Ecological cycling - Null model

Dynamics without ecological structuring

Here batchruns tells us how many realisations of the cycling to run

```
In[ ]:= batchruns = 100;
```

Initialisation

Lifecycle

First define the global pool of strains
Initially in the global pool all strain are in equal frequency = 0.25 and hence all communities are equally probable. We sample on 96 for the experiment.
The time in the wells t_{well} = 20 is already set so we can choose the time spent in the pool phase relative to it.

```
In[ ]:= pooltimes = RecurrenceTable[{a[n + 1] == 2 a[n], a[1] == 0.02}, a, {n, 1, 10}]
         (*Table[2/i,{i,{100,10,1,0.1}}]//N*);
```

```
In[ ]:= counterlist = (*pooltimes*){1.28}
         (*use a specific time relative to t_well = 20 or the above vector pooltimes*);
        startfeedbackat = 9;
```

startfeedbackat is the cycle number where we start the feedback of nutrients in the next pool. Currently the number is set to be larger than cycles so no feedback occurs. Try 0, 5, 9 to generate results for the parameters in the manuscript

```
In[ ]:= meanfrequenciesofstrains = {};
        meandeath = {};
        storeenvs = {};
        comparingextinctions = {};
        For[counter = 1, counter ≤ Length[counterlist], counter ++,
         tpool = counterlist[[counter]];
         storegraphics = {};
         storebarchartdata = {};
         extinction = {};
         For[batchrun = 1, batchrun ≤ batchruns, batchrun ++,
          fourrans = RandomReal[1, 4];
          distribution = If[Length[counterlist] == 1,
            {0.25, 0.25, 0.25, 0.25}, fourrans / Total[fourrans] // N];
          totalcycles = 10;
          cyclegraphic = {};
```

```
finalnumbers = {};
finalpool = {distribution};
countdeath = {};
For[numberofcycle = 0, numberofcycle < totalcycles, numberofcycle++,
 If[distribution ≠ {0, 0, 0, 0}, {
   sampledcommunities =
    Table[RandomChoice[distribution → speciesproperties, 4], 1];
   production = Total[#] & /@ sampledcommunities;
   prodreduced = production /. {2 → 1, 3 → 1, 4 → 1};
   weights = If[numberofcycle < startfeedbackat, weightdistribution,
     Table[Times @@ Table[If[possibleenvssorted⟦j⟧⟦i⟧ == 0,
         1 - distribution⟦i⟧, distribution⟦i⟧], {i, 4}], {j, 16}]];
   randenvs = RandomChoice[weights → possibleenvssorted, 1];

   If[batchruns == 1 && Length[counterlist] == 1, AppendTo[storeenvs, weights]];
   mix = randenvs + prodreduced;
   partitionedmixes = Partition[Riffle[randenvs, prodreduced], 2];

   validmix = {};
   validcols = {};
   For[i = 1, i ≤ Length[mix], i++, If[MemberQ[mix⟦i⟧, 0] == True,
     AppendTo[validcols, White], AppendTo[validcols, Gray];
     AppendTo[validmix, i]]];

   AppendTo[countdeath, 1 - Length[validmix]];

   totrack = partitionedmixes⟦#⟧ & /@ validmix;
   positionstochoosefromdictionary = Table[Position[
       allvalidcombinations, totrack⟦i⟧]⟦1, 1⟧, {i, 1, Length[totrack]}];
   validgrs =
