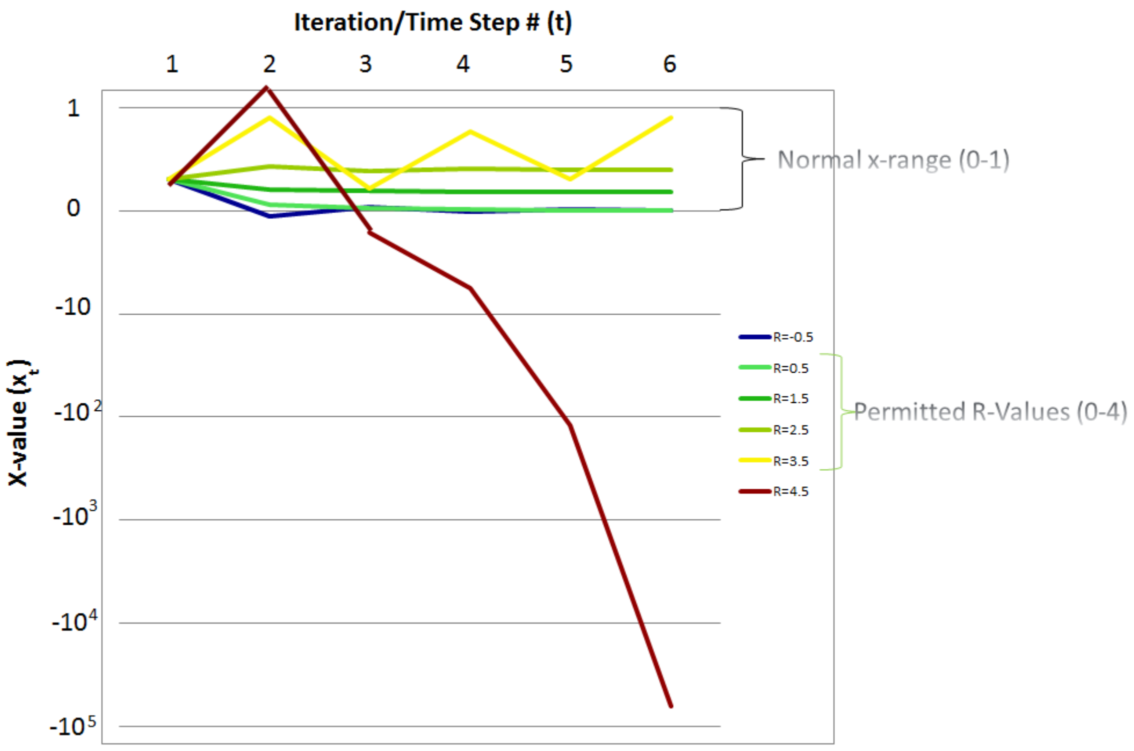
INTRODUCTION TO COMPLEXITY Spring 2014

Unit 2 HW Solutions Guide

Beginner-Level Problems

#1:

The parameter R is assumed to be restricted within the 0 to 4 range in the logistic equation for the x-values to remain stable over multiple iterations. If you try to raise R beyond 4 for most starting values of x, you'll notice after a couple of iterations the value of x moves off the default graph to a value greater than one, followed by your first negative number, and from there shoots down exponentially, running away to negative infinity (i.e. diverging). Setting R in the negative domain will also create anomalous values for x (such as extreme negative numbers that compound out of control). The graph below illustrates these dramatic divergences in x over time from R being just slightly out of range:



