STOR-609 Assessment 1 - Marks and Feedback

March 3, 2025

Name : Sophie Brimble

1 Quality of Source Code

You have provided some very high quality code for this assessment. In particular -

- Your source code is well structured and organised.
- You have made very good use of appropriate representations, data structures and programming idioms provided by base R and other publicly available packages.

Some possible areas for improvement are

- It would be a good idea to add some comments to your code at key points (but don't over do this as good code mostly documents itself). For example, you could give a quick description of what each main data structure represents.
- Make all of your exit points from functions explicit by specifying **return**

Mark (out of 5) : 4.5

2 Solution

- For both problems you have provided example code that produces a correct solution.
- Your code is designed so that it was easy for me to change the problem and produce further correct solutions (i.e. your code is both **re-usable** and **re-runnable**)

Mark (out of 5) : 5

3 Understanding Design Principles

- You have provided an implementation for a generic backtracking algorithm which is immediately recognisable as such when compared with the pseudo code provided.

- You have made good use of appropriate data structures and design patterns (e.g. partial application)
- Your code is flexible, modular, and the major design components are independent and orthogonal. This supports well the quality of **re-usability**.
- You have also considered the computational complexity of your code (i.e. how much extra "computation" is required as your problem size increases).

Notes

1. I noticed that in your generic backtrack algorithm you have

```
if(length(output) != 0) {
      return(output)
    }
```

This precludes the possibility of certain edge cases that might. For example, what would happen if an empty vector was the result of the algorithm ?

2. You discuss the use of global variables and correctly suggest that an improvement would be to eliminate the need for these. This is why the generic backtracking algorithm includes a state P. You could use this (perhaps in the form of a class pattern) to carry around the information you need to modify and keep track of.

Mark (out of 5) : 4.5

4 Quality of Written Communication

- You have provided some excellent supporting information. It is very clear and understandable, and discusses your main design ideas and the representations you used.
- You also provide some useful reflections on your use of global variables and the issues regarding recursion and stack limitations in R.

Suggestion

- It would be really good if you also provided pseudo code for each of your **accept**, **reject**, **first**, and **next** methods.

Mark (out of 5) : 4.5

5 Overall Comments and Marks

- This is a very high standard piece of work. You quite clearly have a good understanding of the R programming system and its main programming idioms and data structures.

- You might like to try experimenting with how you can translate your skills to a different programming system. Python is an obvious candidate but maybe have a look at Julia and - if you really want to push the boat out - try F sharp.

Notes

- The recursion limit is not related to the "power" of the computer. It is a consequence of the design of the R programming system. This has some major implications in the sense that many useful algorithms are best expressed recursively. Of course, any recursive algorithm can be expressed using iteration (and vice versa).

- Do you remember looking at bitwise operations in the **Introductory Python Course** ? How might they be useful in the context of this assignment ?

Overall Mark (out of 20) : 18.5