    Table[envcommunityfitnessmatrix⟦positionstochoosefromdictionary⟦i⟧⟧⟦2⟧,
     {i, 1, Length[positionstochoosefromdictionary]}];

   initconds =
    Table[prodreduced⟦i⟧ x /. Solve[Total[{x, x, x, x} prodreduced⟦i⟧] ==
          inoculumsize, x]⟦1⟧ // N , {i, 1, Length[prodreduced]}];
   initcolors = Blend[base, #] & /@ initconds;
   allgrrates = ReplacePart[ConstantArray[negativegrowthrate, {1, 4}],
     Table[validmix⟦i⟧ → validgrs⟦i⟧, {i, 1, Length[validmix], 1}]];
   finalvalues = {};
   For[i = 1, i ≤ Length[initconds],
    i++, sol = rundynamics[initconds⟦i⟧, allgrrates⟦i⟧]⟦1⟧;
    AppendTo[finalvalues,
     Evaluate[{x1[tmax], x2[tmax], x3[tmax], x4[tmax]} /. sol] // Chop]];

   finalcolors = Table[
     Blend[base, finalvalues⟦i, 1⟧ // Chop], {i, 1, Length[finalvalues]}];
```

```
      AppendTo[cyclegraphic, {initcolors⟦1⟧, finalcolors⟦1⟧}];

      final = Chop[finalvalues, 10^-7];
      finalamounts = {final⟦All, 1, 1⟧ // Total, final⟦All, 1, 2⟧ // Total,
        final⟦All, 1, 3⟧ // Total, final⟦All, 1, 4⟧ // Total};
      finalfractions = If[finalamounts == {0, 0, 0, 0}, {0, 0, 0, 0},
        Chop[finalamounts / Total[finalamounts], 10^-7] // Quiet];
      AppendTo[finalnumbers, finalfractions];
      If[finalfractions == {0, 0, 0, 0}, AppendTo[extinction, numberofcycle]];
      poolsol = pooldynamics[finalfractions, tpool];
      afterppolgrowth = Evaluate[
          {y1[tpool], y2[tpool], y3[tpool], y4[tpool]} /. poolsol]⟦1, 1⟧ // Chop;
      distribution = If[afterppolgrowth == {0, 0, 0, 0}, {0, 0, 0, 0},
        afterppolgrowth / Total[afterppolgrowth] // Chop];
      (*Print[distribution];*)
      AppendTo[finalpool, distribution];},
     AppendTo[finalnumbers, {0, 0, 0, 0}];
     AppendTo[finalpool, {0, 0, 0, 0}];
     AppendTo[countdeath, 1];
     (*Print["here"<>ToString[distribution]];*)
    ]
   ];
   AppendTo[storegraphics, Transpose[cyclegraphic]];
   AppendTo[storebarchartdata,
    {finalnumbers, finalpool, Riffle[finalpool, finalnumbers], countdeath / 1}];
  ];
  AppendTo[comparingextinctions, extinction];
  AppendTo[meanfrequenciesofstrains,
   Transpose[Table[Table[Mean[DeleteCases[#, 0.] & /@
        Transpose[Table[Transpose[storebarchartdata⟦i, 3⟧]⟦j⟧ // N,
           {i, Range[batchruns]}]]⟦k⟧], {k, 1, 2 totalcycles + 1, 1}], {j, 1, 4, 1}]]
    (*Table[Mean[Table[Transpose[storebarchartdata⟦i,3⟧]⟦j⟧//N,
      {i,Range[batchruns]}]],{j,1,4,1}]*)];
  meandeath = AppendTo[meandeath,
    Mean[Table[storebarchartdata⟦i, 4⟧, {i, Range[batchruns]}]]];
 ]
```

Plotting

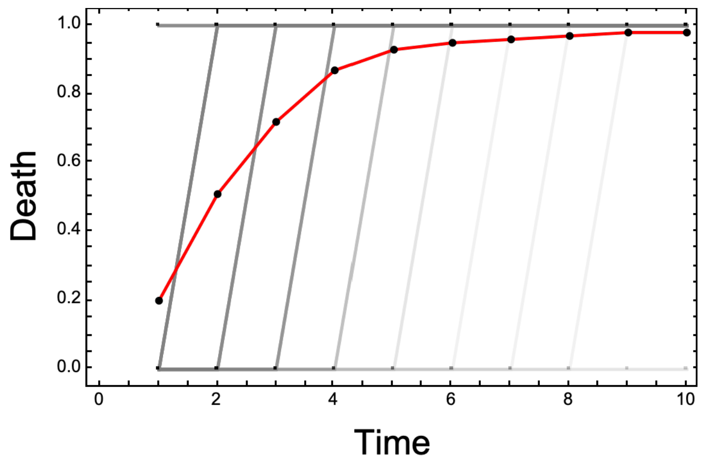
The above lifecycle code can be run in a number of different ways to get the results in the manuscript.
For batchruns = 1 we store the graphics pertaining to each cycle of the the single run.
We can also change the pool times, strain growth rates and the startfeedbacktime to get the relevant figures for single exemplary runs as shown in the manuscript.

