Electronic Supplementary Material 2 for

Elevated nonlinearity as indicator of abrupt change in the dynamics of populations under stress

Vasilis Dakos, Sarah M. Glaser, Chih-hao Hsieh, and George Sugihara

```
% MATLAB script to produce simulated data for the exploitation and agetruncation
% scenarios under environmental and process error
% Estimate CV, AR1 and Nonlinearity DeltaRho and produce basic plots
% 28 April 2016 by V Dakos

clear all
close all

% Stochasticity strength
sr0 = 0.1; % process error
se0 = 0.25; % environmental noise
su0 = 0; % observational noise

% Simulation settings
iter0 = 1000; % number of realisations
steps0 = 100; % length of timeseries

% age truncation scenario
rstart = 0.01; the
rend = 3;
r_values = linspace(rstart,rend,steps0); % vector of control parameter

% exploitation scenario
Fstart = 0;
Fend = 3;
F_values = linspace(Fstart,Fend,steps0); % vector of control parameter

% simulated data
X_agetruncation = Ricker_agetruncation(sr0,se0,su0,iter0,r_values);
X_exploitation = Ricker_exploitation(sr0,se0,su0,iter0,F_values);

##################################################################

function X = Ricker_exploitation(sr0,se0,su0,iter0,T0,initT0,F_values)

% RICKER_EXPLOITATION function to simulate timeseries to estimate AR, VAR and DeltaRho
% 28 April 2016 by V Dakos

% Initialise random number generator
rng('shuffle','twister')

% Simulation settings
T = T0;        % timesteps that are equal to the size of timeseries used
initT = initT0;      % timesteps to get rid of transients

iter = iter0; % number of stochastic realisations
steps = length(F_values); % steps for changing control parameter

% Parameters
r = 0.75; % growth rate
K = 10; % carrying capacity
F = 0; % grazing rate
P = 2; % exponent for sigmoid functional response
H = 0.75; % half-saturation rate

% Stochasticity strength
sr = sr0;     % process error on r
se = se0;     % environmental noise
su = su0;     % measurement error
```

```
% Solving Difference Equation
X = zeros(T,iter*steps);
counter = 1;

for i = 1:steps
    F = F_values(i);
    b = r/K;
    extra = 0;
    k = 1;
    while k <= iter
        x = K;
        for t = 1:initT+T
            r1 = r*exp(sr*randn(1)); % process error
            x = (x*exp(r1-b*x)-F*x^P/(H^P+x^P))*exp(se*randn(1));
            if x<0
                extra = extra+1;
                break
            end

            if t>initT % store values after discarding transient
                X(t-initT,counter) = x*exp(su*randn(1)); % observation error
            end
        end
        if x<0 % if negative solutions, discard and rerun realisation
            k = k;
            counter = counter;
        else
            k=k+1;
            counter = counter+1;
        end
    end
end

savdir = ‚~/save_directory/‚;
filename = sprintf('Ricker_changeF_sr%g_se%g.mat',sr,se);
save(fullfile(savdir,filename),'X','extra_iter'); % save data

#################################################################

function X = Ricker_agetruncation(sr0,se0,su0,iter0,T0,initT0,r_values)

% RICKER_AGETRUNCATION function to simulate timeseries to estimate AR, VAR and DeltaRho
% 28 April 2016 by V Dakos

% Initialise random number generator
rng('shuffle','twister')

T = T0;         % timesteps that are equal to the size of timeseries used
initT = initT0;      % timesteps to get rid of transients

iter = iter0; % number of stochastic realisations
steps = length(r_values); % steps for changing control parameter

% Parameters
r = 0.75; % growth rate
K = 10; % carrying capacity
F = 0; % grazing rate
P = 2; % exponent for sigmoid functional response
H = 0.75; % half-saturation rate

% Stochasticity strength
sr = sr0;     % process error on r
se = se0;     % environmental noise
```

```
su = su0;     % measurement error

% Solving Difference Equation
X = zeros(T,iter*steps);
counter = 1;

for i = 1:steps
    r = r_values(i);
    b = r/K;
    extra = 0;
