Ecography

ECOG-01977
Cobben, M. M. P. and van Noordwijk, A. J. 2016. Stable partial migration under a genetic threshold model of migratory behaviour. – Ecography doi: 10.1111/ecog.01977

Supplementary material

Appendix 1: Additional figures

In this appendix we show some additional supporting figures. In Figure A1 we show values of Equation 1 (A1a) and Equation 2 (A1b) as a function of *x*-location and for different values of the population density. Figure A2 indicates the fraction of migratory individuals at the population level in the area of the landscape around the zone of partial migration. Most interestingly it can be seen that in the scenario without dispersal (A1e) all populations are fixed for either phenotype, while all other scenarios show true partial migration, with individuals of both phenotypes in all populations. In Figure A3 we show the threshold values, averaged per *x*-position, for each of the 100 runs for all scenarios. This gives a good indication of the variation between runs. Here we can clearly see the effect of dispersal, where large dispersal distance (A2c) changes the overall shape of the threshold value curves and no dispersal (A2e) causes genetic fixation in all populations. In Figure A4 we further explore the effects of density dependence in winter mortality on the width of the partial migration zone and its interaction with dispersal and the density dependence in reproduction.

Figure A1a. The actual values of resident winter survival s_r (Equation 1) as a function of x-location and different values of $\frac{Nr_{x,y,t}}{K}$, i.e. population density (dens), for parameter c_dens_s = 0.6 (as used in the base scenario). In addition s_r for c_dens_s = 1, i.e. without density dependence.

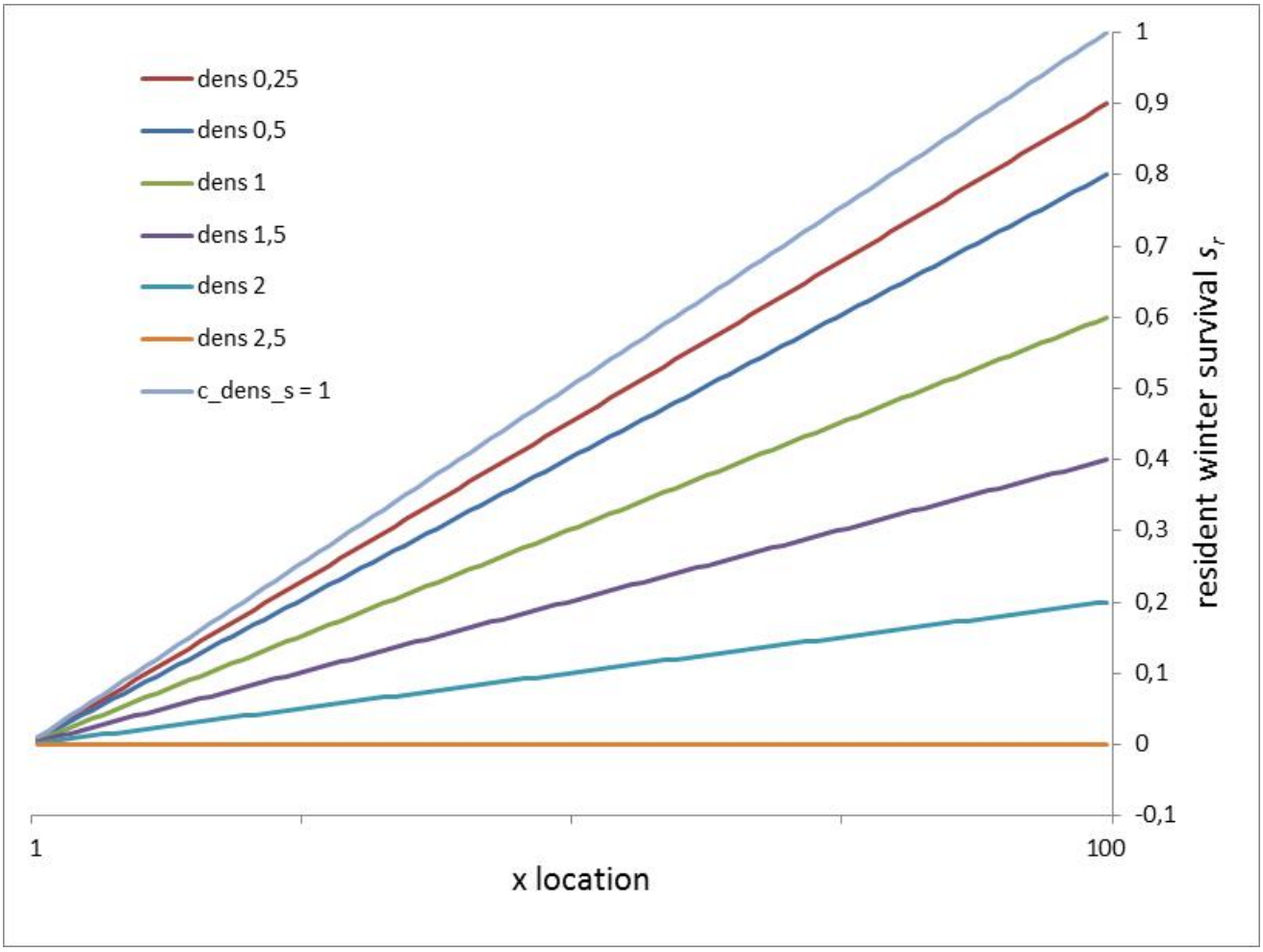


Figure A1b. The reproduction correction factor c_{tot} (Equation 2) as a function of x-location and different values of $\frac{Nr_{x,y,t}}{K}$, i.e. population density (dens), for $c\_dens_r = 0.4$, and $c\_loc_r = 0.8$ (both as used in the base scenario).

