The following is the source code for the simulations reported in Clinchy, Haydon and Smith (Pattern ≠ process: what does patch occupancy really tell us about metapopulation dynamics).

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, StdCtrls, ExtCtrls, Buttons, Clipbrd, OleCtrls,
  Grids, ShellApi, ComCtrls, TeEngine, Series, TeeProcs, Chart;

const
    max_lines = 5000;          {# points in single process + origin}
    y_height=501;
    x_width=501;
    extinction_prob=1;
    density1 = 0.00005;
    NN=74;

type
    EOutput=class(Exception);
    TMainForm = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Exit1: TMenuItem;
    Run1: TMenuItem;
    Go1: TMenuItem;
    Panel1: TPanel;
    Label12: TLabel;
    Screen1: TPaintBox;
    Capture: TButton;
    Memo1: TMemo;
    Start: TButton;
    PatchConfig: TRadioGroup;
    ProgressBar1: TProgressBar;
    samplebox: TEdit;
    Label1: TLabel;
    process: TRadioGroup;
    Prob_occ_box: TEdit;
    Label2: TLabel;
    Chart1: TChart;
    Series1: TBarSeries;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    TestType: TRadioGroup;

    procedure assign_positions_double;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Screen1Paint(Sender: TObject);
    procedure CaptureClick(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure extinguish;
    procedure RefreshBitmap;
    procedure Button1Click(Sender: TObject);
    procedure ClearBitmap;
    procedure update;
    procedure Initialize;
    procedure Button2Click(Sender: TObject);
    procedure extinguish_randomly;
    procedure MW_test_within;
    procedure MW_test_between;
    procedure MW_test(between:boolean);
```

```
    procedure StartClick(Sender: TObject);
    procedure read_data;
  private
    { Private declarations }

  public
    { Public declarations }

  end;

type

 structure = record
             x:double;
             y:double;
             parent:boolean;
             occupied:boolean;
             was_occupied:boolean;
             id:longint;
             end;

 listpoints=array[0..100000] of structure;

var
  MainForm: TMainForm;
  list :listpoints;
  iid,initialized:Boolean;
  line_no,i,j,k,x_offset,y_offset,individual,site,noof_occupied,noof_extinctions,max_points,pound,samples:longint;
  OffScreen,TestBit:TBitmap;
  outfile1,infile,outfile2,out1,out2,Ufile:text;
  was_occupied,occupied,extinction:array[0..max_lines] of boolean;
  undecided,pos,neg,t_value,sample:array[0..10] of real;
  radius,scale,max,z,n,sum1,sum0,mean1,mean0,sum_u1,sum_u0:real;
  extinction_occured:boolean;
  distances,r_distances:array[1..4000] of real;
  whose_r_distances,whose_distances:array[1..4000] of integer;
  n0,n1, max_points1, av_max_points2:integer;
  cluster_radius,prop_affected,prob_occupied :real;
  Ucrit :array[1..20,1..20] of integer;

implementation

{$R *.DFM}

function factorial(n:extended):extended;
var
   sub : extended; i:longint;
begin
if n = 0 then sub := 1
   else
begin
   if n >= 50 then
      sub := 2.506628275 * exp(-n) * exp(ln(n)*(n+0.5))
   else
       begin
           sub := n;
           for i:= trunc(n) downto 2 do
             sub := sub * (i-1);
       end;
   end;
factorial := sub;
end;

procedure prepare_infile;
begin
    assign(infile,'smith.txt');
    reset(infile);
end;

procedure TMainForm.read_data;
```

```
var i,j:integer;
dummy,meanx,meany:real;
dummy_char:char;
begin
  prepare_infile;

  meanx:=0;
  meany:=0;

  for i:=0 to NN-1 do
       begin
          read(infile,dummy);
          read(infile,list[i].x);
          readln(infile,list[i].y);
          meanx:=meanx+list[i].x/NN;
          meany:=meany+list[i].y/NN;
       end;
 for i:=0 to NN-1 do
       begin
         list[i].x:=list[i].x-meanx;
         list[i].y:=list[i].y-meany;
       end;

max_points:=NN-1;

max:=-1;
for i:=0 to max_points do
   if sqrt(sqr(list[i].x)+sqr(list[i].y))>max then
     max:=sqrt(sqr(list[i].x)+sqr(list[i].y));
radius:=max*sqrt(prop_affected);
closefile(infile);
end;

