Problem Set 2

Stochastic Growth

For last week's problem set you performed a numerical simulation of density-independent geometric growth. Those simulations were deterministic: no matter how often you repeated a simulation with a given set of parameter values you will have gotten the exact same result. (2+2 will always equal 4.)

Few things in the real world are ever that constant. Whether it be because of natural variation in biological and abiotic processes or because we have difficulty counting larger numbers of objects or estimating values, the variables and parameters that contribute to the situation we wish to model typically have some degree of variation or uncertainty associated with them. In class we will discuss these sources of uncertainty in terms of environmental and demographic variation (a.k.a. process noise or process error) and observation error.

In this problem set we will start to consider the effects of such uncertainty using stochastic simulations. Simulation models incorporating stochastic variation come in a wide variety of shapes and forms, and its consideration in ecological theory is growing rapidly, but the essence of their importance is relatively consistent: variation and uncertainty in parameter and variable values can lead to stochastic simulation predictions that are different from those of a deterministic simulation representing the otherwise exact same situation. Depending on its nature, stochastic variation can even lead to seemingly non-intuitive results. Stochasticity affects not only the level of certainty we may ascribe to a given prediction, but the prediction itself may be different. It is often easiest to think about these effects in terms of the mean and variance of the prediction.

Part A

Part A.1

In class we will study the model

$$N_{t+1} = \lambda e^{\epsilon_t} N_t \,. \tag{1}$$

Here, ϵ (epsilon) is a normally-distributed random variable with mean $\mu = 0$ (mu), and standard deviation $\sigma > 0$ (sigma). More specifically, $\epsilon_t \sim \mathcal{N}(\mu = 0, \sigma_\lambda)$. You're going to simulate this model and plot its temporal dynamics. That means modifying your geometric-growth script from last time. (Remember to save it under a new name before modifying it!).

Let's start with the following variable definitions in `R`:

```
N(0) = 1
lambda= 1.01
time-steps = 2500
sigma=0.2
```

To get `R` to give you a single normally-distributed random variable — a different value of λ for each time step — use `rnorm()`. The exponential function is written as `exp()`.

Once you have successfully implemented this stochasticity into your population growth `for` loop, you will see a stochastic version of geometric growth in your plot. (Remember to use a log-scale for the y-axis.) It will look like you have randomly perturbed the geometric growth equation,

which is of course exactly what you've done. Each time you run the script, you will get a new realization of the process. Try running it a few times with the same set of parameter values. A fast way to do this is to hit the up arrow on your keypad within the command-console before hitting enter again, which will cause `R` to repeat the previous command. What happens when you change the value of sigma? Check to make sure your simulation is working correctly by setting sigma to zero. Do you get the right answer?

Now modify your code in order to save each of the stochastic λ values experienced by the population (i.e. each of the randomly drawn values) over time. To do this, you must store the value of for each time-step into a vector. This means declaring a new variable (e.g., `rlambda <- numeric()`) and indexing it by the time-step number, just like you did for saving the value of `N` at time-step in your population-size through time vector. What is the mean λ experienced by your population over time? (Apply the `mean()` function to your vector.) How does this value compare to deterministic λ value you specified? Repeat your simulation a few more times to convince yourself of this pattern. What would happen if the deterministic $\lambda < 1$? Try it (after increasing the initial population size)!

Once you have looked at several individual realizations, your next task is to extend your script so that it plots multiple realizations. This will mean adding an additional `for` loop. (You could also use a `while` loop.) Note that every time you invoke the `plot` function it erases whatever is in the graphical output window. So, for all realizations after the first realization, you have to use a different function to add the new data . The easiest way to do this is to use `lines`. To get a clear sense of the behaviour of this model, plot at as many realizations as you can without turning the entire plot black with lines. (150 reps is probably enough.) You will have to adjust the range of your y-axis to see all the realizations. This is done within the plot function by specifying `ylim`. For example, `ylim = c(1, 10^20)`.

Optional: A trick for plots like this is to change to color of the lines to be semi-transparent. Do this by specifying the alpha-level of the line color as follows:

```
# to convert a named colour into its code
   Col<-as.vector(col2rgb('black')/255)
# to convert the color into a semi-transparent colour
# (alpha sets the level of transparency)
   Col<-rgb(Col[1],Col[2],Col[3], alpha=0.5)
# to add new values of N to existing plot
   lines(N,col=Col)
```

Part A.2

As written, your code will only save the vector of λ values for the last realization. What if we want to save them for each realization? To keep the realizations separated we can't just keep adding the λ values to the same extended vector. Rather, we're going to fill a 2-dimensional array. (When an array is two-dimensional it is effectively the same as a matrix.) Thus, instead of initializing a vector using `numeric()`, we're going to initialize an `array`. This requires specifying its dimensions

```
rlambda<-array(NA, dim=c(reps, TimeSteps)}
```

This creates an array with `reps` number of rows and `TimeSteps` number of columns and fills it with `NA` values. We can index this array similar to how we did it for a vector. If we want to put a value in the 2nd row and 4th column, we would say: `rlambda[2,4] <- value`. Use this trick and update your code in order to save the stochastic λ values and N population sizes of each of your

realizations over time. Note that you will have to place $N(0)$ in the first column of the N-array and add 1 extra column to it in order to have the right dimensions. (Think back to the last homework if that doesn't make sense.) You can use `dim()` to see if you have the right number of dimensions. Turn off the lines that plot your output by placing a `#` in front of them before you run your simulation so that you're simply storing the output. You can use `head()` or `tail()` to take a look at the top or bottom rows of your output. Once you've done that, loop through the rows of our N-array to plot the results for each of your realizations again. (The functions `nrow()` and `ncol()` return the number of rows and columns in a matrix, respectively.) Now you can change anything you want to the appearance of your plot without having to run the simulations over and over again!

