# **SI-5 Cleaned Matlab code**

The following Matlab v7.14.0.739 code are the main modules and are bespoke to data file structures and computer set-up.

5.1 Main module – starting the simulations and collating outputs from runs

5.2 The model per run – including outputs for lambda, occupancy, numbers of individuals and extinction rates per run

5.3 The seed flow per year submodule – redistributing the seeds among populations based on:

5.4. The initial seed distribution matrix – run once at the start

5.5. The per year per population parameter draw submodule

5.6. An example parameter definition file

5.1 Main module – starting the simulations and collating outputs from run

IPM Main module for a connected B. oleracea metapopulation, including dispersal

Outputs: lambda's for occupancy and individuals and numbers per year; occupancy and colonisation rates and others ad hoc.

The model simulates population size per year in a point population with a diameter of 12-meters. Dispersal is not-point based but a 1-m 4-way integral

Around populations is not habitable habitat

The model is based on number of plants flowering or not before seed set.

The model starts with seed set and incorporation of the % plants flowering per class

tic

clc

clear all

mkdir('temp')

% General parameters

Sensis = [xx];% fill in the different scenario's, see define file

definition\_function = 'defintion\_name'; % give name

extention = 'name'; % give name

folder = 'definitions/';

definition\_function = [folder,definition\_function];

folder\_out = 'name/'; % give name

if exist(folder\_out) == 0

mkdir(folder\_out)

end

Output.org = {[1,2]};

copyfile (definition\_function, 'define\_function.m')

clear definition\_function

max\_diameter = 100;

histo\_length = max\_diameter +3;

size\_steps = ceil(max\_diameter/10);

cd Results

load('pops&dispersal.mat')

cd ../

max\_density(1) = (max(max(Dorset\_populations.densities))).\*2; % maximum number of adult plants

max\_density(2) = 0.5.\*max\_density(1); % max density tolerance

Oleracea\_parameters\_load;

population\_types = Oleracea\_population\_types;

save('start\_parameters')

Time\_run = 0;

pops\_only = 0;

%% Run the model for every scenario (sensi) provided

for sensis = 1:1:length(Sensis)

sensi = Sensis(sensis);

tic;

save('Output','Output')

run = 1;

define\_function % an example parameter definition file is provided in 5.6

max\_density(3) = (max\_density(1).\*0.5).\*pop\_max;

delete('Prop\_among.mat')

if pop\_max == 3

pops\_only = 1;

end

%% Calculate Kernels

Oleracea\_Seed\_flow\_kernels % see 5.4

save('Prop\_among', 'Prop\_among');

%% Send runs to background processors

output = 1;

job = createJob('configuration', 'Full');

for run = 1:run\_max

createTask(job, @Oleracea\_actual\_simulations, 1,...

{run,population\_types,max\_density,Prop\_among,Sensis,...

max\_diameter,size\_steps,sensi,histo\_length,run\_max,extention});

end

submit(job);

waitForState(job, 'finished');

results = getAllOutputArguments(job);

destroy(job)

%% Collect output per run and do statistics and write to Output file

load('Output.mat')

Output.population\_types = population\_types;

if sensi ~= sensi\_para.baseline

Lambda\_meta\_base = Output.(genvarname(['s',int2str(sensi\_para.baseline)])).Lambda\_meta\_base;

Occup\_meta\_base= Output.(genvarname(['s',int2str(sensi\_para.baseline)])).Occup\_meta\_base;

Lambda\_base = Output.(genvarname(['s',int2str(sensi\_para.baseline)])).Lambda\_base;

end

cd temp

for run = 1:run\_max

name\_file = ['output','\_',int2str(run),'.mat'];

load(name\_file)

Lambda(run, :) = lambda; %#ok<\*SAGROW>

Individualss(run,:,:) = Individuals;

Individualss\_seeds(run, :,:) = Individuals\_seed;

Histo\_individuals\_metas(run,:,:) = Histo\_individuals;

Occupancies(run,:) = Occupancy\_adult;

Occupancies\_seeds(run,:) = Occupancy\_seeds;

Start\_individualss(run,:) = Start\_individuals;

Lambda\_metaY(run,:) = lambda\_metaY;

Lambda\_meta(run) = lambda\_meta;

Occup\_meta(run) = occup\_meta;

Total\_individuals(run,:) = total\_individual;

Empty\_rates(run,:) = Empty\_rate;

Colonisation\_rates(run,:) = Colonisation\_rate;

Extinction\_rates(run,:) = Extinction\_rate;

%% sensitivity

if exist('Do\_sensitivity') ~= 0

if strcmpi('y',Do\_sensitivity) ~= 0

% sensitivity per run for meta population lambda

if sensi ~= sensi\_para.baseline

base = Lambda\_meta\_base(run);

new = lambda\_meta;

if isnan(base) ~= 1 && isnan(new) ~= 1 % This is SENSITIVITY !!!

sens\_tmp = (new/base) -1;

if new < base

sens\_tmp = 1-(1./(base/new));

end

if sensi\_para.change\_factor ~=1

sens\_meta(run) = sens\_tmp./abs(1-sensi\_para.change\_factor);

else

sens\_meta(run) = 0;

end

end

else

sens\_meta(run) = 0;

end

clear sens\_tmp

%sensitivity per run for meta occupancy

if sensi ~= sensi\_para.baseline

base = Occup\_meta\_base(run);

new = occup\_meta;

if isnan(base) ~= 1 && isnan(new) ~= 1

%Lambda\_meta

sens\_tmp = (new/base) -1;

if new < base

sens\_tmp = 1-(1./(base/new));

end

if sensi\_para.change\_factor ~=1

occu\_meta(run) = sens\_tmp./abs(1-sensi\_para.change\_factor);

else

occu\_meta(run) = 0;

end

end

else

occu\_meta(run) = 0;

end

clear sens\_tmp

% sensitivity per run for populations lambda

value = lambda;

value(isnan(value)==1) = [];

new = mean(mean(value));

clear value

if sensi ~= sensi\_para.baseline

value = Lambda\_base(run,:);

value(isnan(value)==1) = [];

base = mean(mean(value));

if isnan(base) ~= 1 && isnan(new) ~= 1

%Lambda\_meta

sens\_tmp = (new/base) -1;

if new < base

sens\_tmp = 1-(1./(base/new));

end

if sensi\_para.change\_factor ~=1

sens\_pop(run) = sens\_tmp./abs(1-sensi\_para.change\_factor);

else

sens\_pop(run) = 0;

end

end

else

sens\_pop(run) = 0;

end

clear sens\_tmp

if strcmpi('y',Do\_sensitivity) == 0

sensi\_para.change\_factor = 1;

end

end

end

diror = ['../','Store\_',extention,'/','outputs','\_',int2str(sensi)];

if exist(diror) == 0

mkdir(diror)

end

dir2 = [diror,'/'];

movefile(name\_file,dir2);

clear All not needed % fill in what is not needed

end % per run

cd ..

