Examination Advanced R Programming

Linköpings Universitet, IDA, Statistik

Course code and name:	732A94 Advanced R Programming
Date:	2025/10/20, 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)

What are the important aspects to take into consideration when designing a good database? Discuss.

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 $\mathbb{R}^{n\times m} \ni \mathbf{A} = [a_{ij}]_{\substack{1\leq i\leq n\\ 1\leq j\leq m}}$ 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) which multiplication table it is closest to and what is the distance from it. Use the below distance function between two tables **(of same dimensionality and both non–zero)**, $\mathbf{A}$ and $\mathbf{B}$,

$$d(\mathbf{A},\mathbf{B}) = \left(\sum_{i=1}^{n}\sum_{j=1}^{m}(a_{ij}-b_{ij})^2\right) / \left(\sum_{i=1}^{n}\sum_{j=1}^{m}((|a_{ij}|+|b_{ij}|))^2\right).$$

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 information on the closest available multiplication table to it with the distance, and a unique identifier.

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

b) (4p) 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 search for the closest multiplication table, 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) (4p) Now implement a function called `find_approximate()` that for an indicated table finds the closest multiplication table to it, and also calculates the distance to it. The function should react appropriately if the indicated table is already a multiplication table, and also if there is no appropriate multiplication table in the collection (e.g., there is no table of appropriate dimensionality). The function should take only one parameter—the unique identifier of the indicated table.

```
## S3 and RC example call
my_tables<-find_approximate(my_tables,id_tab2)
##id_tab2 is the id of approxmulttable_ex2.txt
## if using RC you may also call in this way
my_tables$find_approximate(id_tabs2)
```

In the above example we would have the table from `multable_ex1.txt` as the closest one.

d) (1p) Implement a function that displays the state of your collection of tables. You are free to choose yourself how to report the state! This function has to also work directly with `print()`.

```
# calls to show state of the collection of tables
my_tables; print(my_tables)
```

Problem 3 (6p)

a) (3p) Implement a function that calculates

$$\sum_{j=0}^{n} \binom{n}{j}^2,$$

where $\binom{n}{j}$ is the binomial coefficient equalling

$$\binom{n}{j} = \frac{n!}{j!(n-j)!}, \text{ where } n! = \prod_{i=1}^{n} i, \text{ and } 0! \equiv 1.$$

The function should take as its input one parameter, n.
b) (1p) What is the computational complexity of your implementation?
c) (2p) It is known that

$$\sum_{j=0}^{n} \binom{n}{j}^2 = \binom{2n}{n}$$

Implement unit tests, using `R`'s `choose()` function. You should implement a test where in the comparison a correct answer is expected, an incorrect answer is expected and an error is expected.