procedure TMainForm.assign_positions_double;
var i,j:longint;count_id:longint;
   u,ww,cumulative,x4,y,max_points2,cum_x_sq1,cum_x_sq2,mother_x,mother_y,new_mother_x,new_mother_y,try_x,try_y:double;
begin
    cum_x_sq1:=0;
    count_id:=0;
    list[0].x:=0;
    list[0].y:=0;
    list[0].parent:=true;
    list[0].id:=0;
    cum_x_sq2:=0;
if av_max_points2>0 then
  begin
    ww:=random; max_points2 :=0;
    cumulative:=( exp( ln(av_max_points2)*max_points2)*exp(-av_max_points2))/factorial(max_points2);
       while cumulative<ww do
        begin
          cumulative:=cumulative+( exp(ln(av_max_points2)*max_points2)* exp(-av_max_points2))/factorial(max_points2);
          if cumulative<ww then
             max_points2:=max_points2+1;
       end;
  end
else
  max_points2:=0;
for i:=1 to trunc(max_points2) do
    begin
      new_mother_x:=0-cluster_radius;
      new_mother_y:=0-cluster_radius;
      repeat
        try_x:=new_mother_x+random*2*cluster_radius;
        try_y:=new_mother_y+random*2*cluster_radius;
      until sqrt(sqr(try_x-0)+sqr(try_y-0))<= cluster_radius;
      count_id:=count_id+1;
      list[count_id].x:=try_x;
      list[count_id].y:=try_y;
      list[count_id].parent:=false;
      list[count_id].id:=count_id;
```

```
    end;
for i:=1 to max_points1-1 do
    begin
      repeat u:=random;until u>0;
      x4:=sqrt(-ln(u)/(density1*3.1415926)+cum_x_sq1);
      cum_x_sq1:=sqr(x4);
      repeat u:=random;until u>0;
      y:=u*2*3.1415926;
      count_id:=count_id+1;
      list[count_id].x:=x4*cos(y);
      list[count_id].y:=x4*sin(y);
      list[count_id].parent:=true;
      list[count_id].id:=i;
      cum_x_sq2:=0;
      mother_x:=x4*cos(y);
      mother_y:=x4*sin(y);
      if av_max_points2>0 then
        begin
          ww:=random; max_points2 :=0;
          cumulative:=(exp(ln(av_max_points2)*max_points2)*exp(-av_max_points2))/factorial(max_points2);
          while cumulative<ww do
            begin
              cumulative:=cumulative+( exp(ln(av_max_points2)*max_points2)*exp(-av_max_points2))/factorial(max_points2);
              if cumulative<ww then
                max_points2:=max_points2+1;
            end;
        end
      else
        max_points2:=0;

for j:=1 to trunc(max_points2) do
  begin
    new_mother_x:=mother_x-cluster_radius;
    new_mother_y:=mother_y-cluster_radius;
    repeat
      try_x:=new_mother_x+random*2*cluster_radius;
      try_y:=new_mother_y+random*2*cluster_radius;
    until sqrt(sqr(try_x-mother_x)+sqr(try_y-mother_y))<= cluster_radius;
    count_id:=count_id+1;
    list[count_id].x:=try_x;
    list[count_id].y:=try_y;
    list[count_id].parent:=false;
    list[count_id].id:=count_id;
  end;
end;
max_points:=count_id;
max:=-1;
for i:=0 to max_points do
    if sqrt(sqr(list[i].x)+sqr(list[i].y))>max then
      max:=sqrt(sqr(list[i].x)+sqr(list[i].y));
radius:=max*sqrt(prop_affected);
end;

procedure TMainForm.Initialize;
var i,j:integer;
begin
noof_extinctions:=0;
noof_occupied:=0;
   for i:=0 to max_points do
        begin
          if random<prob_occupied then
            begin
              list[i].occupied:=true;
              list[i].was_occupied:=true;
              noof_occupied:=noof_occupied+1;
            end
          else
            begin
              list[i].occupied:=false;
              list[i].was_occupied:=false;
```

