ADDENDUM 5

MATLAB AND R-CODE

ADDENDUM 5

MATLAB AND R-CODE

This addendum includes the most relevant Matlab and R-scripts developed during the course of this research and has 5 subdivisions with code related to chapters 2 to 6.

Text comments are in italics and gray font and are preceded by % for the Matlab code and by # for the R-code. The script is followed by '…' when the Matlab code continues on the next line.

CODE RELATED TO CHAPTER 2: SPECIES ASSEMBLY RULES

This paragraph summarises the Matlab code for the two swapping algorithms for presence/absence data. The scripts for calculating the co-occurrence indices for the original data matrix are not shown here, since they can be easily derived from the code below.

The code concerns only the feeding type 1A. Adapting the code for the other feeding types or all the data can be easily achieved by replacing '1A' by another feeding type or by 'AllData'.

Swap 1 for presence/absence data

```
%% data needed
% replicodes_1A = file with three columns about data of feeding type 1A
   % name of the replicate// number of the sample// number of replicates in the sample
% matrix_1A = presence/absence matrix with
   % replicates in rows and species in columns

for aantal=1:10              % aantal = number of files with 'aantalRand' null models
    aantalRand=100;              % number of null models in one loop ('aantal')
    % load data
    load('replicodes_1A'); load('matrix_1A');
    replicodes_nieuw=replicodes_1A;
    matrix_nieuw=matrix_1A;
    clear matrix_1A; clear replicodes_1A;
    aant=int2str(aantal);
    [X,Y]=size(matrix_nieuw);
    % Calculate row number in p/a matrix where a new sample starts
    N=1; x=1;
    while x<X
            aantalReps=replicodes_nieuw{1,3}(x);
            rijen(N)=x;
            N=N+1;
            x=x+aantalReps;
    end
```

```
samples=N-1; aantalRep=X; aantalSpec=Y; probleem=0;
 resultaat_null=zeros(aantalRand,8);
 rij=1; rij1=1;
for rand=1:aantalRand
      for n=1:samples % repeat swapping algorithm for all samples
            aantalReps=replicodes_nieuw{1,3}(rijen(n));
            % submatrix = data of 1 sample with 'aantalReps' replicates
                  submatrix=[];
            submatrix=matrix_nieuw(rijen(n):(rijen(n)+aantalReps-1),:);
            MM=1;
            % delete species which do not appear in the sample
             kolomleeg=zeros(Y,1);
             for y=Y:-1:1
                        if submatrix(1:aantalReps,y)==zeros(aantalReps,1)
                        submatrix(:,y)=[];
                        kolomleeg(y,1)=1; % remember the empty species columns
                   end
            end
            [A,B]=size(submatrix);
            swaps=500;
            sw=0;
            aantalSwaps=0;
            if A>1 && B>1     % try swaps if the remaining matrix contains more
                                    % than 1 replicate and more than 1 species
                  while sw<swaps % try 1000 swaps
                        % choose randomly 2 replicates and 2 species
                        randomNr1 = randint(2,1,[1,A]);
                        randomNr2 = randint(2,1,[1,B]);
                        % check if 2 different species and 2 different replicates are chosen
                        if (randomNr1(1)~= randomNr1(2)) && …
                              (randomNr2(1)~= randomNr2(2))
                              sw = sw+1;
                              % check if matrix is like [0 1; 1 0] or [1 0; 0 1]
                              if ((submatrix(randomNr1(1),randomNr2(1))== ...
                                          submatrix(randomNr1(2),randomNr2(2))) && ...
                                          (submatrix(randomNr1(1),randomNr2(2))== ...
                                          submatrix(randomNr1(2),randomNr2(1))) && ...
                                          (submatrix(randomNr1(1),randomNr2(1))~= ...
                                          submatrix(randomNr1(1),randomNr2(2))))
                                    % perform swap
                                    submatrix(randomNr1(1),randomNr2(1)) = ...
                                          abs(submatrix(randomNr1(1),randomNr2(1))-1);
                                    submatrix(randomNr1(1),randomNr2(2)) = ...
                                          abs(submatrix(randomNr1(1),randomNr2(2))-1);
                                    submatrix(randomNr1(2),randomNr2(1)) = ...
                                          abs(submatrix(randomNr1(2),randomNr2(1))-1);
                                    submatrix(randomNr1(2),randomNr2(2)) = ...
```

```
                    abs(submatrix(randomNr1(2),randomNr2(2))-1);
                aantalSwaps=aantalSwaps+1;
            end
        end         %if
    end         %1000 swap attempts
end
% repair the submatrix by adding absent species in submatrix
Mr=1; NN=1;
submatrix2=[];
for y=1:Y
    if kolomleeg(y,1)==1
        submatrix2(1:aantalReps,y)=zeros(aantalReps,1);
    else submatrix2(1:aantalReps,y)=submatrix(1:aantalReps,NN);
        NN=NN+1;
    end
end
clear kolomleeg
if NN-1~=B
    probleem=1;
 end
% put the swapped replicates back in the overall matrix
matrixNull(rijen(n):(rijen(n)+aantalReps-1),:)=submatrix2;
clear submatrix2; clear submatrix;
end

%% calculate the indices on the randomised data matrix
    [X,Y]=size(matrixNull); N=1; x=1;
    Comb=Y*(Y-1)/2; % number of species pairs
% C-score
    NCU=1; CU=0; Tsc=0; Scr=0; checker=0;
    for y1=1:(Y-1)              % species y1
    for y2=(y1+1):Y   % species y2 (different from species y1)
        % C
        ri=sum(matrixNull(:,y1));
        rj=sum(matrixNull(:,y2));
            % calculate how many times species y1 and y2 appear together in a
            % replicate (S)
            som=matrixNull(:,y1)+matrixNull(:,y2);
            verschil=abs(matrixNull(:,y1)-matrixNull(:,y2));
            S=sum(som-verschil)/2;
            CU=CU+(ri-S)*(rj-S);
            Tsc=Tsc + S*(aantalRep+S-ri-rj);        %T
            Scr=Scr+S;                              %S
             if max((som))==1                       %checker
            checker = checker+1;
            end
            NCU=NCU+1;
```

```
            end
            end
            Cscore=CU/(Y*(Y-1)/2);
        Tscore=Tsc/(Y*(Y-1)/2);
        Sscore=Scr/(Y*(Y-1)/2);
        % V-score
        ST2=0;
        Tgem=sum(sum(matrixNull))/aantalRep;
            for j=1:aantalRep
            Tj=sum(matrixNull(j,:));
            ST2=ST2+(Tj-Tgem)^2;
            end
        ST2=ST2/aantalRep;
        sigmai2=0;
        for i=1:aantalSpec
            ni=sum(matrixNull(:,i));
            sigmai2=sigmai2+(ni/aantalRep)*(1-(ni/aantalRep));
        end
        Vscore=ST2/sigmai2;
        clear matrixNull
        resultaat_null(rand,1)=Cscore;
        resultaat_null(rand,2)=Tscore;
        resultaat_null(rand,3)=Sscore;
        resultaat_null(rand,4)=Vscore;
        resultaat_null(rand,5)=checker;
    end
    fileN1=['resultaat_STCV_null_Swap1_1A_' aant];
    save(fileN1,'resultaat_null')
    clear
end
```

