EEB C119B/C219B Discussion 1

1. Introduction to R

If you have not used R or need a refresher, open the tutorial http://tryr.codeschool.com/ and do chapters 1, 2, and 3. This will give you the basics to do the rest of this worksheet.

1.1 For loops

One important thing that is not covered in the first two chapters of the tutorial is for loops. This is what a for loop looks like in R:

```
for (ii in 1:10){
    #do something 10 times
}
```

For example, the for loop below sequentially, for each value of k between 1 and 100, modifies variable phi.

```
phi <- 1
for (k in 1:100) {
  phi <- 1+1/phi
}
print(c(k,phi))
```

```
## [1] 100.000000   1.618034
```

More specifically, at the beginning of the for loop, a vector containing all the integers from 1 to 100 in order is created. Then, k is set to the first element in that vector, i.e., 1. Then the R expression from the { to the } is evaluated. When that expression has been evaluated, k is set to the next value in the vector. The process is repeated until, at the last evaluation, k has value 100.

As an aside, note that the final value of phi is the Golden Ratio, 1.618034.

2. Introduction to modeling in R

First, some terminology

Before we start, here are some common terms State variables are quantities describing the current state of the system. A common state variable in models of population dynamics is the size of the population at time t, often denoted N(t). State variables typically change through time in the model – indeed this is often the point of the model. You will need to make some decision about the starting values for the state variables, which are referred to as the *initial conditions*.

Parameters are quantities that describe the processes affecting the system. Often these will correspond to the rate or probability that a given process occurs, e.g. the birth rate or death rate. For simple modeling exercises, the parameter values are typically assumed to be fixed (but there is no reason why they can't change, due to changing environmental conditions for instance).

Coding a discrete model in R

First we will consider discrete-time models – that is, models where time advances in 'chunks' called timesteps. These models are often appropriate for systems with a strong biological rhythm, such as populations with a single annual birth pulse. One advantage of discrete-time models is that they are extremely easy to program. Consider the geometric growth model:

$$N(t+1) = rN(t)$$

This simple equation gives the update rule that is needed to step from one timestep to the next (this is sometimes called a recursion relation). To simulate the model, we just need to step through the desired number of timesteps, typically using a for loop, and use this rule to update the value of the state variable.

2.1 General layout of modeling scripts

Simple scripts used to simulate dynamic models often have four main sections (credit to Steve Ellner and John Guckenheimer for laying this out so simply):

a. Setup statements, if needed (e.g. loading libraries needed for the program).

b. Input data, set parameter values, and/or set initial conditions.

c. The actual calculations.

d. Display the results by plotting, saving, etc.

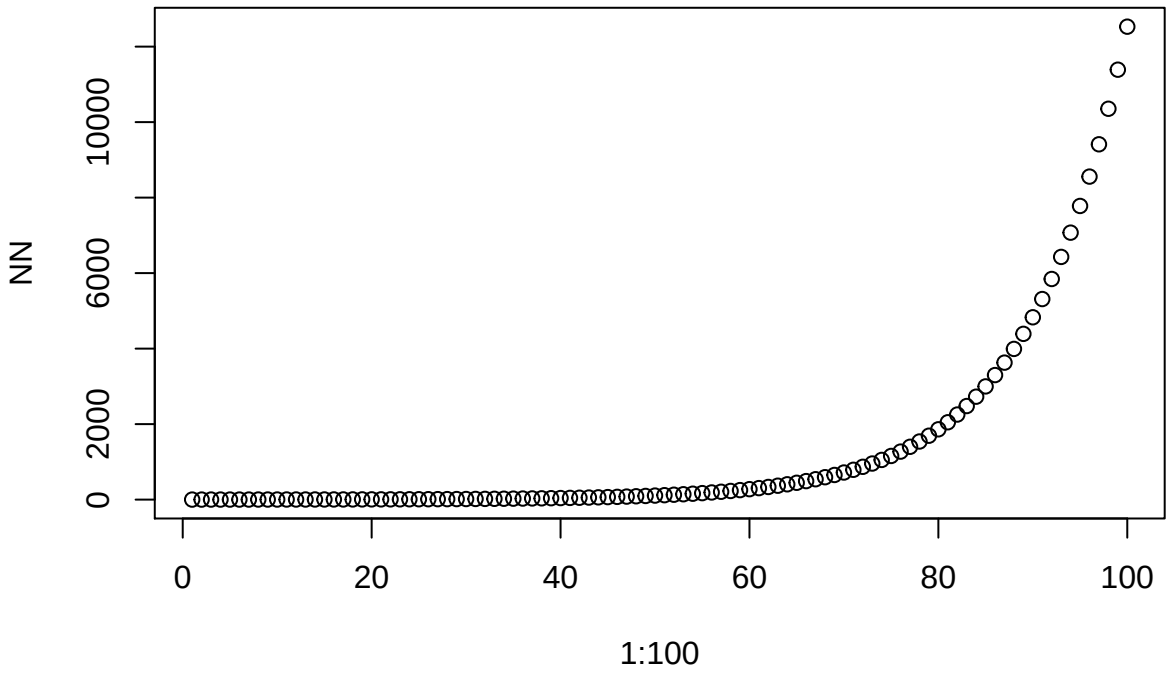
Using this layout isn't mandatory, but it is often a convenient way to structure your thoughts in writing these kinds of scripts.

