Statistical Computing: R - Homework 2023

Koen Plevoets

17-10-2023

Solve this R Homework in an RStudio Project containing an RMarkdown source file and its HTML report. Both files should contain your answers to the questions in separate (sub)sections. There is no need to repeat the questions, just give your answers in code chunks. Note that certain questions ask for output which is part of your HTML report.

For grading your solutions, one should essentially be able to render your report (e.g. by pressing the *Knit* button) in order to see all the results that the questions ask for. That is why it is important that your R code:

- can be sourced without any errors
- only contains the answers to the questions. In other words, code for checking results or exploring certain objects should not be part of your final solutions (just as for any regular exam)
- is reproducible, i.e. it could be used for data with the same structure but different values
- applies elegant coding style

In case you do not feel familiar enough with RMarkdown, you can write your solutions in an R script, but one point will be subtracted. Of course, the same four principles just mentioned also apply to an R script.

Finally, give both your RStudio Project and your RMarkdown file a name structured as `RHW2023_`*First name_Last Name*, in which you replace *First name* by your (principal) first name and *Last name* by your surname. Zip your RStudio Project and upload this on Ufora (in the assignment *R HW 2023* under the menu *Ufora-tools > Assignments*).

Questions

1.

R contains various built-in (high-dimensional) arrays such as e.g. `HairEyeColor` on the association between hair color and eye color for different sexes or `Titanic` with frequencies on the passengers of the *Titanic*. Use these to create two matrices:

- `HairEye` containing the frequencies of hair color X eye color
- `ClassSurvived` containing the frequencies of class X the survived variable

In other words, your matrices should look as follows (if you cannot solve this question, then you can create both matrices manually in order to solve all subsequent questions):

```
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    68   20    15     5
##   Brown   119   84    54    29
##   Red      26   17    14    14
##   Blond     7   94    10    16
```

```
##       Survived
## Class   No Yes
##   1st  122 203
##   2nd  167 118
##   3rd  528 178
##   Crew 673 212
```

2.

Reorder the cells of `HairEye` so both the row sums and columns sums appear in descending order. This can be done in a single command (and there is no need to overwrite the object).

3.

Rank the frequencies in `HairEye` so the lowest value (i.e 5) gets rank 1, the second lowest value (i.e. 7) gets rank 2, etc. Your result should look (exactly) as follows:

```
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    13    9   6.0   1.0
##   Brown    16   14  12.0  11.0
##   Red      10    8   4.5   4.5
##   Blond     2   15   3.0   7.0
```

4.

The three largest values in `HairEye` can be obtained with the single command `c(HairEye["Brown", "Brown"], HairEye["Blond", "Blue"], HairEye["Brown", "Blue"])`. Give another single command, using character indexes, which obtains the same result.

5.

A matrix like `HairEye` can be rearranged into a data frame in long format by first converting the matrix to another class and then converting that object into a data frame. If you include this data frame as an RMarkdown table in your report, then the result looks as follows:

Hair	Eye	Freq
Black	Brown	68
Brown	Brown	119
Red	Brown	26
Blond	Brown	7
Black	Blue	20
Brown	Blue	84

Hair	Eye	Freq
Red	Blue	17
Blond	Blue	94
Black	Hazel	15
Brown	Hazel	54
Red	Hazel	14
Blond	Hazel	10
Black	Green	5
Brown	Green	29
Red	Green	14
Blond	Green	16

Reproduce this result in your RMarkdown report (i.e. include the data frame as an RMarkdown table).

6.