When batchruns > 1 (for example when set to 100) and done for multiple pool times the figures for death and mean death will be relevant.
The averaging of the batch runs per t_{pool}will be done automatically for the plots.

For batchruns > 1

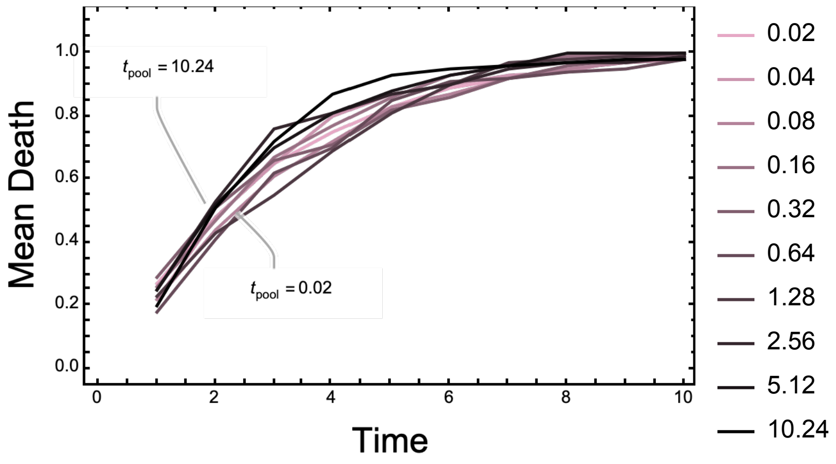
In[ ]:=
```
deathstoplot = Table[storebarchartdata⟦i, 4⟧, {i, Range[batchruns]}];
Show[
 {ListPlot[deathstoplot, Joined → True, PlotStyle → Directive[Gray, Opacity[0.1]],
   PlotRange → {-0.01, 1.01}, Mesh → All, MeshStyle → {Black, Thickness[0.002]},
   Frame → True, FrameStyle → Directive[Black, Thickness[0.002]],
   FrameLabel → {Style["Time", Black, 18], Style["Death", Black, 18]}],
  ListPlot[deathstoplot // Mean, Joined → True, PlotStyle → Red,
   PlotRange → {-0.01, 1.01}, Mesh → All, MeshStyle → {Black, Thickness[0.002]},
   Frame → True, FrameStyle → Directive[Black, Thickness[0.002]],
   FrameLabel → {Style["Time", Black, 18], Style["Death", Black, 18]}]},
 PlotRange → {-0.05, 1.05}]
```

