Code used to generate simulation results for MS: *Predation can select for later and more synchronous arrival times in migrating species*. Harts, Kristensen & Kokko. In *Oikos*

This code was designed for use in Matlab. The code generates a few files based on last generation data, it can be modified to generate data for each generation (not shown).

Words in blue and black are code, annotations are in green.

```
function[]=arrival_time(X,tmax,P0,initarriv,stdarriv,winterMort,nFocal,nTwo,Nterr,a,b,z,alpha,betaP,
    betaG,gen,ms)

% to start a simulation call e.g. arrival_time(850,5,0.05,0.5,0.5,0.1,1000,1000,500,0.5,5,1,0.5,1,2,5000,999)

% X = daily handling time predators have on breeding grounds (a measure of predator abundance)

% tmax = number of arrival days, number of days an individual can arrive on

% P0 = daily mortality during migration period if not arrived yet

% initarriv & stdarriv are for the initial populations arrival time allele, see below for calculation

% migrMort = mortality during migration

% nFocal = total number of individuals of the focal species (initial generation only)

% nTwo = number of alternative prey (i.e. S2) alive at beginning of each spring

% Nterr = total  number of territories

% a = predator preference, 1 = S1and 0 = S2, at 0.5 there is no preference

% b = handling time of S1 (has to be < X)

% z = handling time factor such that bz=handling time of S2

% alpha = proportion good territories (e.g. 0.5 means half is good and half is poor)

% betaP and betaG: number of S1 offspring in poor and good territories respectively (beta gives the mean of the
    poisson distribution which is #offspring per individual)

% gen = number of generations the simulation runs for

% ms = seed number, allows for replicating a specific simulation

rng(ms) % setting seed for repeatability and independent simulations

mutProb=0.1; mutStep=0.01; % mutation probability and determinant of size of mutation.
```

```
%creation of initial population, each individual gets an allele for arrival time (Pop(:,1)) and initial %territorial
    state (Pop(:,2)) NaN means not arrived; 0 means floater, -1 means poor terr owner, 1 %means good terr
    owner

Pop=max(0,min(1,initarriv+stdarriv*(rand(nFocal,1)-rand(nFocal,1))));

Pop(:,2)=NaN*ones([nFocal 1]);

%data collected

meanArriv=NaN*ones([1 gen]); % mean of arrival alleles in focal population (S1)

S1spring=NaN*ones([1 gen]); % number of S1 at start of each generation

nof_arrived=NaN*ones([tmax gen]); % number of S1 arrived this time within each generation

dead_arrived=NaN*ones([tmax gen]); % actual number of S1 predated on breeding grounds / gen.

dead_nonarrived=NaN*ones([tmax gen]); % actual number of S1 predated on nonbreeding grounds /gen.

prodOff=NaN*ones([1 gen]); % number of offspring produced this gen

track_S2=NaN*ones([tmax gen]); % number of S2 on each day t after predation during arrival S1

arrivHist=NaN*ones([1 tmax]); % distribution of arrival allele in the last generation


for g=1:gen % start of new gen.

   S2=nTwo; % each gen starts with the same number of individuals of S2

   RF=0; % RF is number of floaters that have arrived, at start of each gen it is set to 0

   meanArriv(g)=mean(Pop(:,1)); % calculate the mean arrival allele within the population for gen=g

   S1spring(g)=length(Pop(:,1)); % calculate number of S1at start of gen=g


   for t=1:tmax % arrival day within a gen.

      % territory availability update

      aG=alpha*Nterr-sum(Pop(:,2)>0); % good territories still available

      aP=(1-alpha)*Nterr-sum(Pop(:,2)<0); % poor territories still available

      % find those who are arriving now OR are floating, and

      % randomize the order in which they are allowed to take territories
```

```
if t<tmax
   f=vectperm(find(((Pop(:,1)>=(t-1)/tmax) & (Pop(:,1)<t/tmax)) | (Pop(:,2)==0)));
else % on t=tmax
   f=vectperm(find(((Pop(:,1)>=(t-1)/tmax) & (Pop(:,1)<(t+1)/tmax)) | (Pop(:,2)==0)));
end

%arrival and territory assignment (within each time t)
for j=1:length(f)
   if aG>0 % if there are good territories available
      aG=aG-1; % one good territory less available
      Pop(f(j),2)=1; % this indiv. is now the owner of a good terr
   elseif aP>0 % else if there are poor terr available
      aP=aP-1; % one less poor territory available
      Pop(f(j),2)=-1; % this indiv.  is now the owner of a poor terr
   else
      RF=RF+1; % one more floater
      Pop(f(j),2)=0; % this indiv. is now a floater
   end
end
% predation of all individuals that have arrived at breeding site
% S1 denotes the number of indiv. arrived at time t
arrived=find(Pop(:,1)<=(t/tmax)); S1=length(arrived);
nof_arrived(t,g)=S1; % data collection, individuals at breeding ground after arrival

% number of eaten individuals of species 1
est_dead1=min(S1,((a*S1*X)/(1+(a*b*S1)+((1-a)*(b*z*S2))))); % unrounded # of indiv. to die
```

```
prob_dead1=est_dead1-round(est_dead1); % difference between two integers becomes the prob.