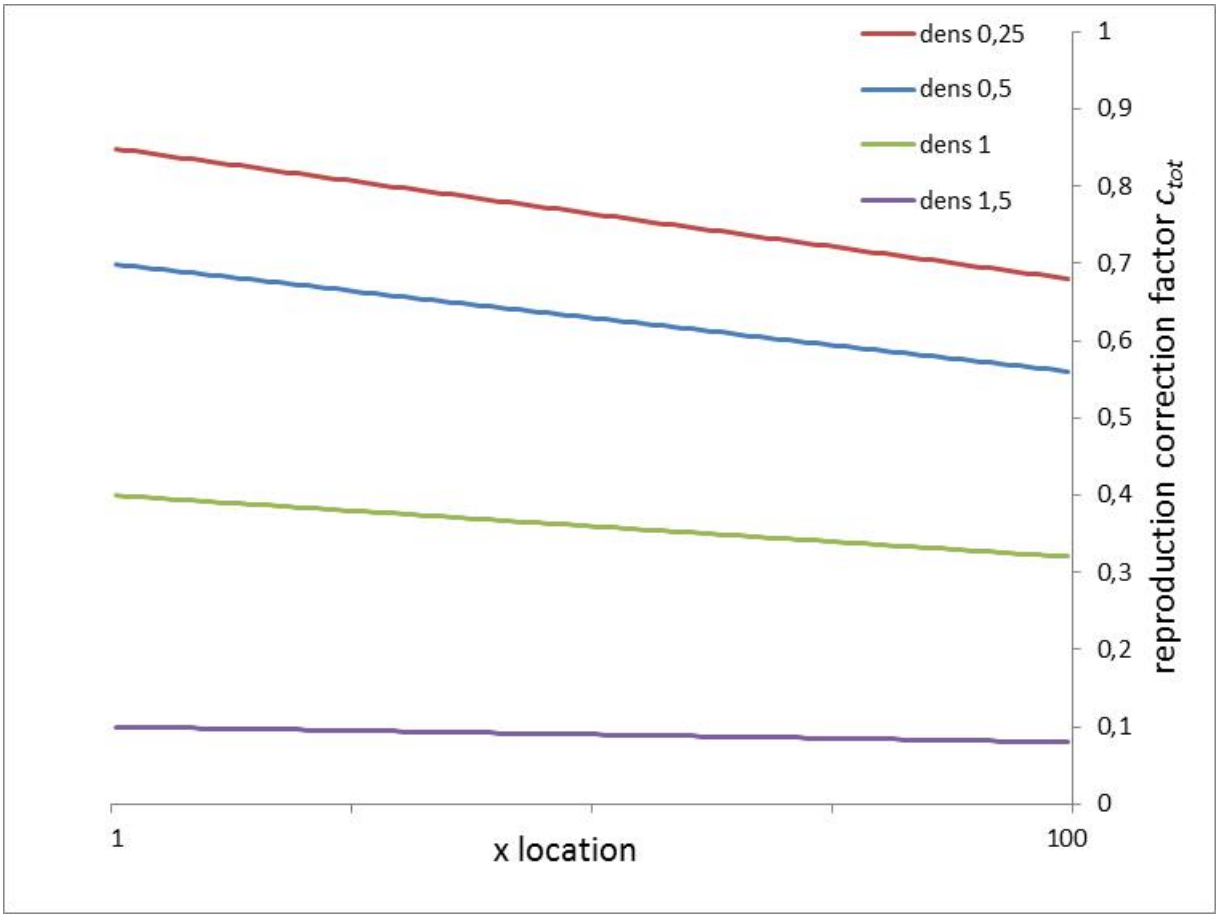


Figure A2. The fraction of migratory individuals in each of the populations at *x*-positions larger than 40 and smaller than 80 in generation 1000 of run 50 of each scenario. The omitted populations all have a fraction of migrants of 1 (for *x*-positions smaller than 40) or 0 (for *x*-positions larger than 80). A white cell indicates an empty (extinct) population.