    k = 1;
    while k <= iter
        x = K;
        for t = 1:initT+T
            r1 = r*exp(sr*randn(1)); % process error
            x = (x*exp(r1-b*x+se*randn(1))-F*x^P/(H^P+x^P)); % environmental noise
             if x<0
                extra = extra+1;
                break
            end

            if t>initT % store values after discarding transient
                X(t-initT,counter) = x*exp(su*randn(1)); % observation error
            end
        end
        if x<0 % if negative solutions, discard and rerun realisation
            k = k;
            counter = counter;
        else
            k = k+1;
            counter = counter+1;
        end
    end
end

savdir = ‚~/save_directory/‚;
filename = sprintf('Ricker_changeR_sr%g_se%g.mat',sr,se)
save(fullfile(savdir,filename),'X','extra_iter'); % save data

#################################################################

% Script to estimate nonlinearity indices using lnlp2010 (C file)

clear all
close all

dir_path = ‚~/read_directory/‚;

% Load original file (raw data)
scenario = {'exploitation', 'age_trunc'};

for i = 1:2
    disp(scenario{i})
    switch scenario{i}
        case {'exploitation'}
            % Scenario 1: exploitation
            filename = sprintf('Ricker_changeF_sr%g_se%g',sr0,se0)
            load([dir_path filename '.mat']);

            % first difference data and normalise
            X_diff = (diff(X));
            SD_diff = std(X_diff);
            AVG_diff = mean(X_diff);
            X_norm_diff = (X_diff - repmat(AVG_diff,T-1,1))./ repmat(SD_diff,T-1,1);
            % save normalised first differenced data
            n_filename = sprintf(['norm_diff_' filename '.txt'])
            dlmwrite([dir_path n_filename],X_norm_diff,'\t'); %% this is the data to use for the nonlinear forecasting

            YY = reshape(X_norm_diff,T-1,iter,steps); % reshape back to 3D matrix
            index_vector = [1:T-1]'; % time vector to add to the timeseries for lnlp

        case {'age_trunc'}
            % Scenario 2: Age truncation
            filename = sprintf('Ricker_changeR_sr%g_se%g',sr0,se0)
            load([dir_path filename '.mat']);

            % first difference data and normalise
            X_diff = (diff(X));
            SD_diff=std(X_diff);
            AVG_diff = mean(X_diff);
            X_norm_diff = (X_diff - repmat(AVG_diff,T-1,1))./ repmat(SD_diff,T-1,1);

            % save normalised first differenced data
            n_filename = sprintf(fullfile(['norm_diff_' filename '.txt']))
            dlmwrite([dir_path n_filename],X_norm_diff,'\t'); %% this is the data to use for the nonlinear forecasting

            YY = reshape(X_norm_diff,T-1,iter,steps); % reshape back to 3D matrix
            index_vector = [1:T-1]'; % time vector to add to the timeseries for lnlp
    end

    %% First estimate the best embedding dimension using simplex
    for ix = 1:steps
        for iy = 1:iter
            sprintf('control parameter = %d , iteration = %d',ix,iy)
            for Elist = 1:3
                data = [index_vector YY(:,iy,ix)];
                lib = [1 round(T/2)];
                pred = [round(T/2)+1 T];
                normType = 2;
                predType = 2; % 1 is for smap, 2 for simplex
                dumpFlag = 0;
                %       Elist = [1];
                Tau = 1;
                TP = 1;
                Blist = Elist+1;
                Theta_epsilon = 1000; % large number means it will include all neighbors
                flag1 = 0;

                [out] = lnlp2010(data, lib, pred,normType, predType, dumpFlag, Elist, Tau, TP, Blist, Theta_epsilon,
flag1);
                % out: a matrix [SampleSize, EmbeddingDimension, Tau, Tp, b, Theta (or Epsilon), MAE, RMSE, rho,
percent_correct, cMAE, cRMSE, crho, cperc]
                % c indicating constant predictor


                % find best embedding dimension