```
            end;
           extinction[i]:=false;
         end;
end;

procedure TMainForm.RefreshBitmap;
var i, j ,k, dot_size,s_dot_size: longint;
Brush:hBrush; sq : Trect;
begin
x_offset:=250;
y_offset:=250;
if max<>0 then
  scale:=(x_width div 2)/max;
dot_size:=2;
s_dot_size:=2;
with OffScreen.Canvas do
    begin
    Font.Size:=10;Font.Name:='Times';

   for i:=0 to max_points do
       begin
         if list[i].occupied then
          begin
            Pen.Color:=RGB((0),(0),(0));
            Brush.Color:=RGB((0),(0),(0));
          end
         else
          begin
            Pen.Color:=RGB((255),(0),(0));
            Brush.Color:=RGB((255),(0),(0));
          end;
             sq:=rect(round(scale*list[i].x+x_offset-s_dot_size),
                 round(scale*list[i].y+y_offset-s_dot_size),
                 round(scale*list[i].x+x_offset+s_dot_size),
                 round(scale*list[i].y+y_offset+s_dot_size));
             ellipse(round(scale*list[i].x+x_offset-s_dot_size),
                 round(scale*list[i].y+y_offset-s_dot_size),
                 round(scale*list[i].x+x_offset+s_dot_size),
                 round(scale*list[i].y+y_offset+s_dot_size));

       end;
   end;
  Screen1.Canvas.StretchDraw(Rect(0, 0, Screen1.Width, Screen1.Height), OffScreen);
 end;

function get_nearest_occupied(to_this_one:integer;var max:real):integer;
var j,it:integer;
begin
max:=1000000000;
for j:=0 to max_points do
  if to_this_one<>j then
    begin
      if (list[j].was_occupied) and (sqrt(sqr(list[to_this_one].x-list[j].x)+sqr(list[to_this_one].y-list[j].y))<max) then
       begin
         max:=sqrt(sqr(list[to_this_one].x-list[j].x)+sqr(list[to_this_one].y-list[j].y));
         it:=j;
       end;
    end;
get_nearest_occupied:=it;
end;

procedure TMainForm.extinguish;
var i,j:integer;x4,u,y:real;
begin

  repeat u:=random;until u>0;
    x4:=u*max;
    repeat u:=random;until u>0;
    y:=u*2*3.1415926;
     list[max_points+1].x:=x4*cos(y);
```

```
    list[max_points+1].y:=x4*sin(y);

    site:=(max_points+1);
    extinction_occured:=false;
    for i:=0 to max_points do
       if sqrt(sqr(list[i].x-list[site].x)+sqr(list[i].y-list[site].y))<=radius then
          if random<extinction_prob then
           begin
             if list[i].occupied=true then
               begin
                 extinction[i]:=true;
                 noof_occupied:=noof_occupied-1;
                 noof_extinctions:=noof_extinctions+1;
                 extinction_occured:=true;
               end;
             list[i].occupied:=false;

           end;
end;

procedure TMainForm.extinguish_randomly;
var i,j,k:integer;
begin
  k:=round(prop_affected*max_points);
  if k<1 then k:=1;
  for i:=1 to k do
    begin
      repeat j:=random(max_points+1) until list[j].occupied=true;
      extinction[j]:=true;
      noof_occupied:=noof_occupied-1;
      noof_extinctions:=noof_extinctions+1;
      extinction_occured:=true;
      list[j].occupied:=false;
    end;
end;

procedure TMainForm.ClearBitmap;
var i, j ,k, dot_size: longint;
Brush:hBrush; sq : TRect;
max,scale:real;
begin
  with OffScreen.Canvas do
    begin
     Brush.Color:=RGB(255,255,255);
     sq:=rect(0,0,x_width,y_height);
     Fillrect (sq);
    end;
  Screen1.Canvas.StretchDraw(Rect(0, 0, Screen1.Width, Screen1.Height), OffScreen);
 end;

procedure read_U_values;
var i,j:integer;
begin
assign(Ufile,'U.txt');
reset(Ufile);
for i:=1 to 20 do
  for j:=1 to 20 do
    if j<20 then
      read(Ufile,Ucrit[i,j])
    else
      readln(Ufile,Ucrit[i,j]);
end;

procedure TMainForm.FormCreate(Sender: TObject);
begin
randomize;
TestType.ItemIndex:=1;
read_U_values;
end;
```

