```
% Script for niche construction model of cultural accumulation
% Laurel Fogarty and Nicole Creanza
% 03/16/2017
% MATLAB

% Based on Fogarty, Wakano, Feldman, and Aoki (2015; 2016); Rendell et al
% (2010)

clear all

%% initialisation
N_vec = [5 100 250]; % number of individuals
E_vec = 1:500; % number of environmental states
h_vec = [1 1.1]; % harshness parameter
beta = 0.9;% probability of learning applied to each trait
r_vec = 1:5;% repeat the sim

for N_ind = 1:numel(N_vec)% cycle through population sizes

    N = N_vec(N_ind);
    c_vec_ref = [1:10:101];
    c_vec = [1*N:10*N:101*N]; % Environment changes every c time steps ADJUSTED
              FOR GENERATIONS.
    nGen = 1000*N; % time steps to run model
    mu = 1/N; % innovation rate

    for hi = 1:numel(h_vec)
        h = h_vec(hi);
        Output_POP = zeros(numel(r_vec),numel(c_vec));% average population output
              for each value of c
        for r = 1:numel(r_vec)
            rep = r_vec(r);
            for c_val = 1:numel(c_vec)

                c=c_vec(c_val);

                INDS_FT = zeros(nGen,N); % matrix of individual fitnesses
                INDS_BH = zeros(numel(E_vec),N,nGen); % matrix of individual
                  behaviours - contains indices where trait is present and 0 where
                  trait is not present

                E = ceil(min(E_vec) + (max(E_vec)-min(E_vec))*rand); % initial
                  random environmental state
                Del = zeros(nGen,N); % min difference between environment and
                  trait for each individual, delta
                E_save = zeros(1,nGen);

                %% main loop
                for i = 1:nGen

                    if i>1
                        INDS_BH(:,:,i) = INDS_BH(:,:,i-1);
                    end
                    % fitness calculations

                    for n = 1:N
```

```
        behaviours = find(INDS_BH(:,n,i)>0);
        if numel(behaviours)>0
            Del(i,n) = min(abs(behaviours-E)); % harshness
        calculation - using closest behaviour to environment
            INDS_FT(i,n) = h.^-Del(i,n);
        else
            Del(i,n) = max(E_vec);
            INDS_FT(i,n) = h.^-Del(i,n);
        end
    end

    % innovation
    nRands = randperm(N);
    for n = nRands % each ind can innovate incuding newborn
        m=rand;
        if m<mu && Del(i,n)>0 % if you don't match the environment
        and chance allows
            available = find(sum(INDS_BH(:,:,i),2)==0);
            if numel(available)>0
                randInd = randperm(numel(available));
                beh = available(randInd(1));
                     INDS_BH(beh,n,i) = 1; % innovate a new trait
                to match a random environment
            end
        end
    end

    % birth and learning
     pRoleModel =  INDS_FT(i,:);
     if any(pRoleModel)
        roleModel = randsample(1:N,1,true,pRoleModel);% choose a
        fit role model
     else
        roleModel = ceil(N*rand);
    end

    roleModel_BEH = INDS_BH(:,roleModel,i);

    % death
    pDeath = ones(size(INDS_FT(i,:)))-
        (INDS_FT(i,:)./max(INDS_FT(i,:)));% probability of death
        proportional to S
    if any(pDeath)
        pDeath(find(isnan(pDeath))) = 0;
        whoDies = randsample(1:N,1,true,pDeath);% chosen in
        proportion to probability of death
    else
        whoDies = ceil(N*rand);
    end

    for b = 1:numel(E_vec)
        if rand<beta
            INDS_BH(b,whoDies,i) = roleModel_BEH(b);% replace
        individual who died with the newborn behaviour
        else
            INDS_BH(b,whoDies,i) = 0;
        end
```

```
                end

                % environmental change
                if rem(i,c)==0
                    En = ceil(numel(E_vec)*rand);% random change
                    E = En;
                end
            end
            Output_POP(r,c_val) =
              numel(find(sum(INDS_BH(:,:,1500:end),2)>0))/500;

        end
    end

    meanOutput_POP = mean(Output_POP);
    minOutput_POP = min(Output_POP);
    maxOutput_POP = max(Output_POP);
    stdOutput_POP = std(Output_POP);
    varOutput_POP = var(Output_POP);
    errorOutput_POP = stdOutput_POP/sqrt(length(stdOutput_POP));

    end
end
```