                % based on highest rho or lowest MAE
                rho(Elist,:) = [Elist out(7)];
                mae(Elist,:) = [Elist out(8)];
            end
            sorted_rho = sortrows(rho,2);
            sorted_mae = sortrows(mae,2);

            Ebest_list_rho(iy,ix) = sorted_rho(end,1);
            Ebest_list_mae(iy,ix) = sorted_mae(1,1);

        end
    end

    save([dir_path filename '_Ebest_list.mat'],'Ebest_list_rho','Ebest_list_mae')

    %% Second estimate nonlinear structure using SMAP
    load([dir_path filename '_Ebest_list.mat'],'Ebest_list_rho','Ebest_list_mae')

    normType = 2;
    predType = 1; % 1 is for smap, 2 for simplex
    dumpFlag = 0;
    Tau = 1;
    TP = 1;
    Blist = 1000; % will include all neighbors
    Theta_epsilon = [0 0.001 0.005 0.01 0.05 0.1 0.2 0.3 0.5 1 2 5]; % a vector of theta's
    flag1 = 0;

    for ix = 1:steps
       for iy = 1:iter
          sprintf('control parameter = %d , iteration = %d',ix,iy)

          % Elist = Ebest_list_rho(iy,ix); % use the highest rho
          Elist = Ebest_list_mae(iy,ix); % use the lowest MAE

          data = [index_vector YY(:,iy,ix)];
          lib = [1 round(T/2)];
          pred = [round(T/2)+1 T];

          [out] = lnlp2010(data, lib, pred,normType, predType, dumpFlag, Elist, Tau, TP, Blist, Theta_epsilon, flag1);

          % nonlinearity indices
          delta_rho = out(:,7) - out(1,7); %rho@theta==0 - best rho (max rho)
          delta_mae = out(1,8) - out(:,8); %MAE@theta==0 - best MAE (lowest MAE)
          [best_delta_rho(iy,ix), ind_rho] = max(delta_rho); % best rho from rho at theta=0
          [best_delta_mae(iy,ix), ind_mae] = max(delta_mae); % best mae from mae at theta=0
          theta_best_rho(iy,ix) = Theta_epsilon(ind_rho);
          theta_best_mae(iy,ix) = Theta_epsilon(ind_mae);

       end
    end

    save([dir_path filename
'_nonlinearity_DIFF_MEASURE_DELTA_RHO.mat'],'best_delta_rho','best_delta_mae','theta_best_rho','theta_be
st_mae')
end

#################################################################

function [out]=lnlp2010(data, lib, pred,normType, predType, dumpFlag, Elist, Tau, TP, Blist, Theta_epsilon, flag1)

% Wrapper function to call lnlp (by Chih-hao Hsieh 01/26/2010)
%
% input:
% data: a [nX2] matrix with first col indicating index
% lib:[initial end], a vector determining the initial and end data ID used as library
% pred:[initial end], a vector determining the initial and end data ID used as prediction
%    NOTE: if lib=pred, the software automatically leave E*Tau out, nan are automatically removed
% normType: 1 = L1 norm, 2 = L2 norm (Euclidean)
% predType: 1 = s-map, 2 = simplex projection, 3 = linear
% dumpFlag: 0 = no, 1 = yes, -1 = suppressing warning
% Elist: Embeding dimension, a vector
% Tp: time step forward in prediction, normally set to 1
% Blist: number of neighborhoods (can be a vector of numeric or a string ('E+1')
%    When predType=1, Blist should be 0 or a big number to use all library
%    When predType=2, Blist is uaually E+1 to from a simplex
%     It also accepts 'E+1', to automatically perform simplex without specifying Blist;
% Theta_epsilon:
%    When predType=1: a column vector indicating nonlinearity parameters. Theta=0 -> linear; Theta>0 ->
nonlinear
%    When predType=2 (epsilon): a column vecto to determine the cut-off values in selecting library data,
%    normally set to a big value in order to use the whole data in the library. If set
%    epsilon=0, the program will use all neighbors
% flag: if flag==0, delete all tempory files used to feed in the block_lnlp C core; otherwise keep the tempory files
%
% output:
% out: a matrix [SampleSize, EmbeddingDimension, Tau, Tp, b, Theta (or Epsilon), MAE, RMSE, rho,
percent_correct, cMAE, cRMSE, crho, cperc]
% c indicating constant predictor
%
% references:
% Sugihara G and RM May (1990) Nature 344: 734-741.
% Sugihara G (1994) Philos. trans. R. Soc. Lond., A. 348: 477-495.