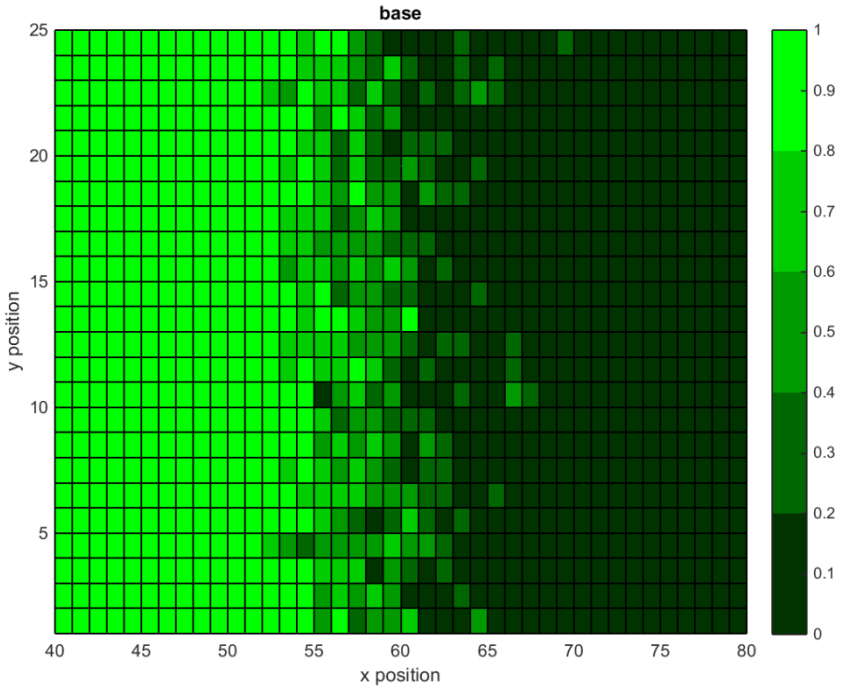


Figure A2a. *Base* scenario

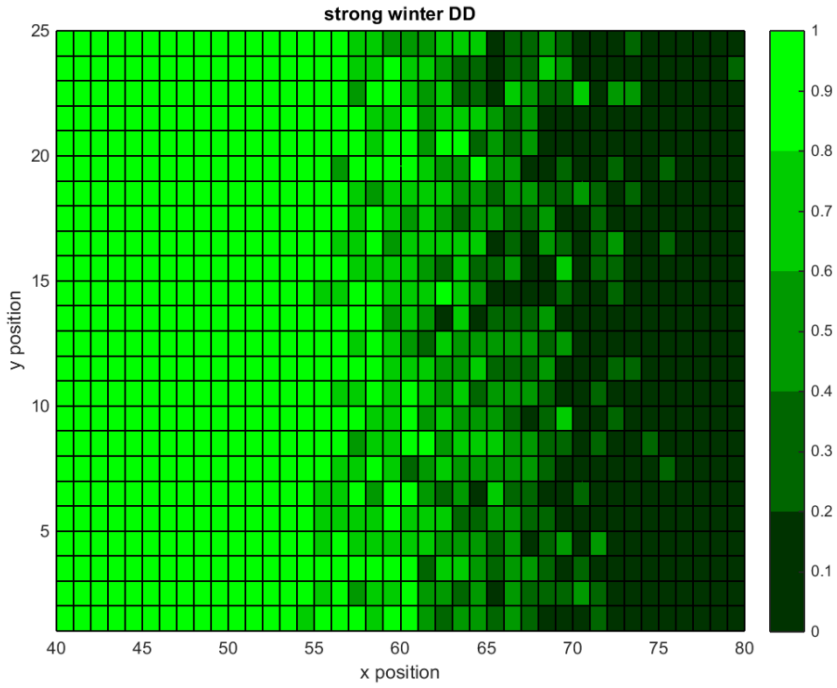


Figure A2b. *Strong winter DD* scenario

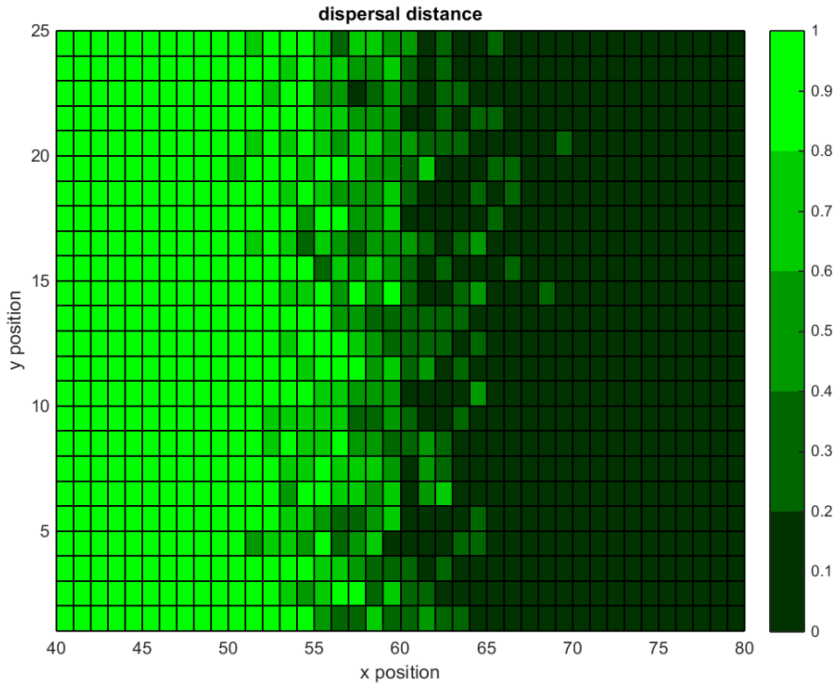


Figure A2c. *Dispersal distance* scenario

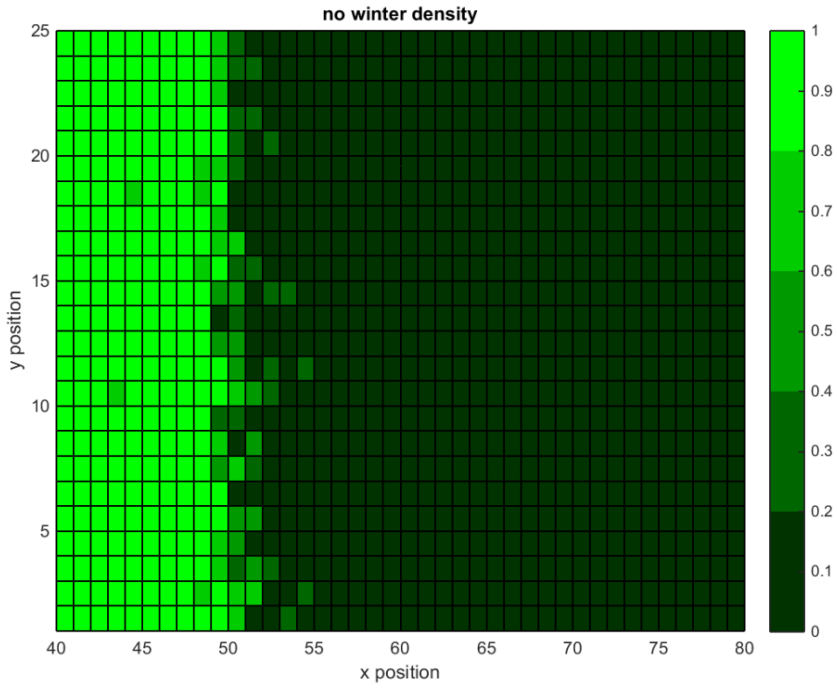


Figure A2d. *No winter DD* scenario

