Scientific Programming Assignment 1

MPhil in Computational Biology

October 12, 2021

If there are errors found, I will update the assignment.

Due date: TBC

Please submit your report to the Moodle website as a single .pdf file. Name your file `spa1_XXX.pdf`, where XXX is your six digit coursework number.

Put a copy of your code (but nothing else) in the appendix of your report. To include R code in latex, see the example code at: `http://www.damtp.cam.ac.uk/user/sje30/teaching/r/rlistings`.

Your report must be a maximum of ten pages, excluding the appendix. This course work will consist of 30% towards your overall mark for this module.

1 Search [5 marks]

Figure 2 shows Joe's pyramid. "Every stone is marked with a different one or two digit positive number. Where a stone rests on two others, its number is the sum of the numbers marked on the two stones on which it rests." (Thanks to the New Scientist for this problem.)

1. How many ways are there of numbering the six blocks on the bottom row? (This requires no code.) [1 mark]
2. Explain restrictions for each of the six numbers on the bottom row. (This requires no code.) [1 mark]
3. Write and run a function that uses these restrictions to search for a solution. What is X? [3 marks]

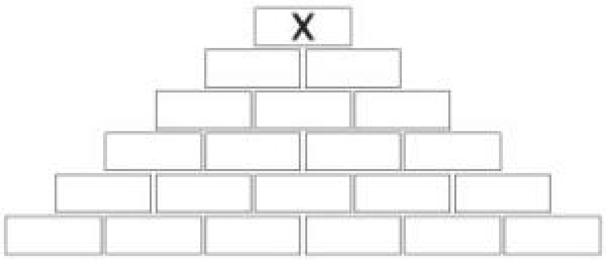
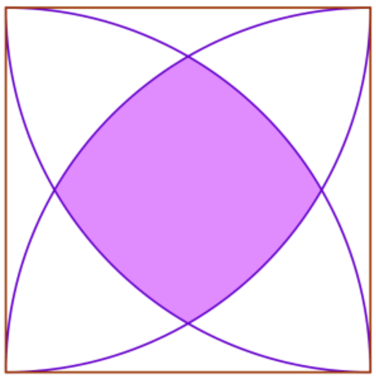


Figure 1: Joe's pyramid. Image taken from the New Scientist.

2 Curved squares [10 marks]

[Source `https://nrich.maths.org/6328`]

A square of side length 1 has an arc of radius 1 drawn from each of its corners, as in the following diagram. The arcs intersect inside the square at four points, to create the shaded region:



(a) Draw the above plot as closely as you can (colours do not need to match) in R using base graphics. [5 marks]

(b) What is the area of the largest square that can be completely contained within the shaded region? Draw the square on top of the plot. [2 marks]

(c) Estimate the area of the shaded region using Monte-Carlo method (e.g. `https://en.wikipedia.org/wiki/Monte_Carlo_method#Overview`). How close is your answer to the analytical solution? [3 marks]

3 A simulation project [15 marks]

(Taken from DMB handbook, section 10).

This section is an optional "capstone" project putting into use the programming skills that have been covered so far. Nothing new about R *per se* is covered in this section.

The exercise is to first write, and then use, a script file that simulates a simple model for density-independent population growth with spatial variation. The model is as follows. The *state variables* are the numbers of individuals in a series of $L = 20$ patches along a line (L stands for "length of the habitat") .

1	2	3	4	...					...	L-1	L

Let $N_j(t)$ denote the number of individuals in patch j ($j = 1, 2, \dots, L$) at time t ($t = 1, 2, 3, \dots$), and let λ_j be the geometric growth rate in patch j. The *dynamic equations* for this model consist of two steps:

1. Geometric population growth within patches:
$$M_j(t) = \lambda_j N_j(t) \quad \text{for all } j. \tag{1}$$

2. Dispersal of some individuals between neighboring patches:
$$N_j(t+1) = (1-2d)M_j(t) + dM_{j-1}(t) + dM_{j+1}(t) \quad \text{for } 2 \le j \le L-1 \tag{2}$$
where $2d$ is the "dispersal rate". We need special rules for the end patches. For this exercise we assume *reflecting boundaries*: those who exit into the void have the sense to come back. That is, there is no leftward dispersal out of patch 1 and no rightward dispersal out of patch L:
$$\begin{aligned} N_1(t+1) &= (1-d)M_1(t) + dM_2(t) \\ N_L(t+1) &= (1-d)M_L(t) + dM_{L-1}(t) \end{aligned} \tag{3}$$

Exercise As a first step, write a function `reflecting` that implements the dispersal phase, equations 2 and . That is, the arguments to the function are a pre-dispersal population vector `Mt` and the dispersal parameter d, and the function returns the population vector after dispersal has taken place.

```
reflecting <- function(Mt,d) {
    ...
    ... your code to compute Nt1, the population vector N at time t+1
    ...
return(Nt1)
}
```

This function takes a vector `Mt` of length L as input, and returns a vector of equal length as output. Equation 2 says that `Nt1[2:(L-1)]` is the sum of 3 terms: individuals that stay in the same patch, individuals that come in from the left (patches 1 to $L-2$), and individuals that come in from the right (patches 3 to L). Think about how you can compute each of these 3 terms without using any loops, and then add them up to compute most values in `Nt1`. After that, you need two more lines to compute the values for patches 1 and L. Note that the code in reflecting has to "figure out" what the value of L is – so what is the R function that tells you the length of a vector?

Exercise Using your `reflecting` function write a script to simulate the model.

• Write your script to start with 5 individuals in each patch at time t=1, iterate the model up to t=100, and graph the log of the total population size (the sum over all patches) over time. Use the following growth rates: $\lambda_j = 0.9$ in the left half of the patches, and $\lambda_j = 1.2$ in the right.

• Write your program so that d and L are parameters, in the sense that the first line of your script file reads

```
d=0.1; L=20;
```

and the program will still work if these are changed to other values.

Note that this model is not totally different from iterating a matrix population model, in that you start with a founding population at time 1 (described by a vector) and use a loop to compute successive populations at times 2,3,4, and so on. So the script can be structured like the scripts for matrix models. After you specify the values of d and L, and create the vector of λ values, you should create a matrix `N` with L rows and 100 columns, so that `N[j,t]` will be $N_j(t)$. The initial population goes into the first column of `N`, and the model is simulated by doing a loop over time (denoted by k in the code below)

```
for(k in 2:200){ # k is the time index
    ... compute Mt from column k-1 of N, and the lambda values
    ... compute Nt1 using the reflecting function
    ... put Nt1 into N, in the right place
}
```

As in the reflecting function, try to vectorize! Vector/matrix operations are much faster than loops. For example, set things up so that in the for-loop, computing the entire `Mt` vector is a one-line statement of the form `a=b*c`.

Exercise Use the model (modified as necessary) to ask how the spatial arrangement of good versus bad habitat patches affects the population growth *rate*. For example, does it matter if all the good sites ($\lambda > 1$) are at one end or in the middle? What if they aren't all in one clump, but are spread out evenly (in some sense) across the entire habitat? **Be a theoretician**: (a) Patterns will be easiest to see if good sites and bad sites are very different from each other. (b) Patterns will be easiest to see if you come up with a nice way to compare growth rates across different spatial arrangements of patches. (c) Don't confound the experiment by also changing the proportion of good versus bad patches, at the same time you're changing the spatial arrangement.