Lab 4: for loops and Compartmental Models

STAT 244NF: Infectious Disease Modeling

SOLUTIONS

08 October, 2021

In our last class, we were introduced to compartmental models. While we haven't learned much about them yet, we have been developing a general set of differencing equations to describe how individuals move from one compartment to another. For an SIR model, the general set of differencing equations (in terms of t and $t+1$) can be written as:

$S_{t+1} = S_t - \lambda_I S_t$

$I_{t+1} = I_t + \lambda_I S_t - \lambda_R I_t$

$R_{t+1} = R_t + \lambda_R I_t$

Given a set of parameters, λ_I and λ_R, a population, N, and an initial infectious individual, we want to simulate the movement of individuals: $S \to I \to R$. The most straightforward tool do this is a loop.

Loops

R is very good at performing repetitive tasks, and if we want to repeat the same task over and over, we can use a loop. There are three main kinds of loops, including for loops, while loops, and repeat loops. We are going to work with the for loops in this lab.

for loops

Example 1

Suppose we would like to print the following:

```
print(paste("Today is October", 6))
```

```
## [1] "Today is October 6"
```

```
print(paste("Today is October", 7))
```

```
## [1] "Today is October 7"
```

```
print(paste("Today is October", 8))
```

```
## [1] "Today is October 8"
```

```
print(paste("Today is October", 9))
```

```
## [1] "Today is October 9"
```

```
print(paste("Today is October", 10))
```

```
## [1] "Today is October 10"
```

Typing these lines (or even using copy/paste) over and over is tedious and inefficient, and there is a better way! As we can see, everything in the above 5 lines is the same, except for the number corresponding to the date. In coding, we would like to avoid repeating ourselves, and we would like to use a for loop to automate this process. To do this, we can write the following:

```
for (day in 6:10){
  print(paste("Today is October", day))
}
```

```
## [1] "Today is October 6"
## [1] "Today is October 7"
## [1] "Today is October 8"
## [1] "Today is October 9"
## [1] "Today is October 10"
```

As you can see, this code produces the same result, and it is much shorter; `day` is a *variable* in this code (in the programming sense of the term), and the argument supplied to `day` is the sequence 6:10. We can call the variable something else if we wish; it is not uncommon to see a generic `i` used, which stands for index:

```
for (i in 6:10){
  print(paste("Today is October", i))
}
```

```
## [1] "Today is October 6"
## [1] "Today is October 7"
## [1] "Today is October 8"
## [1] "Today is October 9"
## [1] "Today is October 10"
```

As with naming objects in R, it is often helpful to use a descriptive name for the variable in your loop so the code is readable.

Example 2

Let's suppose we want to add 1 to the numbers 1 through 10. Write a for loop that accomplishes that task. (There are other ways to do this, but this is meant to help us practice for loops.)

```
for (i in 1:10){
  print(i+1)
}
```

```
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
```

Example 3

In many cases, we would like to also store whatever we are generating through our for loop in an object, which we define prior to executing our for loop. The storage object (`add_storage` below) must be the same size as whatever we are storing. We will define this object for storage in the chunk below, which we will use to store what we executed in Example 2:

```
add_storage <- rep(NA, length(1:10))
```

Rewrite the following as a loop:

```
add_storage[1]  <- 1+1
add_storage[2]  <- 2+1
add_storage[3]  <- 3+1
add_storage[4]  <- 4+1
add_storage[5]  <- 5+1
add_storage[6]  <- 6+1
add_storage[7]  <- 7+1
add_storage[8]  <- 8+1
add_storage[9]  <- 9+1
add_storage[10] <- 10+1
```

```
for (i in 1:10){
  add_storage[i] <- i+1
}
```

Example 4

Modify your code in Example 3, so that instead of adding 1 to 1:10, you add 1 to the previous value of `add_storage[i-1]`. In other words, rewrite the following as loop (*Hint: you will need to execute the first line before you start your for loop. This is called initializing.*):

```
add_storage <- rep(NA, length(1:10))

add_storage[1]  <- 1+1
add_storage[2]  <- add_storage[1]+1
add_storage[3]  <- add_storage[2]+1
add_storage[4]  <- add_storage[3]+1
add_storage[5]  <- add_storage[4]+1
add_storage[6]  <- add_storage[5]+1
```

```
add_storage[7]  <- add_storage[6]+1
add_storage[8]  <- add_storage[7]+1
add_storage[9]  <- add_storage[8]+1
add_storage[10] <- add_storage[9]+1
```

```
add_storage <- rep(NA, length(1:10))

add_storage[1]  <- 1+1
for (i in 2:10){
  add_storage[i] <- add_storage[i-1]+1
}
```

for loops and the SI model

Before we think about the SIR model, which is where we started this discussion, let's think about a slightly simpler model - the SI model. Recall, this is the model we said would be appropriate for HIV, since infectious individuals remain infectious throughout their lives (assuming no ART). The differencing equations for this model would be something like:

