Lab 5: Writing R Functions with Application to Compartmental Models

STAT 244NF: Infectious Disease Modeling

Sample SOLUTIONS

Functions in R

We have encountered built-in functions in this class over the course of the semester, ranging from those in base R (did not require you to load a package, like `rep` or `glm`), to those that are specially loaded when we load a specific package (e.g., `ggplot` from the ggplot2 package). Now, we want to write our own functions in R. This is going to keep us from repeating the same code over and over.

Example 1

If we want to convert a temperature recorded in Celsius to Fahrenheit, we can use the following formula:

$$T_{\text{Fahrenheit}} = T_{\text{Celsius}} \times \frac{9}{5} + 32.$$

Suppose we want to convert 20 degrees Celsius to Fahrenheit. To accomplish this, we could simply write the following:

```
20*9/5+32
```

```
## [1] 68
```

In many cases, we want to be able to apply a formula multiple times, either within the same document or across many reports. So, we are going to write a function. The general setup for a simple function is:

```
THIS IS EXAMPLE CODE BUT SHOULD NOT RUN
function_name <- function(variable){
  object <- formula(variable)
  return(object)
}
```

Below is a function that I wrote to convert Celsius to Fahrenheit:

```
celsius_to_fahrenheit <- function(C_temp){
  F_temp <- C_temp*9/5+32
  return(F_temp)
}
```

Apply this function to convert 20 degrees Celsius to Fahrenheit.

```
celsius_to_fahrenheit(C_temp=20)
```

```
## [1] 68
```

The following function is equivalent, although possibly less readable:

```
celsius_to_fahrenheit <- function(C_temp){
  C_temp*9/5+32
}
```

Example 2

Write a function that converts a temperature in Fahrenheit to Celsius. Call it `fahrenheit_to_celsius`; call the variable `F_temp`.

```
fahrenheit_to_celsius <- function(F_temp){
  C_temp <- (F_temp-32)*5/9
  return(C_temp)
}
```

Make sure your function works - run your function for 68 degrees Fahrenheit. You should get 20 degrees Celsius.

```
fahrenheit_to_celsius(F_temp = 68)
```

```
## [1] 20
```

What happens if you remove the `return(C_temp)` from your function. Try it - what do you conclude?

```
fahrenheit_to_celsius <- function(F_temp){
  C_temp <- (F_temp-32)*5/9
  # return(C_temp)
}

fahrenheit_to_celsius(F_temp=68)
```

The function does not return anything. We need to tell the function to return an object with the line `return(C_temp)`, or in the less readable form, we need to not assign our calculation to an object. In this case R will return the last line that is written that is not assigned to an object. Make sure you are explicit about what you want to return!!

Example 3

You can include more than one variable in your function (just like in built-in functions in R). Let's suppose for some reason we want to calculate body mass index (BMI), which is a function of height and weight. The units are kg/m^2. The formula for BMI is

$$\text{BMI} = \frac{\text{weight}}{(\text{height})^2}.$$

Write a function to calculate BMI; you can call it BMI. This function should have two variables, weight and height.

```r
BMI <- function(weight, height){
  BMI <- weight/height^2
  return(BMI)
}
```

Test your function for a weight of 60 kg and height of 1.7 meters.

```r
BMI(weight=60, height=1.7)
```

```
## [1] 20.76125
```

The BMI is 20.76 kg/m^2.

What happens if you fail to specify a value for height?

```r
# BMI(weight=60)
```

You will get an error that you are missing an argument for height.

Writing a function for an SIS model

Remember the differencing equations for an SI model:

$$S_{t+1} = S_t - \lambda_t S_t$$
$$I_{t+1} = I_t + \lambda_t S_t$$

where $\lambda_t = 1 - \exp\left\{-c_e \times \frac{I_{t-1}}{N}\right\}$ and $c_e = \frac{R_0}{D}$.

As discussed last class, if we have an infectious period, D, defined, then an SI model is not appropriate. Let's suppose an SIS model is more appropriate (we're considering an infection that confers no immunity after an individual stops being infectious). Then, the differencing equations are:

$$S_{t+1} = S_t - \lambda_t S_t + \rho I_t$$
$$I_{t+1} = I_t + \lambda_t S_t - \rho I_t$$

where $\rho = \frac{1}{D}$.

First, write a for loop to simulate from an SIS model. You can assume the following:

```r
N <- 1000
R_0 <- 2
D <- 3 #days
S0 <- 999
I0 <- 1
Time <- 100
```

```r
## function to calculate lambda_t
lambda_t_fcn <- function(R_0, D, I_i, N){
  c_e <- R_0/D
  return(1-exp(-c_e*I_i/N))
}

SIS_df <- data.frame(time=0:Time,
                     S=rep(NA, Time+1),
                     I=rep(NA, Time+1),
                     lambda_t=rep(NA, Time+1))

SIS_df$S[1] <- S0
SIS_df$I[1] <- I0

for (i in 2:(Time+1)){
  SIS_df$lambda_t[i] <- lambda_t_fcn(R_0=R_0, D=D, I_i=SIS_df$I[i-1], N=N)
  SIS_df$S[i] <- SIS_df$S[i-1]-SIS_df$lambda_t[i]*SIS_df$S[i-1]+1/D*SIS_df$I[i-1]
  SIS_df$I[i] <- SIS_df$I[i-1]+SIS_df$lambda_t[i]*SIS_df$S[i-1]-1/D*SIS_df$I[i-1]
}

head(SIS_df)
```

```
##   time        S        I     lambda_t
## 1    0 999.0000 1.000000           NA
## 2    1 998.6676 1.332445 0.0006664445
## 3    2 998.2250 1.775015 0.0008879021
## 4    3 997.6361 2.363888 0.0011826438
## 5    4 996.8531 3.146887 0.0015746843
## 6    5 995.8129 4.187056 0.0020957258
```

