# Examination Advanced R Programming

Linköpings Universitet, IDA, Statistik

| | |
|---|---|
| Course code and name: | 732A94 Advanced R Programming |
| Date: | 2023/10/23, 8–12 |
| Teacher: | Bayu Beta Brahmantio |
| Allowed aids: | The extra material is included in the zip file **exam_help_material_732A94.zip** |
| Grades: | A= $[18-20]$ points<br>B= $[16-18)$ points<br>C= $[14-16)$ points<br>D= $[12-14)$ points<br>E= $[10-12)$ points<br>F= $[0-10)$ points |
| Instructions: | Write your answers in R scripts named according to the pattern **[your exam account]_*.R**<br>The R code should be complete and readable code, possible to run by calling `source()` directly on your `*.R` files. Comment directly in the code whenever something needs to be explained or discussed. Follow the instructions carefully.<br>There are **THREE** problems (with sub–questions) to solve. |

## Problem 1 (4p)

**a) (2p)** Very often a problem can have many algorithms solving it. What should you take into account when choosing the algorithm? Discuss different stages of implementing your solution, and how does the choice of algorithm correspond to each stage.
**b) (2p)** Discuss various ways of improving the speed of `R` code, and how does each one relate to the choice of algorithm.

# 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 are a IX century Viking merchant who through unknown means obtained a computer with `R` installed on it. After teaching yourself S3, S4 and RC programming you realized that you could write an object oriented program that will help you keep track of your sales and purchases.

**a) (3p)** In this task you should use object oriented programming in S3, S4 or RC to write code that keeps track of your sales and purchases. You sail between the two main Baltic ports Birka, and Truso. At each place you buy and sell various products like amber or furs. You should record information for each product where (one of the two ports) you bought it, how much of it, for how much, and how much of it you have in stock. Then, you should also record how much of each product you sold where (one of the two ports) and for how much. Depending on your chosen OO system you can do it through a constructor or by implementing a function `create_stocks()`. The constructing function should not take any arguments. The object should contain for each product the information describing it, in particular a unique id of it, what it is (e.g., amber, fur), its amount (a number), place it was bought in (Truso, Birka), its cost (a number), and how much is left in stock. Furthermore, you should have a database on sales—unique id of sale, what was sold, where (Truso, Birka), in what amount and where.

```
## example call to create a stocks object
my_stock <- create_stocks() # S3
my_stock <- stocks$new() # RC
```

**b) (4p)** Now implement a function called `purchase_goods()` that allows one to add a new file to the archive. The function should have four parameters describing the purchased stock: place of purchase (only Truso, and Birka are allowed), type of goods, its price and amount.

```
## S3 and RC example calls
my_stock <-purchase_goods(my_stock,"Truso","amber",10,100)
## if using RC you may also call in this way
my_stock$purchase_goods("Truso","amber",10,100)
```

**c) (4p)** Now implement a function called `sell_goods()` that is called when you sell something. The function should take four parameters concerning the sale: place (only Truso, and Birka are allowed), product, price and amount. If you do not have enough product in stock, your code should react accordingly. Remember that selling stock reduces it.

```
## S3 and RC example call
my_stock<-sell_goods(my_stock,"Birka","amber",5,200)
## if using RC you may also call in this way
my_stock$sell_goods("Birka","amber",5,200)
```

**d) (1p)** Implement a function that displays the state of your stocks. 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 stock
my_stock; print(my_stock)
```

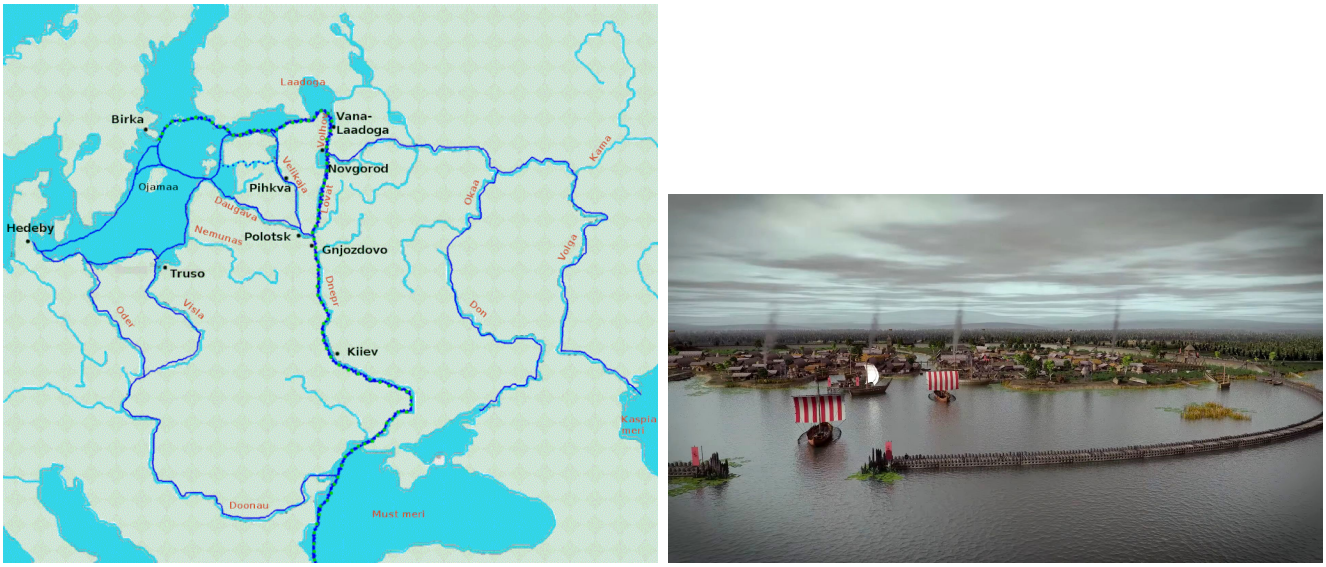


Figure 1: Left: Viking trade routes, Wikimedia Commons, by Minnekon, CC BY-SA 4.0; right: Visualization of the port of Truso, Wikimedia Commons, by the Museum of Archaeology and History in Elbląg, CC BY-SA 3.0.

# Problem 3 (4p)

**a) (2p)** Please implement a function that solves a least square problem, $\vec{y} = \mathbf{X}\vec{\beta}$ through the standard generalized least squares matrix equations,

$$\hat{\beta} = (\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{V}^{-1}\vec{y},$$

that takes as its input the matrices $\mathbf{X}$, $\mathbf{V}$ and the vector $\vec{y}$. Check for correctness of input, in particular do not invert a matrix if it is singular. Please provide **EXAMPLE CALLS** to your function.

**b) (1p)** Assuming that matrix inversion takes $O(n^3)$ time, where $n$ is the length of $\vec{y}$, what is the computational complexity of obtaining $\vec{\beta}$? Assume that the dimension of $\beta$, $k$, is constant (does not depend on $n$), and small with respect to $n$ (in particular $n^{-1}k \to 0$ with $n \to \infty$).

**c) (1p)** Implement a unit test, using that for $k = n$, and $\mathbf{X}$ invertible, we have $\hat{\beta} = \mathbf{X}^{-1}\vec{y}$.