Chapter 7 Practice Questions

Question 7.1:
Consider a modified version of the code that we used in exercise 5 to model population growth

```
r=1;                 # growth rate
dt=1;                          # set time step size
K=1000;                        # carrying capacity
s=-r/K;                        # set s to hold K constant
timevec=seq(0, 20, by=dt);   # vector of time steps
Nvec=numeric(length(timevec)); # empty vector for storing abundances
N=1;                           # initial abundance
t=0;                           # set initial time to zero
i=1;                           # counter for tracking position in N vector

while(i<=length(timevec)) {
  dN=(r+s*N)*N*dt;             # calculate dN
  N=N+dN;                      # add dN to the previous N value
  if(N<0) N=0;                 # prevent negative populations
  Nvec[i]=N;                   # store new N value
  t=t+dt;                      # step forward in time
  i=i+1;                       # add one to counter
}

plot(timevec, Nvec, type="l", # plot results
ylim=c(0, 1400))          # set fixed y-axis limits
```

Use this code to simulate and plot discrete-time population dynamics, i.e. with $\mathrm{d}t = 1$. Try simulating dynamics with varying r (e.g. $r = 1$, $r = 1.8$, $r = 2.2$, $r = 2.5$, $r = 3$). Hold K constant at 1000 across all simulations – to

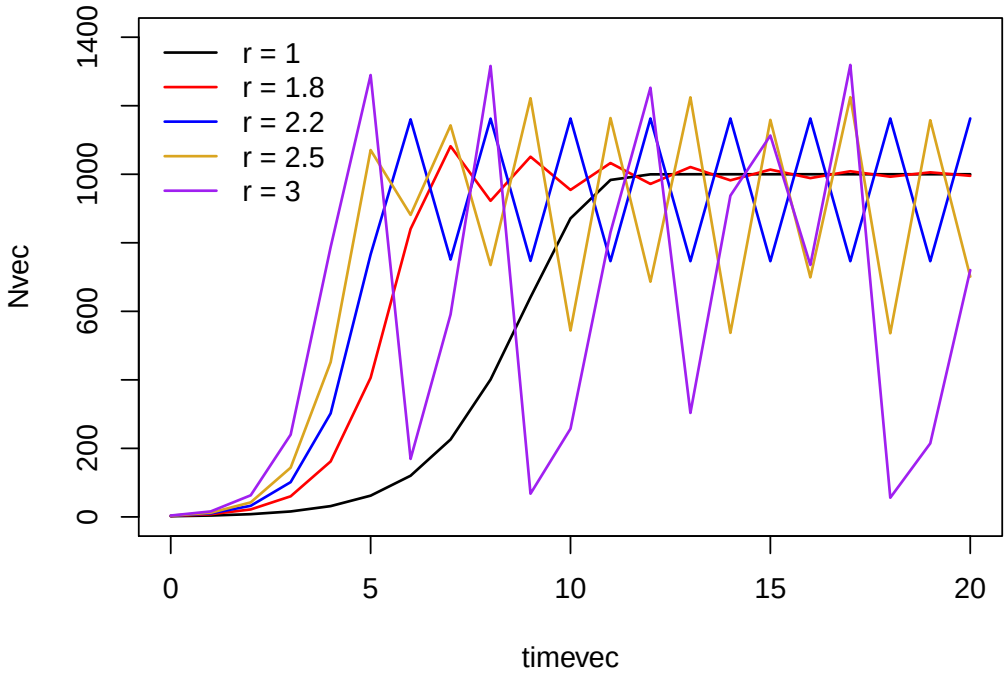
do this, recall that $K = -r/s$, and thus set $s = -r/K$. What do you find as r increases?

Question 7.2:
For the above example with $r = 1.8$, try simulating the system from two different starting points, $N_0 = 1$, and $N_0 = 1.1$. What do you notice about these two simulations? Now, try the same thing, but with $r = 3$. What different type of behavior emerges? What does this suggest about the system?

