Examination Advanced R Programming

Linköpings Universitet, IDA, Statistik

Course code and name:	732A94 Advanced R Programming
Date:	2020/12/02, 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
	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.
	If you also need to provide some hand–written derivations please number each page according to the pattern: Question number . page in question number i.e. Q1.1, Q1.2, Q1.3,. . ., Q2.1, Q2.2, . . ., Q3.1, ...
	Scan/take photos of such derivations preferably into a single pdf file but if this is not possible multiple pdf or .bmp/.jpg/.png files are fine.
	Please do not use other formats for scanned/photographed solutions.
	Please submit all your solutions via LISAM or e–mail.
	If emailing, please email them to **BOTH** krzysztof.bartoszek@liu.se and KB_LiU_exam@protonmail.ch .
	During the exam you may ask the examiner questions by emailing them to KB_LiU_exam@protonmail.ch **ONLY**. Other exam procedures in LISAM.

Problem 1 (5p)

a) (2p) Describe briefly `R`'s coercion mechanism for different variable types.
b) (3p) What will be the results of running `1+"a"`, `1+TRUE` and `1+1L`. Provide and explain the output. What is `R` doing?

Problem 2 (10p)

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

a) (4p) In this task you should use object oriented programming in S3 or RC to write code that simulates a robot lawn mower that mows a rectangular lawn. The lawn mower object has to contain information on its position, the map of the lawn, the current status of the lawn and a mechanism to move the robot to unmowed squares (without bumping into obstacles). The lawn is a rectangle of size $n \times m$ squares. The lawn mower is able to mow a whole square at one go. Each square is defined by a pair of integers (i, j), where $1 \leq i \leq n$ and $1 \leq j \leq n$. At some squares obstacles, e.g. tree or rock are present.

Your goal is to first construct the robot with a map of the lawn. Initially all squares are not mowed. Depending on your chosen OO system you can do it through a constructor (RC) or by implementing a function `create_robot_lawnmower()` (S3). The constructing function should take two arguments—the size of the lawn (a vector of two integers) and the initial location of the robot on the lawn (a pair of integers). The robot object should contain information on each square on the lawn, whether it is mowed or not and whether there is an obstacle on it or not, how many unmowed squares there are, how many obstacles there are and the current location of the robot. You may initially choose which squares have obstacles in any way you like (random, deterministic, your choice). However at least $nm/5$ have to have obstacles and at least $nm/5$ have to be free of obstacles. The robot cannot be placed initially nor enter on a square with an obstacle. Provide some example calls to your code.

```
## example call to create a robot_lawnmower object
robot_lawnmower <- create_robot_lawnmower(nm=c(20,10),current_location=c(1,1)) ## when S3
my_robot_lawnmower <- robot_lawnmower$new(nm=c(20,10),current_location=c(1,1)) ## when RC
```

b) (4p) Now implement a function called `go_mow_squares()`. The function should not take any parameters. The function is to find the closest, in terms of moves of the robot, unmowed square. The robot can move only left, right, up and down. The location has to be updated after each move. After reaching the designated square, the robot mows it, the status of the square has to be changed to mowed (do not forget about the counter of the not mowed squares). Then, the robot has to continue mowing in the same direction it came from until it reaches an obstacle or the edge of the lawn. If when the function is called, all squares are mowed, then an appropriate message has to be provided to the user. Provide some example calls to your code.

```
## S3 and RC call
robot_lawnmower <-go_mow_squares(robot_lawnmower)

## if using RC you may also call in this way
robot_lawnmower$go_clean_tile()
```

c) (2p) Implement a function that displays the state of the lawn. You are free to choose yourself how to report the state! This function has to also work with `print()`.

```
# calls to show state of garden
robot_lawnmower; print(robot_lawnmower)
```

Problem 3 (5p)

a) (2p) Linear regression, where a response depends linearly on a set of predictors, is a key tool in Statistics. Implement a function that takes as its input the model matrix (matrix of predictors' measurements), $\mathbf{X}$, and the response vector $\vec{y}$ and returns the vector of estimates of the regression coefficients. These coefficients have to be calculated directly from the least squares formula

$$\hat{\vec{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\vec{y}.$$

Do not forget to check for correctness of input, including dimensions.

b) (1p) What is the computational complexity of your implementation in terms of the dimensions of $\mathbf{X}$? Assume that for an $m \times m$ matrix inverting it takes $O(m^3)$ time.

c) (2p) Implement a unit test that compares your implementation with direct `R` estimation, i.e. if `X` is your model matrix and `y` is your response vector, then you may extract the regression coefficients from what `lm(y~X)` returns. Due to numerics you *might* have to include some tolerance when comparing for equality.