```
procedure TMainForm.FormDestroy(Sender: TObject);
var
  i ,j : longint;
begin
  try
     OffScreen.Free;
  except
  end;
end;

procedure rank;
var i,j,it,whose:integer;
    min :real;
begin
   for i:=1 to trunc(n) do
     begin
       min:=100000000;
       for j:=1 to trunc(n) do
          if distances[j]<min then
            begin
              it :=j;
              min :=distances[j];
              whose:=whose_distances[j];
            end;
       r_distances[ i]:=min;
       whose_r_distances[ i]:=whose;
       distances[it]:=1000000001;
     end;
end;

procedure TMainForm.MW_test(between:boolean);
var i,j:integer;
    u0,u1:real;
begin
    u0:=0;u1:=0;
    for i:=1 to trunc(n) do
      if whose_r_distances[ i]=0 then
        begin
          sum_u0:=sum_u0+r_distances[ i];
          u0:=u0+i;
        end
      else
        begin
          sum_u1:=sum_u1+r_distances[ i];
          u1:=u1+i;
        end;
    u0:=u0-n0*(n0+1)/2;
    u1:=u1-n1*(n1+1)/2;
    if (n1>20) then
      begin
        z:=(u0-n1*n0/2)/sqrt(n1*n0*(n0+n1+1)/12);
        sample[  round((noof_extinctions/max_points)*10)  ]:=sample[  round((noof_extinctions/max_points)*10)  ]+1;
        if z>1.96 then
           pos[ round((noof_extinctions/max_points)*10)  ]:=pos[  round((noof_extinctions/max_points)*10)  ]+1;
        if z<-1.96 then
           neg[ round((noof_extinctions/max_points)*10)  ]:=neg[  round((noof_extinctions/max_points)*10)  ]+1;
        if (z>-1.96) and (z<1.96) then
           undecided[ round((noof_extinctions/max_points)*10)  ]:=undecided[  round((noof_extinctions/max_points)*10)  ]+1;
      end
    else
     if (between=true) and (n1>10) and (n1<=20) and (n0>1) and (n0<=20)then
      begin
        if u0<=Ucrit[n1,n0] then
          pos[ round((noof_extinctions/max_points)*10)  ]:=pos[  round((noof_extinctions/max_points)*10)  ]+1
        else
          undecided[ round((noof_extinctions/max_points)*10)  ]:=undecided[  round((noof_extinctions/max_points)*10)  ]+1;
        sample[  round((noof_extinctions/max_points)*10)  ]:=sample[  round((noof_extinctions/max_points)*10)  ]+1;
      end;
end;
```

```
procedure TMainForm.MW_test_within;
var i,j:integer;
max,x1,x1_sq,x2,x2_sq,y2_sq,t,se1,se2,pooled_se:real;
begin
   x1:=0;x2:=0;x2_sq:=0;x1_sq:=0;y2_sq:=0;n0:=0;n1:=0;n:=0;sum1:=0;sum0:=0;
   for i:=0 to max_points do
     begin
        if not list[i].occupied then
           begin
              max:=1000000000;
              for j:=0 to max_points do
                if i<>j then
                  begin
                     if (list[j].occupied) and (sqrt(sqr(list[i].x-list[j].x)+sqr(list[i].y-list[j].y))<max) then
                          max:=sqrt(sqr(list[i].x-list[j].x)+sqr(list[i].y-list[j].y));

                  end;
              n0:=n0+1; n:=n+1;
              sum0:=sum0+max;
              distances[trunc(n)]:=max;
              whose_distances[trunc(n)]:=0;
           end;
        if list[i].occupied then
           begin
              max:=1000000000;
              for j:=1 to max_points do
                if i<>j then
                  begin
                     if (list[j].occupied) and (sqrt(sqr(list[i].x-list[j].x)+sqr(list[i].y-list[j].y))<max) then
                          max:=sqrt(sqr(list[i].x-list[j].x)+sqr(list[i].y-list[j].y));
                  end;
              n1:=n1+1;n:=n+1;
              sum1:=sum1+max;
              distances[trunc(n)]:=max;
              whose_distances[trunc(n)]:=1;
           end;

     end;
   rank;
   MW_test(false);
   if (n0>10) and (n1>10) then
    Memo1.Lines.Add(IntToStr(pound)+'   '+FloatToStrF(z,ffFixed,8, 4)+'   '+FloatToStrF(sum0/n0,ffFixed,8, 4)+'   ('+IntToStr(n0)+
')   '+FloatToStrF(sum1/n1,ffFixed,8, 4)+'   ('+IntToStr(n1)+')');
end;