% Hsieh CH, Glaser SM, Lucas AJ, Sugihara G (2005) Nature 435:336-340

% write temporary files
save data.vec data -ascii
fid = fopen('lib.vec','w');fprintf(fid,'%d  %d\n',lib');fclose(fid);
fid = fopen('pred.vec','w');fprintf(fid,'%d  %d\n',pred');fclose(fid);
fid = fopen('Elist.vec','w');fprintf(fid,'%d\n',Elist');fclose(fid);
fid = fopen('Theta_epsilon.vec','w');fprintf(fid,'%f\n',Theta_epsilon);fclose(fid);
fid = fopen('Blist.vec','w');fprintf(fid,'%d\n',Blist');fclose(fid);

% write to lnlp_para
lnlp_para{1}='data.vec';
lnlp_para{2}='lib.vec';
lnlp_para{3}='pred.vec';
lnlp_para{4}='lnlp.out';
lnlp_para{5}=num2str(normType);
lnlp_para{6}=num2str(predType);
lnlp_para{7}=num2str(dumpFlag);
lnlp_para{8}=num2str(1);%%%always allow program to lower # of numbers
lnlp_para{9}='Elist.vec';
lnlp_para{10}=num2str(Tau);
lnlp_para{11}=num2str(TP);
if ischar(Blist)
   lnlp_para{12}=Blist;
else
   lnlp_para{12}='Blist.vec';
end
lnlp_para{13}='Theta_epsilon.vec';

% print out lnlp_para
print_cell(lnlp_para,'lnlp_para','%s\n')

% execute lnlp
!/local_directory % require to have installed the lnlp map- preferably use the rEDM package (https://github.com/
ha0ye/rEDM)

% read results back in
fid = fopen('lnlp.out');
temp = textscan(fid,'%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f','Headerlines',15);
fclose(fid);
for m=1:length(temp)
   out(:,m)=temp{m};
end

% clean up temporary files
if flag1==0
   delete  data.vec lib.vec pred.vec Elist.vec Blist.vec Theta_epsilon.vec lnlp_para lnlp.out* %out.stat
```

```
end

#################################################################

%% Estimating AR1, CV, Nonlinearity summary statistics and plotting

clear all
close all

datadir = ‚~/read_directory/‚;
savdir = ‚~/save_directory/‚;


% datasets:
% 1) changing harvesting rate F (0-3) with r = 0.75,
% 2) changning growth rate r (0.01-3) with F = 0,

% Settings
steps=100;
iter=1000;
T = 100;

% Exploitation
gstart=0;
gend=3;
g_values=linspace(gstart,gend,n_points);
lim_x1 = [-0.1 3.1];
lab1 = {'harvesting rate F'};

% Age truncation
rstart=0.01;
rend=3;
r_values=linspace(rstart,rend,n_points);
lim_x2 = [-0.1 3.05];
lab2 = {'growth rate r'};

% Stochasticity setting
sr = 0.1;
se = 0.25;

namefile=sprintf('Ricker_sr%g_se%g',sr,se)

% Load datasets
filename1 = sprintf([datadir 'Ricker_changeF_sr%g_se%g_c%g'],sr,se,c);
filename2 = sprintf([datadir 'Ricker_changeR_sr%g_se%g_c%g'],sr,se,c);
X1 = load([filename1 '.mat'],'X');
X2 = load([filename2 '.mat'],'X');

% Find limits for transitions
det_shift = 79;
temp = sum(reshape(min(X1.X),iter,n_points)<1);
[aa,bb] = find(temp>0);
g_shift_start = bb(1); % g_value that attractor first shifted
[aa,bb] = find(temp==iter);
g_shift_end = bb(1); % g_value that attractor completely shifted
[aa,bb] = find(temp>=iter/2);
g_shift_50prct = bb(1); % g_value that attractor shifted at least 50% of iterations

% Estimate CV AR1
SD1=std(X1.X);
AVG1=mean(X1.X);
CV1=SD1./AVG1;
AR1_1=[];
AR2_1=[];
```

```
for ii=1:size(X1.X,2)
   cc1=corrcoef(X1.X(2:end,ii),X1.X(1:end-1,ii));
   cc2=corrcoef(X1.X(3:end,ii),X1.X(1:end-2,ii));
   AR1_1=[AR1_1;cc1(1,2)];
   AR2_1=[AR2_1;cc2(1,2)];
end

SD2=std(X2.X);
AVG2=mean(X2.X);
CV2=SD2./AVG2;
AR1_2=[];
AR2_2=[];
for ii=1:size(X2.X,2)
   cc1=corrcoef(X2.X(2:end,ii),X2.X(1:end-1,ii));