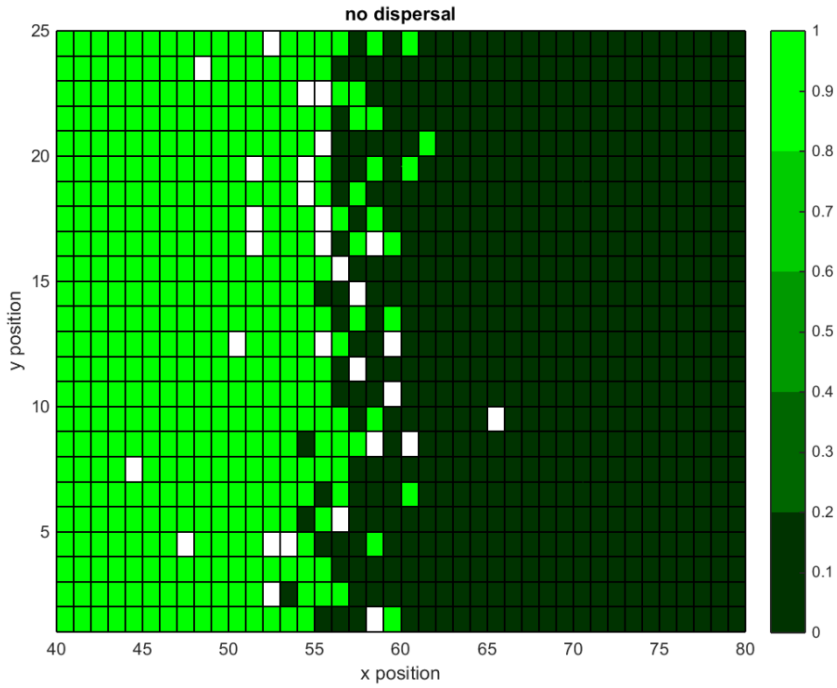


Figure A2e. *No dispersal* scenario

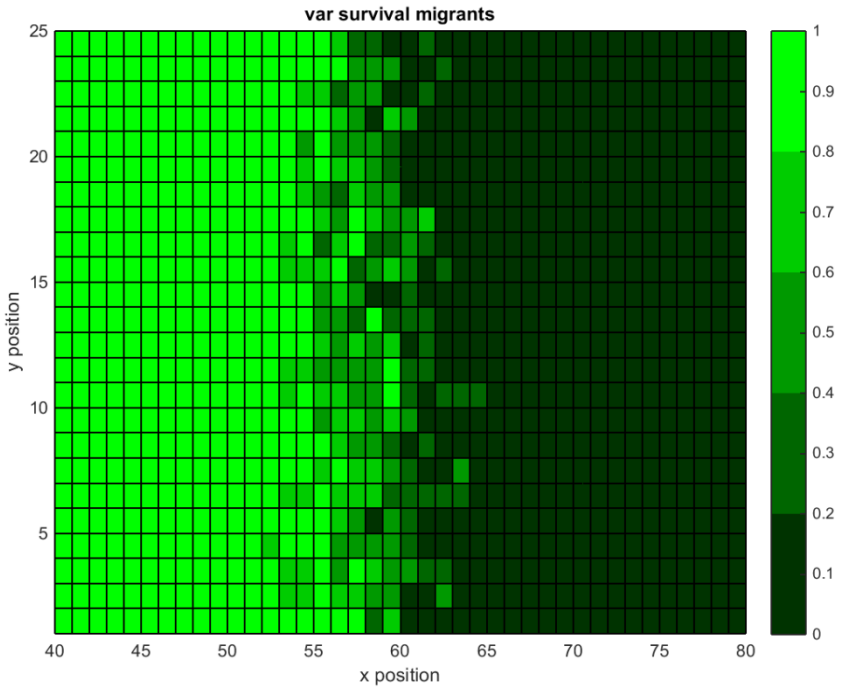


Figure A2f. *Var survival migrants* scenario

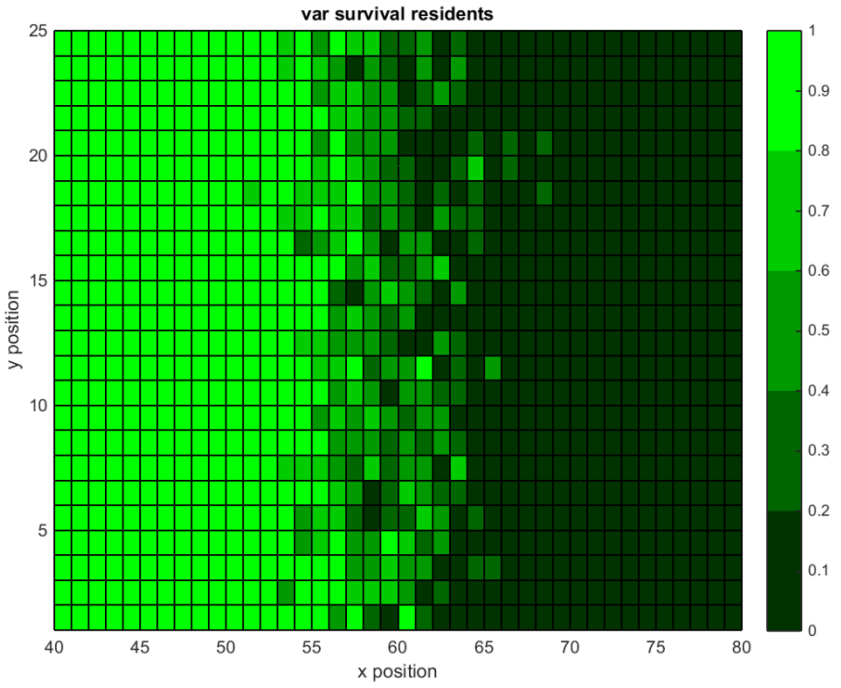


Figure A2g. *Var survival residents* scenario

Figure A3. The average threshold values per *x*-location for each of the 100 runs per scenario.