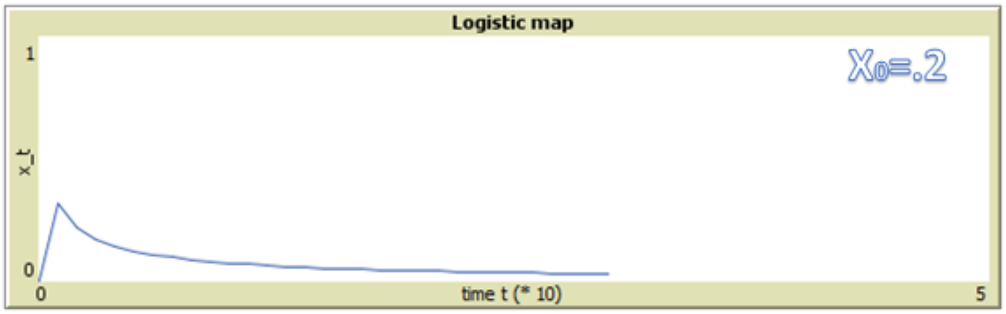
*All seed numbers x_0 starting at 0.5 for convenient comparison. Negative x-values are logarithmically scaled to illustrate dramatic divergence at R=4.5

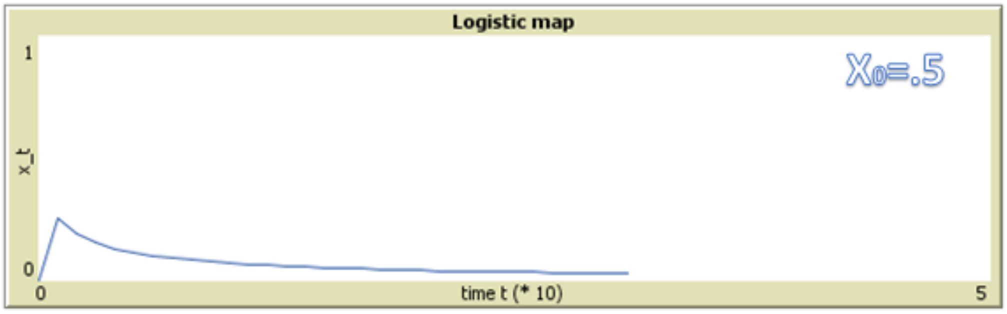
These results also make sense mathematically: The logistic model relies on x to be bounded in the interval 0 and 1. This makes sense in the context of the population growth model: x represents the fraction of carrying capacity, and of course this fraction cannot go above 1. In the mathematical context, x must be kept in the interval 0 to 1 in order to keep the function $x(1-x)$ bounded, since this guarantees neither multiplying factor ("x" nor "$1-x$") exceeds one, preventing subsequent x-values from continuously compounding. Because the highest value normally allowed for $x(1-x)$ is in fact .25 (since $.5*(1-.5)=.25$), R must hence must not go above 4 to guarantee $Rx(1-x) \leq 1$ since $4x(1-x) \leq 1$. If x_t ever exceeds 1 at any step, the subsequent x_{t+1} will be negative due to "$1-x$" being negative and all subsequent negative values blowing up. The non-stop downward spiraling of negative numbers also happens when you allow R to be negative, due to a negative coefficient turning any positive value of $x(1-x)$ into a negative x_{t+1}…

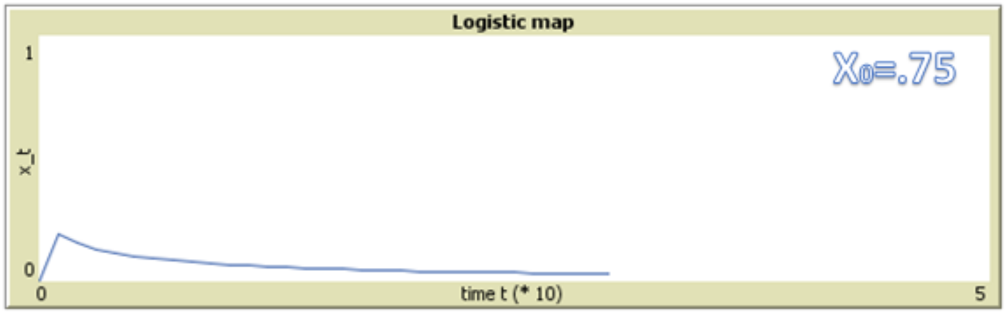
<u>#2:</u>

If $R=1$, the equation $x_{t+1}=Rx_t(1-x_t)$ becomes simplified to $x_{t+1}=x_t(1-x_t)$