   cc2=corrcoef(X2.X(3:end,ii),X2.X(1:end-2,ii));
   AR1_2=[AR1_2;cc1(1,2)];
   AR2_2=[AR2_2;cc2(1,2)];
end

% Get means and std for CV AR1 from all realisations
mean_CV1 = mean(reshape(CV1,iter,n_points));
mean_AR1_1 = mean(reshape(AR1_1,iter,n_points));
std_CV1 = std(reshape(CV1,iter,n_points),0,1);
std_AR1_1 = std(reshape(AR1_1,iter,n_points),0,1);

mean_SD1 = mean(reshape(SD1,iter,n_points));
mean_AVG1 = mean(reshape(AVG1,iter,n_points));
std_SD1 = std(reshape(SD1,iter,n_points),0,1);
std_AVG1 = std(reshape(AVG1,iter,n_points),0,1);

mean_SD2 = mean(reshape(SD2,iter,n_points));
mean_AVG2 = mean(reshape(AVG2,iter,n_points));
std_SD2 = std(reshape(SD2,iter,n_points),0,1);
std_AVG2 = std(reshape(AVG2,iter,n_points),0,1);

mean_CV2 = mean(reshape(CV2,iter,n_points));
mean_AR1_2 = mean(reshape(AR1_2,iter,n_points));
std_CV2 = std(reshape(CV2,iter,n_points),0,1);
std_AR1_2 = std(reshape(AR1_2,iter,n_points),0,1);

% Load Nonlinearity indices
XX1 = load([filename1 '_nonlinearity.mat']);
XX2 = load([filename2 '_nonlinearity.mat']);

% Get means and std for deltaRho
mean_best_delta_rho_1 = mean(XX1.best_delta_rho);
std_best_delta_rho_1 = std(XX1.best_delta_rho,0,1);
std_best_delta_rho_L_1 = std(XX1.best_delta_rho,0,1);
mean_best_delta_rho_2 = mean(XX2.best_delta_rho);
std_best_delta_rho_2 = std(XX2.best_delta_rho,0,1);
std_best_delta_rho_L_2 = std(XX2.best_delta_rho,0,1);


% Estimate standard errors and 95% confidence intervals
nn=1;
ts = tinv([0.025  0.975],iter-1);      % T-Score
SEM_CV1 = std_CV1./sqrt(iter/nn);
CI_CV1 = ts(1)*SEM_CV1;
SEM_CV2 = std_CV2./sqrt(iter/nn);
CI_CV2 = ts(1)*SEM_CV2;
SEM_AR1_1 = std_AR1_1./sqrt(iter/nn);
CI_AR1_1 = ts(1)*SEM_AR1_1;
SEM_AR1_2 = std_AR1_2./sqrt(iter/nn);
CI_AR1_2 = ts(1)*SEM_AR1_2;
```

```
SEM = std_SD1./sqrt(iter/nn);
CI_SD1 = ts(1)*SEM;
SEM = std_SD2./sqrt(iter/nn);
CI_SD2 = ts(1)*SEM;
SEM = std_AVG1./sqrt(iter/nn);
CI_AVG1 = ts(1)*SEM;
SEM = std_AVG2./sqrt(iter/nn);
CI_AVG2 = ts(1)*SEM;
SEM_best_delta_rho_1 = std_best_delta_rho_1./sqrt(iter/nn);
CI_best_delta_rho_1 = ts(1)*SEM_best_delta_rho_1;
SEM_best_delta_rho_2 = std_best_delta_rho_2./sqrt(iter/nn);
CI_best_delta_rho_2 = ts(1)*SEM_best_delta_rho_2;


##################################################################

% Plotting
msize = 3; % resize marker
mark = 'o';
ll = 200; % fix size errorbars


% Plot summary statistics for CV AR1 deltaRho as function of control parameter
figure
set(gcf,'position',[ 222   -17   604   800])

subplot(3,2,1)
hh = errorbar(r_values,mean_CV2,CI_CV2,'kx');
set(hh,'LineStyle','None','Marker',mark,'MarkerEdgeColor','none','MarkerFaceColor','black','MarkerSize',msize)
ylabel('CV')
xlabel(lab2)
axis tight
xlim(lim_x2)
y_ax = get(gca,'ylim');
set(gca,'ylim',[0.95*y_ax(1) 1.05*y_ax(2)]);
hold
plot([r_values(69) r_values(35) r_values(91); r_values(69) r_values(35) r_values(91)],[0.95*y_ax(1) 0.95*y_ax(1)
0.95*y_ax(1); 1.05*y_ax(2) 1.05*y_ax(2) 1.05*y_ax(2)],'r--','LineWidth',1)
title('age-truncation scenario')

subplot(3,2,2)
hh = errorbar(g_values,mean_CV1,CI_CV1,'kx');
set(hh,'LineStyle','None','Marker',mark,'MarkerEdgeColor','none','MarkerFaceColor','black','MarkerSize',msize)
ylabel('CV')
xlabel(lab1)
axis tight
title('exploitation scenario')
xlim(lim_x1)
y_ax = get(gca,'ylim');
set(gca,'ylim',[0.95*y_ax(1) 1.05*y_ax(2)]);
hold