if sensi == sensi\_para.baseline

Output.(genvarname(['s',int2str(sensi\_para.baseline)])).Lambda\_meta\_base = Lambda\_meta;

Output.(genvarname(['s',int2str(sensi\_para.baseline)])).Occup\_meta\_base = Occup\_meta;

Output.(genvarname(['s',int2str(sensi\_para.baseline)])).Lambda\_base = Lambda;

end

%% Statistics and Save output

% overall meta-population lambda Individuals

Output.(genvarname(['s',int2str(sensi)])).Lambda\_meta.mean = (mean(Lambda\_meta))';

Output.(genvarname(['s',int2str(sensi)])).Lambda\_meta.std = (std(Lambda\_meta))';

if exist('Do\_sensitivity') ~= 0

if strcmpi('y',Do\_sensitivity) ~= 0

Output.(genvarname(['s',int2str(sensi)])).Lambda\_meta.sensitivity(1) = mean(sens\_meta);

Output.(genvarname(['s',int2str(sensi)])).Lambda\_meta.sensitivity(2) = std(sens\_meta);

Output.(genvarname(['s',int2str(sensi)])).Occupancy\_meta.sensitivity(1) = mean(occu\_meta);

Output.(genvarname(['s',int2str(sensi)])).Occupancy\_meta.sensitivity(2) = std(occu\_meta);

end

end

% overall meta-population lambda Occupancy

Output.(genvarname(['s',int2str(sensi)])).Occupancy\_meta.occupancy(1) = (mean(Occup\_meta))';

Output.(genvarname(['s',int2str(sensi)])).Occupancy\_meta.occupancy(2) = (std(Occup\_meta))';

if exist('Do\_sensitivity') ~= 0

if strcmpi('y',Do\_sensitivity) ~= 0

Output.(genvarname(['s',int2str(sensi)])).Lambda\_pop.sensitivity(1) = mean(sens\_pop);

Output.(genvarname(['s',int2str(sensi)])).Lambda\_pop.sensitivity(2) = std(sens\_pop);

end

end

% Overview rates and amounts PER YEAR

Output.(genvarname(['s',int2str(sensi)])).Lambda\_meta\_Year.mean = (mean(Lambda\_metaY(:,5:year\_max),1))';

Output.(genvarname(['s',int2str(sensi)])).Lambda\_meta\_Year.std = (std(Lambda\_metaY(:,5:year\_max),0,1))';

Output.(genvarname(['s',int2str(sensi)])).Total\_Individuals.mean = (mean(Total\_individuals,1))';

Output.(genvarname(['s',int2str(sensi)])).Total\_Individuals.std = (std(Total\_individuals,0,1))';

Output.(genvarname(['s',int2str(sensi)])).Occupancy.adult.mean = (mean(Occupancies,1))';

Output.(genvarname(['s',int2str(sensi)])).Occupancy.adult.std = (std(Occupancies,0,1))';

Output.(genvarname(['s',int2str(sensi)])).Occupancy.seeds.mean = (mean(Occupancies\_seeds,1))';

Output.(genvarname(['s',int2str(sensi)])).Occupancy.seeds.std = (std(Occupancies\_seeds,0,1))';

Output.(genvarname(['s',int2str(sensi)])).Individuals.mean = squeeze(mean(Individualss,1));

Output.(genvarname(['s',int2str(sensi)])).Individuals.std = squeeze(std(Individualss,0,1));

Output.(genvarname(['s',int2str(sensi)])).Seeds.mean = squeeze(mean(Individualss\_seeds,1));

Output.(genvarname(['s',int2str(sensi)])).Seeds.std = squeeze(std(Individualss\_seeds,0,1));

Output.(genvarname(['s',int2str(sensi)])).Colonisation.mean = (mean(Colonisation\_rates,1))';

Output.(genvarname(['s',int2str(sensi)])).Colonisation.std = (std(Colonisation\_rates,0,1))';

Output.(genvarname(['s',int2str(sensi)])).Extinction.mean = (mean(Extinction\_rates,1))';

Output.(genvarname(['s',int2str(sensi)])).Extinction.std = (std(Extinction\_rates,0,1))';

Output.(genvarname(['s',int2str(sensi)])).Empty.mean = (mean(Empty\_rates,1))';

Output.(genvarname(['s',int2str(sensi)])).Empty.std = (std(Empty\_rates,0,1))';

clear All not needed % shortened to save text

%% Per population Lambdas

for population = 1:1:pop\_max

Lambda\_pop = Lambda(:,population);

Lambda\_pop(Lambda\_pop ==0) = [];

if isempty(Lambda\_pop) ~= 1

Lambdas(1,population) = mean(Lambda\_pop);

Lambdas(2,population) = std(Lambda\_pop);

else

Lambdas(1:2,population) = NaN;

end

clear Lambda\_pop

end

clear Lambda

value = Lambdas(1,:);

value(isnan(value)==1) = [];

Output.(genvarname(['s',int2str(sensi)])).Lambda\_mean.mean = mean(value);

Output.(genvarname(['s',int2str(sensi)])).Lambda\_mean.std = std(value);

if pop\_max <= 100

Output.Lambda(:,:,sensi) = Lambdas;

end

%% Store parameters used

Output.(genvarname(['s',int2str(sensi)])).kernel = kernel;

Output.(genvarname(['s',int2str(sensi)])).Prop\_among = Prop\_among;

delete('Prop\_among.mat')

sensi\_para = rmfield(sensi\_para,'show');

Output.(genvarname(['s',int2str(sensi)])).Sensi\_para = sensi\_para;

Output.(genvarname(['s',int2str(sensi)])).Sensi\_para.pop\_max = pop\_max;

Output.(genvarname(['s',int2str(sensi)])).Sensi\_para.run\_max = run\_max;

Output.(genvarname(['s',int2str(sensi)])).Sensi\_para.year\_max = year\_max;

Output.(genvarname(['s',int2str(sensi)])).Sensi\_para.max\_diameter = max\_diameter;

Output.(genvarname(['s',int2str(sensi)])).Sensi\_para.size\_steps = size\_steps;

Output.(genvarname(['s',int2str(sensi)])).Sensi\_para.max\_density = max\_density;

Output.Sensi\_para\_baseline = sensi\_para.baseline;

if exist('Do\_sensitivity') ~= 0

Output.(genvarname(['s',int2str(sensi)])).Sensi\_para.Do\_sensitivity = Do\_sensitivity;

end

if exist('zero\_densities')~= 0

Output.(genvarname(['s',int2str(sensi)])).Sensi\_para.zero\_densities = zero\_densities;

end

sensi\_toc = toc;

Output.(genvarname(['s',int2str(sensi)])).time\_run = toc;

toc

Time\_run = Time\_run + toc;

Output.Time\_run = Time\_run;

clear All not needed % shortened to save text

end % SENSIS

%save outputs and delete inbetweens

save('Output','Output')

delete('parameters.mat')

delete ('parameters\_start.mat')

delete('start\_parameters.mat')

rmdir('temp','s')

name\_file = [folder\_out,'Output','\_',extention,'.mat'];

movefile ('Output.mat', name\_file)

clear all

clc

5.2 The model per run – including outputs for lambda, occupancy, numbers of individuals and extinction rates per run

The actual model runs for all populations and all years in for loops

Outputs: lambda's for occupancy and individuals and numbers per year; occupancy and colonisation rates and others ad hoc.