This version has special properties that guarantee x always approaches zero. Simply put, because the value $x_t$is always confined between 0 and 1 by the model, and so by extension, multiplying the current x_t by $1-x_t$ (also confined between 0 and 1 by logical extension), resulting in continuously shrinking x's that nonetheless never go below zero. You can observe this by setting R=1 in the Logistic Map NetLogo Model, and notice x continuously gets smaller, creeping closer to zero, as you keep clicking the "step" button. Below are a couple of examples of starting initial values (x_0's) that all eventually converge to zero (or at least asymptotically approach it)…







It can also be shown algebraically that this is where the only stable point that exists (i.e. where the x-value x_{t+1} is stuck at the same value as in its previous iteration x_t) by setting $x_{t+1} = x_t$ and solving for x_t:

$$x_{t+1} = x_t(1 - x_t) = x_t$$

Reduces to solving for this equation

$$\Rightarrow x_t(1 - x_t) = x_t$$

(expanded out from factored form)

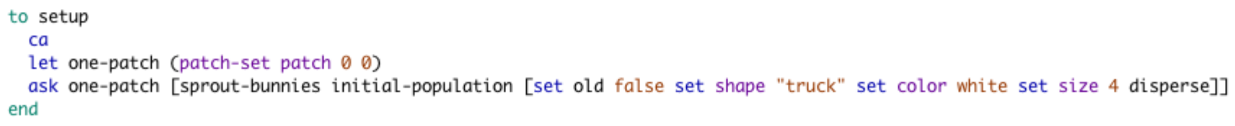
$$x_t - x_t^2 = x_t$$

Then removing x_t from both sides of the equation

$$x_t^2 = 0$$

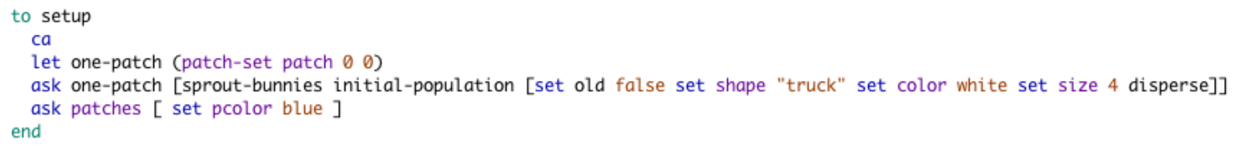
$\Rightarrow x_t = 0$ is the stable point where x repeats itself indefinitely…

Beginner 3.
This simply amounts to going to the “Code” tab in the NetLogo program and changing the “set shape” within “setup.”
For example, I changed the shape to “truck,” as shown in the image below:



```
to setup
  ca
  let one-patch (patch-set patch 0 0)
  ask one-patch [sprout-bunnies initial-population [set old false set shape "truck" set color white set size 4 disperse]]
end
```

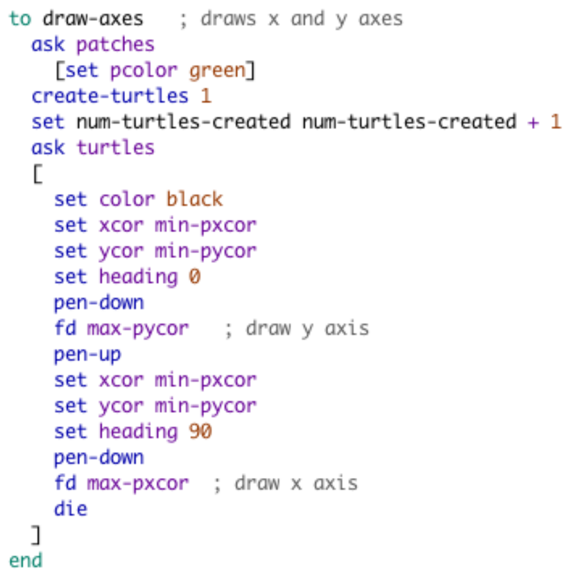
To change the color, again within “setup,” I added the code (without quotations) “ask patches [set pcolor blue]” as shown below:



```
to setup
  ca
  let one-patch (patch-set patch 0 0)
  ask one-patch [sprout-bunnies initial-population [set old false set shape "truck" set color white set size 4 disperse]]
  ask patches [ set pcolor blue ]
end
```

Try it out!