plot([g_values(g_shift_50prct); g_values(g_shift_50prct)],[0.95*y_ax(1); 1.05*y_ax(2)],'r--', 'LineWidth',1)

subplot(3,2,3)
hh=errorbar(r_values,mean_AR1_2,CI_AR1_2,'kx');
set(hh,'LineStyle','None','Marker',mark,'MarkerEdgeColor','none','MarkerFaceColor','black','MarkerSize',msize)
ylabel('AR1')
xlabel(lab2)
axis tight
xlim(lim_x2)
y_ax = get(gca,'ylim');
set(gca,'ylim',[1.05*y_ax(1) 1.05*y_ax(2)]);
hold
plot([r_values(69) r_values(35) r_values(91); r_values(69) r_values(35) r_values(91)],[1.05*y_ax(1) 1.05*y_ax(1)
1.05*y_ax(1); 1.05*y_ax(2) 1.05*y_ax(2) 1.05*y_ax(2)],'r--','LineWidth',1)

subplot(3,2,4)
hh = errorbar(g_values,mean_AR1_1,CI_AR1_1,'kx');
set(hh,'LineStyle','None','Marker',mark,'MarkerEdgeColor','none','MarkerFaceColor','black','MarkerSize',msize)
ylabel('AR1')
xlabel(lab1)
axis tight
xlim(lim_x1)
y_ax = get(gca,'ylim');
set(gca,'ylim',[0.95*y_ax(1) 1.05*y_ax(2)]);
hold
plot([g_values(g_shift_50prct); g_values(g_shift_50prct)],[0.95*y_ax(1); 1.05*y_ax(2)],'r--','LineWidth',1)%

subplot(3,2,5)
hh = errorbar(r_values,mean_best_delta_rho_2,CI_best_delta_rho_2,'kx')
set(hh,'LineStyle','None','Marker',mark,'MarkerEdgeColor','none','MarkerFaceColor','black','MarkerSize',msize)
ylabel('\delta\rho')
xlabel(lab2)
axis tight
xlim(lim_x2)
y_ax = get(gca,'ylim');
set(gca,'ylim',[0.95*y_ax(1) 1.05*y_ax(2)],'xscale','linear');
hold
plot([r_values(69) r_values(35) r_values(91); r_values(69) r_values(35) r_values(91)],[0.95*y_ax(1) 0.95*y_ax(1)
0.95*y_ax(1); 1.05*y_ax(2) 1.05*y_ax(2) 1.05*y_ax(2)],'r--','LineWidth',1)

subplot(3,2,6)
hh = errorbar(g_values,mean_best_delta_rho_1,CI_best_delta_rho_1,'kx')
set(hh,'LineStyle','None','Marker',mark,'MarkerEdgeColor','none','MarkerFaceColor','black','MarkerSize',msize)
ylabel('\delta\rho')
xlabel(lab1)
axis tight
xlim(lim_x1)
y_ax = get(gca,'ylim');
set(gca,'ylim',[0.95*y_ax(1) 1.05*y_ax(2)],'xscale','linear');
hold
plot([g_values(g_shift_50prct); g_values(g_shift_50prct)],[0.95*y_ax(1); 1.05*y_ax(2)],'r--', 'LineWidth',1)

####################################################################

% Plot rescaled AR1, CV, deltaRho
figure
set(gcf,'position',[222     2   929   781])
n_points=100;
sc_mean_AVG1 = rescale(mean_AVG1);
sc_mean_AVG2 = rescale(mean_AVG2);
sc_mean_CV1 = rescale(mean_CV1);
sc_mean_CV2 = rescale(mean_CV2);
sc_mean_AR1_1 = rescale(mean_AR1_1);
sc_mean_AR1_2 = rescale(mean_AR1_2);
sc_mean_SD1 = rescale(mean_SD1);
sc_mean_SD2 = rescale(mean_SD2);
sc_mean_best_delta_rho_1 = rescale(mean_best_delta_rho_1);
sc_mean_best_delta_rho_2 = rescale(mean_best_delta_rho_2);

ptileX1=prctile(reshape(mean(X1.X),iter,n_points),[5 50 95]);
ptileX2=prctile(reshape((X2.X),iter*100,n_points),[5 50 95]);

subplot(2,2,1)
plot(r_values,ptileX2(1,:),'k--',r_values,ptileX2(2,:),'k-',r_values,ptileX2(3,:),'k--')
ylabel('average fish biomass (a.u.)')
xlabel(lab2)
axis tight
```

```
xlim(lim_x2)
y_ax = get(gca,'ylim');
set(gca,'ylim',[0.95*y_ax(1) 1.05*y_ax(2)]);
hold
plot([r_values(69) r_values(35) r_values(91); r_values(69) r_values(35) r_values(91)],[0.95*y_ax(1) 0.95*y_ax(1)
0.95*y_ax(1); 1.05*y_ax(2) 1.05*y_ax(2) 1.05*y_ax(2)],'r--','LineWidth',0.5)
title('age-truncation scenario')

subplot(2,2,2)
plot(g_values,ptileX1(1,:),'k--',g_values,ptileX1(2,:),'k-',g_values,ptileX1(3,:),'k--')
ylabel('fish biomass (a.u.)')