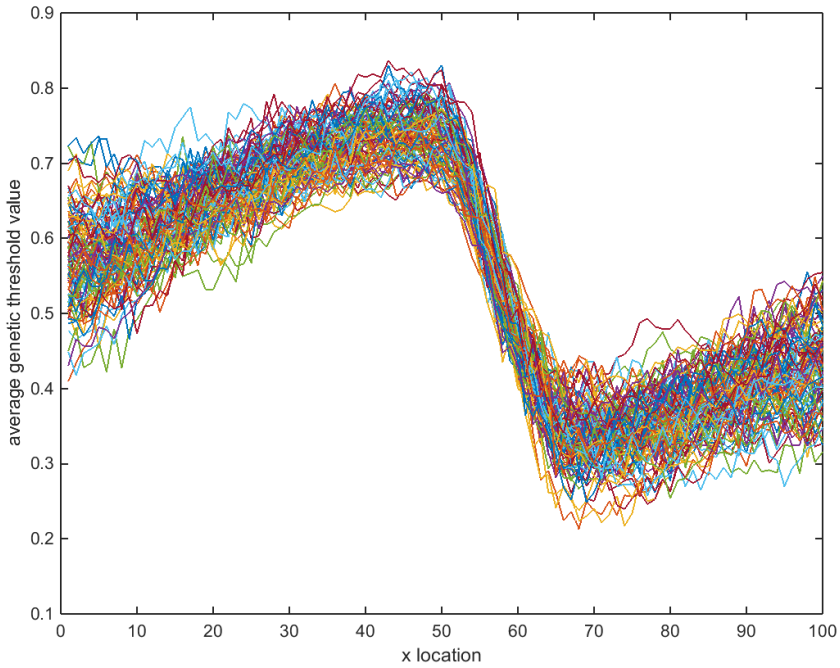


Figure A3a. *Base* scenario

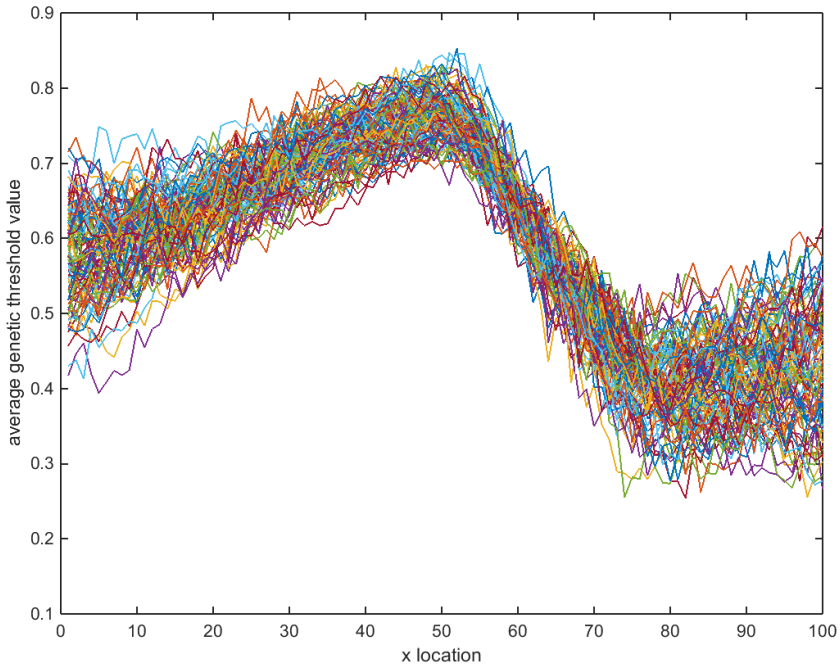


Figure A3b. *Strong winter DD* scenario

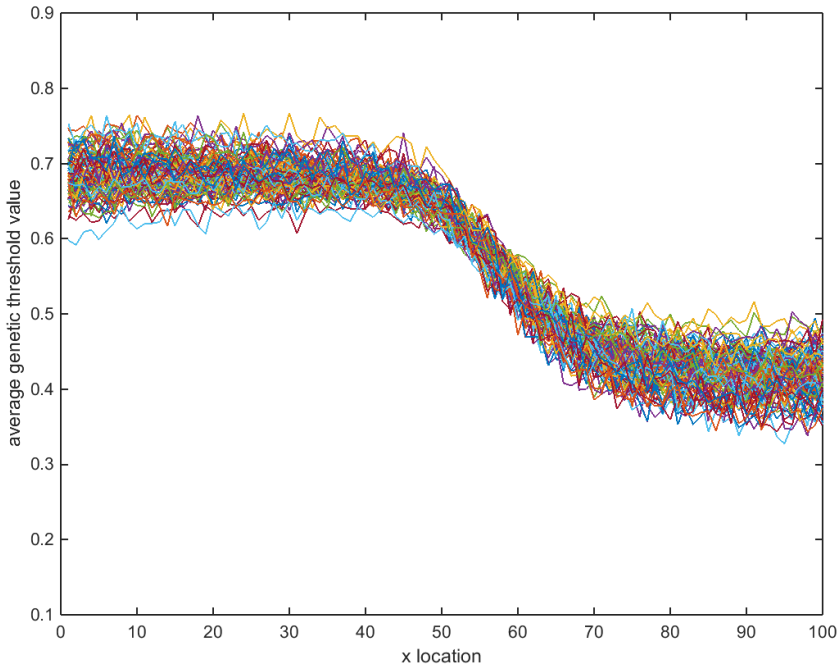


Figure A3c. *Dispersal distance* scenario

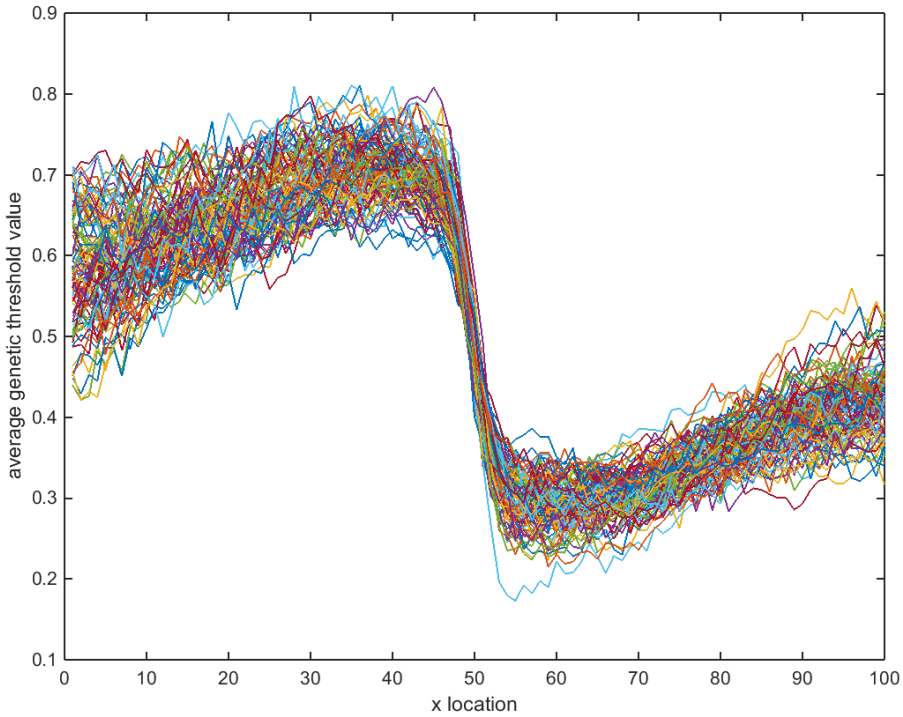


Figure A3d. *No winter DD* scenario

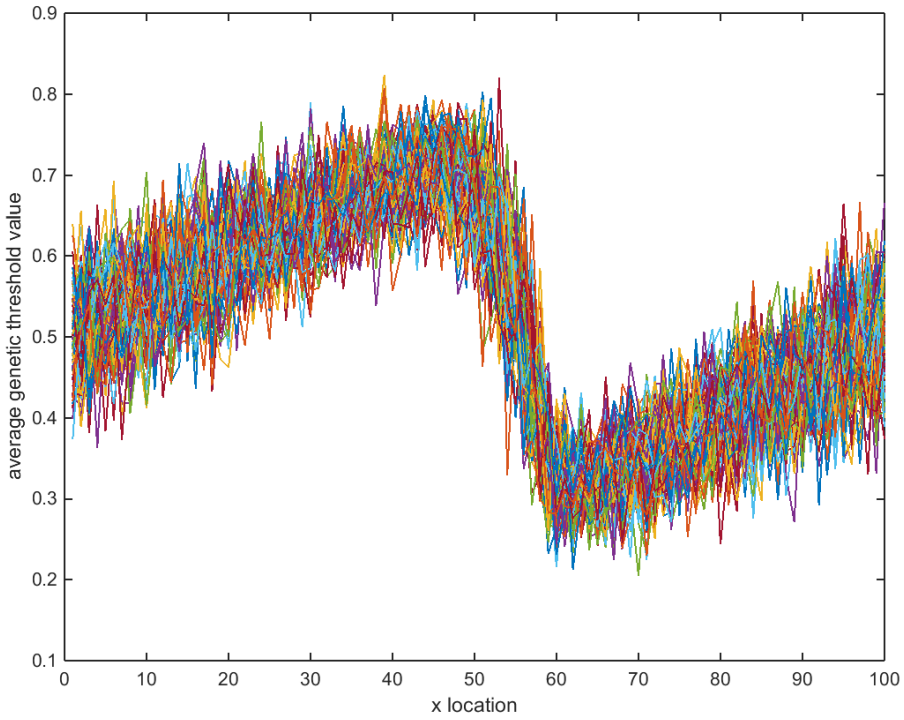


Figure A3e. *No dispersal* scenario

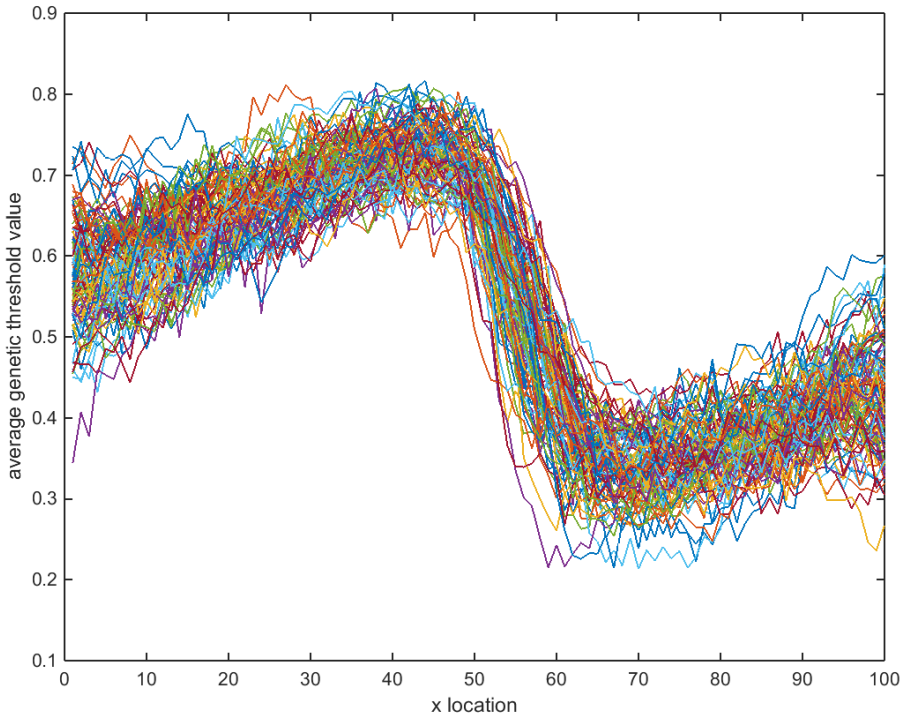


Figure A3f. *Var survival migrants* scenario

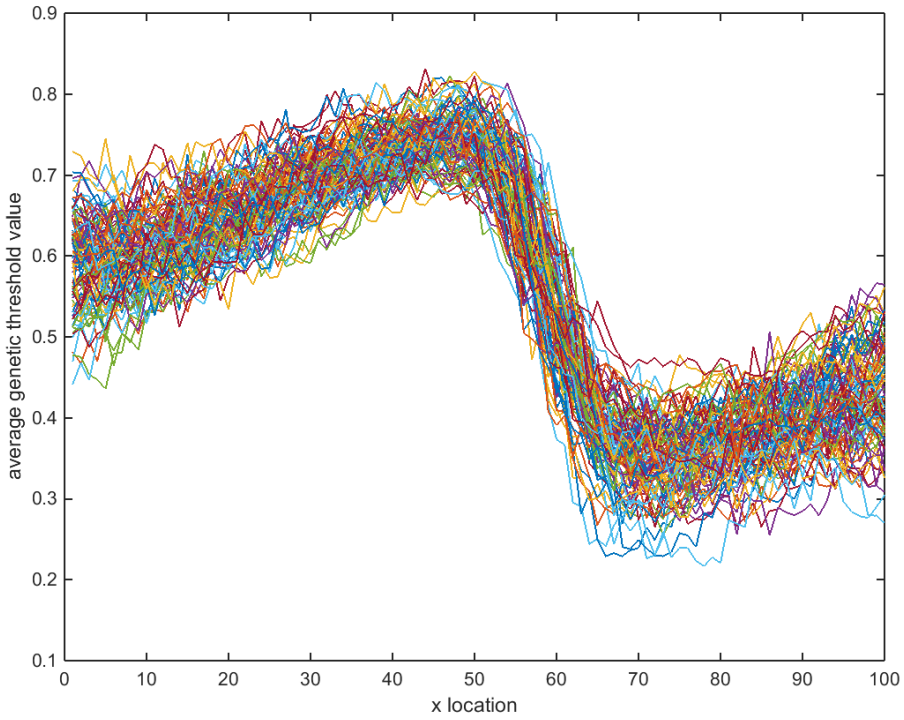
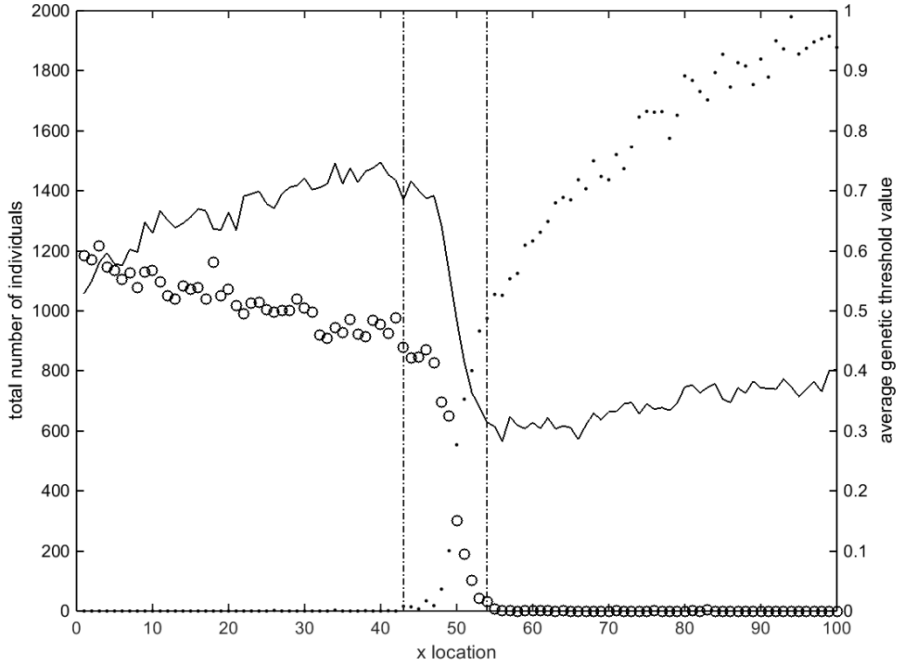
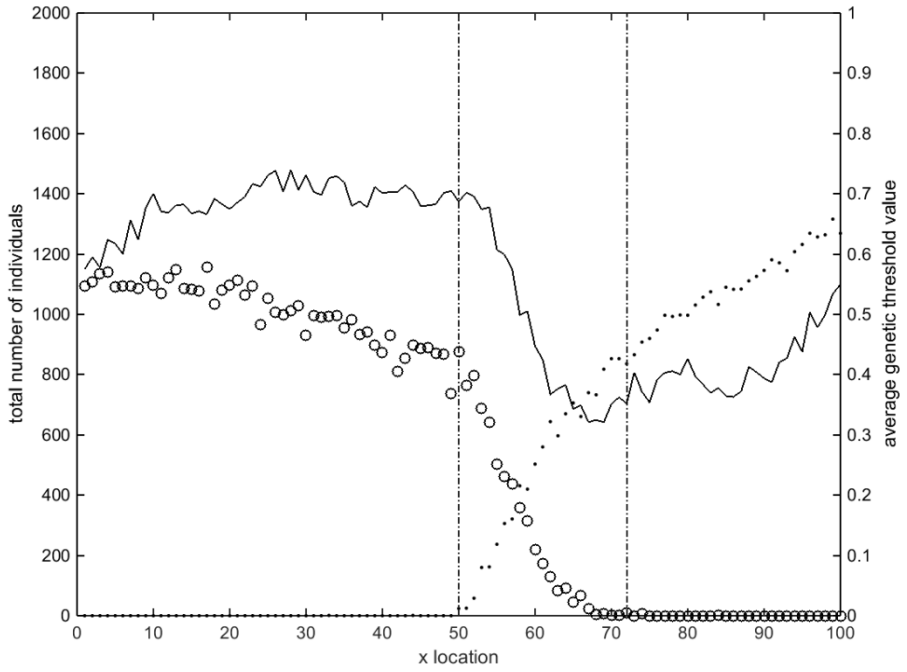


Figure A3g. *Var survival residents* scenario

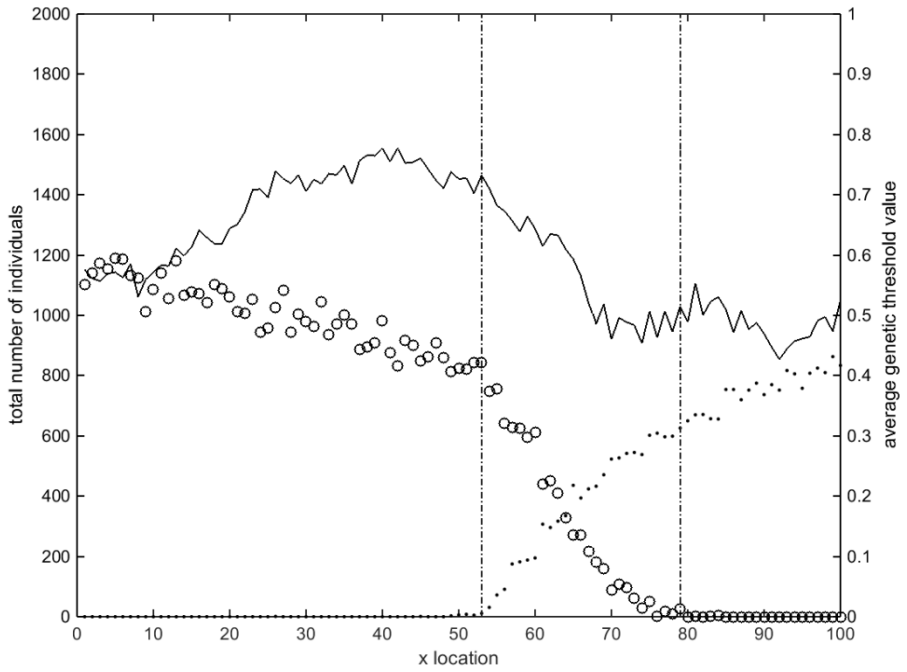
Figure A4. The number of residents (black dots • and left *y*-axis) and the number of migrants (open circles ○ and left *y*-axis) after the winter after 1,000 generations, across the landscape summed per *x*-location (*x*-axis). The black vertical lines (· –) indicate the *x*-position of the partial migration zone as defined in the method section. The solid black line (–) is the average genetic threshold value per *x*-location (right *y*-axis) for a. the *no winter DD* scenario, with c_dens_s is 1, b. the base scenario with c_dens_s is 0.6, c. the *strong winter DD* scenario with c_dens_s is 0.3, d. as in c but with additional strong density dependence in reproduction of c_dens_r is 0.1, and e. as in d but without dispersal, all for a single run.