Beginner 4.
To make the background of the plot green, add the code (without quotations) “ask patches [set pcolor green]” to the “to draw-axes” procedure, as shown in the image below:



```
to draw-axes   ; draws x and y axes
  ask patches
    [set pcolor green]
  create-turtles 1
  set num-turtles-created num-turtles-created + 1
  ask turtles
  [
    set color black
    set xcor min-pxcor
    set ycor min-pycor
    set heading 0
    pen-down
    fd max-pycor   ; draw y axis
    pen-up
    set xcor min-pxcor
    set ycor min-pycor
    set heading 90
    pen-down
    fd max-pxcor  ; draw x axis
    die
  ]
end
```

Now, to label the coordinates: within the “to iterate” procedure there are two “ask turtle” commands, one for x0 and one for x0’. Within the procedure for x0, add (without the first set of quotations) “set label (word x-current ", " x-new)” and within the procedure for x0’, add “set label (word x-current' ", " x-new')”. The command “word” tells the program that we are concatenating the instances of the variables with a string, namely the string “, ” so that the coordinate shows up. (Why should we use x-current and x-new, and x-current’ and x-new’, instead of x-cor?)

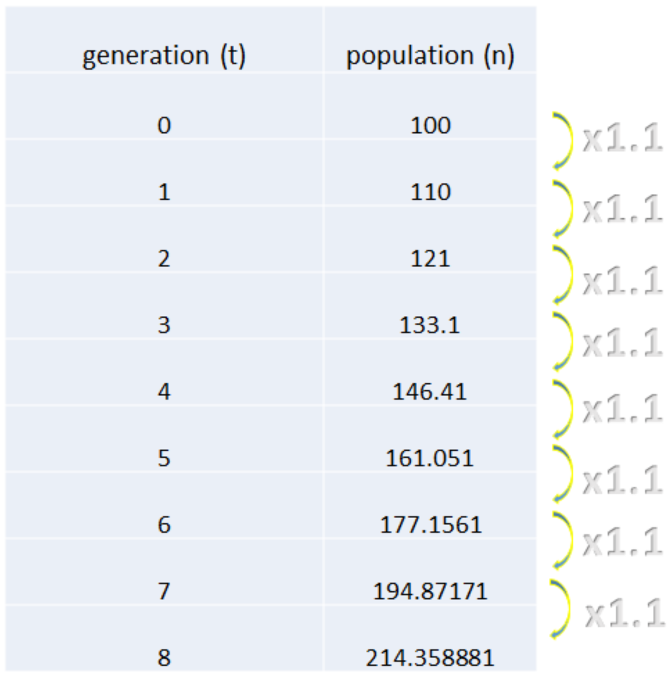
The code then looks like the image below:

```
to iterate
  set x-current x-new
  set x-new (R * x-current * (1 - x-current))  ; one iteration of logistic map
  ask turtle turtlex0-who
  [
      set xcor (x-current * max-pxcor)  ; update coordinates for turtle representing first initial condition
      set ycor (x-new * max-pycor)
      set label (word x-current ", " x-new)
  ]
  set x-current' x-new'
  set x-new' (R * x-current' * (1 - x-current'))  ; one iteration of logistic map
  ask turtle turtlex0'-who
    [
      set xcor (x-current' * max-pxcor)  ; update coordinates for turtle representing second initial condition
      set ycor (x-new' * max-pycor)
      set label (word x-current' ", " x-new')
    ]
end
```