xlabel(lab1)
axis tight
title('exploitation scenario')
xlim(lim_x1)
y_ax = get(gca,'ylim');
set(gca,'ylim',[0.95*y_ax(1) 1.05*y_ax(2)]);
hold
plot([g_values(g_shift_50prct); g_values(g_shift_50prct)],[0.95*y_ax(1); 1.05*y_ax(2)],'r--')

subplot(2,2,4)
plot(g_values,sc_mean_CV1,'k-',g_values,sc_mean_AR1_1,'k:',g_values,sc_mean_best_delta_rho_1,'k-.','Linewi
dth',1)
ylabel('rescaled indicators')
xlabel(lab1)
legend('CV','AR1','\delta\rho','location','best')
legend boxoff
axis tight
xlim(lim_x1)
y_ax = get(gca,'ylim');
set(gca,'ylim',[0.95*y_ax(1) 1.05*y_ax(2)]);
hold
plot([g_values(g_shift_50prct); g_values(g_shift_50prct)],[0.95*y_ax(1); 1.05*y_ax(2)],'r--')%, ...

subplot(2,2,3)
plot(r_values,sc_mean_CV2,'k-',r_values,sc_mean_AR1_2,'k:', ...
   r_values,sc_mean_best_delta_rho_2,'k-.','Linewidth',1)
ylabel('rescaled indicators')
xlabel(lab2)
axis tight
xlim(lim_x1)
y_ax = get(gca,'ylim');
set(gca,'ylim',[0.95*y_ax(1) 1.05*y_ax(2)]);
hold
plot([r_values(69) r_values(35) r_values(91); r_values(69) r_values(35) r_values(91)],[0.95*y_ax(1) 0.95*y_ax(1)
0.95*y_ax(1); 1.05*y_ax(2) 1.05*y_ax(2) 1.05*y_ax(2)],'r--','LineWidth',0.5)

##################################################################

% Plot distributions of AR1, SD, DeltaRho for exploited vs unexploited simulated time series
cv_all_1 = reshape(CV1,iter,n_points);
cv_all_2 = reshape(CV2,iter,n_points);
AR1_all_1 = reshape(AR1_1,iter,n_points);
AR1_all_2 = reshape(AR1_2,iter,n_points);

figure
set(gcf,’position',[  479    63   685   720])
far = 1:33; sfar=length(far);
close_to = 34:(g_shift_50prct); sclose=length(close_to);
sample=1:1000;
ss=1000;

% Rescale indicators
sc_cv_all_1 = rescale(cv_all_1(:,1:g_shift_50prct));
```

```
sc_AR1_all_1 = rescale(AR1_all_1(:,1:g_shift_50prct));
sc_XX1.best_delta_rho = rescale(XX1.best_delta_rho(:,1:g_shift_50prct));

data0 = NaN.*ones(max(ss*sfar,ss*sclose),6);
data0(1:ss*sfar,4)=reshape(sc_AR1_all_1(sample,far),ss*sfar,1);
data0(1:ss*sclose,3)=reshape(sc_AR1_all_1(sample,close_to),ss*sclose,1);
data0(1:ss*sfar,2)= reshape(sc_cv_all_1(sample,far),ss*sfar,1);
data0(1:ss*sclose,1) = reshape(sc_cv_all_1(sample,close_to),ss*sclose,1);
data0(1:ss*sfar,6)=reshape(sc_XX1.best_delta_rho(sample,far),ss*sfar,1);
data0(1:ss*sclose,5)=reshape(sc_XX1.best_delta_rho(sample,close_to),ss*sclose,1);

far = 1:69; sfar=length(far);
close_to = 70:100; sclose=length(close_to);
sc_cv_all_2 = rescale(cv_all_2(:,[far close_to]));
sc_AR1_all_2 = rescale(AR1_all_2(:,[far close_to]));
sc_XX2.best_delta_rho = rescale(XX2.best_delta_rho(:,[far close_to]));

data1 = NaN.*ones(max(ss*sfar,ss*sclose),6);
data1(1:ss*sfar,4)=reshape(sc_AR1_all_2(sample,far),ss*sfar,1);
data1(1:ss*sclose,3)=reshape(sc_AR1_all_2(sample,close_to),ss*sclose,1);
data1(1:ss*sfar,2)= reshape(sc_cv_all_2(sample,far),ss*sfar,1);
data1(1:ss*sclose,1) = reshape(sc_cv_all_2(sample,close_to),ss*sclose,1);
data1(1:ss*sfar,6)=reshape(sc_XX2.best_delta_rho(sample,far),ss*sfar,1);