The per year loop is split in two in which populations are combined for seed flow.

function [run\_max] =...

Oleracea\_actual\_simulations(run,population\_types,max\_density,Prop\_among,Sensis,...

max\_diameter,size\_steps,sensi,histo\_length, run\_max,extention)

rng('shuffle')

load('Prop\_among.mat')

define\_function

% an example parameter definition file is provided in 5.6

%% redefine histogram with a seedbank and 2 seedlings classes

Histo\_base\_model(1:3,pop\_max) = 0;

Histo\_base\_model(4:histo\_length,pop\_max)= 0;

%% Initiate output parameters

for population = 1:1:pop\_max

lengths = populations.(genvarname(['population','\_',int2str(population)])).histo\_org\_length;

Histo\_base\_model(4:(3+lengths),population) = ...

populations.(genvarname(['population','\_',int2str(population)])).Histo\_base;

Start\_individuals(population) = sum(Histo\_base\_model(4:histo\_length,population));

end

% Base model Cabbage model per population per year

%% THIS IS THE MODEL per year!!!!!!!!!!!!!!

% Make all random matrices first

adult\_test = rand(year\_max,pop\_max);

seed\_test = rand(year\_max,pop\_max);

class1\_test = rand(year\_max,pop\_max);

class2\_test= rand(year\_max,pop\_max);

reached = zeros(pop\_max,1);

lambda\_meta = 0;

occup\_meta = 0;

for year = 1:1:year\_max

% Selectk the year of the parameters

if strcmpi('y',Do\_sensitivity) ~= 0

population\_year = population\_years(year);

else

if strcmpi('y',sensi\_para.all\_years) ~= 0

population\_year = 7;

else

population\_year = 0;

while population\_year == 0

population\_year = randi(6);

end

end

end

%% From here per population!!!

%% Initiate parameters and variables

draws.org = {[1,2]};

Colonisation = zeros(pop\_max,1);

Extinction = zeros(pop\_max,1);

for population = 1:1:pop\_max

%Initiate variables

if year == 1

Histo\_individuals(:,population) = Histo\_base\_model(:,population)'; %#ok<\*AGROW>

Individuals(1,population) = Start\_individuals(population);

total\_individual(1) = sum(Individuals);

Individuals\_all = sum(round(Histo\_individuals(2:histo\_length,population)));

Occupancy\_adult(1) = (length(find(Start\_individuals> sensi\_para.threshold)))./pop\_max;

Occupancy\_seeds(1) = (length(find( Histo\_individuals(1) > sensi\_para.threshold)))./pop\_max;

if Individuals\_all > 0 % existing populations

mortality(population) = 0;

lambda(population) = 1;

Colonisation(population) = 1;

elseif Individuals\_all == 0 % Non existing populations

mortality(population) = 1;

lambda(population) = 0;

end

end

% draw individual function parameters, see 5.5

% here the function draws are made per year and sensitivities within

draws = Oleracea\_draws(run, population,populations,Histo\_sizes,...

sensi\_para,draws,population\_year,Do\_sensitivity,max\_diameter,sensi,histo\_length);

end % end of per population

%% per population Fecundity (autumn)

for population = 1:pop\_max

Fecundity = draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Fecundity;

Flowering\_like = …

draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Flowering\_like;

Flowering\_plants = Histo\_individuals(4:histo\_length,population).\* Flowering\_like(2,:)'; %year t-1

Seeds\_total(population) = sum(Flowering\_plants.\*Fecundity(2,:)'); %year t-1

% Round seeds to Threshold numbers: Make sure there are not infitive small number of seeds produced

Seeds\_total(population)= round(Seeds\_total(population).\*(1/sensi\_para.threshold)).\*sensi\_para.threshold;

end

%% For all populations: add seed flow from outside minus seed flow to outside

if strcmpi('y',sensi\_para.wind\_dispersal\_present) ~= 0 || strcmpi('y',sensi\_para.human\_dispersal\_present) ~= 0

Oleracea\_Seed\_flow\_year; % 5.3

end

%% Per population: Growth part (spring)

for population = 1:pop\_max

Histo\_stop = 0;

SBincorp = draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).SBincorp;

SBrate = draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).SBrate;

Gs\_undis = draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Gs\_undis;

Ss\_undis = draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Ss\_undis;

Gs\_dis = draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Gs\_dis;

Ss\_dis = draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Ss\_dis;

Adult\_surv = ...

(draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Adult\_surv).\*(1-...

sensi\_para.mortality\_disturbance\_Adult);

Adult\_increase =...

draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Adult\_increase;

Histo\_individuals\_meta = Histo\_individuals(:,population);

Seed\_total = Seeds\_total(population).\*SBincorp; %year t-1

% Seedbank survival from t-1 to t

seedbank = Histo\_individuals\_meta(1).\* SBrate;

% Split between Disturbed and Undisturbed

Histo\_tmp\_dis = Histo\_individuals\_meta(1:3).\* sensi\_para.proportion\_disturbance;

Histo\_tmp\_undis = Histo\_individuals\_meta(1:3).\* (1- sensi\_para.proportion\_disturbance);

Seed\_total\_dis = Seed\_total.\* sensi\_para.proportion\_disturbance;

Seed\_total\_undis = Seed\_total.\* (1- sensi\_para.proportion\_disturbance);

%Direct germination and to seedbank

%DISTURBED

to\_first\_year\_dis = ((Gs\_dis(1).\*Ss\_dis(1)).\* Seed\_total\_dis)+ (Gs\_dis(2).\*Seed\_total\_dis); %

to\_seedbank\_dis = Seed\_total\_dis- ((Gs\_dis(1).\* Seed\_total\_dis)+ (Gs\_dis(2).\*Seed\_total\_dis));

germ\_dis = (((Gs\_dis(3)+Gs\_dis(5))./2)+Gs\_dis(4)).\*Histo\_tmp\_dis(1);

to\_second\_year\_dis = Histo\_tmp\_dis(2).\*(Ss\_dis(2).\*Ss\_dis(3));

to\_adult\_loop\_dis = Histo\_tmp\_dis(3).\*(Ss\_dis(4).\*Ss\_dis(5));