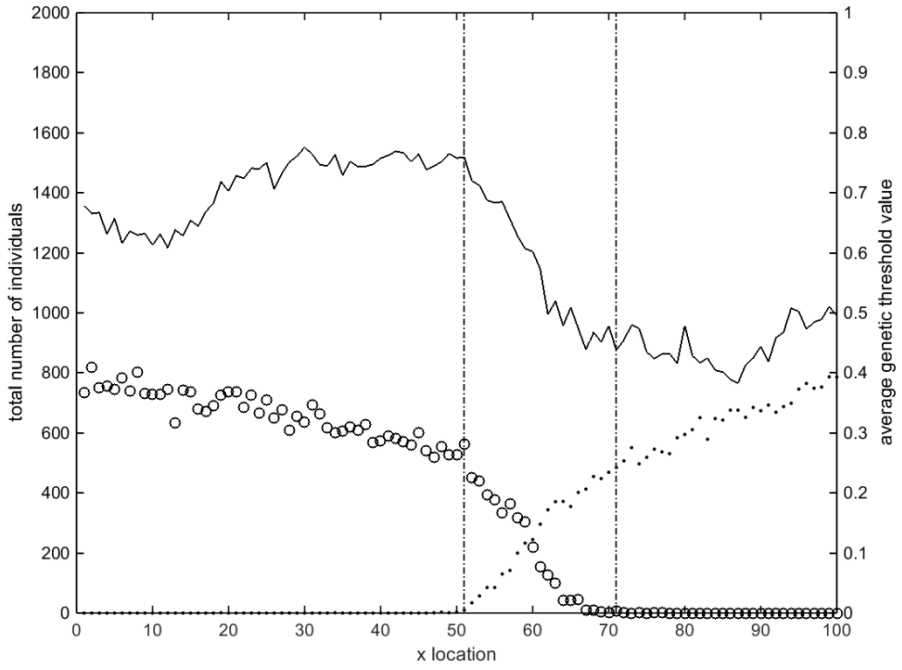
a.


total number of individuals
average genetic threshold value
x location
0
200
400
600
800
1000
1200
1400
1600
1800
2000
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10
20
30
40
50
60
70
80
90
100

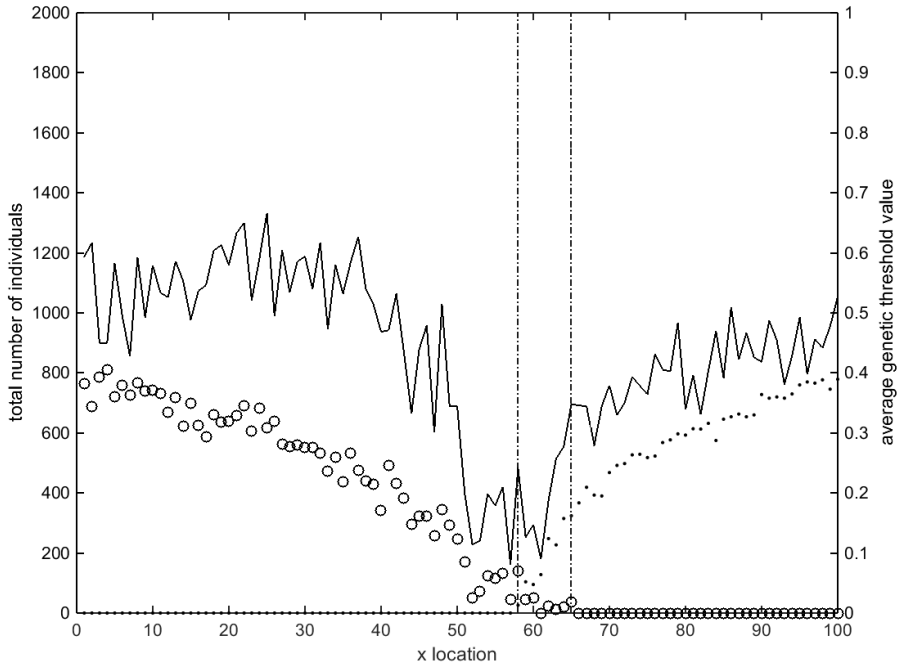
b.


total number of individuals
average genetic threshold value
x location
0
200
400
600
800
1000
1200
1400
1600
1800
2000
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10
20
30
40
50
60
70
80
90
100

c.


total number of individuals
average genetic threshold value
x location
0
200
400
600
800
1000
1200
1400
1600
1800
2000
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70
80
90
100

d.


total number of individuals
average genetic threshold value
x location
0
200
400
600
800
1000
1200
1400
1600
1800
2000
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70
80
90
100

e.

