-1],Tr=Tr,slope=slope,t=t)
        }else if(fitness=="ex"){
          w<-mapply(wfunc,ma[,g-1],Tr=Tr,lambda=lambda,c=c)
        } else if(fitness=="logistic"){
          w<-mapply(wfunc,ma[,g-1],Tr=Tr,k=k,infl=infl,wmax=wmax,wres=wres
)
        }else if(fitness=="stabilizing"){
          w<-mapply(wfunc,ma[,g-1],w2=w2,w.mean.ma=w.mean.ma)
        }
          w<-w/max(w)

# Alleles in next generation are sampled from those of previous generation
with probability given by fitness w

      selected<-sample(x = 1:N,size = 2*npairs,replace = F,prob = w)

      if(isTRUE(assort)){
        ma.pos <- ma
        ma.pos[which(ma<0)] <- 0
        Pdist <- as.matrix(abs(dist(ma.pos[selected,g-1])))

# Calculate mating probabilities using FND function
        matprobs <- FND(D=Pdist,Dmax,C=C,s=s)
# Set diagonal to 0 such that self mating does not occur
        diag(matprobs) <- 0

# remaining is a vector with the selected individuals from which to iterat
ively remove the individuals
# that have already been paired.
        remaining <- selected
        P1<-NULL # vector contining ids of parent 1
        P2<-NULL # corresponding vector containing ids of parent 2
        for(i in 1:(npairs)){
# 1 Parent is randomly chosen from the "selected" population
          P1[i] <- sample(remaining,1)
# The parent with which this individual will mate is sampled from the sele
cted population using the mating probabilities corresponding to this indiv
idual and all its potential pairs
          P2 [i] <- sample(x = selected ,size = 1,replace = F,prob = matpr
obs[,which(selected==P1[i])])
# Paired individuals are removed from the remaining vector to ensure sampl
ing without replacement
          remaining <- remaining[-c(which(remaining==P1[i]),which(remainin
g==P2[i]))]
# The probability of mating with an individual that has already paired = 0
```

```
          matprobs[,c(which(selected==P1[i]),which(selected==P2[i]))] <- m
atprobs[c(which(selected==P1[i]),which(selected==P2[i])),] <- 0
        }

# Organise pairs into a list which facilitates use in mating function bell
ow
        pairs <- cbind(P1,P2)
        pairs<-rep(split(pairs, 1:npairs),each=op)


        pairs<-cbind(selected[1:npairs],selected[ (npairs+1):(2*npairs)])
        pairs<-split(pairs, rep(1:nrow(pairs), each = ncol(pairs)))
        pairs<-pairs[sample(x = rep(1:npairs,each=op),size = N,replace = F
)]

        alleles<-lapply(pairs,mating,a=all.a[[g-1]],b=all.b[[g-1]],n=n.loc
i)
        alleles<-do.call("rbind",alleles)
        a<-alleles.a<-alleles[c(TRUE,FALSE),] # get odd rows. these are a
alelleles
        b<-alleles.b<-alleles[c(FALSE,TRUE),] # get even rows. these are b
alleles

######  MUTATION

# Determine which alleles will undergo mutation:
      mut.a<-matrix(data = sample(x = c(1,0),size = N*n.loci,replace = T,p
rob = c(mu,1-mu)),nrow = N,ncol = n.loci)
      mut.b<-matrix(data = sample(x = c(1,0),size = N*n.loci,replace = T,p
rob = c(mu,1-mu)),nrow = N,ncol = n.loci)

# Mutate those alleles by sampling from allele effects distribution
      while(any(a[mut.a==1]==alleles.a[mut.a==1])){ # while ensures no mut
ation to same value
        a[mut.a==1]<- a[mut.a==1] + round(rnorm(sum(mut.a),0,mu.eff))
        b[mut.b==1]<- b[mut.b==1] + round(rnorm(sum(mut.b),0,mu.eff))
      }

###### GENOTYPE -> PHENOTYPE

      offspring<-apply(a+b,1,sum) # the value of each offspring is the sum
of alleles a and b summed over all loci
      env.deviates<-rnorm(n = N,mean = mE,sd = sdE)
      offspring<-offspring+env.deviates

# If the generation at which a file should be saved has been reached, save
file and assign current the data of current generation as the first elemen
t of the follwing one
      if(g==save.every+1){
        print(i)
        saving.name<-paste0(save.points[sv]-save.every,"_",save.points[sv]
,"_","simulation","_",simulation,"_",name,".rda")
        print(saving.name)
        saveRDS(object = list(ma=ma,allele.a=all.a,allele.b=all.b,threshol
```

```
d=Trm),
                file=saving.name)

# the first element of every object becomes the current one (i.e save gen+
1 becomes g==1 of next file)

        all.a[[1]]<-a
        all.b[[1]]<-b
        ma[,1]<-offspring
        Trm[,1]<-Tr

      }else{

# Otherwise  save the data and proceed to the next generation

        all.a[[g]]<-a
        all.b[[g]]<-b
        ma[,g]<-offspring
        Trm[,g] <- Tr
        print(g)
      }
    }
  }
}
}
```