Examination Advanced R Programming

Linköpings Universitet, IDA, Statistik

Course code and name:	732A94 Advanced R Programming
Date:	2024/12/02, 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)

Discuss the pros and cons of using vector based calculations in `R`, i.e., the `apply` family of functions.

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!**
You want to help your grandparents with preparation of jams. Jams will be made from fruit either in a jam making machine (which is faster but can take less fruit—it makes jam from up to 2kg of fruit in 2hours) or a pot (takes more fruit but also more time, up to 10kg of fruit in 6hours). You want to write an object oriented (S3, S4, or RC) program that will keep track of the amount of available fruit, made jams and suggest how to divide fruit between the jam making machines and pots.

a) (3p) In this task you should use object oriented programming in S3, S4 or RC to write code that keeps track of the amount of available fruit, made jams and suggests how to divide fruit between the jam making machines and pots. You should record how many kilograms of each fruit your grandparents have, how many kilograms of each jam they have made, and how many pots and jam making machines they have. Depending on your chosen OO system you can do it through a constructor or by implementing a function `create_pantry()`. The constructing function should take two arguments—the number of available jam making machines and pots. The object should contain for each fruit type how many kilograms of it are available and how many kilograms of jam were made from it.

```
## example call to create a stocks object
my_pantry <- create_pantry(machines=2,pots=4) # S3
my_pantry <- pantry$new(machines=2,pots=4) # RC
```

b) (4p) Now implement a function called `add_fruit()` that allows your grandparents to add new fruit. The function should have three parameters describing it: name, amount (in kilograms), and how many kilograms of jam can be made from 1kg of the fruit.

```
## S3 and RC example calls
my_pantry<-add_fruit(my_pantry,"strawberry",10,0.5)
my_pantry<-add_fruit(my_pantry,"black_cherry",15,0.75)
my_pantry<-add_fruit(my_pantry,"apple",10,0.6)
## if using RC you may also call in this way
my_pantry$add_fruit("strawberry",10,0.5)
my_pantry$add_fruit("black_cherry",15,0.75)
my_pantry$add_fruit("apple",10,0.6)
```

c) (4p) Now implement a function called `make_jam()` that is called when your grandparents want to make some jam. It should take as its input the fruit and desired amount of jam. Check if this is possible and react appropriately. Then, in the output propose how many jam making machines and pots are needed (notice that it can happen that some need to be washed and reused—describe this in the output) and how long will it take to make the jam. Do not forget to update the pantry—the amount of fruit and jams.

```
## S3 and RC example call
my_pantry<-make_jam(my_pantry,"apple",3)
## if using RC you may also call in this way
my_pantry$make_jam("apple",3)
```

d) (1p) Implement a function that displays the state of your grandparent's pantry. 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 pantry
my_pantry; print(my_pantry)
```

Problem 3 (6p)

a) (2p) Each pixel of astronomical data can take brightness values between 0 and 100000 (for each of the RGB channels). This resolution is however far too large for presenting on user's PC, where RGB channels take values between 0 and 255. Your task is to implement a function that takes a vector of numbers (which will correspond to a colour channel), and rescales it linearly so that 0 corresponds to the minimum value (say it is a) in it and 255 to the maximum value (say it is b) in it, i.e., an original value $x_{highres}$ is changed according to

$$x_{lowres} = \frac{255}{b-a}x_{highres} + (255 - \frac{255b}{b-a}).$$

You should implement the above formula manually.
b) (2p) What is the computational complexity of your solution? Is it better to use loops or vector based calculations here? Would these have different complexities?
c) (2p) Implement a unit test, using the **scales**`::rescale()` function.