Appendix 2: Matlab program sources

Main program

```
clear all;
close all;
clc;
dbstop if error;
location='P:\Marleen\Migration\matlab'; % where the .m file is stored

%% location and scenarios input

basefolder='P:\Marleen\Migration\matlab\output\rep_runs\'; % where the
different scenario folders are stored
scens=dir([basefolder 'scen_*']); % reads all the scenario folders

for scen=1:length(scens)
    datafolder=([basefolder scens(scen,1).name '\']);

    %% load parameters
    cd(datafolder);
    par_file=['par_' scens(scen,1).name '.m']; % each scenario folder contains
a parameter file that starts with 'par_' and then the scenario name, identical
to the folder name
    run(par_file); % reads the spedific parameters
    cd(location);

    %% make base deme matrix / format to use 'reshape'
    % and make nearest neighbor matrix
    deme_count_base=repmat(ones(y_max,1),1,x_max);
deme_count_base=cumsum(deme_count_base,2);
deme_count_base=reshape(deme_count_base,y_max*x_max,1);
    deme_count_base(:,2)=repmat(cumsum(ones(y_max,1)),x_max,1);
deme_count_base(:,3)=zeros(y_max*x_max,1);
    A=deme_count_base;
    A(:,A(3,:)==0)=[];
    B=A;
    idNN=rangesearch(A,B,disp_dist,'Distance','chebychev');
    all_NN=cellfun('length',idNN);
    NN_mat=zeros(x_max*y_max,max(all_NN));
    for i=1:x_max*y_max
        NN_mat(i,1:all_NN(i))=idNN{i};
    end

    %% loop over runs
    for run=1:100

        %% initialisation
        ind=zeros(N_ind,6);      % [1.inherited threshold value, 2.x_loc deme,
3. y_loc deme, 4. migration, 5. nr of offspring, 6. dispersal]
        ind(:,1)=rand(N_ind,1); % initialisation with random allele values
        ind(:,2)=ceil(rand(N_ind,1)*x_max); % initialisation with random x
location
```

```
        ind(:,3)=ceil(rand(N_ind,1)*y_max); % initialisation with random y
location
        %% loop over years
        for year=1:1000
            if mod(year,100)==0 % to keep track of progress during running
                scen
                run
                year
            end
        ind_w=zeros(N_ind,6);    % initiating the winter survival matrix

        %% winter survival
        % first separate migrants from residents
        migr_prob = std_migr * randn + (ind(:,3)/100); %throughout the code:
no for-loops over individuals, but calculations covering the whole matrix at
once through indexing
        index1=ind(:,1) > migr_prob; %then migration
        ind(index1,4)=1; % to indicate whether the individual migrated

        [deme, ~, deme_m,
~]=get_deme_info(y_max,x_max,ind,deme_count_base,'deme_m_only');  % goes to
get_deme_info function to sort this information into populations at all xy
combinations

        index2=rand(N_ind,1) < (std_s_migr *randn + s_migr); % applying
migration survival to ALL individuals
        index=index1+index2; % combines surviving individuals with migrating
individuals
        index=index==2; clear  index2 % selects the individuals that have
survived migration
        ind_w(index,1:4)=ind(index,1:4); % and puts them in the ind_w matrix

        index1=index1==0; % these are the inds that did not migrate
        res_dens=(deme-deme_m)/K; % calculating local population density

        surv_prob = std_surv *randn + (ind(:,3)/100); % survival prob based on
a random number from a normal distribution with average = y-loc and a non-
variable std, so modelling bad and good years rather than individual
differences
        index2=rand(N_ind,1) < (surv_prob.*(1-rc_s*res_dens(y_max*(ind(:,2)-
1)+ind(:,3)))); % survival 0/1 based on y-location, corrected for local
resident density for ALL individuals
        index=index1+index2;  % combining survival and residency
        index=index==2; clear index1 index2; % selecting resident individuals
that have survived winter
        ind_w(index,1:3)=ind(index,1:3); clear index % and puts them in the
ind_w matrix too

        ind_w(ind_w(:,1)==0,:)=[]; % all individuals that have survived
winter, regardless of migratory behaviour

        %% output %% after winter survival
        if
year==1||year==50||year==100||year==200||year==300||year==400||year==500||year
==600||year==700||year==800||year==900||year==1000
            ind=ind_w;
```

```
            resultfile=[datafolder scens(scen,1).name '_run' num2str(run)
'_yr' num2str(year)];
            save(resultfile,
'ind','disp_rate','disp_dist','m_rate','s_factor','std_surv','std_s_migr');
        end

        %% dispersal

        N_ind_w=size(ind_w,1);
        ind_d=ind_w; clear ind_w % initiating dispersal matrix
        index1=rand(N_ind_w,1) < disp_rate; % selecting dispersing individuals
        ind_d(index1,6)=1; % to indicate whether the individual dispersed
        index=find(index1); % index of the dispersing individuals
        column=ind_d(index,2); % x-loc of the dispersing individuals
        row=ind_d(index,3); % y-loc of the dispersing individuals
        patch_number=sub2ind(size(deme),row,column); % patch number of the
dispersing individuals = row in NN_mat

        nr_NN=cellfun('length',idNN(y_max*(ind_d(:,2)-1)+ind_d(:,3)))-1; %
number of possible destination cells, -1 to compensate for the same xy
position
        index2=ceil(rand(N_ind_w,1).*nr_NN)+1;   % selecting a destination
cell (for ALL individuals). the first is the source population, so this should
be position 2 and further in idNN
        NN_mat_column=index2(index); % combining the numbers of the
destination cells and dispersing individuals
        ind_d_new=ind_d;

patch_column=A(NN_mat(sub2ind(size(NN_mat),patch_number,NN_mat_column)),1); %
for all dispersing individuals determine destination x from the destination
cell number

patch_row=A(NN_mat(sub2ind(size(NN_mat),patch_number,NN_mat_column)),2); % for
all dispersing individuals determine destination y from the destination cell
number
        ind_d_new(index,2)=patch_column;
        ind_d_new(index,3)=patch_row; clear index index1 index2
        ind_d=ind_d_new;

        [deme, ~, ~,
~]=get_deme_info(y_max,x_max,ind_d,deme_count_base,'light');

        %% reproduction

        N_ind_d=size(ind_d,1);
        ind_r=ind_d; clear ind_d    %   ind_r(:,4) is from now indicating
whether the individual's mother was a migrant
        tot_dens=deme/K; % calculating population density
        ind_r(:,5)=(1-ind_r(:,3)*rc_rl).*(1-rc_rd*tot_dens(y_max*(ind_r(:,2)-
1)+ind_r(:,3))); % total reproduction correction factor, incl. both location
and density correction
        % base value nr of offspring = 7 %
        ind_r(:,5)=floor(ind_r(:,5).*(rand(N_ind_d,1)*7)); %random nr of
offspring [0..6], corrected
        clear N_ind_d
```

```
        % find the indices per nr of offspring
        index1=ind_r(:,5)==1;
        index2=ind_r(:,5)==2;
        index3=ind_r(:,5)==3;
        index4=ind_r(:,5)==4;
        index5=ind_r(:,5)==5;
        index6=ind_r(:,5)==6;
        ind_r=[repmat(ind_r(index1,:),1,1); repmat(ind_r(index2,:),2,1);
repmat(ind_r(index3,:),3,1);
repmat(ind_r(index4,:),4,1);repmat(ind_r(index5,:),5,1);
repmat(ind_r(index6,:),6,1)]; % multiply by the nr of offspring
        clear index1 index2 index3 index4 index5 index6

        % mutation
        mut= rand(size(ind_r,1),1); % a random new threshold value for ALL
individuals
        index_m= rand(size(ind_r,1),1) < m_rate; % select the mutating
individuals
        ind_r(index_m,1)=mut(index_m); % replace the mutated threshold values


        ind=ind_r; clear ind_r
        N_ind=size(ind,1);


        end; clear year
    end; clear run
end; clear scen
```

Function get_deme_info

```
function [deme, deme_t, deme_m,
deme_r]=get_deme_info(y_max,x_max,ind,deme_count_base,method)

% get deme info
% deme is the number of individuals per deme
% deme_t is the average threshold per deme
% deme_m is the number of migrated individuals per deme
% deme_r is the number of offspring per deme

[~, sorti]=sortrows(ind(:,2:3)); y=ind(sorti,2:3); ind=ind(sorti,:);
index_c = find([true;sum(diff(y),2)~=0]);
values = y(index_c,:); instances = diff(index_c); % instances = nr of
individuals with the same xy combination.
instances=[instances; size(y,1)-sum(instances)];
deme_count=[values instances]; clear values instances y

deme_count_base_str=1000*deme_count_base(:,1)+deme_count_base(:,2);
deme_count_str=1000*deme_count(:,1)+deme_count(:,2);
index=ismember(deme_count_base_str,deme_count_str); clear deme_count_base_str
deme_count_str

deme=deme_count_base; deme(index,3)=deme_count(:,3); clear deme_count
deme=reshape(deme(:,3),y_max, x_max);

switch method
    case 'complete'
        deme_t=zeros(y_max,x_max); deme_m=zeros(y_max,x_max);
deme_r=zeros(y_max,x_max);
        for dm=1:length(index_c)-1
            row=mean(ind(index_c(dm):index_c(dm+1)-1,3));
            column=mean(ind(index_c(dm):index_c(dm+1)-1,2));
            deme_t(row,column)=mean(ind(index_c(dm):index_c(dm+1)-1,1));
%the individuals sorted by population nr, so averaging over the individuals
per population
            deme_m(row,column)=sum(ind(index_c(dm):index_c(dm+1)-1,4));
            deme_r(row,column)=sum(ind(index_c(dm):index_c(dm+1)-1,5));
        end;
        deme_t(end)=mean(ind(index_c(dm+1):end,1));
        deme_m(end)=sum(ind(index_c(dm+1):end,4));
        deme_r(end)=sum(ind(index_c(dm+1):end,5)); clear dm index_c row column

    case 'light'
        deme_t=zeros(y_max,x_max); deme_m=zeros(y_max,x_max);
deme_r=zeros(y_max,x_max);

    case 'deme_m_only'
        deme_m=zeros(y_max,x_max);
deme_t=zeros(y_max,x_max);deme_r=zeros(y_max,x_max);
        for dm=1:length(index_c)-1
            row=mean(ind(index_c(dm):index_c(dm+1)-1,3));
            column=mean(ind(index_c(dm):index_c(dm+1)-1,2));
            deme_m(row,column)=sum(ind(index_c(dm):index_c(dm+1)-1,4));
        end;
        deme_m(end)=sum(ind(index_c(dm+1):end,4)); clear dm index_c row column
```

```
end
```