Scientific Programming Assignment 2

MPhil in Computational Biology

November 3, 2018

If there are errors found, I will update the assignment on the web at `http://github.com/sje30/rpc2018`

Due date: 2018-11-22 23:45

Please submit your report to Moodle as a Rnw: NO OTHER FORMATS ARE ACCEPTED.

Name your file `spa2_XXX.Rnw`, where XXX is your USN. The files will be marked anonymously so please refrain from writing your name (or CRSid) in the document.

You must ensure that the file can be compiled into a PDF by someone else using:

```
require(knitr); knit2pdf('spa2_XXX.Rnw')
```

Your .Rnw file should dynamically compute and report answers, rather than you writing the code and then typing your answers manually into the latex part of the document. In particular, it should run on subliminal, using only the system packages installed. If in doubt about whether you can use a given package, ask me in advance. (You may use .Rmd, rather than .Rnw as long as you indicate clearly how to convert it to a pdf.)

Your report must be a maximum of ten pages, excluding the appendix. This course work will consist of 25% towards your overall mark for this module.

1 Skyscraper [10 marks]

`https://www.brainbashers.com/skyscrapers.asp`. Your job is to solve the skyscraper puzzle.

1. Write the function "skyplot" that draws the problems as shown above. Use your function to draw the two grids shown in Figure 1. It should have the following structure:

```
skyplot <- function(edges, soln, ...) {
  ## EDGES is a vector of length 4N containing the numbers clockwise
  ## around the grid, starting in the top left corner.  SOLN is a
  ## vector of length N^2, storing on a row-by-row basis, starting in
  ## top-left corner.  Any elements containing 0 are meant to indiciate blank.
  ## N could be between 4 and 9.
  ...
}
```

2. Show all the solutions to the problems in Figure 2. Show your answers using the skyplot() function to draw sky plots. [5 marks]

Hints: to solve this problem, the search space is small enough that you can exhaustively search all possible solutions. For drawing primitives, look at the functions: *segments()*, *text()*. You may use the following function.

```
par(mfrow=c(1,2),mar=c(2,2,1.5,1))
skyplot( 1:20, 1:25, main='A')
skyplot( c(2,0,0,0,2,  0,0,0,2,0,  0,0,3,4,0,  0,1,0,2,0),
        c(0,0,0,0,0,  0,0,0,0,0,  0,0,0,0,5,  5,0,0,0,4, 0,0,0,0,0),
        main='B')
```

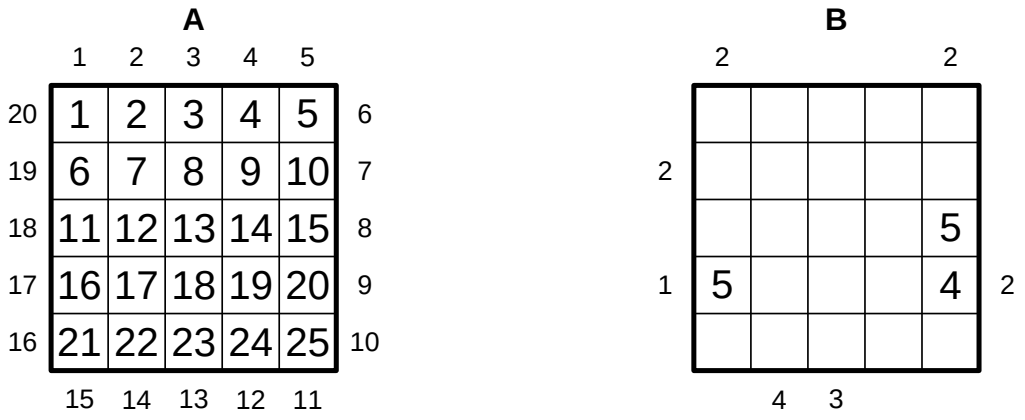

Figure 1: Layout of the skyscraper problem. Panel A shows the indexing used. Panel B shows a typical 5x5 problem.

```
par(mfrow=c(1,2),mar=c(2,2,1.5,1))
e1 = scan(comment.char="#", quiet=T,
          'https://raw.githubusercontent.com/sje30/rpc2018/master/a2/e1.dat')
skyplot(e1, rep(0,16), main='A')
e2 = c(3,0,1,0, 0,0,1,0, 0,0,0,0, 0,0,0,2)
skyplot(e2, rep(0,16), main='B')
```