procedure TMainForm.MW_test_between;
var i,j:integer;
max,x1,x1_sq,x2,x2_sq,y2_sq,t,se1,se2,pooled_se:real;
begin
   x1:=0;x2:=0;x2_sq:=0;x1_sq:=0;y2_sq:=0;n0:=0;n1:=0;n:=0;sum1:=0;sum0:=0;
   for i:=0 to max_points do
     begin
        if (not list[i].occupied) and (list[i].was_occupied) then
           begin
              max:=1000000000;
              for j:=0 to max_points do
                if i<>j then
                  begin
                     if (list[j].occupied) and (sqrt(sqr(list[i].x-list[j].x)+sqr(list[i].y-list[j].y))<max) then
                          max:=sqrt(sqr(list[i].x-list[j].x)+sqr(list[i].y-list[j].y));

                  end;
              n0:=n0+1; n:=n+1;
              sum0:=sum0+max;
              distances[trunc(n)]:=max;
              whose_distances[trunc(n)]:=0;
           end;
        if list[i].occupied then
           begin
              max:=1000000000;
```

```
                for j:=1 to max_points do
                  if i<>j then
                    begin
                      if (list[j].occupied) and (sqrt(sqr(list[i].x-list[j].x)+sqr(list[i].y-list[j].y))<max) then
                          max:=sqrt(sqr(list[i].x-list[j].x)+sqr(list[i].y-list[j].y));
                    end;
                n1:=n1+1;n:=n+1;
                sum1:=sum1+max;
                distances[trunc(n)]:=max;
                whose_distances[trunc(n)]:=1;
              end;

      end;
    rank;
    MW_test(true);
    if (n0+n1>20) then
      Memo1.Lines.Add(IntToStr(pound)+'   '+FloatToStrF(z,ffFixed,8, 4)+'   '+FloatToStrF(sum0/n0,ffFixed,8, 4)+'   ('+IntToStr(n0)+
')   '+FloatToStrF(sum1/n1,ffFixed,8, 4)+'   ('+IntToStr(n1)+')');
end;

procedure TMainForm.Screen1Paint(Sender: TObject);
begin
    Screen1.Canvas.StretchDraw(Rect(0, 0, Screen1.Width, Screen1.Height), OffScreen);
end;

procedure TMainForm.CaptureClick(Sender: TObject);
begin
  Clipboard.Assign(OffScreen);
end;

procedure TMainForm.Exit1Click(Sender: TObject);
begin
   close;
end;

procedure TMainForm.update;
var i:integer;
begin
   for i:=0 to max_points do
      if list[i].occupied=false then
        list[i].was_occupied:=false;
end;

procedure TMainForm.Button1Click(Sender: TObject);
begin
   pound:=0;
   repeat
     if iid then
       extinguish_randomly
     else
       extinguish;
     inc(pound);
     if extinction_occured then
      if TestType.ItemIndex=0 then
        MW_test_within
      else
        MW_test_between;
     update;
   until noof_occupied<trunc(max_points*0.15) ;
   ClearBitmap;
   RefreshBitmap;
end;

procedure TMainForm.Button2Click(Sender: TObject);
var i:integer;
    sum_neg,sum_pos,sum_all:real;
begin
rewrite(outfile2,'Mout0x.txt');
sum_neg:=0;
sum_pos:=0;
```

```
sum_all:=0;
for i:=0 to 10 do
 begin
   sum_neg:=sum_neg+neg[i];
   sum_pos:=sum_pos+pos[i];
   sum_all:=sum_all+sample[i];
    if sample[i]>0 then
      begin
        t_value[i]:=t_value[i]/sample[i];
        neg[i]:=neg[i]/sample[i];
        pos[i]:=pos[i]/sample[i];
        undecided[i]:=undecided[i]/sample[i];
      end
 end;
  Memo1.Lines.Add('Overall Prop. of Positive Results '+FloatToStrF(sum_pos/sum_all,ffFixed,5, 3));
  Memo1.Lines.Add('Overall Prop. of Negative Results '+FloatToStrF(sum_neg/sum_all,ffFixed,5, 3));
  Memo1.Lines.Add('Total number of extinction events '+FloatToStrF(sum_all,ffFixed,5, 3));