Question 7.3:
Consider the case with $r = 3$ and $N_0 = 1$. The plot of N vs. t looks almost random. Importantly, however, these dynamics are *chaotic*, and therefore totally deterministic (i.e. governed by fixed underlying and repeatable rules). There are several ways that we can plot these data that reveal this underlying structure. Make such a plot, and explain what it shows.

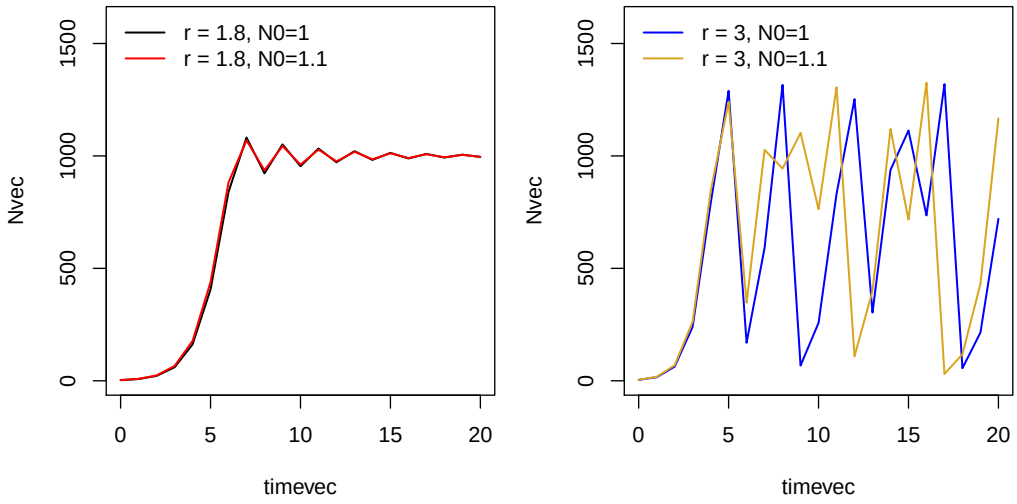
Answers:

Question 7.1:
Increasing r leads to different kinds of oscillatory and limit cycle dynamics. For $r = 1$, the system population K relatively smoothly. For $r = 1.8$, we see damped oscillations (i.e. the population over-shoots the carrying capacity initially, but eventually the oscillations die down and the population approaches carrying capacity). For $r = 2.2$, we have a fixed limit cycle with two points between which the system oscillates. For $r = 2.8$, a four-point limit cycle emerges by around time-step 10. Lastly, for $r = 3$, there is no obvious pattern that emerges – they system just seems to be fluctuating wildly. This last dynamic is a hallmark of chaos.

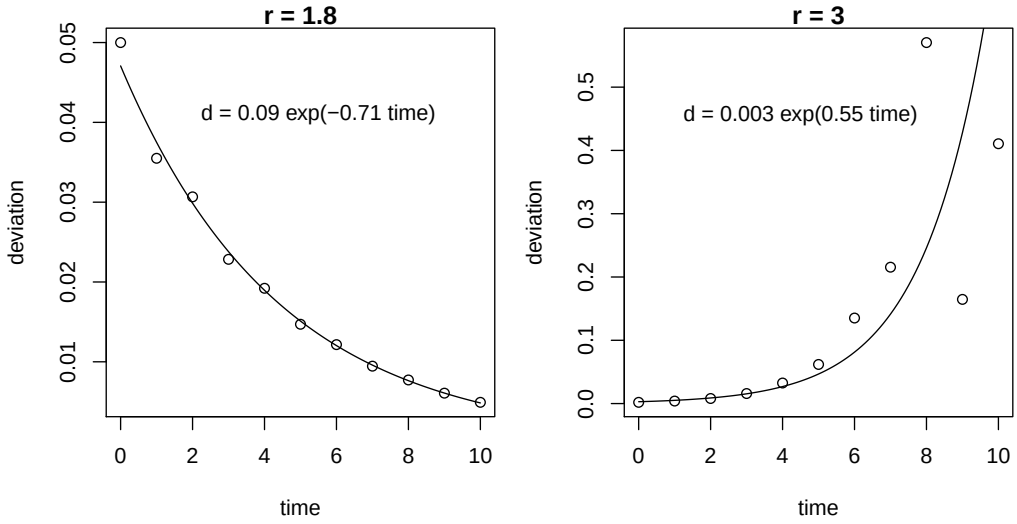


Question 7.2:
For $r = 1.8$, the two simulations follow almost exactly the same trajectory, and quickly converge to the same value. For $r = 3$, the trajectories quickly diverge, such that by about time-step 5, they are following completely different patterns. This sensitivity to starting conditions suggests that the dynamics