Intermediate-Level Problems

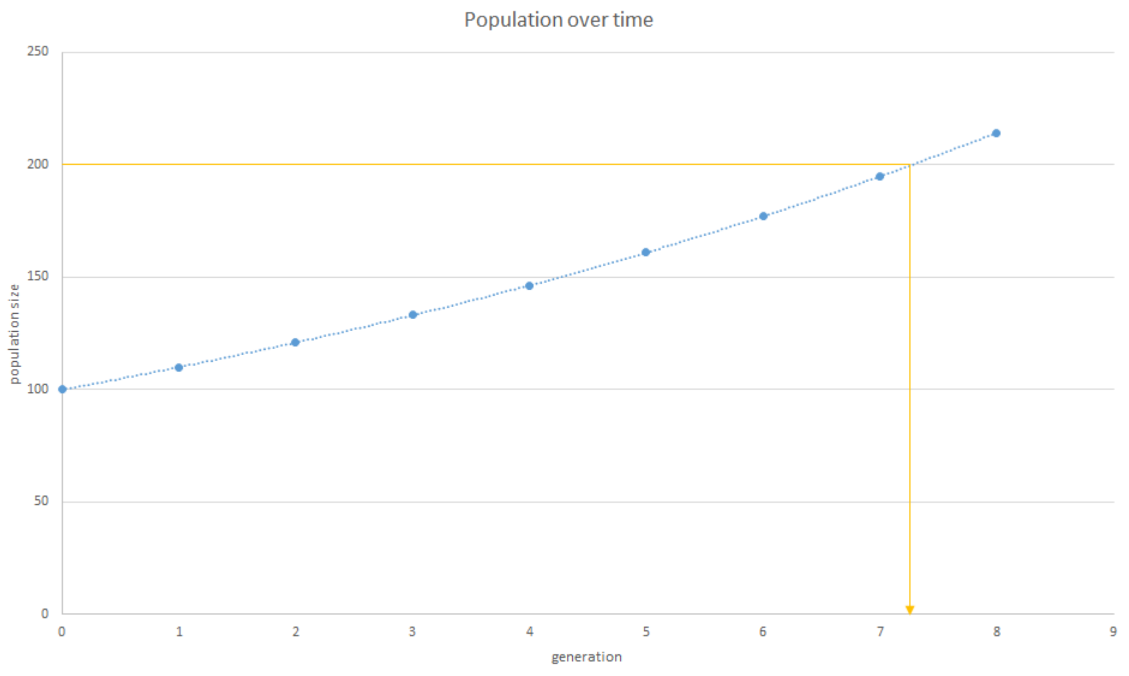
#1:

This is a simple exponential function where the population increases by a factor of 1.1 (i.e. by 10%) per generation from its previous value, as illustrated by the table and plotted on the graph.



generation (t)	population (n)
0	100
1	110
2	121
3	133.1
4	146.41
5	161.051
6	177.1561
7	194.87171
8	214.358881

x1.1
x1.1
x1.1
x1.1
x1.1
x1.1
x1.1
x1.1



We can see that the population doesn't surpass 200 until after the 8th generation. In terms of continuous time units, it appears (visually tracing the point population crosses 200 on the graph above to the corresponding value for t) that this occurs around t = 7.3 generations

We can also get a somewhat more precise numerical answer by algebraically deriving the value of t where n=200.

$$n = 100(1.1)^t = 200$$
$$100(1.1)^t = 200$$

$$1.1^t = 2$$

(we need to take the inverse function of exponent base 1.1 to isolate t here, which is the logarithm base 1.1)

$$t = \log_{1.1} 2$$

Thus, $t \approx 7.2725$ when n=200

#2:

Similar to problem #2 from the beginner's section, the fixed point can be found by solving for $x_{t+1} = x_t$, this time in the case of $R = 2$:

$$x_{t+1} = 2x_t(1 - x_t) = x_t$$

$$\Longrightarrow 2x_t(1 - x_t) = x_t$$

$$2x_t - 2x_t^2 = x_t$$

(Setting equation to zero on one side often simplifies these problems)

$$x_t - 2x_t^2 = 0$$

$$2x_t^2 = x_t$$

$$x_t^2 = 0.5x_t$$

(Divide both sides by x_t)

$$x_t = 0.5$$

Intermediate 3.

This one is more involved as it requires, in addition to including code for x’’, adding a new slider for x0’’, two new monitors for x’’{t} and x’’{t+1}, and editing the logistic map plot of “x_t” and “time t(*10)” to include x’’{t}. The NetLogo file with these modifications can be found here.