Swap 2 for presence/absence data

In this paragraph only the code about the swapping algorithm is shown. The rest of the code is equal to what is shown above.

```
[A,B]=size(submatrix);
swaps=1000;
sw=0;
aantalSwaps=0;
if A>1 && B>1
    while sw<swaps
        % choose randomly 2 replicates and 1 species
        randomNr1 = randint(2,1,[1,A]);
        randomNr2 = randi(B);
        % check if it concerns 2 different replicates
```

```
        if (randomNr1(1)~= randomNr1(2))
            sw = sw+1;
            w1=submatrix(randomNr1(1),randomNr2);
            w2=submatrix(randomNr1(2),randomNr2);
            % perform swap
             submatrix(randomNr1(1),randomNr2) = w2;
            submatrix(randomNr1(2),randomNr2) = w1;
            % check if no empty replicates are present
            % if one of the replicates contains no species, restore the submatrix
            if sum(submatrix(randomNr1(1),:))==0 || ...
                sum(submatrix(randomNr1(2),:))==0
                submatrix(randomNr1(1),randomNr2) = w1;
                submatrix(randomNr1(2),randomNr2) = w2;
            end
            clear w1; clear w2;
        end             % if
    end                 % 1000 swap attempts
end
```

Histograms

```
groep='1A'                          % define feeding type
Nrandom=100;                        % number of null models in one file
NrandomTot=1000;                    % total number of null models
fileN1=['resultaat_ori_' groep '.mat'];     % values of original data matrix
load(fileN1)

% load all values of null models
for aantal=1:10
    aant=int2str(aantal)
    fileN1=['result_random_' aant '_' groep];
    load(fileN1)
    resultaat((aantal-1)*Nrandom+1:aantal*Nrandom,:)=resultaat_null;
    clear resultaat_null
end

fid = fopen('labels_hor_as.txt');
labelslijst = textscan(fid,'%s');       % names of labels of x-axis
fclose(fid);

% calculate the two-sided 95% confidence interval of the 1000 null models
quant97=ceil(NrandomTot*97.5/100);
[X,Y]=size(resultaat);
resultaat(1000,:)=[];
for kolom=1:Y
    B=sort(resultaat(:,kolom));
    grens(kolom,1)=B(quant97);
```

```
    clear B
end
for kolom=1:Y
    B=sort(resultaat(:,kolom),'descend');
    grens2(kolom,1)=B(quant97);
    clear B
end

for kolom=1:Y
    % define minimum and maximum value of x-axis
    maximum=max(resultaat(:,kolom));
    minimum=min(resultaat(:,kolom));
    if resultaat_ori(kolom)>maximum
            maximum=resultaat_ori(kolom);
    end
    if resultaat_ori(kolom)<minimum
            minimum=resultaat_ori(kolom);
    end
    maximum=(maximum-minimum)/20 + maximum;
    minimum= -(maximum-minimum)/20 + minimum;
    hist(resultaat(:,kolom));          % plot histogram
    h = findobj(gca,'Type','patch');
    set(h,'FaceColor',[0.7,0.7,0.7],'EdgeColor',[0.3,0.3,0.3])
    h2 = findobj(gca,'Type','axes');
    set(h2,'FontSize',30)
    set(gca,'xlim',[minimum maximum]);
    hold on
    x=resultaat_ori(kolom)*ones(1,11);
    y=0:20:200
    x2=grens(kolom,1)*ones(1,11);
    y2=0:20:200
    x3=grens2(kolom,1)*ones(1,11);
    y3=0:20:200
    % plot original value on histogram
    plot(x,y,'color','k','LineWidth',3)
    hold on
    % plot upper border of 95% CI on histogram
    plot(x2,y2,'color','k','LineWidth',3,'LineStyle','--')
    hold on
    % plot lower border of 95% CI on histogram
    plot(x3,y3,'color','k','LineWidth',3,'LineStyle','--')
    xlabel(labelslijst{1,1}{kolom,1},'FontSize',38)
    ylabel('Frequency','FontSize',36)
    fname=['hist_dens_' int2str(kolom) '_' groep '_190411.jpg' ];
    print('-r100','-djpeg',fname)
    hold off
end
```

CODE RELATED TO CHAPTER 3: ARTIFICIAL NEURAL NETWORKS

Find optimal number of neurons

The optimal network transfer functions and learning methods where at this point already defined. Finding the optimal total network can be realised by adding two more loops (one for the transfer function and one for the learning method) within the neuron loop.

```
%% calculate neural networks for different neurons
% samples with geographical coordinates (replicate ID, latitude, longitude)
stations = load('stations_011007_4.csv');
% data matrix with environmental variables in columns,replicates in rows
Pfile = load('omgeving011007_4.csv');
% target: target values (biodiversity indices) in columns, replicates in rows
Tfile = load('target_compl_sampleIndependent_011007_4.csv');
% file with stratified data: column 8 assigns fold for replicate
fanny=load('fanny_clustering_011007_4.csv');
[Pr,Pk]=size(Pfile);
coeff = 0.5;        % minimum correlation coefficient to retain neural network
neuronStart=1; neuronInterval=1; neuronEind=6;
datum=date;             % date is used in file name
NZdata=Pfile;
[Ta1,Ta2]=size(Tfile);

% Preprocess environmental variables: mean = 0 and standard deviation = 1
P1 = NZdata.';
[P2,P2info] = mapstd(P1);
% reduce number of environmental variables by principal component analysis
% keep only those variables which contribute more than 1% of the variation % in the data
[P2,P2infoPCA] = processpca(P2,0.01);
[P2R,P2K]=size(P2);
 Tst=Tfile.';
folds=10;
for index=1:Ta2                     % find neural networks for every target (diversity index)
    indx=int2str(index);
    N=1;
    result=zeros(1000,26);
    T1=Tst(index,:);
    ind=int2str(index);
    [T2,T2info] = mapminmax(T1);            % transform target to interval [-1 1]
    for neuron = neuronStart:neuronInterval:neuronEind      % find # neurons
            for fold=1:folds        % tenfold cross-validation
            fld=int2str(fold);
            clear P; clear M; clear val; clear T; clear rij; clear test;
            P=P2; T=T2; M=1; Mt=1;
            % split up data in three sets: test, validation & training set
```

```
for rij=Pr:-1:1
      if fanny(rij,9)==fold
            val.P(:,M)=P2(:,rij);
            P(:,rij)=[];
             val.T(1,M)=T(1,rij);
            T(:,rij)=[];
            M=M+1;
            elseif fanny(rij,9)==fold+1 || (fold==10 && fanny(rij,9)==1)
            test.P(:,Mt)=P2(:,rij);
            P(:,rij)=[];
            test.T(1,Mt)=T(1,rij);
            T(:,rij)=[];
            Mt=Mt+1;
            end
end
for keer=1:20                    % construct 20 neural networks
            kr=int2str(keer);
            % NET = NEWFF creates a new network: first transfer function = tansig
            % transfer function of ultimate layer = purelin
            net1=newff(minmax(P), [neuron, 1],{'tansig', 'purelin'},'trainlm');
            net1.trainParam.goal=0.0005;
            net1.trainParam.max_fail=10;
            % train network
            [net, tr] = train(net1, P, T,[],[],val,test);
            % simulate data with network
            aL=sim(net,P);
            % backtransform output of network to original range
            aL=mapminmax('reverse',aL,T2info);
            TL=mapminmax('reverse',T,T2info);
            % calculate performance parameters for training data
            RL= corrcoef(aL,TL);
            rSpearmanL = corr(aL.',TL.','type','spearman');
            nL = length(aL);
            RMSEL= sqrt((1/nL)*sum((aL-TL).^2));
            RMSELgem=sqrt((1/nL)*sum((aL-TL).^2))/mean(T1);
            RSEL= sqrt(sum((aL-TL).^2)/sum((TL-mean(T1)).^2));
            EL= (1/nL)*sum((aL-TL).^2);

      % calculate performance parameters for independent test data
            aT=sim(net,test.P);
            aT=mapminmax('reverse',aT,T2info);
            testT=mapminmax('reverse',test.T,T2info);
            RT= corrcoef(aT,testT);
            rSpearmanT = corr(aT.',testT.','type','spearman');
            nT = length(aT);
            RMSET= sqrt((1/nT)*sum((aT-testT).^2));
            RMSETgem=sqrt((1/nT)*sum((aT-testT).^2))/mean(T1);
```

```
                RSET= sqrt(sum((aT-testT).^2)/sum((aT-mean(T1)).^2));
                ET= (1/nT)*sum((aT-testT).^2);

                % calculate performance parameters for all data
                aAlles=sim(net,P2);
                aAlles=mapminmax('reverse',aAlles,T2info);
                Ralles= corrcoef(aAlles,T1);
                rSpearmanalles = corr(aAlles.',T1.','type','spearman');
                nalles = length(aAlles);
                RMSEalles= sqrt((1/nalles)*sum((aAlles-T1).^2));
                RMSEallesgem=sqrt((1/nalles)*sum((aAlles-T1).^2))/mean(T1);
                RSEalles= sqrt(sum((aAlles-T1).^2)/sum((aAlles-mean(T1)).^2));
                Ealles= (1/nalles)*sum((aAlles-T1).^2);

                % test normality of residuals
                residuelen=aAlles-T1;
                [hL pL lL cL]=lillietest(residuelen);
                if lL>cL
                NormResi=0;
                else NormResi=1;
                end
                result(N,1)=index;          result(N,2)=neuron;
                result(N,3)=fold;               result(N,4)=keer;
                result(N,5)=RL(2,1);        result(N,6)=rSpearmanL;
                result(N,7)=RMSEL;          result(N,8)=RMSELgem;
                result(N,9)=RSEL;           result(N,10)=EL;
                result(N,11)=RT(2,1);       result(N,12)=rSpearmanT;
                result(N,13)=RMSET;         result(N,14)=RMSETgem;
                result(N,15)=RSET;          result(N,16)=ET;
                result(N,17)=Ralles(2,1);       result(N,18)=rSpearmanalles;
                result(N,19)=RMSEalles;         result(N,20)=RMSEallesgem;
                result(N,21)=RSEalles;      result(N,22)=Ealles;
                result(N,23)=NormResi;          result(N,24)=cL;
                result(N,25)=lL;                result(N,26)=pL;
                N=N+1;
                % save the network for later use
                nm=['Res\netT_' datum '_K' kr '_F' fld '_T' indx '.mat'];
                save(nm,'net');
                clear net, clear tr, clear net1, clear av,
            clear Rv, clear nv
        end             % keer
        end                 % fold
end                     % neuron
name = [Res\result' indx '_' datum];
save(name,'result')
clear result; clear T1; clear ind; clear T1tr; clear transformatie;
clear indexTr; clear T; clear T2info;
```

```
end       % target
```

