Conditionals, custom functions, and game theory

Brian Kissmer

USU Department of Biology

Sept. 19th, 2024

Learning objectives

1. Learn how to use conditional statements and custom functions in R
2. Understand the applications of game theory to studies in biology

Today's outline

1. Primer on effective coding
2. Prisoner's dilemma coding project

Coding effectively

I noticed a lot of folks were struggling not just with the code, but with having a good workflow for dealing with code. Follow these steps to improve your ability to work through coding problems

Understand the goal BEFORE starting to code

- If you don't FULLY understand the problem before you start writing code, it will be very difficult to write code to solve that problem. Try writing the problem down and looking for things you don't understand. If you're stuck with something, see whether you can answer more basic questions about the topic. For example:

Why do we use the binomial distribution when simulating genetic drift?

What is the binomial distribution?

What is genetic drift?

Review previous material

- If you don't know how to do something, it is probably located in previous course materials. For example

How do I access the fifth row and third column of a matrix named Y?

Make sure you fully understand what code does

- I give many examples for how to execute certain kinds of code in our notes (sampling a probability distribution, writing for loops, custom functions, etc.). You are free to copy/paste the structure, but if you don't know what it means then you cannot properly modify it to suit your goals. Try figuring out with your group what every part of every line in this code is doing.

```
x <- rep(NA,50)
for(i in 1:50){
    x[i] <- rbinom(n = 1, size = 15, prob = 0.3)
    if (x[i] > 3) {
        print(“X is larger than 3”)
    }
}
```

Play with your code

If you don't understand what something does, mess around with it. Put different values into the arguments and see how it changes. Do this to understand what the output is, and how you can change the arguments to get what you need. For example

If you want to find out what `rbinom()` does, try:

```
rbinom(n = 1, size = 100, prob = 0.3)
rbinom(n = 100, size = 100, prob = 0.3)
rbinom(n =100, size = 100, prob = 0.9)
```

This is an essential part of learning how coding tools work. I can provide examples in class but you should try messing around with the tools we use, and be curious with them!

Plan your code before you start writing it

- ➢ Come up with a game plan. Write down what you need your code to do.
- ➢ Then, think back to your available tools to get it done.
- ➢ Think about what each of those needs (i.e. custom functions)
- ➢ If you struggle with this, reference the previous planning steps

Carefully look at your errors and warnings

If you're stuck on something and keep getting errors, remember that R errors contain information about what is going wrong. Reading these errors carefully can help you understand what needs fixing. For example:

```
> x <- 1:5
> for(i in 1:5){
+    print(x[j])
+ }
Error: object 'j' not found

> rbinom(n = -1,size = 5,prob = .3)
Error in rbinom(n = -1, size = 5, prob = 0.3) : invalid arguments

> rbinom(n = 10,size = ,prob = .3)
Error in rbinom(n = 10, size = , prob = 0.3) :
  argument "size" is missing, with no default
```

Constantly check your results

R lets you look at variables and objects. If you are doing something that has several steps, make sure you look at them after each step to ensure the right thing is happening:

```
> Y <- matrix(NA,nrow = 5, ncol = 5)
> Y
     [,1] [,2] [,3] [,4] [,5]
[1,]   NA   NA   NA   NA   NA
[2,]   NA   NA   NA   NA   NA
[3,]   NA   NA   NA   NA   NA
[4,]   NA   NA   NA   NA   NA
[5,]   NA   NA   NA   NA   NA
> Y[,1] <- 0.1
> Y
     [,1] [,2] [,3] [,4] [,5]
[1,]  0.1   NA   NA   NA   NA
[2,]  0.1   NA   NA   NA   NA
[3,]  0.1   NA   NA   NA   NA
[4,]  0.1   NA   NA   NA   NA
[5,]  0.1   NA   NA   NA   NA
```

```
head(Y)
View(Y)
```

Look things up, you’re on a computer!

- ➢ The internet is a great place to look for help
- ➢ Even professional coders constantly look up solutions to their bugs, errors, etc.
- ➢ Google, StackExchange, even ChatGPT (if you’re careful) will have good results. Either tutorials, or forums where someone had the same issue that you had

In summary, be organized and thorough

- ➢ This takes practice, but is a great tool for learning any topic, not just coding
- ➢ Code needs to be exactly correct, so if you don’t know what you’re doing or why you’re doing it, it’s really easy to get errors.
- ➢ Some code won’t give you errors, but if you don’t know what you should expect you’ll have no idea if your code is working properly

If you still want more help with the basics, use tutorials

- ➢ Remember the tutorials we talked about in Week 1? Now you may have spent enough time with code to see where you're struggling
- ➢ Swirl() package in R, interactive tutorial
- ➢ Learning R textbook, information is in syllabus
- ➢ ChatGPT (again if done correctly)
- ➢ Online tutorials for specific topics

Prisoners dilemma in ecology

See the Programming Project 2 Handout