MAS115: SEMESTER 2 MINI-PROJECT SKETCH SOLUTION

TIM HEATON

1. Eating a Mouse

We created a function which took as arguments:

- `N` — the number of times we are going to start our mouse walking around,
- `start` — the starting square.

Our function would then create a vector of length `N` which would store the time until death required for each realisation. The main body of the function consists of a repeat loop which breaks as soon as the current `room` becomes 5 - i.e. the mouse is eaten. Within this loop we have a series of nested `if` statements which determine where the mouse will move next conditional upon the current `room`. Each time the mouse moves we also increment the time `Time`. We then return the mean survival time. Our end code was as follows:

```
1  # Function for N random walks storing time until enter room 5
2  # Arguments:
3  # N - the total number of random walks
4  # start - the room in which the mouse starts
5  # Output:
6  # Time - vector length N with the time until eaten for each iteration
7  RWMouseEat <- function(N, start = 1)
8  {
9    Time <- rep(0, N)
10   for(i in 1:N) {
11     room <- start # The current room
12     repeat {
13       # Exit if in room 5 as mouse is eaten
14       if (room == 5) break
15
16       # Otherwise find where the nouse moves to next
17       Time[i] <- Time[i] + 1 # Increment the time by 1 minute
18       if (room == 1) {
19         room <- sample(c(2,3), 1) # Move to room 2 or 3
20       } else if (room == 2) {
21         room <- sample(c(1,3), 1) # Move to room 1 or 3
22       } else if (room == 3) {
23         room <- sample(c(1,2,4,5), 1) # Move to room 1,2,4 or 5
```

```
24          } else { # Must be in room 4
25            room <- sample(c(3,5), 1) # Move to room 3 or 5
26          }
27        }
28      }
29      return(mean(Time))
30  }
```

We chose 10000 realisations for each starting square to give a reliable estimate of the true value — you need to choose something relatively large since to account for the randomness in any individual realisation. This gave the answers:

Starting Room	1	2	3	4
Simulated mean moves until death	8.03	8.03	5.96	4.02

1.1. **Theory - Eating the mouse.** By following through the example of working out the conditional expectation you should come up with a system of simultaneous equations which can be written in the following matrix form:

$$\begin{pmatrix} 1 & -0.5 & -0.5 & 0 \\ -0.5 & 1 & -0.5 & 0 \\ -0.25 & -0.25 & 1 & -0.25 \\ 0 & 0 & -0.5 & 1 \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

We can solve these equations using the following code in **R**:

```
1 # Create matrix
2 P <- matrix(c(1,-0.5,-0.5,0,-0.5,1,-0.5,0), nrow = 2, byrow = TRUE)
3 P <- rbind(P, c(-0.25, -0.25, 1, -0.25)) # Add row 3
4 P <- rbind(P, c(0, 0, -0.5, 1)) # Add row 4
5
6 solve(P, rep(1,4))
```

This gives the exact expected number of rolls (which we should compare to our simulated versions) of

Starting Room	1	2	3	4
Expected number of moves	8.00	8.00	6.00	4.00

2. Probability of Escape

In order to create code for the second part of the project (the probability of the mouse escaping into the garden) I called the garden room 6. I then modified my above code so that the mouse could move around the house as before but also permitting it to move into room 6 (i.e. the garden). I then stopped my random walk as soon as the mouse either entered room 5 (escaped) or room 6 (eaten). Dependent upon which of these happened I either said that `Escape` was `TRUE` (if he entered the garden) of `FALSE` (if he entered room 5). Again I created a lot of random walks and counted the proportion of times that the mouse was eaten. My listing is below:

```
########## Now for the probability of escape #########
RWMouseEscape <- function(N, start = 1)
{
  Escape <- rep(NA, N) # Initialise with NAs so know if its working
  for(i in 1:N) {
    room <- start # Start the mouse off
    repeat {
      # Create break conditions if outside or in room 5
      if (room == 5) {
        Escape[i] <- FALSE # Mouse is eaten - No escape
        break
      } else if (room == 6) {
        Escape[i] <- TRUE # Mouse has escaped
        break
      }

      # Otherwise find where the mouse moves to next
      if (room == 1) {
        room <- sample(c(2,3), 1) # Move to room 2 or 3
      } else if (room == 2) {
        room <- sample(c(1,3), 1) # Move to room 1 or 3
      } else if (room == 3) {
        room <- sample(c(1,2,4,5), 1) # Move to room 1, 2, 4 or 5
      } else { # Must be in room 4
        room <- sample(c(3,5,6), 1) # Move to room 3,5 or 6 (outside)
      }
    }
  }
  return(mean(Escape))
}
```