Out[ ]=

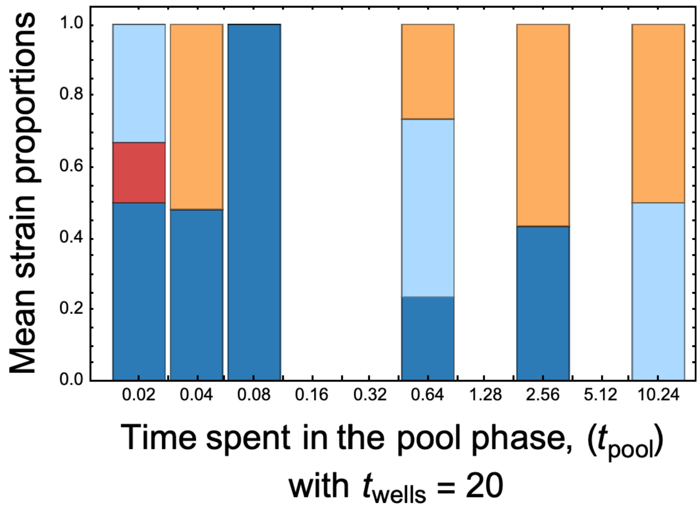


In[∘]:=
```
ListPlot[Table[If[i == 1 || i == Length[counterlist],
    Callout[meandeath⟦i⟧, "t_pool = " <> ToString[counterlist⟦i⟧],
     If[i == 1, {3, 0.38}, {1, 0.8}]], meandeath⟦i⟧],
   {i, 1, Length[meandeath]}], PlotLegends → counterlist⟦1 ;;⟧,
  PlotStyle → Table[{Darker[Hue[0.92, 0.27, 1.], i/Length[counterlist]],
     i/Length[counterlist]}, {i, 1, Length[counterlist]}]
   (*{{Red,Full},{Red,Dashed},{Red,DotDashed}}*), PlotRange → {-0.05, 1.05},
  Joined → True, MeshStyle → {Black, Thickness[0.002]},
  Frame → True, FrameStyle → Directive[Black, Thickness[0.003]],
  FrameLabel → {Style["Time", Black, 18], Style["Mean Death", Black, 18]}(*,
  GridLines→{None, {0.875}},GridLinesStyle→Directive[Red, Dashed,Thick]*)]
```

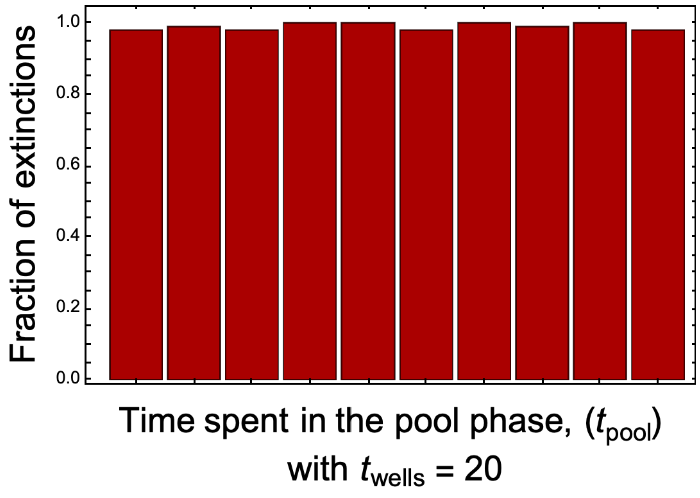
Out[∘]=



```
In[•]:= BarChart[Normalize[#, Total[#, ∞] &] & /@ Table[
           meanfrequenciesofstrains⟦i⟧ // Last, {i, Length[meanfrequenciesofstrains]}],
         ChartLayout → "Stacked", ChartStyle → base, Frame → True,
         FrameStyle → Directive[Black, Thickness[0.002]],
         FrameLabel → {Style["Time spent in the pool phase, (tpool)\n with twells = " <>
             ToString[tmax], Black, 18], Style["Mean strain proportions", Black, 18]},
         ChartLabels → {counterlist, None}(*,
         PlotLabel→Style["Equilibrium proportions",Black,24]*)]
Out[•]=
```



```
In[•]:= BarChart[Length[#]/batchruns & /@ comparingextinctions, ChartStyle → Darker[Red],
         Frame → True, FrameStyle → Directive[Black, Thickness[0.002]],
         FrameLabel → {Style["Time spent in the pool phase, (tpool)\n with twells = " <>
             ToString[tmax], Black, 18], Style["Fraction of extinctions", Black, 18]},
         ChartLabels → {counterlist, None}(*,
         PlotLabel→Style["Equilibrium proportions",Black,24]*),
         PlotRange → {All, {-0.01, 1.05}}]
Out[•]=
```