of this model are chaotic for $r = 3$, but not for $r = 1.8$.



This example leads us to an interesting point about how to quantify chaos. In general, this is done using something called a Lyapunov exponent. If we plot the distance between the two trajectories as a function of time and fit an exponential relationship to the pattern, we find the following



Note that for $r = 3$ the distance between the points, d increases exponentially, i.e. $d = d_0 e^{\lambda t}$, with growth rate $\lambda \approx 0.55$ (n.b. here we mean

the exponential growth rate of the *distance*, not the abundance). In general, any case where the exponential growth rate $\lambda > 0$ is considered chaotic. In contrast, for the case with $r = 1.8$, $\lambda = -0.71$, which implies convergence among trajectories and therefore non-chaotic dynamics.

Question 7.3:
As noted in the question, plots of N vs. t look almost random. However, we can make several other plots that reveal the structure behind these chaotic dynamics.

First, an option discussed in the textbook is to plot $\frac{1}{N}\frac{\Delta N}{\Delta t}$ vs. N, as this reveals the linear relationship between N and per-capita growth rate. Similarly, one could plot the population growth rate ($\frac{\Delta N}{\Delta t}$), which is simply the per-capita relationship times abundance N. Because we are multiplying a linear relationship by the explanatory variable, this results in a quadratic relationship. Finally, our last example, "Nt" vs. "N_tp1" shows the relationship between $N(t)$, and $N(t+1)$. Recall that in each time step, we calculate $N(t+1) = N(t) + (r + sN(t))N(t)$ (we can ignore dt, since it is set to 1 in our example). Thus, $N(t+1)$ must be related to $N(t)$ following a simple quadratic relationship. This final method, of plotting time-lagged abundance vs. abundance in the current time step, is a common and often effective way for identifying underlying patterns in complex population dynamics.

The code and resulting figure for plotting these patterns are shown below. Any answer that demonstrates an underlying pattern in the data can be considered correct.

```
## Example 1: N vs. t
plot(timevec, Nvec, type="l")    # plot N vs. t

## Example 2: dNNdt vs. N
#note diff(x) function calculates x[i+1]-x[i] for all i
dN = diff(Nvec);                 #get dN
dt = diff(timevec);              # get dt
Nt = Nvec[1:(length(Nvec)-1)];   #N(t)
# note - we need to remove the last element, since
# there is no corresponding future observation from
# which to calculate dNNdt
dNNdt = 1/Nt*dN/dt;              # calculate per-capita growth rate
plot(Nt, dNNdt);
```

```
abline(h=0, lty=2);                # add line at dNNdt = 0
abline(a=r, b=s);                  # add line following dNNdt = r+sN

## Example 3: dNdt vs. N
dNdt = dN/dt;                      # calculate population growth rate
plot(Nt, dNdt);
abline(h=0, lty=2);                # add line at dNNdt = 0
#dN/Ndt = r+sN
#dN/dt = (r+sN)*N
Nseq<-seq(0, 1400, by=1)           # a sequence of N values for plotting
dNdt_est<-(r+s*Nseq)*Nseq          # analytical estimate of dNdt
lines(Nseq, dNdt_est)              # add line

## Example 4: N(t) vs. N(t-1)
Nt = Nvec[1:(length(Nvec)-1)];     #N(t)
N_tp1 = Nvec[2:length(Nvec)];      #N(t+1)
plot(Nt, N_tp1)
lines(Nseq, Nseq+(r+s*Nseq)*Nseq)
```