%UNDISTURBED

to\_first\_year\_undis = ((Gs\_undis(1).\*Ss\_undis(1)).\* Seed\_total\_undis)+ (Gs\_undis(2).\*Seed\_total\_undis); %

to\_seedbank\_undis = Seed\_total\_undis- ((Gs\_undis(1).\* Seed\_total\_undis)+ (Gs\_undis(2).\*Seed\_total\_undis));

germ\_undis = (((Gs\_undis(3)+Gs\_undis(5))./2)+Gs\_undis(4)).\*Histo\_tmp\_undis(1);

to\_second\_year\_undis = Histo\_tmp\_undis(2).\*(Ss\_undis(2).\*Ss\_undis(3));

to\_adult\_loop\_undis = Histo\_tmp\_undis(3).\*(Ss\_undis(4).\*Ss\_undis(5));

%JOIN

to\_first\_year = to\_first\_year\_dis + to\_first\_year\_undis;

to\_seedbank = to\_seedbank\_dis + to\_seedbank\_undis;

germ = germ\_dis + germ\_undis;

to\_second\_year = to\_second\_year\_dis + to\_second\_year\_undis;

to\_adult\_loop = to\_adult\_loop\_dis + to\_adult\_loop\_undis;

clear All not needed % shortened to save text

%subtract Germ from seed bank and add new seeds

seedbank = seedbank-germ; % is seedbank at t

seedbank = seedbank + to\_seedbank;

% add germination to first year seedlings

to\_first\_year = to\_first\_year + germ; % first year seedlings at t

%within adult\_loop

Adult\_plants = zeros(4,(histo\_length-3));

Adult\_plants(1,:) = Adult\_increase(1,:);

Adult\_plants(2,:) = Histo\_individuals\_meta(4:histo\_length).\* (Adult\_surv(2,:)');

Adult\_plants(3,:) = floor(Adult\_plants(1,:).\*Adult\_increase(2,:));

Adult\_plants(4,:) = 0;

% restrictions

Adult\_plants(3, Adult\_plants(3,:)>100) = max\_diameter;

Adult\_plants(3, Adult\_plants(3,:)<2) = 2;

for f = 1:1:max\_diameter

Adult\_plants(4, (Adult\_plants(3,f)-1)) = Adult\_plants(2,f) + Adult\_plants(4, (Adult\_plants(3,f)-1));

end

Adult\_plants(2,:) = Adult\_plants(4,:);

Adult\_plants(4,:) = [];

Adult\_plants(3,:) = [];

% collate all transitions

Histo\_individuals\_meta(4:histo\_length) = Adult\_plants(2,:);

Histo\_individuals\_meta(1)= seedbank;

Histo\_individuals\_meta(2)= to\_first\_year;

Histo\_individuals\_meta(3) = to\_second\_year;

Histo\_individuals\_meta(4) = Histo\_individuals\_meta(4)+ to\_adult\_loop;

% Note individuals at May of year +1

%% Minimum density correction

% make sure populations are not infinitively small, using all adults as

% one main class

Indi\_meta\_pop = sum(Histo\_individuals\_meta(4:histo\_length));

if Indi\_meta\_pop < sensi\_para.threshold

if Indi\_meta\_pop > adult\_test(year,population)\*sensi\_para.threshold

histo\_tmp = ((Histo\_individuals\_meta(4:histo\_length)./Indi\_meta\_pop)).\*sensi\_para.threshold;

Histo\_individuals\_meta(4:histo\_length) = histo\_tmp;

Histo\_stop = 0;

else

Histo\_individuals(:,population) = Histo\_individuals\_meta';

Histo\_individuals\_meta(4:histo\_length) = 0;

Histo\_stop = 1;

end

Indi\_meta\_pop = sum(Histo\_individuals\_meta(4:histo\_length));

end

% Idem Make sure there are not infinitively small number of seeds and

% seedlings

if Histo\_individuals\_meta(1) < sensi\_para.threshold

if Histo\_individuals\_meta(1) > seed\_test(year,population)\*sensi\_para.threshold

Histo\_individuals\_meta(1) = sensi\_para.threshold;

else

Histo\_individuals\_meta(1) = 0;

end

end

if Histo\_individuals\_meta(2) < sensi\_para.threshold

if Histo\_individuals\_meta(2) > class1\_test(year,population)\*sensi\_para.threshold

Histo\_individuals\_meta(2) = sensi\_para.threshold;

else

Histo\_individuals\_meta(2) = 0;

end

end

if Histo\_individuals\_meta(3) < sensi\_para.threshold

if Histo\_individuals\_meta(3) > class2\_test(year,population)\*sensi\_para.threshold

Histo\_individuals\_meta(3) = sensi\_para.threshold;

else

Histo\_individuals\_meta(3) = 0;

end

end

Indi\_meta\_pop\_juv = sum(Histo\_individuals\_meta(2:3));

%% Maximum density correction and lambda calculation

Individuals\_all = sum((Histo\_individuals\_meta(2:histo\_length)));

% Calculate lambda and mortality vs establishment

if Individuals\_all > 0

% correct for maximum density

if Indi\_meta\_pop >= max\_density(1)

histo\_tmp = ((Histo\_individuals\_meta(4:histo\_length)./Indi\_meta\_pop)).\*max\_density(1);

Histo\_individuals\_meta(4:histo\_length) = histo\_tmp;

clear histo\_tmp

Indi\_meta\_pop = sum(Histo\_individuals\_meta(4:histo\_length));

if reached == 0

lambda(population) = (Indi\_meta\_pop./ (Start\_individuals(population))) ^(1/((year+1)-1)) ;

end

reached(population) = 1;

elseif Indi\_meta\_pop < max\_density(2) && reached(population) == 1

reached(population) = 0; % reset the reached

end

if mortality(population) == 0

if Indi\_meta\_pop > 0

if reached(population) == 0

lambda(population) = (Indi\_meta\_pop./ (Start\_individuals(population))) ^(1/((year+1)-1)) ;

end

end

Colonisation(population) = 0;

elseif mortality(population) == 1

if Indi\_meta\_pop > 0

mortality(population) = 0;

Colonisation(population) = 1;

Start\_individuals(population) = sum(Histo\_individuals\_meta(4:histo\_length));

end

end

elseif Individuals\_all == 0 && mortality(population) == 0; % Dissapearing population

mortality(population) = 1;

Extinction(population) = 1;

end

if Histo\_stop == 0

Histo\_individuals(:,population) = Histo\_individuals\_meta';

end

Indi\_meta(population) = Indi\_meta\_pop;