Run the above code multiple times by changing the startfeedback at for batchsize>1 and store the results in the following variables to generate the histogram and the piechart

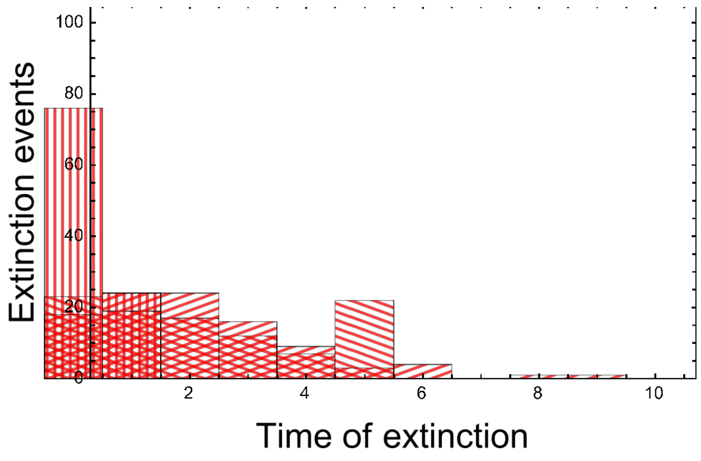
```
In[ ]:= feedbackat9 = extinction;
```

```
In[ ]:= feedbackat5 = extinction;
```

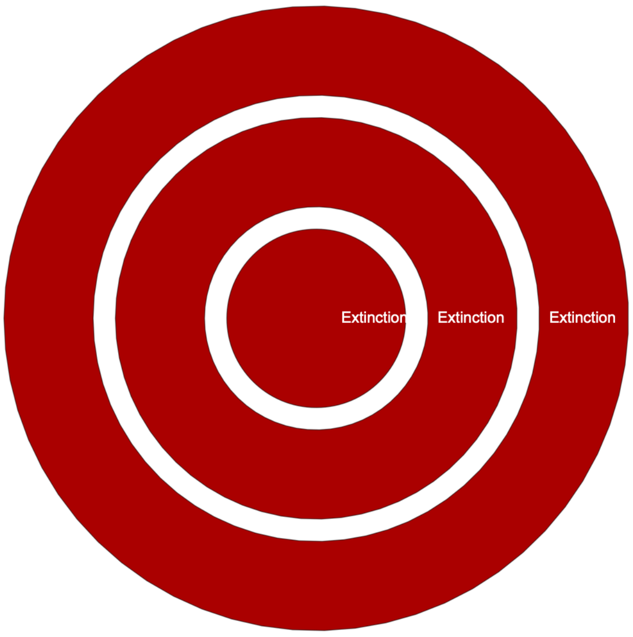
```
In[ ]:= feedbackat0 = extinction;
```

```
Histogram[{feedbackat0, feedbackat5, feedbackat9},
 Frame → True, FrameStyle → Directive[Black, Thickness[0.003]],
 FrameLabel → {Style["Time of extinction", Black, 18],
   Style["Extinction events", Black, 18]}, PlotRange → {{0.5, 10.5}, {0, 100}},
 ChartStyle → {{HatchFilling[90 Degree, 1, 4], HatchFilling[-20 Degree, 1, 4],
    HatchFilling[20 Degree, 1, 4]}, {Darker[Red], Darker[Red], Darker[Red]}}]
```

Out[ ]=



```
In[∘]:= PieChart[{{100 - Length[feedbackat0], Length[feedbackat0]},
          {100 - Length[feedbackat5], Length[feedbackat5]},
          {100 - Length[feedbackat9], Length[feedbackat9]}},
        ChartLabels → {"", Style["Extinction", White, 8]},
        ChartStyle → {White, Darker[Red]}, Background → White]
Out[∘]=
```


Extinction
Extinction
Extinction