Perturb

```
stations = load('stations.csv');  % load station data
Pfile = load('omgeving.csv');         % load environmental data
Tfile = load('target.csv');         % load target data
neuron=2; folds=10;               % best model has 2 neurons & 10-fold cross-validation was
applied
intervallen=20;                   % number of intervals considered in variable contribution
Bestes=10;                        % Find the ten best models
NZdata=Pfile;
[NZ1,NZ2]=size(NZdata);
P1 = NZdata.';
 [P2,P2info] = mapstd(P1);            % preprocess environmental variables
[P2,P2infoPCA] = processpca(P2,0.01);
targets=16;
for target=1:targets
      target
      ind=int2str(target);
      trgt=int2str(target);
      datumF='24-Oct-2007';
      name=['Res\result_T' trgt '_N2_' datumF];
      load(name);
      clear name;
      temp=result;
      temp(:,36)=temp(:,13);
      % find the ten best neural networks based on RMSE of test set
      [Te1,Te2]=size(temp);
      for fold=1:10
                  min=10000;
                  for te1=1:Te1
                  if temp(te1,3)==fold && temp(te1,36)<min
                              min=temp(te1,36);
                              besteNN(fold,:)= temp(te1,:);
                  end
                  end
      end

      % add increasing levels of noise to the variable of interest and monitor
      % the effect on the output
      for fold=1:10
            beste=1;
            bst=int2str(beste);
            keer=besteNN(fold,4);
            fld=int2str(fold);
            krOK=int2str(keer);
            naamN=['Res\netT_' datumF '_K' krOK '_F' fld '_T' trgt '_N2'];
```

```
        load(naamN)
        clear naamN;
        Tst=Tfile.';
        T1=Tst(target,:);
        [T2,T2info] = mapminmax(T1);
        for nz2=1:NZ2                % do this loop for every environmental variable
            NZdata=Pfile;
            variabele=NZdata(:,nz2);
            MIMA=minmax(variabele.');
            % add different levels of noise to the environmental variable under
            % consideration
            for noise=1:(intervallen+1)
                    ruis=randn(NZ1,1).';
                    [ruis2,ruisInfo]=mapminmax(ruis);
                    ruis2T=ruis2.';
                var4=variabele+ruis2T*(MIMA(2)-MIMA(1))*(noise-1)/(2*intervallen);
                    var4=mapminmax(var4.',MIMA(1),MIMA(2)).';
                    NZdata(:,nz2)=var4;
                    P1 = NZdata.';
                    P2 = mapstd('apply',P1,P2info);
                    P2=processpca('apply',P2,P2infoPCA);
                    % simulate output with altered variables data
                    a=sim(net,P2);
                    % backtransform
                    a=mapminmax('reverse',a,T2info);
                    Rt= corrcoef(a,T1);
                    rSpearman = corr(a.',T1.','type','spearman');
                    nt = length(a);
                    % calculate root mean squared error on output
                    RMSEt= sqrt((1/nt)*sum((a-T1).^2));
                    resultNoise(target,nz2,noise,fold)=RMSEt;
            end
            end
    end
end
datum=date;
naamR=['Res\resultNoise_somVLT_10folds_N2' datum];
naamB=['Res\besteNN_somVLT_10folds_N2' datum];
save(naamR, 'resultNoise')
save(naamB, 'besteNN')
datum='06-Nov-2007';
naamR=[Res\resultNoise_somVLT_10folds_N2' datum];
load(naamR)

[P1,P2]=size(Pfile);
[T1,T2]=size(Tfile);
[N1,N2]=size(resultNoise);
```

```
 % calculate the relative increase in RMSE by adding noise to each
% environmental variable
for target=1:targets                    % loop for targets (biodiversity indices)
    for noise=1:(intervallen+1)             % loop for level of noise
            for nz2=1:NZ2                       % loop for each environmental variable
            for fold=1:10               % loop for each fold
                matrix4(target,nz2,noise,fold)= (resultNoise(target,nz2,noise,fold)- ...
                resultNoise(target,nz2,1,fold))*100/resultNoise(target,nz2,1,fold);
            end
            end
    end
end

% calculate the average increase for each target and each variable
for target=1:targets
    for var=1:NZ2
        kolom=1
        for fold=1:10
                temp(1:5,kolom)=matrix4(target,var,17:21,fold);
                kolom=kolom+1;
        end
        [Te1,Te2]=size(temp);
        temp2 = reshape(temp,Te1*Te2,1);
        gemiddelde(var,target)=mean(temp2);
        standdev(var,target)=std(temp2);
        clear temp; clear temp2;
    end
end
ondergrens=gemiddelde-standdev;
for target=1:targets
    somT=sum(gemiddelde(:,target));
    gemiddeldeP(:,target)=gemiddelde(:,target)*100/somT;
    standdevP(:,target)=standdev(:,target)*100/somT;
    clear temp; clear temp2;
end
ondergrensP=gemiddeldeP-standdevP;
save('Res\SENSperturb_LVT_B10F_gemP','gemiddeldeP')
save('Res\SENSperturb_LVT_B10F_stdP','standdevP')
```