Indi\_meta\_juv(population) = Indi\_meta\_pop\_juv;

Indi\_seeds(population) = Histo\_individuals\_meta(1);

end % end of per POPULATION

%% Metapopulation growth rates

total\_individual(year+1) = sum(Indi\_meta) ;

Individuals(year+1,:) = Indi\_meta;

Individuals\_seed(year+1,:) = Indi\_seeds; %#ok<NASGU>

Occupancy\_adult(year+1) = (length(find(Indi\_meta > 0)))./pop\_max;

Occupancy\_seeds(year+1) = (length(find(Indi\_seeds > 0)))./pop\_max;

if total\_individual(year+1) > 0 && (Occupancy\_adult(year+1) < 0.95 || pop\_max == 3)

occup\_meta = (Occupancy\_adult(year+1)./Occupancy\_adult(1))^(1/((year+1)-1));

end

if total\_individual(year+1) > 0 && (total\_individual(year+1) <max\_density(3) || pop\_max == 3)

if (year+1)>= 4

lambda\_meta = (sum(Indi\_meta)./sum(Individuals(4,:)))^(1/((year+1)-5));

else

lambda\_meta = (sum(Indi\_meta)./sum(Start\_individuals)) ^(1/((year+1)-1)) ;

end

if isinf(lambda\_meta) ~= 0

lambda\_meta = (sum(Indi\_meta)./sum(Start\_individuals)) ^(1/((year+1)-1)) ;

end

end

if total\_individual(year+1) > 0

if (year+1)>= 4

lambda\_metaY(year+1) = (sum(Indi\_meta)./sum(Individuals(year,:)));

else

lambda\_metaY(year+1) = NaN;

end

if isinf(lambda\_metaY(year+1)) ~= 0

lambda\_metaY(year+1) = NaN;

end

end

clear Seeds\_total

%% Extinction and Colonisation rates

if year == 1

Empty\_rate(year) = sensi\_para.zero\_densities./pop\_max;

Colonisation\_rate(year) = (1-sensi\_para.zero\_densities);

end

Empty\_rate(year+1) = (sum(mortality))./pop\_max;

Colonisation\_rate(year+1) = (sum(Colonisation))./pop\_max;

Extinction\_rate(year+1) = (sum(Extinction))./pop\_max; %#ok<NASGU>

end % end of per YEAR

cd temp

name\_file = ['output','\_',int2str(run),'.mat'];

save(name\_file,'lambda','Individuals','Histo\_individuals',...

'Occupancy\_adult','Occupancy\_seeds','Individuals\_seed',...

'Start\_individuals','lambda\_meta','Prop\_among',...

'sensi\_para','total\_individual', 'occup\_meta','Empty\_rate',...

'Colonisation\_rate','Extinction\_rate','lambda\_meta2','lambda\_metaY')

cd ..

5.3 The seed flow per year submodule – redistributing the seeds among populations

Seed flow per year based on earlier calculated proportions (see 5.4)

Seed\_present = Seeds\_total;

val = rand(pop\_max,4);

%% first wind dispersal; second human dispersal in that order

for population = 1:pop\_max

% lost

Seed\_flow\_tmp = Seed\_present(population).\*Prop\_among.wind\_lost;

Seed\_flow\_tmp(Seed\_flow\_tmp<val(population,1) & Seed\_flow\_tmp<1) = 0;

Seed\_flow\_tmp(Seed\_flow\_tmp>0 & Seed\_flow\_tmp<1) = 1;

Lost\_seeds(population)= sum(Seed\_flow\_tmp);

clear Seed\_flow\_tmp

Seed\_flow\_tmp = Seed\_present(population).\*Prop\_among.wind\_lost\_within\_source;

Seed\_flow\_tmp(Seed\_flow\_tmp<val(population,2) & Seed\_flow\_tmp<1) = 0;

Seed\_flow\_tmp(Seed\_flow\_tmp>0 & Seed\_flow\_tmp<1) = 1;

Lost\_within\_seeds(population)= sum(Seed\_flow\_tmp);

clear Seed\_flow\_tmp

Seed\_flow\_tmp = Seed\_present(population).\*Prop\_among.wind\_edge(population);

Seed\_flow\_tmp(Seed\_flow\_tmp<val(population,3) & Seed\_flow\_tmp<1) = 0;

Seed\_flow\_tmp(Seed\_flow\_tmp>0 & Seed\_flow\_tmp<1) = 1;

Edge\_seeds(population)= sum(Seed\_flow\_tmp);

clear Seed\_flow\_tmp

Seed\_flow\_tmp = Seed\_present(population).\*Prop\_among.wind\_export(population);

Seed\_flow\_tmp(Seed\_flow\_tmp<val(population,4) & Seed\_flow\_tmp<1) = 0;

Seed\_flow\_tmp(Seed\_flow\_tmp>0 & Seed\_flow\_tmp<1) = 1;

Export\_seeds(population)= sum(Seed\_flow\_tmp);

% Import

Seed\_flow\_tmp = Seed\_present(population).\*(Prop\_among.wind\_annulus(population,:));

Seed\_flow\_tmp(Seed\_flow\_tmp<val(population) & Seed\_flow\_tmp<1) = 0;

Seed\_flow\_tmp(Seed\_flow\_tmp>0 & Seed\_flow\_tmp<1) = 1;

Recieved\_seeds\_tmp(population,:)= Seed\_flow\_tmp;

clear Seed\_flow\_tmp

end

Recieved\_seeds = sum(Recieved\_seeds\_tmp,1);

Exported\_seeds = (Export\_seeds + Lost\_seeds + Edge\_seeds + Lost\_within\_seeds);

Seed\_present\_w = Seed\_present - Exported\_seeds + Recieved\_seeds;

clear val

%% human dispersal

% calculate the number of seeds that will be available on the patch and

% picked up.

prop\_area = sensi\_para.area\_size/((sensi\_para.area\_size)^2); % note the area is considered square and the path 1 m wide

total\_available = Seed\_present\_w.\* prop\_area.\* (sensi\_para.Pu.\* sensi\_para.pick\_up\_sensitivity);

val = rand(size(Prop\_among.human,1),1);

for population = 1:pop\_max

Seed\_flow\_tmp = total\_available(population)'.\*(Prop\_among.human(population,:));

Seed\_flow\_tmp(Seed\_flow\_tmp<val(population) & Seed\_flow\_tmp<1) = 0;

Seed\_flow\_tmp(Seed\_flow\_tmp>0 & Seed\_flow\_tmp<1) = 1;

Seed\_flow\_h(population,:)= Seed\_flow\_tmp;

Seed\_flow\_export(population) = Prop\_among.human\_export(population).\*total\_available(population);

