Coding in R

Dr Josh Hodge

Brief

These exercises have been designed for you to build on your understanding of the coding commands introduced in lecture. These will be:

- if…else statements
- ifelse statements
- for loops

The exercises have been designed to help you develop your interpretation of code, debugging of and writing code to solve some abstract and real-world problems. The exercises may require you to integrate other functions, such as min(), max() and abs(), so do use a search engine and/or the help function. Support is provided via MS Teams and I suggest you collaborate within your groups set-up - programming outside of academia is completed collaboratively!

The practical is split into three sections. Sections one and two are the core material and could creep up in an exam (if MSc) – there are some very challenging questions. Section three is there for more challenging exercises and for those that want to expand their knowledge further and introduces you to some new content.

Section One: Interpretation and Debugging

For this section, I want you to interpret the code and answer the associated quiz on Blackboard. This is purely to check your understanding and you can complete the quiz multiple times if needed. After completion of the quiz, written feedback is provided for each question for you to consult.

1. What would be the output of?

```
ifelse(sqrt(9)<2,sqrt(9),0)
```

2. What would be the output of?

```
ifelse(sqrt(100)>9,sqrt(100),0)
```

3. What would be the output of?

```
z<-6
if(z<0){
  y<-z*3
```

```
  } else{
    y<-z*5
    y
}
```

4. What would be the output of?

```
x<-15
y<-3
if(is.numeric(x)){
if(is.numeric(y) & y!=0){
  z=x/y
  z
}}
```

5. What would be the output of?

```
x<-letters[20]
if(is.numeric(x)) {
print('is numeric')} else{
if(is.character(x)){
print('is character')
}}
```

6. The following code chunk is counting the number of colors in the R palette that are red. We can search for this by counting the characters that include the string “red”. Your job is to debug the following code so that is works (str_detect() is a function part of the package stringr so look it up). There are five errors in total.

```
require(stringr)

## Loading required package: stringr

count<- 0
all<- colors()
for(colour within all)(
if(str_detect(color, "red")){
  count <- cnt+1
}
  print(count)
)
```

7. We’re going to load a dataset from the Park Grass Experiment that has measured community composition over a time series for a whole host of sites. Your job is to debug the following for loop that is trying to count the recordings for each species. There are five errors in total.

```
parkgrass <- read.csv("parkgrass.csv")
species_list <- unique(parkgrass$species)

for(species in species_list){
  print(paste(There are, total(species=parkgrass$species), "of the species:
", species_list)
}
```

8. To calculate total species richness for each site in the year 2000, your supervisor has developed the following code that needs debugging. There are five errors in total.

```
results <- data.frame(site=character(), count=numeric())
for(i in unique(parkgrass$plot)){
  temp_file <- subset(parkgrass, parkgrass$plot!=i&parkgrass$year!=2000)
  speciesrichness <- length(temp_file$species)
  temp <- data.frame(site=speciesrichness, count=i)
  results<- rbind(results, temp)
}
```

9. To calculate the mean composition per species in 1993, your supervisor has developed the following code that needs debugging. There are three errors in total.

```
results<- data.frame(species=character(), mean.comp=numeric())
parkgrass_1993<- subset(parkgrass, parkgrass==1993)
for(species in unique(parkgrass_1993$species)){
  temp_file <- subset(parkgrass_1993, parkgrass_1993$species==i)
  mean<- average(temp_file$comp)
  temp<- data.frame(species=i, mean.comp=mean)
  results<- cbind(results, temp)
}
```

10. From question 7, *Holcus lanatus* was identified as having the most recordings across the years and you are now going to examine this data a little closer. A research assistant has inputted the data from the original recording sheets into an csv file for you to use in your programming. The goal of chunk of code is to find the means for the sites that have received no fertiliser or liming treatment (sites 2/2a, 3a, and 12a) across the time series. You'll notice the data reads in with weird column names (e.g. X1991) and this is because numbers as column names aren't handle well by R. Your job is to interpret the following code that is calculating the total percentage for *Holcus lanatus* for the sites of 2/2a, 3a and 12a across the recorded years and answer the questions your supervisor has about the specific parts of the code.

```
holcus <- read.csv("~/Library/Mobile
Documents/com~apple~CloudDocs/Imperial/Teaching/PG/Biological Computing in
R/20:21/2. Tuesday/Practical/holcus.csv")
reference <- c("2/2a", "3a", "12a")

for(a in reference){
```

```
  ifelse(sum(a==holcus$plot)>0, print(paste("The total percentage coverage of
Holcus lanatus in site",a,"is",sum(holcus[holcus$plot==a, 3:16]), "%")),
print(paste("No recording")))
}
```

a. What kind of data structure is `reference`?
b. What does this block of code mean: `sum(holcus[holcus$plot==a, 3:16]`?
c. Explain in plain English what the for loop is doing.