Profile

The first part of the code is identical to the first part of the previous paragraph and will not be repeated here.

```
% load data
% preprocess environmental variables
 [V1,W1]= size(P1);
```

```
perc = 100*(0:0.2:1);       % define percentile values
[PE1,PE2]=size(perc);
for v1=1:V1                 % Calculate percentile values for each environmental variable
    percentiles = prctile(P1(v1,:),perc);
    waarden(v1,1:PE2)=percentiles;
end
profile=zeros(NZ2,abN+1,targets,Bestes);
for target=1:targets
    target
    ind=int2str(target);
    trgt=int2str(target);
    datumF='24-Oct-2007';
    % load result file from previous analysis (first paragraph)
    name=['Res\result_T' trgt '_N2_' datumF];
    load(name);
    clear name;
    [Res1,Res2]=size(result);
    temp=result;
    temp(:,36)=temp(:,13)
     % find the ten best neural networks based on RMSE of test set
    [Te1,Te2]=size(temp);
    for fold=1:10
            min=10000;
            for te1=1:Te1
            if temp(te1,3)==fold && temp(te1,36)<min
                min=temp(te1,36);
                    besteNN(fold,:)= temp(te1,:);
            end
            end
    end
    for fold=1:Folds
        [RE1,RE2] =size(temp);
        keer=besteNN(fold,4);
        fld=int2str(fold);
        krOK=int2str(keer);
        % load the network corresponding with the best result
        naamN=['Res\netT_' datumF '_K' krOK '_F' fld '_T' trgt '_N2'];
        load(naamN)
        clear naamN;
        Tst=Tfile.';
        T1=Tst(target,:);
         [T2,T2info] = mapminmax(T1);
        [V,W]= size(P1);
        % create dataset where 1 environmental variable changes
            for vari=1:V        % do this for all environmental variables
            m=waarden(vari,1);
            M=waarden(vari,PE2);
```

```
            abI=(M-m)/abN;
            for interv=1:(abN+1)  % do this for 'abN' intervals
                  NZab=waarden;
                  Interval=m+abI*(interv-1);
                  % keep percentile values of all environmental variables
                  % replace data of 1 variable by constant value
                  NZab(vari,:)=Interval*ones(1,PE2);
                  NZab = mapstd('apply',NZab,P2info);
                  NZab = processpca('apply',NZab,P2infoPCA);
                  % calculate output for altered environmental variables matrix
                  aNZab=sim(net,NZab);
                  aNZ=mapminmax('reverse',aNZab,T2info);
                  profile(vari,interv,target,fold)=mean(aNZ);
            end         %intervals
            end               % variables
      end
end                     %target
save('Res\SENSprofile_10BF','profile')
load('Res\SENSprofile_10BF')
% calculate influence of environmental variables
for target=1:targets
      for beste=1:Bestes
            prof=profile(:,:,target,beste);
            extremen=minmax(prof);
            verschil=extremen(:,2)-extremen(:,1);
            resultaat(:,target,beste)=verschil;
            clear verschil;
      end
end
save('Res\SENSprofile_10BF_res','resultaat')
load('Res\SENSprofile_10BF_res')

% convert result to percentages
somK=sum(resultaat,1)
for target=1:targets
      for beste=1:10
            resultaatP(:,target,beste)=resultaat(:,target,beste)*100./somK(1,target,beste);
      end
end
gemiddeld=mean(resultaatP,3);
standdev=std(resultaatP, 0, 3);
ondergrens=gemiddeld-standdev;
[PR1,PR2,PR3,PR4]=size(profile);

% check if influence of environmental variable on diversity index is positive or negative
for target=1:targets
      for beste=1:10
```

```
            prof=profile(:,:,target,beste);
        extremen=minmax(prof);
            for vari=1:PR1
            MAX=NaN; MIN=NaN;
            for interv=1:(abN+1)
                    if profile(vari,interv,target,beste)==extremen(vari,1)
                    MIN=interv;
                    elseif profile(vari,interv,target,beste)==extremen(vari,2)
                    MAX=interv;
                    end
            end
            if MAX - MIN<0
                posneg(vari,target,beste)=-1;
            elseif MAX - MIN>0
                posneg(vari,target,beste)=1 ;
            else posneg(vari,target,beste)=NaN;
            end
            end
    end
end
save('Res\SENSprofile_10BF_posneg','posneg')

for vari=1:V1
    for target=1:targets
            result10B(vari, target,1)=mean(resultaat(vari,target,:));
            result10B(vari, target,2)=std(resultaat(vari,target,:));
    end
end
save('Res\SENSprofile_10BF_gemiddelde','result10B')

for vari=1:V1
    for target=1:targets
            resultPN10B(vari, target,1)=mean(posneg(vari,target,:));
            resultPN10B(vari, target,2)=std(posneg(vari,target,:));
    end
end
save('Res\SENSprofile_10BF_posnegGem','resultPN10B')
```

Modified Profile

In this paragraph only the part where the code differs from the original 'Profile method' is shown.

```
% check if influence of environmental variable on diversity index
% is positive or negative
% create environmental dataset where 1 environmental variable changes
% but the other variables are kept at their original value
```

```
for vari=1:V        % do this for all environmental variables
    mM=minmax(P1(vari,:));
    abI=(mM(2)-mM(1))/abN;
    for interv=1:(abN+1)                    % do this for 'abN' intervals
            NZab=P1;
            Interval=mM(1)+abI*(interv-1);
            NZab(vari,:)=Interval*ones(W,1);        % change the values of 1
                                                    % environmental variable
            NZab = mapstd('apply',NZab,P2info);
            NZab = processpca('apply',NZab,P2infoPCA);
            aNZab=sim(net,NZab);
            aNZ=mapminmax('reverse',aNZab,T2info);
            waarde(vari,interv,fold,target)=mean(aNZ);
    end         %intervals
    end             % variables
```

CODE RELATED TO CHAPTER 4: GEOSTATISTICS

In this paragraph only the R-code for the generalised least squares model is shown. More Matlab code is developed to transform output data of the geostatististical models to arcgis maps, but this is not shown in this chapter.

Generalised least squares model

This paragraph contains R-code instead of Matlab Code!

```
rm(list = ls())
getwd()
setwd("D:\\Bmerckx\\MyMatlab\\kaart_MM_SB\\meio_art")
library(nlme)
meio=read.table("rekenR2.txt", header=TRUE)          # load training data
validatie=read.table("valR.txt",header=TRUE)         # load validation data
dimensies=dim(meio)
meio2=scale(meio[,1:(dimensies[2])])                 # rescale both matrices
dimensies2=dim(validatie)
center2=mat.or.vec(dimensies2[1], dimensies2[2])
scale2=mat.or.vec(dimensies2[1], dimensies2[2])
for (rij in 1:dimensies2[1]){
    center2[rij,]=attr(meio2,"scaled:center")[1:dimensies2[2]]
    scale2[rij,]=attr(meio2,"scaled:scale")[1:dimensies2[2]]
    validatie2=(validatie - center2)/scale2
}

meio2=as.data.frame(meio2)
attach(meio2)
meio.train=meio2
```

```
# choose which column to model
index = 2
divIndex= meio.train[,11+index]

## before variable selection
d502 <- d50^2
mud2 <- mud^2
TSM_max2 <- TSM_max^2
TSM_min2 <- TSM_min^2
TSM_mean2 <- TSM_mean^2
chl_max2 <- chl_max^2
chl_min2 <- chl_min^2
chl_mean2 <- chl_mean^2
diepte2 <- diepte^2
meio.gls=gls(divIndex~ E+N+d50+mud+TSM_mean+TSM_max+TSM_min+chl_mean
      + chl_max+chl_min+diepte+year+d502+mud2+TSM_mean2
      + TSM_max2+TSM_min2+chl_mean2+chl_max2+chl_min2+diepte2
      + d50:mud + d50:TSM_mean + d50:TSM_min + d50:chl_mean
      + d50:chl_min + d50:diepte + mud:TSM_mean
      + mud:TSM_min + mud:chl_mean + mud:chl_min + mud:diepte,
      cor=corSpher(form= ~EL_UTMuniek+NB_UTMuniek),data=meio.train)
summary(meio.gls)

# after variable selection
mud2=mud^2
meio.gls=gls(divIndex~ mud + TSM_mean + mud2,
      cor=corSpher(form= ~EL_UTMuniek+NB_UTMuniek),data=meio.train)
summary(meio.gls)

# residual analysis
er=resid(meio.gls)
er[abs(er)>3*sd(er)]
e1=er[abs(er)<3*sd(er)]
shapiro.test(e1)

## validation
attach(meio2)
meio2$mud2=meio2$mud^2
meio.lm.predR=predict(meio.gls,newdata=meio2)
meio.lm.residR=residuals(meio.gls,newdata=meio2)
attach(validatie2)
validatie2$mud2=validatie2$mud^2
meio.lm.predV=predict(meio.gls,newdata=validatie2)
meio2=scale(meio[,1:(dimensies[2]-1)])
predictionV=
```

```
    meio.lm.predV*attr(meio2,"scaled:scale")[index+11]+attr(meio2,"scaled:center")[index
+11]
predictionR=
    meio.lm.predR*attr(meio2,"scaled:scale")[index+11]+attr(meio2,"scaled:center")[index
+11]
write.csv(predictionV, "meioGlsV_ES25.csv")
write.csv(predictionR, "meioGlsR_ES25.csv")

# calculate quality parameters of prediction
meio.lm.MSPE=mean((meio.lm.pred-validatie2[,11+index])^2)
meio.lm.MSPE
meio.lm.spear=cor(meio.lm.pred,validatie2[,11+index],method="spearman")
meio.lm.pearson= cor(meio.lm.pred,validatie2[,11+index],method="pearson")
meio.lm.spear
meio.lm.pearson
MSE[subst,index] = meio.lm.MSPE
spear[subst,index] = meio.lm.spear
pear[subst,index] = meio.lm.pearson
```

