Scientific Programming Assignment 1

MPhil in Computational Biology

October 9, 2023

If there are errors found, I will update the assignment on the web at
`http://github.com/sje30/rpc2016`

Due date: 2023-10-24 23:45

Please submit your report to the Moodle website as a single .pdf file. Name your file `spa1_XXXXXX.pdf`, where XXXXXX is your unique 6-digit ID.

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 25% towards your overall mark for this module.

1 Shakespeare [5 marks]

Download the data file `https://www.gutenberg.org/files/100/old/shaks12.txt`

(a) Read the file into R on a line by line basis. (readLines)

(b) Break each line into words, where words are simply any characters separated by the space character (strsplit).

(c) Remove any punctuation from words using the following hint:

```
> gsub("[[:punct:]]", '', c("MOTH.", "Thrice-worthy", "gentleman!"))
[1] "MOTH"         "Thriceworthy" "gentleman"
```

This means for example that the words "he'll" and "lords!-why" from the file will become the words "hell" and "lordswhy" respectively.

(d) Convert everything to lower case.

(e) What are the five most common words in the file, and how often does each of them occur? (Hint: use `table()`).

(f) What are the six longest words in this file? (Hint: you should find 3 of the 6 longest words come from the preamble of the file, not from Shakespeare.)

Show your code and the output that it generates, keeping the output concise.

2 Examination marking [10 points]

The data for this exercise is in `grading` folder.

Twelve students have sat a multiple-choice exam. The exam had 100 questions, and each answer was one of a,b,c,d,e. The file `crib.dat` stores the correct answer for each question (in order). The students had to answer 30 questions from the 100. Your job is to write a script that will mark each student's performance, and produce a data.frame which stores the results:

```
> results <- data.frame(student=1:num.students, score=correct,
                        grade=alpha.grades, rank=rank)
> print(results)
   student score grade rank
1        1    19     B  6.0
2        2    xx     x  xxx
3        3    xx     x  xxx
4        4    xx     x  xxx
5        5    xx     x  xxx
6        6    xx     x  xxx
7        7    xx     x  xxx
8        8    xx     x  xxx
9        9    xx     x  xxx
10      10    xx     x  xxx
11      11    xx     x  xxx
12      12    xx     x  xxx
```

To help you get started, you can see that student 1 got 19/30 correct, their rank was 6/12 (1st rank for highest) and their grade was B. Grades are determined using the datafile `grade.txt`; convert the score into a percentage, take the floor() to convert percentage to an integer, then find which grade band the score falls in.

Hints: use read.table(, header=TRUE) to read in a student file. The name of the datafile to read in should be generated using paste(). You can use scan() to read in the crib.dat file.

The invigilator of the exam suspects that a student was cheating, but cannot recall which student it was. Write a program that will automatically check whether a pair of students have similar results; what do you conclude?

3 Cryptarithms [10 marks]

[See `http://www.logicville.com/cryptarithm.htm` for background reading. Note that leading digits in each number will not be zero.]

(a) Write a function that solves the following maths problem: [5 marks]

```
 A B
 * C
 ---
 D E
+F G
 ---
 H I
```

where the symbols A–I correspond to distinct digits, 0-9. Show all solutions.

(b) Write a function that takes one argument, a string, and returns all corresponding solutions. Demonstrate it working on the following three cases: [5 marks]

- `"send + more = money"`
- `"snow + rain = sleet"`
- **(Advanced:)** `"one + two + two + three + three = eleven"`

To solve this problem you might find the following function useful.

```
## ---- permutations
## 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)
    }
}
```

For the advanced section, you might find the following tip useful:

```
eval(parse(text="rnorm(3)"))

## [1]  1.6300217 -0.1902119  1.4706708
```