Seed\_flow\_edge(population) = Prop\_among.human\_edge(population).\*total\_available(population);

clear test

clear Seed\_flow\_tmp

end

clear val

%Recalculate the seeds after dispersal

Recieved\_seeds\_h = sum(Seed\_flow\_h,1);

Exported\_seeds\_h = Seed\_flow\_export + Seed\_flow\_edge;

Seed\_present\_h = Seed\_present\_w - Exported\_seeds\_h + Recieved\_seeds\_h;

Seeds\_total = Seed\_present\_h;

clear All not needed % shortened to save text

5.4. The initial seed distribution matrix – run once at the start

Calculations of kernels and among population proportions of seed dispersal for wind and human dispersal. Including proportions to outside the meta-population

cd Results

load('pops&dispersal.mat')

cd ../

%% Create the combined wind dispersal kernel for area size intervals

%initiate matrix

clear Prop\_among

kernel = 0;

max\_distance = sensi\_para.area\_size\*(pop\_max-1);

%Make unique distance matrices

%Steps are distance centre

step = sensi\_para\_integral\_steps;

size = sensi\_para.area\_size;

% creating bins from Wichman et al. 2009, following Soons

% et al. 2004

tot\_bins = ceil(max(Wind\_dispersal\_Soons.extreme))./step; % so this is the lower edge;

prop\_disp(1:tot\_bins) = hist(Wind\_dispersal\_Soons.extreme,[tot\_bins])./50000;

prop\_disp((tot\_bins+1):step:(max\_distance.\*2)) = 0;

kernel.wind(1,:) = prop\_disp;

total = length(kernel.wind(1,:));

kernel.wind(2,1:total) = 0:step:(step\*(total-1));

%% Human mediated dispersal KERNEL following Wichmann et al. 2009

% create Kernel for size\_area intervals

% Make the attachment into a PDF, with the CDF adding to "a". By

% calculating which proportion of the original drops of during the patch.

% A small proportion beyond the populations disappears,

% if instead of x + area\_size "inf" is added, the cum\_sum of this

% PDF is 'a'.

Human\_disp\_a = 0.947; % power exponential from Wichmann et al 2009

Human\_disp\_b = 0.165; % power exponential from Wichmann et al 2009

fun\_num = @(x) Human\_disp\_a\*(exp(-(x.^Human\_disp\_b)));

% the proportion of seeds attached at start patch at distance x

for i = 1:1:(max\_distance.\*2)

x = i-step;

kernel.attached(1,i) = fun\_num(x);

kernel.attached(2,i) = x;

end

% Human dispersal is from the edge of the patch to a next patch, where

% it is dropped off at every interval. Hence the drop rate per

% interval is summed as integral with stepsize 'step', but no integral

% over x1 is taken, since only the end is important = size -step

% Within the patch follows the same logic, only seeds are picked off

% and dropped at every interval in the patch. This is seen as a

% directional movement over x-intervals, resulting in a value still attached at the end

% of the patch.

for i = 1:1:length(Distance\_matrix.oneD.centre)

count = 0;

x2 = (Distance\_matrix.oneD.centre(i)); % beginning of the patch

x3 = (Distance\_matrix.oneD.centre(i)+size)-step; % end of the patch

for j = step:step:(size)

x1 = (Distance\_matrix.oneD.centre(1)+size)-j; % distance from source

dist\_1 = find(kernel.attached(2,:)==(x2-x1));

dist\_2 = find(kernel.attached(2,:)==(x3-x1));

if dist\_1 ~= dist\_2

count = count +1;

disp(count) = (kernel.attached(1,dist\_1) - kernel.attached(1,dist\_2));

% Note no correction for the annulus, since this is one

% directional already

end

end

kernel.dispersal\_human(i) = sum(disp)/count;

clear j

clear disp

end

%% calcute the human dispersal matrix throughout the meta\_population from Kernel

for i = 1:pop\_max

for y = 1:1:pop\_max

x = Distance\_matrix.centre(i,y);

if x > 0

Prop\_among.human(i,y) = kernel.dispersal\_human(find(Distance\_matrix.oneD.centre == x));

end

end

end

% calculate two sided disappearances

Prop\_among.human\_export = sum(Prop\_among.human,1);

% exported is to two sites so for non edge patches it does double, this is

% correct since seeds that are dropped within the patch can have a

% tertiary dispersal in another direction.

human\_edge\_1 = max(Prop\_among.human\_export)-Prop\_among.human\_export;

human\_inf = fun\_num((max(max(Distance\_matrix.centre)))+step);

Prop\_among.human\_edge = human\_edge\_1 + human\_inf;

clear disapearance\_tmp

clear x

clear i

clear y

%% Wind dispersal following Soons et al. 2004

% as 4-way integral over every x and every y of source and target.

maximum\_distance = ((ceil(max(Wind\_dispersal\_Soons.extreme)./size)+1));

if length(Distance\_matrix.oneD.centre) < maximum\_distance

maximum\_distance = length(Distance\_matrix.oneD.centre);

end

for i = 1:1:maximum\_distance

count = 0;

for x1 = 0:step:((size)-step)

for y1 = 0:step:((size)-step)

for x2 = Distance\_matrix.oneD.centre(i):step:((Distance\_matrix.oneD.centre(i)+size)-step)

for y2 = 0:step:((size)-step)

count = count + 1;

distance = sqrt(((x1-x2).^2) + ((y1-y2).^2));

distance = (ceil(distance./step)).\*step;

dist = find(kernel.wind(2,:)==distance);

disp(count) = kernel.wind(1,dist)./((size.\*(1/step)).^2);

% correction for number of x, y points of source

% note not of target, since those need to be summed

% over the 144 values there

if distance ~= 0

disp\_annu(count) = disp(count) \* (step./((2\*pi).\*distance)); % correction for annulus

end

end

end

end

end

clear x1

clear x2

clear y1

clear y2

kernel.dispersal\_wind\_annulus(i) = sum(disp\_annu);

kernel.dispersal\_wind\_lost(i) = (sum(disp)./(size\*(1/step)))-(2\*kernel.dispersal\_wind\_annulus(i));

clear disp

clear disp\_annu

end

kernel.dispersal\_wind\_annulus(i+1:pop\_max+1) = 0;

kernel.dispersal\_wind\_lost(i+1:pop\_max+1) = 0;

%% calculate the wind dispersal matrix throughout the meta\_population from Kernel

for i = 1:pop\_max

for y = 1:1:pop\_max

x = Distance\_matrix.centre(i,y);

if x > 0

Prop\_among.wind\_annulus(i,y) = kernel.dispersal\_wind\_annulus((find(Distance\_matrix.oneD.centre == x)));

if x == size

Prop\_among.wind\_annulus(i,y) = Prop\_among.wind\_annulus(i,y);

end

end

end

Prop\_among.wind\_lost = sum(kernel.dispersal\_wind\_lost((1+1):length(kernel.dispersal\_wind\_lost)));

clear wind\_edge\_1

clear wind\_edge\_2

end

clear x

clear i

clear y

% calculate two sided disappearances

disapearance\_tmp = sum( Prop\_among.wind\_annulus,1);