It would be preferable to write a function so we can change the sample size, basic reproductive number, and all the other quantities above by simply specifying different arguments for variables in a function. Here is a rough outline of what you should aim for in your function:

```
SIS_simulation <- function(N, S0, I0, plus some other things){
  Create your storage
  Calculate your c_e

  Add your for loop in here

  Return your storage
}
```

```r
SIS_simulation <- function(N, S0, I0, R_0, D, Time){
  SIS_df <- data.frame(time=0:Time,
                       S=rep(NA, Time+1),
                       I=rep(NA, Time+1),
                       lambda_t=rep(NA, Time+1))

  SIS_df$S[1] <- S0
  SIS_df$I[1] <- I0
```

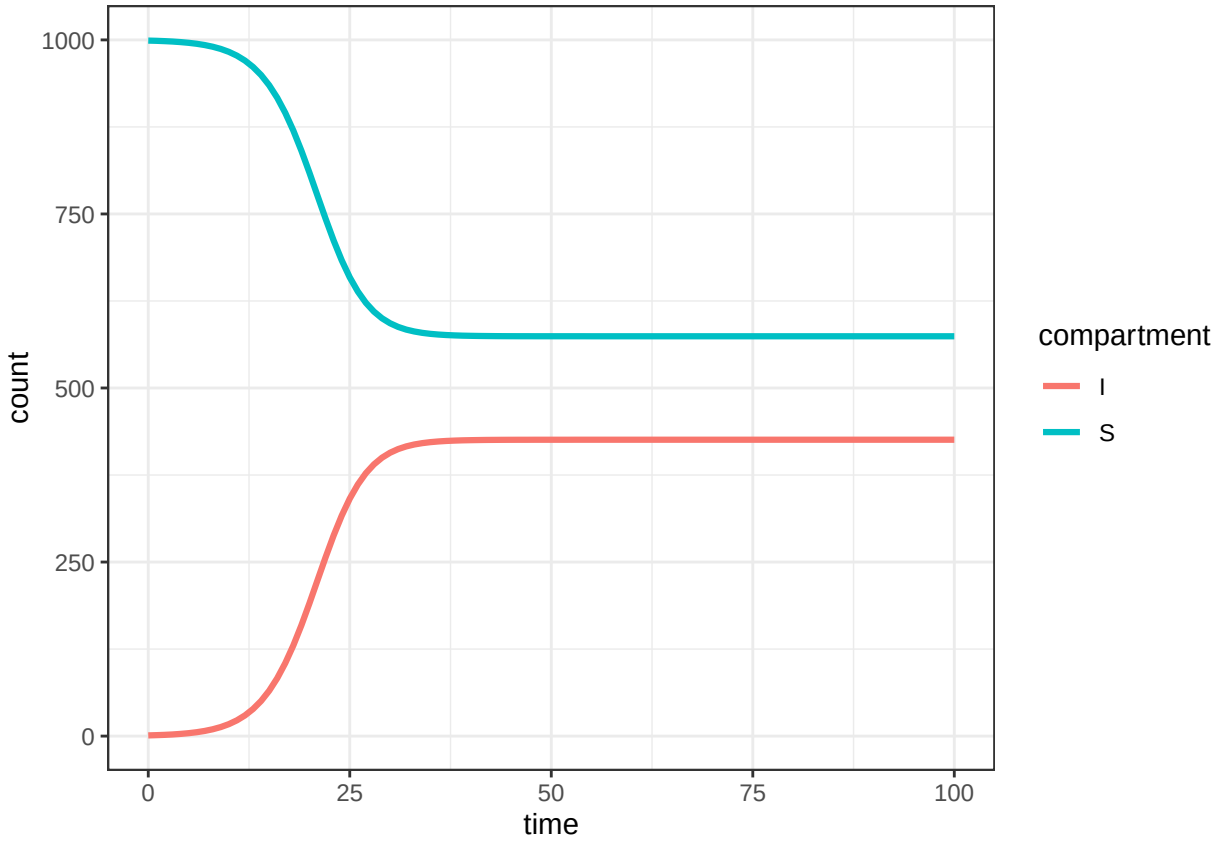
```
  for (i in 2:(Time+1)){
    SIS_df$lambda_t[i] <- lambda_t_fcn(R_0=R_0, D=D, I_i=SIS_df$I[i-1], N=N)
    SIS_df$S[i] <- SIS_df$S[i-1]-SIS_df$lambda_t[i]*SIS_df$S[i-1]+1/D*SIS_df$I[i-1]
    SIS_df$I[i] <- SIS_df$I[i-1]+SIS_df$lambda_t[i]*SIS_df$S[i-1]-1/D*SIS_df$I[i-1]
  }

  return(data.frame(time=rep(0:Time, 2),
                    compartment=rep(c("S","I"), each=(Time+1)),
                    count=c(SIS_df$S, SIS_df$I)))
}
```

Test out your function by using the same arguments for the variables that you specified when you wrote your for loop. You should get the same result.

```
sim1 <- SIS_simulation(N=N, S0=S0, I0=I0, R_0=R_0, D=D, Time=Time)
```

Now that you have written this function, you can test it out for other population sizes, reproductive numbers, infectious period, and time duration. Depending on how much time you have, you can try varying just one of these values, or varying multiple parameters. Plot your results so you can compare them.

```
## same conditions
ggplot(data=sim1, aes(x=time, y=count)) +
  geom_line(aes(color=compartment), linewidth=1.1) +
  theme_bw()
```



You should have other plots where you change one or more of your arguments. Note, $S_t + I_t$ should always add up to N for all values of t!