Write a script to simulate and plot the discrete geometric growth model.

```
NN = rep(NA,100)
NN[1] = 1
R = 1.1
tt = seq(2,100)

for (time in tt){
    NN[time]=R*NN[time-1]
}

plot(1:100,NN)
```



3. Another discrete model

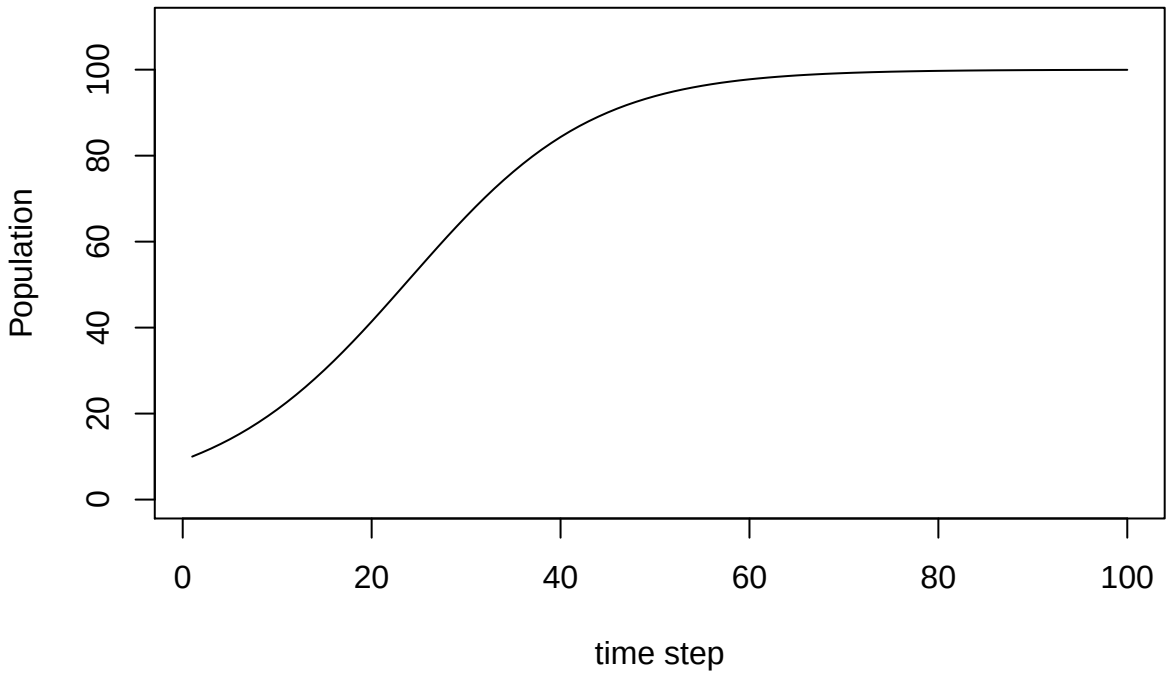
Make a new script called discrete_logistic.R that simulates the discrete logistic growth model:

$$N(t+1) = N(t)(1 + r_d(1 - \frac{N(t)}{K}))$$

```
NN = rep(NA,100)
NN[1] = 10
K=100
R = .1
tt = seq(2,100)

for (time in tt){
    NN[time]=NN[time-1]*(1+R*(1-(NN[time-1]/K)))
}

plot(1:100,NN,type='l',ylim=c(0,K+10),ylab="Population",xlab="time step")
```


Population
0
20
40
60
80
100
0
20
40
60
80
100
time step