Prop\_among.wind\_edge = ((1./(disapearance\_tmp./max(disapearance\_tmp)))-1).\*(sum(kernel.dispersal\_wind\_annulus(2:length(kernel.dispersal\_wind\_annulus))));

clear wind\_edge\_a

clear wind\_edge\_b

clear disapearance\_tmp

Prop\_among.wind\_lost\_within\_source = 1-(sum(kernel.dispersal\_wind\_annulus) + Prop\_among.wind\_lost + …

kernel.wind(1,1));

Prop\_among.wind\_export = sum(Prop\_among.wind\_annulus,1);

%% Exclusions

% Exclude human dispersal if asked to.

if strcmpi('y',sensi\_para.human\_dispersal\_present) == 0

Prop\_among.human(:,:) = 0;

Prop\_among.human\_export(:) = 0;

Prop\_among.human\_edge(:) = 0;

end

% Exclude wind dispersal if asked to.

if strcmpi('y',sensi\_para.wind\_dispersal\_present) == 0

Prop\_among.wind\_annulus(:,:) = 0;

Prop\_among.wind\_edge(:) = 0;

Prop\_among.wind\_lost = 0;

Prop\_among.wind\_lost\_within\_source = 0;

end

5.5. The per year per population parameter draw submodule

Life cycle parameters draws per population per year.

Overall population parameters (K, OH, W) are pre-assigned (not shown)

function draws = Oleracea\_draws(run, population,populations,Histo\_sizes,...

sensi\_para,draws,population\_year,Do\_sensitivity,max\_diameter,sensi,histo\_length)

% read base parameters per population (see 5.6)

PSB\_mean = populations.(genvarname(['population','\_',int2str(population)])).PSB\_mean;

Gs\_mean\_undis =populations.(genvarname(['population','\_',int2str(population)])).Gs\_mean\_all(1,:);

Gs\_mean\_dis =populations.(genvarname(['population','\_',int2str(population)])).Gs\_mean\_all(2,:);

Ss\_mean\_undis =populations.(genvarname(['population','\_',int2str(population)])).Ss\_mean\_all(1,:);

Ss\_mean\_dis =populations.(genvarname(['population','\_',int2str(population)])).Ss\_mean\_all(2,:);

Adult\_surv\_mean = populations.(genvarname(['population','\_',int2str(population)])).Adult\_surv\_mean\_all;

Adult\_increase\_mean = populations.(genvarname(['population','\_',int2str(population)])).Adult\_increase\_mean\_all;

Flowering\_like\_mean = populations.(genvarname(['population','\_',int2str(population)])).Flowering\_like\_mean\_all;

Fecundity\_mean\_fn = populations.(genvarname(['population','\_',int2str(population)])).Fecundity\_mean\_fn\_all;

Fecundity\_mean\_c = populations.(genvarname(['population','\_',int2str(population)])).Fecundity\_mean\_c\_all;

%% Seedbank

% function needs to be within limits 0-100%

SBincorp = PSB\_mean(1);

if SBincorp < 0

SBincorp = 0;

end

if SBincorp > 1

SBincorp = 1;

end

% function needs to be within limits 0-100%

SBrate = PSB\_mean(2);

if SBrate < 0

SBrate = 0;

end

if SBrate > 1

SBrate = 1;

end

clear PSB\_mean

clear PSB\_std

%% Emergence and seedling survival

for f= 1:1:length(Gs\_mean\_dis)

% function needs to be within limits 0-100%

Gs\_dis(f) = Gs\_mean\_dis(f);

if Gs\_dis(f) < 0

Gs\_dis(f) = 0;

end

if Gs\_dis(f) > 1

Gs\_dis(f) = 1;

end

end

clear GS\_mean\_dis

clear GS\_std\_dis

for f= 1:1:length(Gs\_mean\_undis)

% function needs to be within limits 0-100%

Gs\_undis(f) = Gs\_mean\_undis(f);

if Gs\_undis(f) < 0

Gs\_undis(f) = 0;

end

if Gs\_undis(f) > 1

Gs\_undis(f) = 1; %#ok<\*AGROW>

end

end

clear addition

clear GS\_mean\_dis

clear GS\_std\_dis

for f= 1:1:length(Ss\_mean\_dis)

% function needs to be within limits 0-100%

Ss\_dis(f) = Ss\_mean\_dis(f);

if Ss\_dis(f) < 0

Ss\_dis(f) = 0;

end

if Ss\_dis(f) > 1

Ss\_dis(f) = 1;

end

end

clear Ss\_mean\_dis

clear Ss\_std\_dis

Ss\_dis(5) = Ss\_dis(3);

for f= 1:1:length(Ss\_mean\_undis)

% function needs to be within limits 0-100%

Ss\_undis(f) = Ss\_mean\_undis(f);

if Ss\_undis(f) < 0

Ss\_undis(f) = 0;

end

if Ss\_undis(f) > 1

Ss\_undis(f) = 1;

end

end

clear addition

clear Ss\_mean\_dis

clear Ss\_std\_dis

Ss\_undis(5) = Ss\_undis(3);

%% Adult survival likelihoods

s\_max = 2;

% function needs to be within limit below 100%

while s\_max > 1

Bc = Adult\_surv\_mean(population\_year,1);

B1 = Adult\_surv\_mean(population\_year,2);

B2 = Adult\_surv\_mean(population\_year,3);

for i = 1:1:max\_diameter

x = log10(Histo\_sizes(i,1)+1);

Adult\_surv(2,i) = Bc + (B1\*(x)) + (B2\*(x^2));

Adult\_surv(1,i) = Histo\_sizes(i,1);

if Adult\_surv(2,i) < 0

Adult\_surv(2,i) = 0;

end

if Adult\_surv(2,i) >1

Adult\_surv(2,i) = 1;

end

end

s\_max = max(Adult\_surv(2,:));

end

clear All not needed % shortened to save text

%% Adult size increase

s\_max = 2; % think about limits, if applicable

while s\_max >1

Bc = Adult\_increase\_mean(population\_year,1);

B1 =Adult\_increase\_mean(population\_year,2);

for i = 1:1:max\_diameter

x = log10((Histo\_sizes(i,1))+1);

Adult\_increase(2,i) = (Bc + (B1\*(x)))+1; % Note that % increase rate is transforned to % rate (so + 1)

