Examination Advanced R Programming

Linköpings Universitet, IDA, Statistik

Course code and name:	732A94 Advanced R Programming
Date:	2025/12/04, 8–12
Teacher:	Krzysztof Bartoszek
Allowed aids:	The extra material is included in the zip file **exam_help_material_732A94.zip**
Grades:	A= $[18-20]$ points
	B= $[16-18)$ points
	C= $[14-16)$ points
	D= $[12-14)$ points
	E= $[10-12)$ points
	F= $[0-10)$ points
Instructions:	Write your answers in R scripts named according to the pattern **[your exam account]_*.R** The R code should be complete and readable code, possible to run by calling `source()` directly on your `*.R` files. You may have a separate file for each question, or answer everything in one file. Comment directly in the code whenever something needs to be explained or discussed. Follow the instructions carefully. There are **THREE** problems (with sub–questions) to solve.

Problem 1 (2p)

Provide examples of functions/problems of different types of computational complexity—with their computational complexity. What complexity is acceptable from a practical point of view, and what is not acceptable?

Problem 2 (12p)

READ THE WHOLE QUESTION BEFORE STARTING TO IMPLEMENT! Remember that your functions should **ALWAYS** check for correctness of user input! For each subquestion please provide **EXAMPLE CALLS!**
We say that a matrix of real numbers $\mathbb{R}^{n\times m} \ni \mathbf{A} = [a_{ij}]_{\substack{1\le i\le n \\ 1\le j\le m}}$ is a *multiplication table* if its entries satisfy:

$$a_{ij} = \frac{a_{i,\cdot}a_{\cdot,j}}{a_{\cdot,\cdot}},$$

where

$$\begin{aligned} a_{i1} + a_{i2} + \ldots + a_{im} &= a_{i,\cdot}, \\ a_{1j} + a_{2j} + \ldots + a_{nj} &= a_{\cdot,j}, \\ \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij} &= a_{\cdot,\cdot}. \end{aligned}$$

You are a data scientist that is interested in pairs of variables that form multiplication tables. You want to write an object oriented (S3, S4, or RC) program that will help you keep track of your table collection.
a) (3p) In this task you should use object oriented programming in S3, S4 or RC to write code that keeps track of your table collection. You should record for each table information on it: a short description, its dimensionality and whether it is a multiplication table. If the table is not a multiplication table you should record (this will be implemented in **c)** but plan out the object structure now and prepare an appropriate implementation) how it deviates from being a multiplication table. Depending on your chosen OO system you can do it through a constructor or by implementing a function `create_table_collection()`. The constructing function should not take any arguments. The object should contain the number of tables present and for each table its description, dimensionality, the table itself, a flag if it is a multiplication table and if not how it deviates from one, and a unique identifier.

```
## example call to create a table object
my_tables <- create_table_collection() # S3
my_tables <- table_collection$new() # RC
```

b) (2p) Now implement a function called `add_table()` that allows one to add a new table to the collection. The function should take two parameters: the name of the text file containing the table and short description of it. The function should read in the table from the file and store it according to the description. You should not yet calculate the deviations, this will be done in **c)**. Below are example calls to three tables provided with the exam.

```
## S3 and RC example calls
my_tables <-add_table(my_tables,"multable_ex1.txt","a random multiplication table")
## file multable_ex1.txt contains a multiplication table
my_tables <-add_table(my_tables,"approxmulttable_ex2.txt","world records in running")
## file approxmulttable_ex2.txt is not a multiplication table,
## but is actually well approximated by multable_ex1.txt
my_tables <-add_table(my_tables,"notmulttable_ex3.txt","University of Delaware students")
## file notmulttable_ex3.txt is not a multiplication table
## if using RC you may also call in this way
my_tables$("multable_ex1.txt","a random multiplication table")
my_tables$("approxmulttable_ex2.txt","world records in running")
my_tables$("notmulttable_ex3.txt","University of Delaware students")
```

c) (6p) Now implement a function called `deviation_from_multable()` that for an indicated table (if it is not a multiplication table) calculates how far away each entry is from being a multiplication table entry value. The function should react appropriately if the indicated table is already a multiplication table. The table of deviations should be stored in the table object. We recall that a table is a multiplication table if its values satisfy

$$a_{ij} = \frac{a_{i,\cdot}a_{\cdot,j}}{a_{\cdot,\cdot}}.$$

Therefore, the deviation of the value from being a multiplicative value can be calculated as

$$d(a_{ij}, \frac{a_{i,\cdot}a_{\cdot,j}}{a_{\cdot,\cdot}}),$$

for some chosen distance function $d(\cdot,\cdot)$. A function $d(\cdot,\cdot)$ is a distance function if it satisfies

1. $d(x,y) = 0$ if and only if $x = y$,
2. $d(x,y) = d(y,x)$, symmetricity,
3. $d(x,y) \le d(x,z) + d(z,y)$, triangle inequality.

The function should take two parameters—the unique identifier of the indicated table and a function to calculate the distance between two numbers. You may **not** use the example distance function.

```
f_dist<-function(x,y){(abs(x^{3}-y^{3}))^{1/3}}
## S3 and RC example call
my_tables<-deviation_from_multable(my_tables,id_tab2,f_dist)
##id_tab2 is the id of approxmulttable_ex2.txt
## if using RC you may also call in this way
my_tables$deviation_from_multable(id_tabs2,f_dist)
```

d) (1p) Implement a function that displays the deviations from multiplication tables. You are free to choose yourself how to report these! This function has to also work directly with `print()`.

```
# calls to present deviations
my_tables; print(my_tables)
```

Problem 3 (6p)

a) (3p) Implement a function that checks if a provided natural number is a Kaprekar (Dattatreya Ramchandra Kaprekar—Indian mathematician) number. A Kaprekar number is one such that its square can be split into two blocks of digits, such that the numbers formed by these digits, when summed result in the Kaprekar number. Take for example $297^2 = 88209$, then $88 + 209 = 297$. Importantly, 0 can be dropped if needed, e.g., $4879^2 = 23804641$, and $238 + 4641 = 4879$ (i.e., the 0 in front of 4641 is dropped, as it would be a leading 0). Mathematically we can write that if N is a Kaprekar number, then there exist natural numbers k, Q, R such that $N^2 = Q \cdot 10^k + R$ so that $N = Q + R$. The function should take as its input one parameter, n.
TIP: You can consider using the functions `as.numeric()`, `unlist()` and `strsplit()`.
b) (3p) The following are the first five Kaprekar numbers

1	9	$9^2 = 81$	$8 + 1 = 9$
2	45	$45^2 = 2025$	$20 + 25 = 45$
3	297	$297^2 = 88209$	$88 + 209 = 297$
4	4879	$4879^2 = 23804641$	$238 + 4641 = 4879$
5	538461	$538461^2 = 289940248521$	$289940 + 248521 = 538461$

Implement unit tests, using the above information. You should implement a test where a a Kaprekar number is found, when it is not and one where your function throws an error.