Workshop 6: Stochasticity

Everything we have done so far has been deterministic, meaning that if we start out with the same initial values, we get the same output, every time. Even chaotic dynamics in the Logistic Growth model are deterministic. In nature, however, many events are stochastic (e.g., a healthy individual might get stepped on and die, or a storm may kill many individuals). Here, we will learn how to incorporate various stochastic processes into models.

```
rbinom(n=1, size=80, prob=0.7)
```

You can read the above code as \Give me a single draw from a binomial distribution with a

Vector arguments

A useful feature of R is that you can pass vectors in as arguments to probability functions. For example, let's go back to our survival example from the Binomial section. Suppose, instead of a single survival probability, we wanted to compare the number of survivors for 10 di erent survival probabilities (0.1, 0.2, ..., 1). We could use the following:

```
my.probs <- 1:10/10
rbinom(n=10, size=100, prob=my.probs)
```

Here, for the rst of the `n=10` draws, `rbinom` uses the rst probability, `my.probs[1]`. For the second of the `n=10` draws, it uses the second probability, `my.probs[2]`, and so on. If `n` is less than `length(my.probs)=10`, not all of the probabilities will be used and if it is greater, then it will recycle values, starting at the beginning.

1. Increase the replication to verify that the values output from `rbinom` do, indeed, match those speci ed by the probability vector. You can do this by simply increasing `n`, but you will need to gure out how to handle the output. Hint: the `matrix` and `rowMeans` commands might be helpful here.

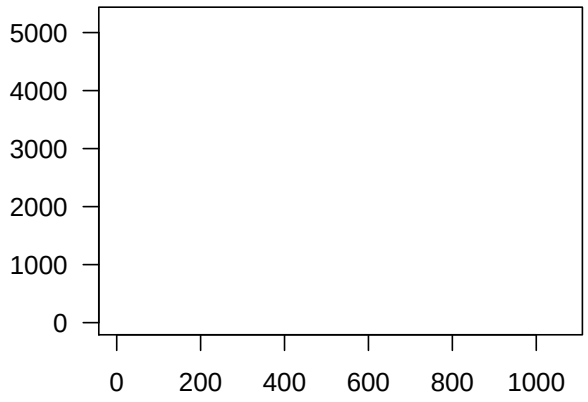
Lotka-Volterra predator-prey model

As we can see, the system cycles with increasing amplitude until, eventually, the predators drive the prey extinct, and then they follow suit and also go extinct. To ensure that my population sizes never became negative, I added two `max` commands (one for predators and one for prey) inside my for loop:

```
for(t in 2:num.iter) {
  H[t] <- max(0, [prey equation here] )
  P[t] <- max(0, [pred equation here] )
}
```

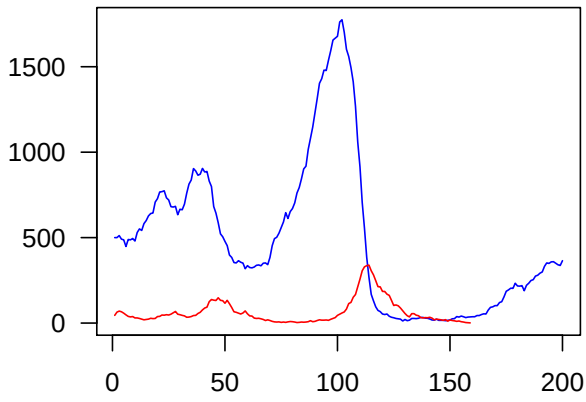
This ensures that any negative values are replaced with zero (e.g., extinction).

2. Make another plot, using the same model output, but now plotting the number of predators vs the number of prey, through time. E.g.,

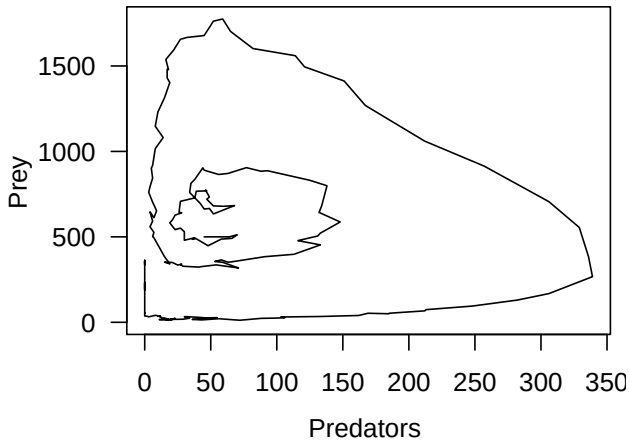


3. If you've stored your output in two vectors, `H` and `P`, see if you can run the following:

```
plot(NA,
     xlim=c(0,max(P)), ylim=c(0,max(H)),
     xlab='Predators', ylab='Prey', las=1)
for(i in 1:(nrow(out)-1))
  lines(2, las=1)
```



In the above, the predators actually went extinct rst and then the prey grew exponentially (I'm only showing the rst 200 generations - prey exponential growth happend after that). This contrasts what we had above without stochasticity, where the prey went extinct rst, which then led to a gradual extinction of predators. Try recreating your gures a few times to compare iterations.



Setting the seed

It is often useful to be able to recreate the same set of stochastic draws (e.g., in case you want to identify a bug in your code that only happens sometimes). To do this, you can set R's random seed, using the `set.seed` command. E.g., try:

```
set.seed(1) ## set the seed
runif(5)
set.seed(1) ## recreate the above 5 draws using the same seed
runif(5)
set.seed(2) ## try a new seed
runif(5)
```

```
set.seed(2) ## recreate the above 5 draws using the same seed
runif(5)
```