Adult\_increase(1,i) = Histo\_sizes(i,1);

end

s\_max = 0.5;

end

clear All not needed % shortened to save text

%% Likelihood of flowering

% function needs to be within limits of 0 and 100%

s\_max = 2;

s\_min = -1;

while s\_max > 1 || s\_min <0

Bc = Flowering\_like\_mean(population\_year,1);

B1 = Flowering\_like\_mean(population\_year,2);

for i = 1:1:max\_diameter

x = Histo\_sizes(i,1);

Flowering\_like(2,i) = 1/(1+(exp(Bc+(B1.\*x))));

Flowering\_like(1,i) = x;

end

s\_max = max(Flowering\_like(2,:));

s\_min = min(Flowering\_like(2,:));

end

clear All not needed % shortened to save text

%% Fecundity

% Function has a lower limit of 0 and no maximum

s\_min = -1;

while s\_min <0

for i = 1:1:max\_diameter

x = Histo\_sizes(i,1);

Fecundity(1,i) = Histo\_sizes(i,1);

% Constant function

Fecundity(2,i) = -1;

while Fecundity(2,i) < 0

Fecundity(2,i) = Fecundity\_mean\_c(population\_year);

end

end

s\_min = min(Fecundity(2,:));

end

%% Sensitivity

% change targetted parameters with a certain fraction

if exist('Do\_sensitivity') ~= 0

if strcmpi('y',Do\_sensitivity) ~= 0

Oleracea\_Sensitivy\_runss;

end

end

%% Make it arrays per population

draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Flowering\_like = Flowering\_like;

draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Fecundity = Fecundity;

draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).SBincorp = SBincorp;

draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).SBrate = SBrate;

draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Gs\_undis = Gs\_undis;

draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Ss\_undis = Ss\_undis;

draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Gs\_dis = Gs\_dis;

draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Ss\_dis = Ss\_dis;

draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Adult\_surv = Adult\_surv;

draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).Adult\_increase = Adult\_increase;

draws.(genvarname(['r',int2str(run)])).(genvarname(['p',int2str(population)])).histo\_length = histo\_length;

5.6. An example parameter definition file

Example of a definition function file, multiple variations are possible depending on the goal

%% Set Base variables

pop\_max = 1000;

run\_max = 250;

year\_max = 100;

sensi\_para.area\_size = 12;

sensi\_para\_integral\_steps = 1;

sensi\_para.wind\_dispersal\_present = 'Y';

sensi\_para.human\_dispersal\_present = 'Y';

sensi\_para.pick\_up\_sensitivity = 1;

sensi\_para.threshold = 1;

sensi\_para.Pu = 0.023;

sensi\_para.show = ceil(100./pop\_max);

sensi\_para.proportion\_disturbance = 0.25; % add here the proportion disturbance per site

sensi\_para.all\_years = 'N'; % all years together (yes) or draw from single years (N)

sensi\_para.mortality\_disturbance\_Adult = 0;

sensi\_para.zero\_densities = 1-(0.35);

sensi\_para.baseline = 1; % baseline run for sensitivity calculations

Do\_sensitivity = 'N';

sensi\_para.change\_factor = 1;

% % set histogram diagram

Histo\_base = zeros(max\_diameter,pop\_max);

Histo\_sizes(:,1) = 2:(max\_diameter+1);

%% Set distances

% create 1D pop\_max populations, at area\_size distance apart, starting with 0.

Distances\_centre = 0:sensi\_para.area\_size:(sensi\_para.area\_size\*(pop\_max-1));

for i = 1:pop\_max

for y = 1:1:pop\_max

Distance\_matrix.centre(i,y)= abs(Distances\_centre(i)-Distances\_centre(y));

end

end

Distance\_matrix.oneD.centre = Distances\_centre;

clear Distances\_centre

clear i

clear y

%% create populations

cd Results

load('pops&dispersal.mat')

cd ../

mean\_densities = mean(Dorset\_populations.densities,2);

mean\_densities = mean\_densities./(prctile(mean\_densities,95));

mean\_densities(mean\_densities>1) = 1;

histo = hist(mean\_densities(mean\_densities>0),size\_steps);

histo(2,:) = 1/size\_steps:(1/size\_steps):1;

pdf\_pops = histo(1,:)/sum(histo(1,:));

cdf\_pops = cumsum(pdf\_pops);

if sensi ~= sensi\_para.baseline

cd temp

name\_file = ['sensitivity\_factors','\_',int2str(run),'.mat'];

load(name\_file)

cd ..

end

count\_zero = 0;

again = 1;

while again ~= 0

for population = 1:1:pop\_max

type\_draw = ceil(rand\*3);

if sensi ~= sensi\_para.baseline

type\_draw = types\_draw(population);

else

types\_draw(population) = type\_draw;

end

hist\_draw = min(find(cdf\_pops>= rand));

if sensi ~= sensi\_para.baseline

hist\_draw = hist\_draws(population);

else

hist\_draws(population) = hist\_draw;

end

zero\_draw = rand;

if sensi ~= sensi\_para.baseline

zero\_draw = zero\_draws(population);

else

zero\_draws(population) = zero\_draw;

end

if type\_draw == 1

populations.(genvarname(['population','\_',int2str(population)])) = population\_types.Kim;

elseif type\_draw == 2

populations.(genvarname(['population','\_',int2str(population)])) = population\_types.OH;

elseif type\_draw == 3

populations.(genvarname(['population','\_',int2str(population)])) = population\_types.Win;

end

if zero\_draw <= sensi\_para.zero\_densities

populations.(genvarname(['population','\_',int2str(population)])).Histo\_base =...

populations.(genvarname(['population','\_',int2str(population)])).Histo\_base.\*0;

count\_zero = count\_zero +1;

else

populations.(genvarname(['population','\_',int2str(population)])).Histo\_base = ...

populations.(genvarname(['population','\_',int2str(population)])).Histo\_base.\*histo(2,hist\_draw);

end

end

zero\_rate = count\_zero./pop\_max;

if zero\_rate < (sensi\_para.zero\_densities - (1/pop\_max)) || zero\_rate > (sensi\_para.zero\_densities + ...

(1/pop\_max)) || zero\_rate == 1

again = 1;

zero\_rate = 0;

count\_zero = 0;

else

again = 0;

end

end

%% set changed parameters: EXAMPLE

if sensi == 1

sensi\_para.proportion\_disturbance = 0.4; % add here the proportion disturbance per site

elseif sensi== 2

% .............

end

if sensi == sensi\_para.baseline

if strcmpi('y',sensi\_para.all\_years) ~= 0

population\_years (year\_max) = 7;

else

population\_years = randi([1,6],year\_max);

end

cd temp

name\_file = ['sensitivity\_factors','\_',int2str(run),'.mat'];

save(name\_file,'population\_years', 'types\_draw', 'hist\_draws','zero\_draws')

cd ..

end