CODE RELATED TO CHAPTER 5: NULL MODELS MAXENT

In this paragraph only the code to create random subsets with a minimum distance between them will be given. The code concerning model selection in Maxent can be found in the next paragraph.

Create five subsets with a minimum distance between these subsets

```
% read station data with three columns:
   % speciesnumber  // Easting (UTM_coordinates) // Northing (UTM_coordinates)
lijst=csvread('maxent_lijst.csv');
fid = fopen('specieslist2.txt');        % read text file with 1 column: species names
soortenlijst = textscan(fid,'%s'); fclose(fid);
distances=[5000, 10000];            % choose for which distances you want to create subsets
NDist=length(distances); [X,Y]=size(lijst);
[Sp1,Sp2]=size(soortenlijst{1,1});
soortenlijst=soortenlijst{1,1};
for dist=1:NDist                    % create cross-validation files for different distances
    mindist=distances(dist);
    mdint=round(mindist/1000);
    md=int2str(mdint);
    for species=1:Sp1                   % create different  files for each species
            minATOK=0;
            N=1;
            tempSp=[];
        for rij=1:X                 % find data for each species
            if lijst(rij,1)==species
                    tempSp(N,:)=lijst(rij,:); N=N+1;
```

```
    end
end
stations=tempSp; [S1,S2]=size(stations); [T1,T2]=size(tempSp);
pogingen=0; geslaagd=0;
% create subsets with 1 element, which are at least 5 or 10 km separated from each
% other
temp1OK=[]; temp2OK=[]; temp3OK=[]; temp4OK=[]; temp5OK=[];
    while pogingen<1000 && geslaagd==0  % if necessary try 1000 times to make
                                        % subsets
    coord=tempSp(:,2:3);
    random2=randperm(T1);
    aantalSt=T1;
    temp1=[]; temp2=[]; temp3=[]; temp4=[]; temp5=[];
    aantal=1; reserve=0; aantalT=0;
    % create 5 subsets, each with 1 station and at least 'mindist' distance apart
    while aantal<6 && aantalT<T1
        aantalT=aantal+reserve;
        duo=coord(random2(aantalT),1:2);
            afstand1nieuw=0; afstand2nieuw=0;
        afstand3nieuw=0; afstand4nieuw=0; afstand5nieuw=0;
        success=0;
        if isempty(temp1)
            temp1(1,:)=duo; aantal=aantal+1; success=1;
        end
        afstand1nieuw = sqrt((temp1(1,1)-duo(1))^2+(temp1(1,2)-duo(2))^2);
        if isempty(temp2)&& ~isempty(temp1) && afstand1nieuw > mindist
            temp2(1,:)=duo; aantal=aantal+1; success=1;
        end
        if ~isempty(temp2)
            afstand2nieuw = sqrt((temp2(1,1)-duo(1))^2+(temp2(1,2)-duo(2))^2);
        end
        if isempty(temp3)&& ~isempty(temp2) && afstand2nieuw > …
            mindist && afstand1nieuw > mindist
            temp3(1,:)=duo;  aantal=aantal+1; success=1;
        end
        if ~isempty(temp3)
            afstand3nieuw = sqrt((temp3(1,1)-duo(1))^2+(temp3(1,2)-duo(2))^2);
        end
        if isempty(temp4)&& ~isempty(temp2) && ~isempty(temp3) && …
            afstand3nieuw > mindist && afstand2nieuw > mindist && ...
            afstand1nieuw > mindist
            temp4(1,:)=duo;  aantal=aantal+1; success=1;
        end
        if ~isempty(temp4)
            afstand4nieuw = sqrt((temp4(1,1)-duo(1))^2+(temp4(1,2)-duo(2))^2);
        end
        if isempty(temp5) && ~isempty(temp2) && ~isempty(temp3) && ...
```

```
            ~isempty(temp4) && afstand4nieuw > mindist && afstand3nieuw > ...
            mindist && afstand2nieuw > mindist && afstand1nieuw > mindist
            temp5(1,:)=duo;  aantal=aantal+1; success=1;
      end
      if success==0
            reserve=reserve+1;
      end
end
for t1=T1:-1:1          % delete the assigned stations from list
            duo=coord(t1,1:2);
            if isequal(temp1,duo) || isequal(temp2,duo) || …
            isequal(temp3,duo) || isequal(temp4,duo) || isequal(temp5,duo)
            coord(t1,:)=[];
            end
end
aantal=5; aantal2=1; reserve=0; aantalT=0; pogingsets=0;
r1=2; r2=2; r3=2; r4=2; r5=2;
random3=randperm(T1-5);
RG1=1;
reservegroep=[];
% add randomly stations to the 5 subsets while considering the minimum
% distance constraints
while aantal<=T1 && aantalT<T1-5
      aantalT=aantal2+reserve;
      duo=coord(random3(aantalT),1:2);  % choose a random pair from
                                        % coordinate list
      mdeerste(1:5)=ones(1,5)*10^20;
      success=0;
      [d11,d12]=size(temp1);
      % calculate distance
      for r=1:d11
            afstand=sqrt((temp1(r,1)-duo(1))^2+(temp1(r,2)-duo(2))^2);
            if afstand<mdeerste(1)
                  mdeerste(1)=afstand;
            end
      end
            [d21,d22]=size(temp2);
      for r=1:d21
            afstand=sqrt((temp2(r,1)-duo(1))^2+(temp2(r,2)-duo(2))^2);
            if afstand<mdeerste(2)
                  mdeerste(2)=afstand;
            end
      end
       [d31,d32]=size(temp3);
      for r=1:d31
            afstand=sqrt((temp3(r,1)-duo(1))^2+(temp3(r,2)-duo(2))^2);
            if afstand<mdeerste(3)
```

```
            mdeerste(3)=afstand;
        end
    end
    [d41,d42]=size(temp4);
    for r=1:d41
        afstand=sqrt((temp4(r,1)-duo(1))^2+(temp4(r,2)-duo(2))^2);
        if afstand<mdeerste(4)
                mdeerste(4)=afstand;
        end
    end
    [d51,d52]=size(temp5);
    for r=1:d51
        afstand=sqrt((temp5(r,1)-duo(1))^2+(temp5(r,2)-duo(2))^2);
        if afstand<mdeerste(5)
            mdeerste(5)=afstand;
        end
    end
    %for which subset is the minimum distance found
        Min1=min(mdeerste);
        Min2=10^10;
        for k=1:5
        if mdeerste(k)>Min1 && mdeerste(k)<Min2
                Min2=mdeerste(k);
                m2=k;
        elseif mdeerste(k)==Min1
                m1=k;
        end
    end
    tr=zeros(1,5);
    if Min1==0
        reserve=reserve+1;
            %  if the pair is distant enough from all sets,
            % assign to a back-up group
        elseif Min1>mindist
        reservegroep(RG1,1:2)=duo;
        RG1=RG1+1;
        aantal2=aantal2+1;
    % if the distance of newly chosen pair falls within 'mindist' range
    % from one subset, but is not too close to any other of the four sets
    %  then assign the pair
    elseif Min1<=mindist && Min1>0 && Min2>mindist
        if m1==1 && r1<(ceil(aantalSt/5) + 1)
            temp1(r1,:)=duo; r1=r1+1; success=1; aantal2=aantal2+1;
            elseif m1==2 && r2<(ceil(aantalSt/5) + 1)
            temp2(r2,:)=duo; r2=r2+1; success=1; aantal2=aantal2+1;
            elseif m1==3 && r3<(ceil(aantalSt/5) + 1)
            temp3(r3,:)=duo; r3=r3+1; success=1; aantal2=aantal2+1;
```

```
                elseif m1==4 && r4<(ceil(aantalSt/5) + 1)
                temp4(r4,:)=duo; r4=r4+1; success=1; aantal2=aantal2+1;
                elseif m1==5 && r5<(ceil(aantalSt/5) + 1)
                temp5(r5,:)=duo; r5=r5+1; success=1; aantal2=aantal2+1;
                else success=0;
                reserve=reserve+1;
                end
            elseif success==0;
            reserve=reserve+1;
            end
        end
        % assign the pair in the back-up group to the subset with the least data
        % and check if the 'mindist' condition still holds
        if ~isempty(reservegroep)
            [RGR,RGK]=size(reservegroep);
            for rijRG=1:RGR
                duo=reservegroep(rijRG,:);
                % calculate distance
                mdeerste(1:5)=ones(1,5)*10^20;
                [d11,d12]=size(temp1);