Part A.3

Having all the output stored like this allows us to analyze the results as well. Create a function to calculate the geometric mean of a vector as follows:

```
gmean <- function(x){ exp ( mean (log ( x ) ) ) }
```

Use it to calculate the geometric mean of the stochastic λ values you observed across all realizations. Is the arithmetic mean (calculated using the `mean()` function) or the geometric mean closer to the deterministic λ that you specified?

Add the following to your plot of stochastic realizations using `lines()`: (a) the population size over time expected if the deterministic value of λ were equal to the mean observed arithmetic λ, and (b) the population size over time expected if the deterministic value of λ were equal to the mean observed geometric λ. (Set `lwd=3` and `col='red'` or `'green'` to make these lines more visible).

Which is closer to the observed population size over time averaged across the realizations? Why??

Another way of consider this is to look at the distribution of final population sizes. That is, we want to plot a frequency distribution of N_T, where T is the total number of time-steps in your simulation. Extract the final population sizes from your N-array, take the *log* of this vector, and use `hist()` to plot a histogram of the final population sizes. We will show in class that the expected value (the mean) of $log(N_T)$ (where $T >> 0$) can analytically be solved as

$$\mathbb{E}[\log(N_T)] = \log(N_0) + \log(\bar{\lambda_g})T\,,$$

where the logarithm of the geometric mean lambda (λ_g) is equivalent to the arithmetic mean of the log-transformed λ_t values. The variance around that expectation is

$$Var[\log(N_T)] = T\sigma^2_\lambda\,.$$

Use these expectations to test for errors in your code by calculating the arithmetic mean and variance of your log-transformed final population sizes using the functions `mean()` and `var()`. How well do they compare? Use `abline()` to indicate the observed mean final population size and the expected final population size on your histogram (differentiating the lines using color and a thicker line width). How close are they to each other? Could they be closer? How?

Part B

In class we're also going to study the dynamics of a popular formulation of density-dependent growth: the discrete logistic model. This model simply penalizes a population's intrinsic growth

rate (r_d) by an amount proportional to its population size (αN). We can write the logistic equation in two ways:

$$N_{t+1} = N_t + r_d\,(1 - \alpha N_t)\,N_t \tag{2}$$

$$N_{t+1} = N_t + r_d\left(1 - \tfrac{N_t}{K}\right)N_t\,. \tag{3}$$

Modify your script from Problem Set 1 to perform a deterministic simulation of Eqn. 3, specifying $r_d = 1$ and $K = 100$. Plot the dynamics of a population starting at $N(0) = 1$ and show that population size converges to K over time (whereupon $N(t+1) = N(t)$ for all time thereafter). How does varying r_d and $N(0)$ affect the population's final abundance in your simulations? Now solve Eqn. 3 at steady-state (i.e. when $N(t+1) = N(t)$) to analytically show that N will reach a value of K for any intrinsic growth rate and initial population size (other than $N(0) = 0$) .

Using what you've learned in Part A, investigate how the addition of a small amount of normally-distributed stochastic temporal variation in the population's intrinsic growth rate affects the dynamics and equilibrium population size of the population? (Think about how adding normally-distributed variation to a variable is actually different to what we did in adding variation to λ in Part A.) How does adding a small amount of normally-distributed stochastic temporal variation in the population's carrying capacity affect dynamics and equilibrium population size? (Plot 3 or 4 different realizations of each per figure.) What is the effect of combining variation in the two. What are we implicitly assuming by modeling the variation in r and K in this way?

For each of the above (i.e. varying r, K or rK) produce a pair of side-by-side plots illustrating (a) the observed change in population size between time t and $t+1$ as a function of population size at time t, and (b) the observed per capita change in population size between time t and $t+1$ as a function of population size at time t.

You can plot multiple plots in the same figure using the `mfrow` or `mfcol` options of the `par` function. The `par` function specifies higher-order instructions for your plots. For example,

```
# create a plotting area that is divided into one row and two columns
  par(mfrow = c(1, 2))
# plots 10 normally distributed vs. 10 uniformly distributed numbers in red
  plot(rnorm(10), runif(10), col='red')
```

That was a lot, I know. But...something else to try, just if you're still itching for more: As we talked about at the beginning of the class, and as is likely to come up at a few more times in the future, all models should be considered "wrong" (despite the fact that they can be very useful). The logistic model is no exception. To see an example of why, try running Eqn. 3 with $N(0) > K$ and a negative intrinsic growth rate ($r < 0$)!