11. We're going to calculate Shannon-Wiener diversity index using a for loop for the year 1999. The formula of the index is as followed if the data is recorded as counts:

$$H' = -\sum\left(\frac{n_i}{N} * ln\frac{n_i}{N}\right)$$

or if the data is recorded as proportions:

$$H' = -\sum p_i ln p_i$$

The code is currently developed is not working so it is up to you to debug it. There are five errors in total.

```
parkgrass_1999<- subset(parkgrass, parkgrass$year=1999)
results<- data.frame(plot=character(), shannon=numeric())
for(site in unique(parkgrass_1999$site)) {
  tempfile <- subset(parkgrass_1999, parkgrass_1999$plot==site)
  prop<- temp_file$comp/100
  shannon <- sum((prop*log(prop)))
  temp<- data.frame(plot=site, shannon=shannon)
  results<- rbind(result, temp)
}
```

Section Two: Writing Code

This section will require you to write snippets of code. Please please use a script file rather than typing your answers straight into the console. You can use the complex

12. Write a for() loop that prints the first four numbers of this sequence: x <- c(7, 4, 3, 8, 9, 25)

13. Write a for loop that iterates over the numbers 1 to 7 and prints the cube of each number using print().

14. Write a portion of code that returns the absolute value of a numeric (HINT: you can choose your own number for the vector).

15. Write a portion of code that calculates the square root of a given numeric (choose any number at random), if the value in x is negative it should return NA.
16. Write a portion of code that returns the amount of values that are larger than the mean of a vector (Hint: You'll need to use the mean() function).

17. In Question 8, you debugged a code that calculated the species richness of each site at the Park Grass Experiment for the year 2000. Using that code as a template, write a portion of code that calculates the species richness for every site for every year.

18. Once you have completed Q13, you'll notice that some sites have 0 recordings for species, for example 2/2a for the year 1991. You'll need to rewrite your answer to Q13 to account for this (HINT: think of how you can integrate an if statement).

19. We can use coding in R build mechanistic models in ecology. One example of this is Levin's model of meta-population dynamics. It mathematically conceptualises a population in which individuals reproduce and die within local patches of the habitat and their offspring disperse into other patches. There are four main assumptions:

    - There are an infinite number of patches, which are all equal in size and quality.
    - Each habitat patch has only two possible states: occupied vs vacant.
    - All habitat patches are equally connected, and spatial distribution is not a factor.
    - Patches produce colonists and extinctions independently and are not affected by other patches.

This has been translated into the differential equation:

$$\frac{dp}{dt} = cp(1-p) - ep$$

Where colonisation rate is *cp(1-p)*, extinction rate is *ep*, patch occupancy (p) is the fraction of patches that are occupied, colonisation (c) is rate in which patches produce colonists and extinction (e) is the probability that a patch goes extinct.

For now, you are required to simulate extinction rate over a range of values (0 to 1) for e and p. Create a for loop that does just this.

Section Three: Making Functions

We can use conditional expressions to create our own functions that will be stored in the RStudio "Global Environment". The basic structure of these are:

1. the function name
2. the function() command
3. Arguments need to be specified in the brackets
4. {} with code inside the braces

Here is a quick function to calculate the proportion of each value in a vector.

```
proportion<- function(vector){
  total<- sum(vector)
  vector/total
}
```

Now we have created this function. We can apply it to any vector we like.

```
proportion(c(1,2,3,4))

## [1] 0.1 0.2 0.3 0.4

proportion(seq(from=1, to=24, by=3))

## [1] 0.01086957 0.04347826 0.07608696 0.10869565 0.14130435 0.17391304
0.20652174
## [8] 0.23913043

proportion(sample(1:100, 5, replace = TRUE))

## [1] 0.24538259 0.26121372 0.23482850 0.05804749 0.20052770
```

Thinking about the construction of functions, write your own functions to answer the following question. Write these in a script file.

20. Create a function that will return the sum of 2 numbers.

21. Create a function that given a vector and an numeric value will return how many times the integer appears inside the vector.

22. Create a function that given a data frame and a vector, will add the vector (if the vector's length matches with the rows number of the data frame) as a new variable to the data frame.

23. Create a function that given a numeric vector, sorts this in ascending order and multiples the result by 2.

24. Create a function for Levin's model of meta-population dynamics.