                for r=1:d11
                afstand=sqrt((temp1(r,1)-duo(1))^2+(temp1(r,2)-duo(2))^2);
                if afstand<mdeerste(1)
                    mdeerste(1)=afstand;
                end
                end
            [d21,d22]=size(temp2);
            for r=1:d21
                afstand=sqrt((temp2(r,1)-duo(1))^2+(temp2(r,2)-duo(2))^2);
                if afstand<mdeerste(2)
                    mdeerste(2)=afstand;
                end
            end
            [d31,d32]=size(temp3);
            for r=1:d31
                afstand=sqrt((temp3(r,1)-duo(1))^2+(temp3(r,2)-duo(2))^2);
                if afstand<mdeerste(3)
                    mdeerste(3)=afstand;
                end
            end
            [d41,d42]=size(temp4);
            for r=1:d41
                afstand=sqrt((temp4(r,1)-duo(1))^2+(temp4(r,2)-duo(2))^2);
                if afstand<mdeerste(4)
                        mdeerste(4)=afstand;
                end
            end
```

```
[d51,d52]=size(temp5);
for r=1:d51
    afstand=sqrt((temp5(r,1)-duo(1))^2+(temp5(r,2)-duo(2))^2);
    if afstand<mdeerste(5)
        mdeerste(5)=afstand;
    end
end
% find subset where the minimum distance is found
    Min1=min(mdeerste);
    Min2=10^10;
    for k=1:5
    if mdeerste(k)>Min1 && mdeerste(k)<Min2
            Min2=mdeerste(k);
            m2=k;
    elseif mdeerste(k)==Min1
            m1=k;
    end
    end

    % check the number of data in each subset
    aantalEl(1,1)=numel(temp1)/2; aantalEl(2,1)=numel(temp2)/2;
aantalEl(3,1)=numel(temp3)/2; aantalEl(4,1)=numel(temp4)/2;
aantalEl(5,1)=numel(temp5)/2;
    aantalEl(1:5,2)=1:5;
    aantalEl=sortrows(aantalEl,1);
% if the pair is distant enough from all sets, assign to the subset
% with the least data
if (Min1>mindist && Min1>0)
        nummer=aantalEl(1,2);
        switch nummer
        case 1
                temp1(d11+1,:)=duo;
        case 2
                temp2(d21+1,:)=duo;
        case 3
                temp3(d31+1,:)=duo;
        case 4
                temp4(d41+1,:)=duo;
        case 5
                temp5(d51+1,:)=duo;
        end
% if the distance of newly chosen pair falls within 'mindist' range
% from one subset, but is not too close to any other of the
% four sets then assign the pair
elseif (Min1<=mindist && Min1>0 && Min2>mindist)
        switch m1
        case 1
```

```
                        temp1(d11+1,:)=duo;
                    case 2
                        temp2(d21+1,:)=duo;
                    case 3
                        temp3(d31+1,:)=duo;
                    case 4
                        temp4(d41+1,:)=duo;
                    case 5
                        temp5(d51+1,:)=duo;
                    end
                end
            end
        end
        % check how many pairs have been assigned in total
        AT(1)=numel(temp1)/2; AT(2)=numel(temp2)/2;   AT(3)=numel(temp3)/2;
        AT(4)=numel(temp4)/2; AT(5)=numel(temp5)/2;
        minAT=min(AT);
        % if more data have been assigned to the 5 subsets then during the previous

        % attempt, then remember the new subsets
        if minAT>minATOK
            temp1OK=temp1; temp2OK=temp2; temp3OK=temp3;
                temp4OK=temp4; temp5OK=temp5; minATOK=minAT;
        end
        pogingen=pogingen+1;
    end

        clear temp1; clear temp2; clear temp3; clear temp4; clear temp5;
        temp1=temp1OK;      temp2=temp2OK; temp3=temp3OK; temp4=temp4OK;
    temp5=temp5OK;
        AT(1)=numel(temp1)/2; AT(2)=numel(temp2)/2; AT(3)=numel(temp3)/2;
    AT(4)=numel(temp4)/2; AT(5)=numel(temp5)/2;
        minAT=min(AT);
    geselecteerd=(numel(temp1)+numel(temp2)+numel(temp3)+numel(temp4)+...
        numel(temp5))/2;
    if pogingen==1000 && minATOK>1
        geslaagd=1;
        else geslaagd=0;
        end
        MiMa=minmax(AT);
        % subsets should have the same number of data ±1
        if MiMa(2)-MiMa(1)>1
        while numel(temp1)/2> MiMa(1)+1;
            RP=randperm(numel(temp1)/2);
            temp1(RP(1),:)=[];
        end
        while numel(temp2)/2> MiMa(1)+1;
```

```
        RP=randperm(numel(temp2)/2);
        temp2(RP(1),:)=[];
    end
    while numel(temp3)/2> MiMa(1)+1;
        RP=randperm(numel(temp3)/2);
        temp3(RP(1),:)=[];
    end
    while numel(temp4)/2> MiMa(1)+1;
        RP=randperm(numel(temp4)/2);
        temp4(RP(1),:)=[];
    end
    while numel(temp5)/2> MiMa(1)+1;
        RP=randperm(numel(temp5)/2);
        temp5(RP(1),:)=[];
    end
end
% save the 5 cross-validation files in .csv-format
    if geslaagd==1
    randSt=randperm(ceil(aantalSt/5));
    rand5=randperm(5); tllr=1;
    VR(1)=numel(temp1)/2; VR(2)=numel(temp2)/2; VR(3)=numel(temp3)/2;
    VR(4)=numel(temp4)/2; VR(5)=numel(temp5)/2;
    ST=0; VRS(1)=0;
    for su=1:5
        ST=VR(su)+ST;
        VRS(su+1)=ST;
    end
    temp=[];
    temp((VRS(1)+1):VRS(2),:)=temp1; temp((VRS(2)+1):VRS(3),:)=temp2;
    temp((VRS(3)+1):VRS(4),:)=temp3; temp((VRS(4)+1):VRS(5),:)=temp4;
    temp((VRS(5)+1):VRS(6),:)=temp5;
    AS=int2str(aantalSt);
    [T1,T2]=size(temp);
    for sub=1:5
        subs=int2str(sub);
        spec_name=[soortenlijst{species,1} '_' subs];
        spec_nameF=[soortenlijst{species,1}];
        testset(1:VR(sub),:)=temp((VRS(sub)+1):VRS(sub+1),:);
        tempx=temp;
        for rijx=VRS(sub+1):-1:(VRS(sub)+1)
            tempx(rijx,:)=[];
        end
        rekenset=tempx;
        file_name_test=['crossfilesAC_km' md '\' spec_name '_test' '.csv'];
        lijn1='species, dd long, dd lat';
        dlmwrite(file_name_test, lijn1, 'delimiter','');
        [X1,X2]=size(testset);
```

```
                    for rijT=1:X1
                        long=num2str(testset(rijT,1));
                        lat=num2str(testset(rijT,2));
                        lijn2=[spec_name ',' long ',' lat];
                        dlmwrite(file_name_test, lijn2, 'delimiter','','newline', 'pc', '-append');
                        clear lijn2; clear long; clear lat;
                    end
                    clear file_name_test
                    file_name_reken=['crossfilesAC_km' md '\' spec_name '_reken' '.csv'];
                    dlmwrite(file_name_reken, lijn1, 'delimiter','');
                    clear lijn1
                    [Y1,Y2]=size(rekenset);
                    for rijR=1:Y1
                        long=num2str(rekenset(rijR,1));
                        lat=num2str(rekenset(rijR,2));
                        lijn2=[spec_name ',' long ',' lat];
                        dlmwrite(file_name_reken, lijn2, 'delimiter','', 'newline', …
                        'pc', '-append');
                        clear lijn2; clear long; clear lat;
                    end
                    clear file_name_reken; clear spec_name; clear spec_nameF; clear subs;
                    clear testset; clear rekenset;
                end
            end
                lijstac(species,1)=species;
                lijstac(species,2)=geslaagd;
                if geslaagd
                lijstac(species,3)=sum(VR); lijstac(species,4)=VR(1); lijstac(species,5)=VR(2);
                lijstac(species,6)=VR(3); lijstac(species,7)=VR(4); lijstac(species,8)=VR(5);
                else
                lijstac(species,3)=0; lijstac(species,4)=0; lijstac(species,5)=0; lijstac(species,6)=0;
                lijstac(species,7)=0; lijstac(species,8)=0;
            end
        end
    end
```