if rand(1)<prob_dead1 % if random number is smaller than difference (probability)
   dead1=round(est_dead1)+1; % than it will become the larger integer
else
   dead1=round(est_dead1); % else the smaller integer
end
dead_arrived(t,g)=dead1; % data collection, S1 indiv. died per t from predation
doomed=picksurv(ones([1 S1]),dead1);  % pick the indiv. that will be predated
Pop(arrived(doomed),:)=[]; % remove these individuals from population

% the 2nd species dies in a simpler manner as identities don't have to be tracked
dead2=min(S2,round(((1-a)*S2*X)/(1+(a*b*S1)+((1-a)*(b*z*S2)))));
S2=S2-dead2; % mortality updates the number of second species present
track_S2(t,g)=S2; % data collection, tracking S2

% finally, daily mortality of individuals that have not yet arrived on breeding grounds
if t<tmax % because everybody has arrived on day tmax
   notarrived_and_dead=find(Pop(:,1)>=(t/tmax) & rand([size(Pop,1) 1])<P0);
   dead_nonarrived(t,g)=length(notarrived_and_dead); % data collection
   Pop(notarrived_and_dead,:)=[]; % remove
end

if t==tmax % last time step t, fill remaining territories with floaters
   bG=alpha*Nterr-sum(Pop(:,2)>0); % good terr. available
   bP=(1-alpha)*Nterr-sum(Pop(:,2)<0); % poor terr. available
   fl_tmax=vectperm(find(Pop(:,2)==0)); % to create random selection of floaters to good and poor
```

```
        for k=1:length(fl_tmax)
            if bG>0 % if there are good territories available
                bG=bG-1; % one good territory less available
                Pop(fl_tmax(k),2)=1; % this ind is now the owner of a good terr
            elseif bP>0 %else if there are poor terr available
                bP=bP-1; %one less poor territory available
                Pop(fl_tmax(k),2)=-1; % this nFocal is now the owner of a poor terr
            end
        end
    end
end
% Post arrival:
% reproduction & mutation
allNew=[]; % vector to collect new indiv. in
for q=1:length(Pop(:,1))
    nOff=0; New=[];
    if Pop(q,2)>0  % individuals with a good breeding territrory
        nOff=poissrnd(betaG); %number of offspring produced at a good site, average=betaG
    elseif Pop(q,2)<0  % individuals with a poor breeding territrory
        nOff=poissrnd(betaP); %number of offspring produced at a poor site, average=betaP
    end
    if nOff>0
      New=max(0,min(1,((Pop(q,1)*ones([nOff 1]))+((rand(nOff,1)<mutProb).*(randn(nOff,1)*mutStep)))));
      % offspring inherit parent's arrival time & mutation may occur
        New(:,2)=zeros([nOff 1]); % offspring start as floater
        allNew=[allNew;New];
    end
```

```
    end
    prodOff(g)=length(allNew);
    Pop=[Pop;allNew]; % offspring join adult population

    % mortality in the migration & wintering season
    Pop(rand(size(Pop,1),1)<winterMort,:)=[]; %mortality during migration & overwintering

    % get the distribution of arrival time allele in the last generation
    if g==gen
       for p=1:tmax
          if p<tmax
             arrivDoc=find((Pop(:,1)>=(p-1)/tmax) & (Pop(:,1)<p/tmax));
          else
             arrivDoc=find((Pop(:,1)>=(p-1)/tmax) & (Pop(:,1)<(p+1)/tmax));
          end
          arrivHist(p)=length(arrivDoc);
       end
    end
     % the annual cycle ends by all individuals getting status NaN, i.e. they have not arrived yet
    Pop(:,2)=NaN;
    % save datafiles
    save('meanArriv','meanArriv'); save('S1spring','S1spring');
    save('nof_arrived','nof_arrived'); save('dead_arrived','dead_arrived');
    save('dead_nonarrived','dead_nonarrived'); save('prodOff','prodOff');
    save('track_S2','track_S2'); save('arrivHist','arrivHist');
end
```