Examination Advanced R Programming

Linköpings Universitet, IDA, Statistik

Course code and name:	732A94 Advanced R Programming
Date:	2019/11/28, 8–12
Teacher:	Krzysztof Bartoszek
Allowed aids:	The extra material is included in the zip file **exam_material.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 an R script file named **[your exam account].R** The R code should be complete and readable code, possible to run by copying directly into a script. 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 (5p)

a) (2p) In the lecture a number of so–called normal forms for databases were discussed. There is an underlying thread to all of them, i.e. the main reason for database normalization. What is this reason? What implications does a denormalized database have its maintainers?
b) (3p) You were provided with an `.RData` file that contains two (related) data frames. The first one is meant to record all the courses a student took and contains the variables (columns) `liuid`, `surname`, `name`, `course_code` and `course_name`. The second data frame is meant to record the grades of the students for the courses and contains the variables (columns) `liuid`, `course_code`, `course_name` and `grade`. What problems can you imagine when maintaining this data structure, e.g. when adding new students or a course name changes? Think of other possibilities. How can you improve the data structure?

Problem 2 (10p)

READ THE WHOLE QUESTION BEFORE STARTING TO IMPLEMENT! Remember that your functions should **ALWAYS** check for correctness of user input!

a) (2p) In this task you should use object oriented programming in S3 or RC to write code that simulates a futuristic wardrobe. The wardrobe is to contain a data structure on its contents and also automatically provide clothes appropriate for the given weather.

Your goal is to first construct an initial wardrobe. The `create_wardrobe()` function should take one argument, `max_number_clothes`, the maximum number of clothing items it can store. The wardrobe object should contain for each clothing item information describing it, in particular a unique id of the item, a text name (you may have the same one e.g. `"a"` for every item) and under what weather conditions it may be worn. You may assume that weather conditions are described by numbers or single letter or whatever else is easiest for you. Provide some example calls to your code.

```
## S3 and RC call to future_wardrobe() function
future_wardrobe <- create_wardrobe(max_number_clothes=5)
```

b) (3p) Now implement a function called `add_to_wardrobe()` that allows one to add a new clothing item to the wardrobe. The function should have two parameters: the name of the item and its weather type. The id is to be automatically generated. Do not forget that the wardrobe has a maximum capacity! Provide some example calls to your code.

```
## S3 and RC call
future_wardrobe <-add_to_wardrobe(future_wardrobe,"raincoat","rain")

## if using RC you may also call in this way
future_wardrobe$add_to_wardrobe("raincoat","rain")
```

c) (4p) Now implement a function called `obtain_clothes()` that allows a user to obtain clothing items for the current weather type. The function should take two parameters the weather type and the desired number of clothing items. You decide how to react when there are not enough clothing items for the given weather type. Do not forget that the clothing items have to leave the wardrobe! Inside the `obtain_clothes()` function implement informing the user what clothes were provided. Provide some example calls to your code.

```
## S3 and RC call
future_wardrobe <-obtain_clothes(future_wardrobe,"rain",5)

## if using RC you may also call in this way
future_wardrobe$obtain_clothes("rain",5)
```

d) (1p) Implement a plot **OR (NO NEED TO DO BOTH!)** print function that informs about the contents of the wardrobe. You are free to choose yourself how to report the content!

```
# Plotting and printing calls
plot(future_wardrobe); print(future_wardrobe)
```

Problem 3 (5p)

a) (2p) In statistics it is often important to generate a matrix that has all its values equal to a constant value. Implement a function that takes as its input a value and the dimension of the matrix. The function is to create a square matrix (with the provided dimension) that has every entry equal to the provided one. You may only use `for` loops to generate the matrix. In particular you may not use `matrix()` nor similar functions. Do not forget that your function should check for correctness of input and react appropriately.
b) (1p) What is the complexity of your solution in terms of the matrix's dimension.
c) (2p) Of course the same can be achieved using `R`'s inbuilt `matrix(value, dimension, dimension)`. Implement a unit test that compares your implementation with direct `matrix` creation. Furthermore, write tests that check how your function behaves when something goes wrong, e.g. there is a problem with the input.