CODE RELATED TO CHAPTER 6: FIND OPTIMAL HSM FOR DIFFERENT DENSITIES

In this paragraph the code for the forward selection of features of the Maxent models in shown.

```
% The only parameter which needs adjustment after one loop of variable selection is
completed is
% 'Reeks'.
```

```
Reeks=1;
varselectie=[int2str(Reeks) 'varFW'];
Nspecies=6; Nfeatures=5; TotaalAantalVar=9;
NrVar=Reeks+1;
fid = fopen('omgeving_SB.txt');
maxent_omgeving_asc = textscan(fid,'%s');
fclose(fid);

% open text file with one column: the species names
fid = fopen('specieslist.txt');
soortenlijst = textscan(fid,'%s');
fclose(fid);
teller=1;

% open text file with one column: the features names
fid = fopen('features.txt');
featuresF = textscan(fid,'%s');
fclose(fid);
Nfolds=4;

% name the frequencies for which you want to make models
% (100=code for RA)
ondergrenzen=[0,1,5,10,100];
Nthresholds=length(ondergrenzen);

% read the html output files of Maxent
auc_lijst=zeros(Nthresholds *Nfeatures*Nspecies*TotaalAantalVar*Nfolds,8);
for PA=1:Nthresholds                    % loop for models with different frequencies
    pa=int2str(ondergrenzen(PA));
    for features=1:Nfeatures        % loop for models with different features
                                    % (i.e. linear, quadratic)
        ftrs=int2str(features);
            for species=1: Nspecies         % loop for different species
                for var=1:TotaalAantalVar       % loop for different variables
                for fold=1:Nfolds           % loop for different folds in cross-validation
                    fld=int2str(fold);
                    folder=[int2str(NrVar-1) 'varFW\' pa 'PA\' ftrs  'Features\'
                    maxent_omgeving_asc{1,1}{var,1}(1:4)];
                    N=1;
                    spec_name=[soortenlijst{1,1}{species,1}];
                    spec_nameF=[folder '\' soortenlijst{1,1}{species,1}  '_' fld];
                     file_name_html=[folder '\' spec_name '_' fld '.html'];
                    % read the .html file
                    fid = fopen(file_name_html);
                    if fid>0
                            tekst = textscan(fid,'%s');
                            [T1,T2]=size(tekst{1,1});
```

```
                                trshld=tekst{1,1}{335,1};
                                [s1,s2]=size(trshld);
                                aanpassing=[];
                                for a=1:(100-s2)
                                aanpassing=[' ' aanpassing];
                                end
                                treshold(teller,1:100)=[tekst{1,1}{335,1}  aanpassing];
                                spec(teller)=species;
                                t1=(T1-200);
                                verderdoen=1;
                                % find the line concerning the AUC of test and training data
                                % in the file (the line is different for every file)
                                while verderdoen
                                if sum(size(tekst{1,1}{t1,1}))==9 && …
                                      sum(size(tekst{1,1}{t1+1,1}))==4 && ...
                                      strcmp('training', tekst{1,1}{t1,1}(1:8)) && ...
                                      strcmp('AUC', tekst{1,1}{t1+1,1}(1:3))
                                           verderdoen=0;
                                else t1=t1+1;
                                end
                                end
                                auc_test_temp=tekst{1,1}{t1+17,1};
                                auc_training_temp=tekst{1,1}{t1+3,1};
                                test_temp=str2double(auc_test_temp(1:5));
                                train_temp=str2double(auc_training_temp(1:5));
                                auc_lijst(teller,1)=teller;          auc_lijst(teller,2)=species;
                                auc_lijst(teller,3)=var;                  auc_lijst(teller,4)=fold;
                                auc_lijst(teller,5)=train_temp; auc_lijst(teller,6)=test_temp;
                                auc_lijst(teller,7)=PA;              auc_lijst(teller,8)=features;
                                teller=teller+1;
                                fclose(fid);
                        else
                                auc_lijst(teller,1)=teller;          auc_lijst(teller,2)=species;
                           auc_lijst(teller,3)=var;                  auc_lijst(teller,4)=fold;
                           auc_lijst(teller,5)=-1;              auc_lijst(teller,6)=-1;
                           auc_lijst(teller,7)=PA;              auc_lijst(teller,8)=features;
                                teller =teller+1;
                        end
                        clear trshld; clear file_name_html;
                        end
                end
            end
        end
end
csvwrite(['Nthresholds_auc_lijst_6spec_' varselectie '.csv'],auc_lijst)

%% calculate the average value for the fivefold cross-validation
```

```
data=auc_lijst;
[R,K]=size(data);
rij=1;
spec=data(rij,2);
resultaat=zeros(Nspecies*TotaalAantalVar*Nthresholds*Nfeatures,8);
rijr=1;
for PA=1:Nthresholds
    pa=int2str(ondergrenzen(PA));
    for features=1:Nfeatures
        ftrs=int2str(features);
        for species=1:Nspecies
            for var=1:TotaalAantalVar
            N=1;
            temp=[];
            for rijtje=1:R
                if auc_lijst(rijtje,7)==PA && auc_lijst(rijtje,8)==features && ...
                auc_lijst(rijtje,3)==var && auc_lijst(rijtje,2)==species
                temp(N,:)=data(rijtje,5:6)
                N=N+1;
                end
            end
            reken2=[]; test2=[];
            N1=1; N2=1; N3=1; N4=1;
            for t1=1:Nfolds
                if temp(t1,1)>0
                reken2(N1)=temp(t1,1);
                N1=N1+1;
                end
                if temp(t1,1)>0
                test2(N2)=temp(t1,2);
                N2=N2+1;
                end
            end
            resultaat(rijr,1)=species;          resultaat(rijr,2)=var;
            resultaat(rijr,3)=mean(reken2);     resultaat(rijr,4)=N1-1;
            resultaat(rijr,5)=mean(test2);      resultaat(rijr,6)=N2-1;
            resultaat(rijr,7)=PA;               resultaat(rijr,8)=features;
            rijr=rijr+1 ;
            end
        end
    end
end
csvwrite(['Nthresholds_auc_CV_6spec_' varselectie '.csv'],resultaat)