  with Series1 do
    begin
      AddXY(20,pos[2],'15-25',clRed);
      AddXY(30,pos[3],'25-35',clRed);
      AddXY(40,pos[4],'35-45',clRed);
      AddXY(50,pos[5],'45-55',clRed);
      AddXY(60,pos[6],'55-65',clRed);
      AddXY(70,pos[7],'65-75',clRed);
      AddXY(80,pos[8],'75-85',clRed);
   end;

  writeln(outfile2,t_value[1],chr(9),pos[1]:8:5,chr(9),neg[1]:8:5,chr(9),
  undecided[1]:8:5,chr(9),sample[1]:2:1,chr(9),'0.05-0.15');
  writeln(outfile2,t_value[2],chr(9),pos[2]:8:5,chr(9),neg[2]:8:5,chr(9),
  undecided[2]:8:5,chr(9),sample[2]:2:1,chr(9),'0.15-0.25');
  writeln(outfile2,t_value[3],chr(9),pos[3]:8:5,chr(9),neg[3]:8:5,chr(9),
  undecided[3]:8:5,chr(9),sample[3]:2:1,chr(9),'0.25-0.35');
  writeln(outfile2,t_value[4],chr(9),pos[4]:8:5,chr(9),neg[4]:8:5,chr(9),
  undecided[4]:8:5,chr(9),sample[4]:2:1,chr(9),'0.35-0.45');
  writeln(outfile2,t_value[5],chr(9),pos[5]:8:5,chr(9),neg[5]:8:5,chr(9),
  undecided[5]:8:5,chr(9),sample[5]:2:1,chr(9),'0.45-0.55');
  writeln(outfile2,t_value[6],chr(9),pos[6]:8:5,chr(9),neg[6]:8:5,chr(9),
  undecided[6]:8:5,chr(9),sample[6]:2:1,chr(9),'0.55-0.65');
  writeln(outfile2,t_value[7],chr(9),pos[7]:8:5,chr(9),neg[7]:8:5,chr(9),
  undecided[7]:8:5,chr(9),sample[7]:2:1,chr(9),'0.65-0.75');
  writeln(outfile2,t_value[8],chr(9),pos[8]:8:5,chr(9),neg[8]:8:5,chr(9),
  undecided[8]:8:5,chr(9),sample[8]:2:1,chr(9),'0.75-0.85');
  writeln(outfile2,t_value[9],chr(9),pos[9]:8:5,chr(9),neg[9]:8:5,chr(9),
  undecided[9]:8:5,chr(9),sample[9]:2:1,chr(9),'0.85-0.95');
  closefile(outfile2);
end;

procedure TMainForm.StartClick(Sender: TObject);
var i,w:integer;
begin
randomize;
with memo1 do clear;
with Series1 do clear;
   for i:=0 to 10 do
    begin
       t_value[i]:=0;
       sample[i]:=0;
       neg[i]:=0;
       pos[i]:=0;
       undecided[i]:=0;
   end;
  sum_u1:=0;sum_u0:=0;
  OffScreen:=TBitmap.Create;
  OffScreen.Width := x_width;
  OffScreen.Height := y_height;
  randomize;
  samples:=StrToInt(SampleBox.text);
```

```
 ProgressBar1.max:=samples;
 prob_occupied:=StrToFloat(Prob_occ_box.text);
 case process.Itemindex of
   0: iid:=true;
   1: begin
        iid:=false;
        prop_affected:=0.005;
      end;
   2: begin
        iid:=false;
        prop_affected:=0.01;
      end;
   3: begin
        iid:=false;
        prop_affected:=0.016;
      end;
   4: begin
        iid:=false;
        prop_affected:=0.05;
      end;
   5: begin
        iid:=false;
        prop_affected:=0.1;
      end;
 end;

 for w:=1 to samples do
   begin
    ClearBitmap;
    ProgressBar1.position:=w;
    case PatchConfig.Itemindex of
     0:begin
        max_points1 := 74;
        av_max_points2 := 0;
        assign_positions_double;
       end;
     1:begin
        max_points1 := 8;
        av_max_points2 := 9;
        cluster_radius := 70;
        assign_positions_double;
       end;
     2:begin
        max_points1 := 8;
        av_max_points2 := 9;
        cluster_radius := 40;
        assign_positions_double;
       end;
     3:read_data;
    end;
    Initialize;
    Button1Click(MainForm);
   end;
  Button2Click(MainForm);
end;

end.
```