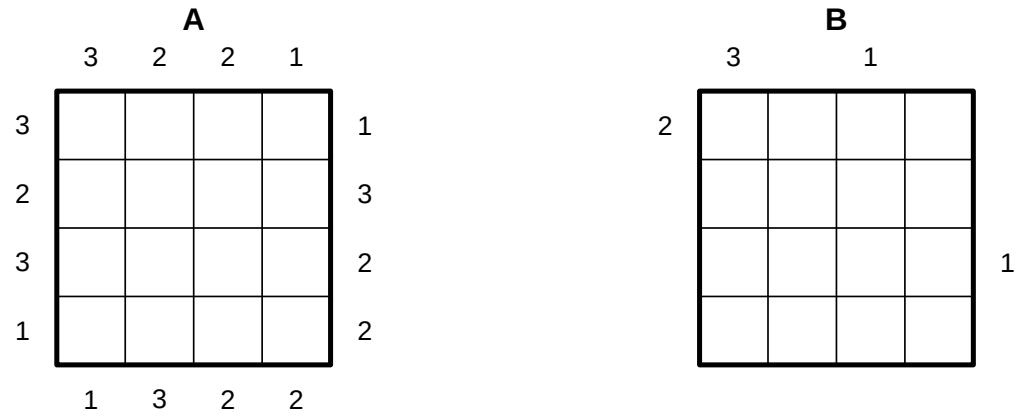

Figure 2: Solve these problems. Ensure that you read in e1 as above, as I **may** change this file when compiling your pdf. (If it changes, you can assume it will be a valid 4x4 problem.)

```
## Taken from: http://stackoverflow.com/questions/11095992
permutations <- function(n){
    if(n==1){
        return(matrix(1))
    } else {
```

```
        sp <- permutations(n-1)
        p <- nrow(sp)
        A <- matrix(nrow=n*p,ncol=n)
        for(i in 1:n){
            A[(i-1)*p+1:p,] <- cbind(i,sp+(sp>=i))
        }
        return(A)
    }
}
```

2 Roulette [15 marks]

Complete question 22.2 (playing Roulette) from (Jones, Maillardet & Robinson, 2009), answering the questions that they set. (The question is appended at the end of this assignment.) In particular, your knitr document must dynamically generate the tables requested in the question.

(You should assume that there is only one zero on the roulette wheel; i.e. there is no double zero.)

22.2 Roulette

At the Crown Casino in Melbourne, Australia, some roulette wheels have 18 slots coloured red, 18 slots coloured black, and 1 slot (numbered 0) coloured green. The red and black slots are also numbered from 1 to 36. (Note that some of the roulette wheels also have a double zero, also coloured green, which nearly doubles the house percentage.)

You can play various 'games' or 'systems' in roulette. Four possible games are:

- A. Betting on Red

  This game involves just one bet. You bet \$1 on red. If the ball lands on red you win \$1, otherwise you lose.

- B. Betting on a Number

  This game involves just one bet. You bet \$1 on a particular number, say 17; if the ball lands on that number you win \$35, otherwise you lose.

- C. Martingale System

  In this game you start by betting \$1 on red. If you lose, you double your previous bet; if you win, you bet \$1 again. You continue to play until you have won \$10, or the bet exceeds \$100.

- D. Labouchere System

  In this game you start with the list of numbers (1, 2, 3, 4). You bet the sum of the first and last numbers on red (initially \$5). If you win you delete the first and last numbers from the list (so if you win your first bet it becomes (2,3)), otherwise you add the sum to the end of your list (so if you lose your first bet it becomes (1, 2, 3, 4, 5)). You repeat this process until your list is empty, or the bet exceeds \$100. If only one number is left on the list, you bet that number.

Different games offer different playing experiences, for example some allow you to win more often than you lose, some let you play longer, some cost more to play, and some risk greater losses. The aim of this assignment is to compare the four games above using the following criteria:

1. The expected winnings per game;
2. The proportion of games you win;
3. The expected playing time per game, measured by the number of bets made;
4. The maximum amount you can lose;
5. The maximum amount you can win.

22.2.1 Simulation

For each game write a function (with no inputs) that plays the game once and returns a vector of length two consisting of the amount won/lost and how many bets were made. Then write a program that estimates 1, 2, and 3, by simulating 100,000 repetitions of each game. Note that a game is won if you make money and lost if you lose money.

22.2.2 Verification

For games A and B, check your estimates for 1 and 2 by calculating the exact answers. What is the percentage error in your estimates for 100,000 repetitions?

For each game, work out the exact answers for 4 and 5. Of course, if this is not close to the answer given by your simulation, then you should suspect that either your calculation or your program is erroneous.

22.2.3 Variation

Repeat the simulation experiment of Part 22.2.1 five times. Report the minimum and maximum values for 1, 2, and 3 in a table as follows:

Game	Exp. winnings min–max	Prop. wins min–max	Exp. play time min–max
A			
B			
C			
D			

Modify your program from Part 22.2.1 so that in addition to estimating the expected winnings, expected proportion of wins, and expected playing time, it also estimates the *standard deviation* of each of these values. (You may use the built-in function `sd(x)` to do this.) For a single run, consisting of 100,000 repetitions of each game, report your results in a table as follows:

Game	Winnings mean, std dev	Prop. wins mean, std dev	Play time mean, std dev
A			
B			
C			
D			

For which game is the amount won most variable?

For which game is the expected playing time most variable?