%% Find the variable which entails the highest average AUC-valuefor each threshold, each
% species and each feature
N=1;
```

```
[BB1,BB2]=size(resultaat);
for PA=1:Nthresholds
    pa=int2str(ondergrenzen(PA));
    for features=1:Nfeatures
            ftrs=int2str(features);
            for species=1:Nspecies
            NN=1
            for rijtje=1:BB1
                    if resultaat(rijtje,7)==PA && resultaat(rijtje,8)==features && ...
                    resultaat(rijtje,1)==species
                    temp(NN,1:8)=resultaat(rijtje,1:8);
                    NN=NN+1;
                    end
            end
            maxi=max(temp);
            M=1;
            per=randperm(TotaalAantalVar);
            for rijp=1:TotaalAantalVar
                    rij=per(rijp)
                    if temp(rij,5)==maxi(5) && M==1
                    selectie(N,1:2)=temp(rij,1:2);
                    selectie(N,3:4)=temp(rij,7:8);
                    selectie(N,5)=temp(rij,5);
                    N=N+1;
                    M=M+1;
                    elseif isnan(maxi(5)) && M==1
                    selectie(N,1:2)=temp(rij,1:2);
                    selectie(N,3:4)=temp(rij,7:8);
                    selectie(N,5)=NaN;
                    M=M+1; N=N+1;
                    end
            end
            end
    end
end
 csvwrite(['Nthresholds_auc_besteTestWaarde_6spec_' varselectie '.csv'],selectie)

%% read the results from the previous forward selection of the variables
for reeks=1:Reeks
    rks=int2str(reeks);
    selectie=csvread(['Nthresholds_auc_besteTestWaarde_6spec_' rks 'varFW.csv'])
    [Se1,Se2]=size(selectie);
    volledigeLijst(1:Se1,reeks+3)=selectie(1:Se1,2);
    volledigeLijstAUC(1:Se1,reeks+3)=selectie(1:Se1,5);
end
volledigeLijst(1:Se1,1)=selectie(1:Se1,3);
volledigeLijst(1:Se1,2)=selectie(1:Se1,4);
```

```
volledigeLijst(1:Se1,3)=selectie(1:Se1,1);
volledigeLijstAUC(1:Se1,1)=selectie(1:Se1,3);
volledigeLijstAUC(1:Se1,2)=selectie(1:Se1,4);
volledigeLijstAUC(1:Se1,3)=selectie(1:Se1,1);
[VL1,VL2]=size(volledigeLijst);

%% make a new list of variables to use in the model, thus with all the
% previously selected variables and one extra variable
N=1;
for PA=1:Nthresholds
    pa=int2str((PA));
    for features=1:Nfeatures
            ftrs=int2str(features);
            for species=1:Nspecies
            temp=[];
            for rij=1:VL1
                    if volledigeLijst(rij,1)==PA && volledigeLijst(rij,2)==features && ...
                    volledigeLijst(rij,3)==species
                    temp=volledigeLijst(rij,1:VL2);
                    temp2=volledigeLijstAUC(rij,1:VL2);
                    end
            end
            if ~isempty(temp)
                    for vars=1:TotaalAantalVar
                    OK=true;
                    for kolom=4:VL2
                            if vars==temp(1,kolom) || isnan(temp2(1,kolom))
                            OK=false;
                            end
                    end
                    if OK==true
                            lijst(N,1:VL2)=temp;
                            lijst(N,VL2+1)=vars;
                            N=N+1;
                    end
                end
            end
        end
    end
end

%% write the batch file with Maxent command lines, to automatically run the
% different models
 all_var=maxent_omgeving_asc;
[AV1,AV2]=size(all_var{1,1})
file_name_bat=['Nthresholds_maxent_6spec_' int2str(NrVar) 'FW.bat'] ;
[Sp1,Sp2]=size(soortenlijst{1,1});
```

```
N=1;
[Lij1,Lij2]=size(lijst);
for PA=1:Nthresholds
    pa=int2str(ondergrenzen(PA));
    mapCF=['crossfilesOndergrens\' pa '\'];
    for features=1:Nfeatures
            ftrs=int2str(features);
            featureStr=[];
            if features==1
            featureStr=[];
            else
            for fe=1:features
                featureStr=[featureStr ' ' featuresF{1,1}(fe,:)  ];
            end
            end
            for species=1:Nspecies
                LLL=1;
                tlijst=[];
                for rijl=1:Lij1
                if lijst(rijl,1)==PA && lijst(rijl,2)==features && lijst(rijl,3)==species
                        tlijst(LLL,:)=lijst(rijl,:);
                        LLL=LLL+1;
                end
            end
            if ~isempty(tlijst)
                    all_var=maxent_omgeving_asc;
                    specname=soortenlijst{1,1}(species,:)
                    lijstvar=1:TotaalAantalVar;
                    NrModel=TotaalAantalVar-NrVar+1
                    temp=[];
                    temp=tlijst(:,4:Lij2);
                    for rij=1:(TotaalAantalVar-NrVar+1)
                    eruit=1:TotaalAantalVar;
                    for vars=1:(Lij2-3)
                            eruit(temp(rij,vars))=0;
                    end
                    nieuwErin=temp(rij,Lij2-3);
                    omgevingStr=[];
                    for av2=1:AV1
                        if eruit(av2)>0
                            omgevingStr=['togglelayerselected='  …
                            all_var{1,1}{av2,1}(1:4) ' ' omgevingStr];
                        end
                    end
                    map=[int2str(NrVar) 'varFW\' pa 'PA\' ftrs 'Features\' ...
                    maxent_omgeving_asc{1,1}{nieuwErin,1}(1:4)];
                    mkdir(map)
```

```
                    spec=int2str(species);
                    for sub=1:Nfolds
                            sb=int2str(sub);
                            lijn1=['java -mx512m -jar maxent.jar ...
                            environmentallayers=C:\Bmerckx\nemspec4\kaarten_OK ' ...
                            omgevingStr 'samplesfile=C:\Bmerckx\nemspec4\' mapCF ...
                            specname{1,1} '_' sb '_reken.csv ...
                            testsamplesfile=C:\Bmerckx\nemspec4\' mapCF ...
                            specname{1,1} '_' sb '_test.csv ...
                            outputdirectory=C:\Bmerckx\nemspec4\' map  ' ' ...
                            featureStr ' noplots invisible nowarnings ...
                            nooutputgrids autorun'];
                        dlmwrite(file_name_bat, lijn1, 'delimiter', ...
                            '','newline', 'pc', '-append');
                        clear lijn1;
                    end
                    clear spec;
                    lijst1(N,1:NrVar) = temp(1:NrVar);
                    N=N+1;
                    end
                end
            end
    end
end
```