R Scripting

Exercises for unit 3 - Structured programming

Marcus Wurzer

Please solve the following problems!

1. Coursebook exercises:

   a. Use vapply() to:

      - Compute the standard deviation of every column in a numeric data frame (Hint: Type `help(package = "datasets")` and choose a suitable data set).
      - Compute the standard deviation of every numeric column in a mixed data frame. (Hint: you'll need to use `vapply()` twice.)

   b. What does `replicate()` do? What sort of for loop does it eliminate? Why do its arguments differ from `lapply()` and friends?

   c. What's the relationship between `which()` and `Position()`?

2. Simulate some sales figures data of a shop using the following lines of code:

```r
set.seed(1)
sf <- matrix(round(rnorm(144, 500, 200)), ncol = 12)
rownames(sf) <- 2011:2022
colnames(sf) <- abbreviate(month.name, 3, named = FALSE)
```

   a. Use a functional to compute the average monthly sales figures over the whole period!

   b. Now generate a data frame to have the data represented in a different way:

```r
sf.df <- data.frame(sales = c(t(sf)),
                    month = factor(colnames(sf), levels = colnames(sf)))
```

   Apply another functional on `sf.df` to get the same average monthly sales figures as above!

3. The Fibonacci sequence is a famous sequence in mathematics. The first two elements are defined as [1, 1]. Subsequent elements are defined as the sum of the preceding two elements. For example, the third element is 2 (= 1+1), the fourth element is 3 (= 1+2), the fifth element is 5 (= 2+3), and so on.

   To obtain the first 12 Fibonacci numbers in R, we can use

```r
Fibonacci <- numeric(12)
Fibonacci[1] <- Fibonacci[2] <- 1
for (i in 3:12) Fibonacci[i] <- Fibonacci[i - 2] + Fibonacci[i - 1]
```

   Modify the code to generate the Fibonacci sequence in the following ways:

   a. Change the first two elements to 2 and 2.
   b. Change the first two elements to 3 and 2.

c. Change the update rule from summing successive elements to taking differences of successive elements. For example, the third element is defined as the second element minus the first element, and so on.
d. Change the update rule so that each element is defined as the sum of three preceding elements. Set the third element as `1` in order to start the process.

4. A digitized black and white image can be represented by a binary matrix, with "0" symbolizing color value "white" and "1" color value "black".

   a. Create an 800x600 matrix and randomly fill it with black and white values.
   b. Compute the grey value (= mean color value) for each row and each column of the matrix, respectively, using a loop.
   c. Compute the grey value (= mean color value) for each row and each column of the matrix, respectively, using a functional.

5. Data set `mtcars` is included in R:

```r
data(mtcars)
?mtcars
```

   a. Coerce variable `vs` to a factor with meaningful labels, then use a functional to return a summary of variable `disp` (displacement) - separately for cars with V-shaped and straight engines and given in ccm (cubic centimeters) instead of cubic inches (conversion formula: ccm = cu.in / 0.061024).
   b. Coerce variable `am` to a factor with meaningful labels, too, then use a functional to return the mean fuel consumption (variable `mpg`) - separately for all four combinations of motor and engine types and given in liters/km instead of miles per gallon (conversion formula: liters/km = 378.5411784 / (1.609344 * mpg)).

6. Generate a 31x10 matrix called `temps`, containing temperature measurements (in °C) for the 31 daily average high temperatures of July in ten cities. The criteria are:

   - In each city, the temperatures should follow a normal distribution with a randomly generated mean temperature (`?rnorm`) that lies between 20 and 30 degrees and a standard deviation of 5.
   - The row names should be created using the following scheme: July 1, . . . , July 2, . . . , July 3, . . .
   - The column names should be created using the following scheme: City A, . . . , City B, . . . , City C, . . .

   Unfortunately, many of the entries have been lost when reading the data. . . :

```r
temps[sample(1:length(temps), length(temps) / 10)] <- NA
```

   a. Coerce the matrix to a data frame. Then, use a functional to coerce this data frame to some suitable data structure that only contains the non-missing values for each city and the respective names (= dates). (Note: Use the functional that retains the names, i.e., it is not necessary to make use of the `names()` function!)
   b. Once this new data structure has been created, operate over it to identify the days that showed the highest July temperature for each city. These days should be presented using the simplest possible data structure.