Again I chose 10000 realisations for each starting room to give a reliable estimate of the true value. This gave the answers:

Starting Room	1	2	3	4
Simulated probability of escape	0.20	0.20	0.21	0.41

2.1. **Theory - The mouse escaping.** You need to use the same idea as before conditioning on the room the mouse will move to next but with equation for conditional probability rather than expectation (also adding a room 6 or outside). For example if we let p_i be the probability of escape given the mouse is currently in room i then we find, for a mouse currently in room 1 and conditioning on where they might move next, that

$$p_1 = \frac{1}{2}p_2 + \frac{1}{2}p_2.$$

Continuing this through for the other rooms we get e.g.

$$p_4 = \frac{1}{3}p_3 + \frac{1}{3}p_5 + \frac{1}{3}p_6$$

but we now that $p_5 = 0$ (as he will be eaten) and $p_6 = 1$ (as this is when he's escaped). Doing this for all the rooms you should come up with a system of simultaneous equations which can be written in the following matrix form:

$$\begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ -\frac{1}{4} & \frac{1}{4} & 1 & \frac{1}{4} \\ 0 & 0 & -\frac{1}{3} & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{3} \end{pmatrix}.$$

We can solve these equations using the following code in **R**:

```
# Create probability matrix
P <- matrix( c(1,-0.5,-0.5,0,-0.5,1,-0.5,0), nrow = 2, byrow = TRUE)
P <- rbind(P, c(-0.25, -0.25, 1, -0.25)) # Add row 3
P <- rbind(P, c(0, 0, -1/3,  1)) # Add row 4

solve(P, c(0,0,0,1/3))
```

This gives the exact expected number of rolls (which we should compare to our simulated versions) of

Starting Room	1	2	3	4
Expected number of moves	0.2	0.2	0.2	0.2

3. Other solution approaches

There are other, equally valid, ways of writing code which solves the problem so you will need to check yourself that the code of the projects you are marking does what it is supposed to do. This is good practice since it you will learn how to read other people's code and hopefully realise how important good comments are.

One alternative approach for example gets rid of the if statements by creating a `list` describing the floor plan. To move around the house you then jut sample from the relevant column in the `list`. Below is how this code might work for the probability of escape but hopefully you can see how to modify it for a solution to the expected time until death for the first part too.

```
simEscape<-function(start,ngoes) {
 fate<-rep(NA,ngoes)
 # specify neighbours of rooms 1,2,3 and 4
 u<-c(2,3)
 v<-c(1,3)
 w<-c(1,2,4,5) # 5 is cat's room
 x<-c(3,5,6) # 6 indicates escape through exit
 neighbourList<-list(u,v,w,x)
 for(j in 1:ngoes){
  t<-0
  currentRoom<-start
  Z <- -1
  while(Z==-1){
    newNeighbours<-neighbourList[[currentRoom]]
    currentRoom<-sample(newNeighbours,1)
    t <- t+1
    if (currentRoom==5) {Z <- 0} else{
      if (currentRoom==6) {Z <- 1}
    }
  }
  fate[j]<-Z
 }
 return(sum(fate)/ngoes)
}
```

4. Marking

You should give all of the projects you are given a mark out of 4. It is up to yo to decide what mark you think is justified but I would perhaps suggest a mark scheme along the lines of:

- 0 — a poor attempt with neither the simulation or theory parts really correct.
- 1 — a reasonable attempt but quit a lot of stuff wrong. I would give this mark to someone who had e.g. got the theory part correct but had not even attempted the simulation sections or had got them totally wrong.
- 2 — a fairly good report. I would give this mark for example to someone who had got all the theory parts correct and had maybe got one of the two computer simulations but not the other.
- 3 — a very good report (i.e. a good first). I would expect to give this mark if all of the theory and coding is correct but perhaps the presentation isn't quite as good as it could be.
- 4 — an excellent report. Not only is everything correct but it is also well written, presented and explained.