$S_{t+1} = S_t - \lambda_I S_t$

$I_{t+1} = I_t + \lambda_I S_t$

Write code to simulate from an SI model. You can assume the following:

```
N <- 1000
lambdaI <- 0.02
S0 <- 999
I0 <- 1
```

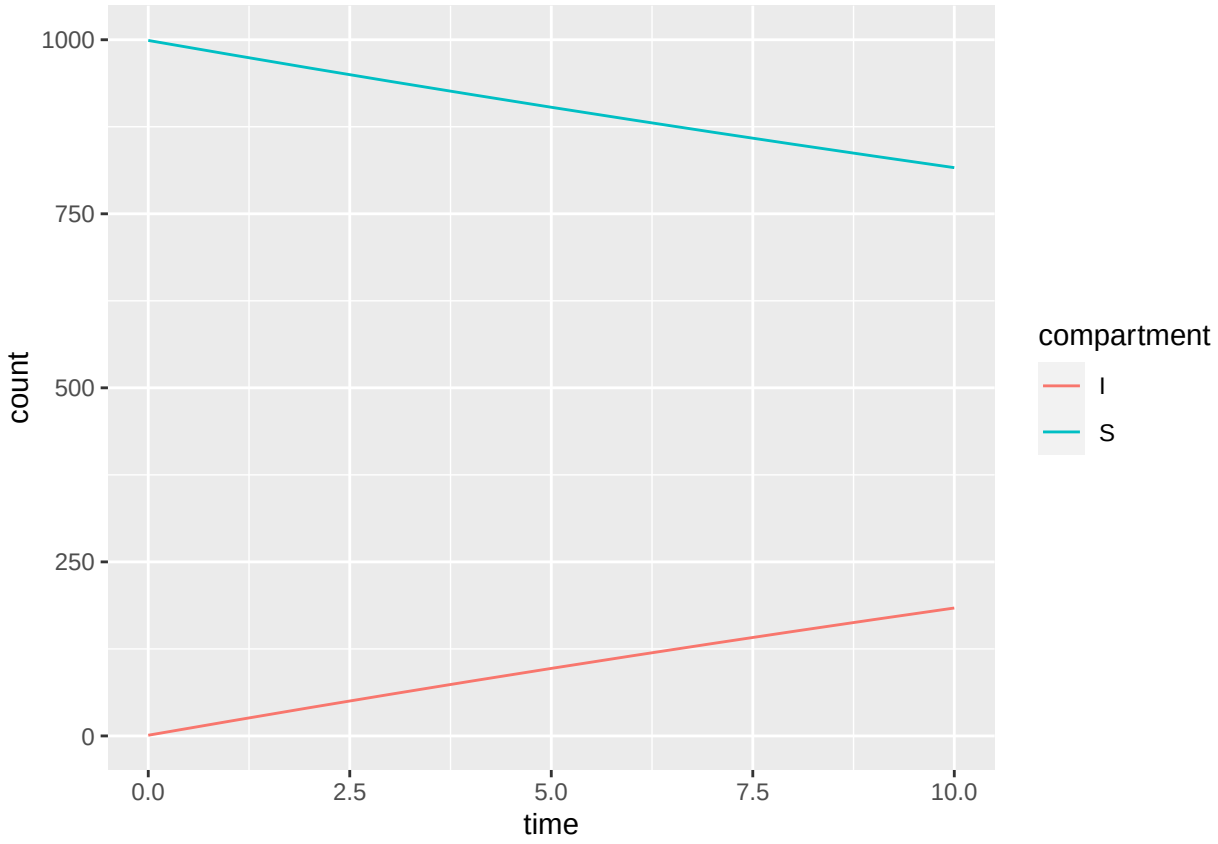
You will need to define storage for S and I before writing your for loop. Assume that we observe this process for 10 time points.

```
S <- rep(NA, length(1:11))
I <- rep(NA, length(1:11))

S[1]  <- S0
I[1]  <- I0
for (i in 2:11){
  S[i] <- S[i-1]-lambdaI*S[i-1]
  I[i] <- I[i-1]+lambdaI*S[i-1]
}
```

Combine your S and I vectors into a data frame, along with a vector for time (1:10) and use ggplot to plot these two curves (one for S and one for I). In this data frame, you will want three columns, one for time, one to denote the compartment (S or I) and one to hold the count values for each of S and I. The data frame should have 20 rows and 3 columns. (22 rows if you include time 0).

```
SI_df <- data.frame(time = rep(0:10, 2),
                    compartment = rep(c("S", "I"), each=11),
                    count = c(S, I))

library(ggplot2)
```

```
ggplot(data=SI_df, aes(x=time, y=count)) +
  geom_line(aes(color=compartment))
```



Time permitting:

We would like to modify our code so that λ_I depends on time, since it should be influenced by the number of infectious individuals in the population. Let's brainstorm about how to do this.

No time left – we will get to this next week.