data1(1:ss*sclose,5)=reshape(sc_XX2.best_delta_rho(sample,close_to),ss*sclose,1);

% t-test to compare distributions
alpha=0.05;
vartype='unequal';
tail='both';
[h,p_expl_CV] = ttest2(data0(:,1),data0(:,2),alpha,tail,vartype);
[h,p_expl_AR1] = ttest2(data0(:,3),data0(:,4),alpha,tail,vartype);
[h,p_expl_deltarho] = ttest2(data0(:,5),data0(:,6),alpha,tail,vartype);
[h,p_trunc_CV] = ttest2(data1(:,1),data1(:,2),alpha,tail,vartype);
[h,p_trunc_AR1] = ttest2(data1(:,3),data1(:,4),alpha,tail,vartype);
[h,p_trunc_deltarho] = ttest2(data1(:,5),data1(:,6),alpha,tail,vartype);

sp_id = [{'CV expl'},{'CV nexpl'},{'AR1 expl'},{'AR1 nexpl'},{[char(948) char(961) ' expl']},{[char(948) char(961) '
nexpl']}];

subplot(2,1,1)
hh=boxplot(data1,{sp_id},'colors',repmat('br',1,3),'factorgap',[1], ...
    'labelverbosity','minor','symbol','k*','outliersize',0.5,'positions',[0.5, 0.7, 1, 1.2, 1.5,1.7]);
p = prctile(data1,[5 25 75 95]);
modify_boxplot(hh,p,0,0)
set(gca,'ylim',[-0.05 1.05])
ylabel('indicators (rescaled)')
xlabel('')
title('age-truncation scenario')
legend(['pCV' num2str(p_trunc_CV,'%.2')],['pAR1' num2str(p_trunc_AR1,'%.2')],['pdR'
num2str(p_trunc_deltarho,'%.2')])

subplot(2,1,2)
hh=boxplot(data0,{sp_id},'colors',repmat('rb',1,3),'factorgap',[1], ...
    'labelverbosity','minor','symbol','','outliersize',0.5,'positions',[0.5, 0.7, 1, 1.2, 1.5,1.7]);
p = prctile(data0,[5 25 75 95]);
modify_boxplot(hh,p,0,0)
set(gca,'ylim',[-0.05 1.05])
ylabel('indicators (rescaled)')
xlabel('')
title('exploitation scenario')
legend(['pCV' num2str(p_expl_CV,'%.2')],['pAR1' num2str(p_expl_AR1,'%.2')],['pdR' num2str(p_expl_deltarho,’%.
2’)])\
##################################################################

function y = rescale(x,a,b)

% RESCALE function for data in [a,b]

if nargin<2
   a=0;
end

if nargin<3
   b=1;
end

m=min(x(:));
M=max(x(:));

if M-m<eps
   y=x;
else
   y = (b-a)*(x-m)/(M-m) + a;
end

##################################################################

function modify_boxplot(h,p,outliers,horizontal)

% MODIFY_BOXPLOT function for modifying the percentiles plotted by boxplot

s = p;

% Modify upper whisker's length

if horizontal
set(h(1,:),{'Xdata'},num2cell(s(end-1:end,:),1)')

% Modify lower whisker's length
set(h(2,:),{'Xdata'},num2cell(s(2:-1:1,:),1)')

% Modify upper whisker's endbar
set(h(3,:),{'Xdata'},num2cell(s([end end],:),1)')

% Modify lower whisker's endbar
set(h(4,:),{'Xdata'},num2cell(s([1 1],:),1)')

% Modify body
set(h(5,:),{'Xdata'},num2cell(s([2 3 3 2 2],:),1)')

else
  % Modify upper whisker's length
set(h(1,:),{'Ydata'},num2cell(s(end-1:end,:),1)')

% Modify lower whisker's length
set(h(2,:),{'Ydata'},num2cell(s(2:-1:1,:),1)')

% Modify upper whisker's endbar
set(h(3,:),{'Ydata'},num2cell(s([end end],:),1)')

% Modify lower whisker's endbar
set(h(4,:),{'Ydata'},num2cell(s([1 1],:),1)')

% Modify body
set(h(5,:),{'Ydata'},num2cell(s([2 3 3 2 2],:),1)')
```

```
end


set(h(7,:),{'Visible'},{'off'})

if outliers
hold on
plot(1,.95,'+r',2,.1,'+r')
end
```