For each cell in a matrix like `HairEye` a 2x2 matrix can be created where the (1, 1) cell contains the value of the (1, 1) cell of `HairEye` and where the other cells contains the *sums* of the values not in the same column and/or row. For instance, for the (1, 1) cell (i.e. 68) of `HairEye` the 2x2 matrix is:

```
##      [,1] [,2]
## [1,]   68   40
## [2,]  152  332
```

Here, the (1, 2) cell contains the sum of the values in the first row of `HairEye` but *not* in the first column (i.e. 20, 15, 5). The (2, 1) cell contains the values in the first column of `HairEye` but *not* the first row (i.e. 119, 26, 7) and the (2, 2) cell contains the sum of (*all*) the remaining cells. Similarly, the 2x2 matrix of the (2, 1) cell in `HairEye` is:

```
##      [,1] [,2]
## [1,]  119  167
## [2,]  101  205
```

Write some code which returns the 2x2 matrices for all (16) cells of `HairEye` in a list of 16 components, called `all2x2`.

7.

The previous exercise can be solved with `for` loops or meta-functions etc. Now give another solution to obtain the same result. Time both computations in microseconds using the **microbenchmark** package. Note that you can wrap a whole block of R code in `{ }` and pass that R expression as the first argument to the `microbenchmark()` function. Add the following sentence to your RMarkdown report in which you replace the `...` by the necessary R code (for which you can use the summary function in **microbenchmark**):

The average time of the first computation is . . . microseconds and the average time of the second computation is . . . microseconds.

8.

The *odds ratio* in a 2x2 table is computed as:

$$\frac{N_{1,1} * N_{2,2}}{N_{2,1} * N_{1,2}}$$

where $N_{i,j}$ is the frequency in the cell on row i and column j of the table. As a consequence, odds ratios can be computed for any IxJ matrix based on their "partial 2x2 matrices", which you computed in the previous two exercises. Write a function `GlobalOddsRatio()` which first checks whether its input argument is a matrix (and raises an error when it isn't) and then computes this IxJ matrix. In other words, the `GlobalOddsRatio()` function should work as follows:

```
GlobalOddsRatio(1:12)
```

```
## Error in GlobalOddsRatio(1:12): is.matrix(x) is not TRUE
```

```
GlobalOddsRatio(HairEye)
```

```
##        Eye
## Hair         Brown      Blue     Hazel     Green
##   Black 3.71315789 0.3368298 0.8395368 0.3496791
##   Brown 1.44631529 0.5555136 1.5935013 0.8737076
##   Red   0.97388316 0.5135615 1.3741950 2.3136842
##   Blond 0.06901408 8.0981718 0.3933683 1.2522523
```

Note: If you do not find the solution for the whole IxJ matrix, then give a solution for a simple 2x2 matrix: that will earn you 1 point.

9.

The *log odds ratio* is:

$$log\left(\frac{N_{1,1} * N_{2,2}}{N_{2,1} * N_{1,2}}\right) = log\left(N_{1,1}\right) + log\left(N_{2,2}\right) - log\left(N_{2,1}\right) - log\left(N_{1,2}\right)$$

Write a function `LogOddsRatio()` which computes the log odds ratios for an IxJ matrix such as returned in the previous exercises. Since it makes use of the logarithm, it should have two additional arguments:

- `base` specifying the base of the logarithm: its default value should be Euler's number e
- `adjust` specifying the constant to be added to the cells with zero counts (and *only* those cells) before taking the logarithm: its default value should be a half

Note: Again, if you do not find the solution for the whole IxJ matrix, then give a solution for a simple 2x2 matrix: that will earn you 1 point.

10.

Create an S3 class system `TwoWay` in which the functions `GlobalOddsRatio()` and `LogOddsRatio()` are embedded as the methods `GOR()` and `LOR()`, respectively. An instance of the class `TwoWay` should be created with the function `CreateTwoWay()` which does the following:

- It checks whether its input argument is a matrix (and raises an error if it is not)
- It returns the input with class `TwoWay`

Apply this class and methods to the matrix `ClassSurvived` and reproduce these results in your report:

```
Titanic2Way <- CreateTwoWay(ClassSurvived)
class(Titanic2Way)
```

```
## [1] "TwoWay"
```

```
GOR(Titanic2Way)
```

```
##       Survived
## Class         No       Yes
##   1st  0.2231729 4.4808313
##   2nd  0.6343505 1.5764155
##   3rd  1.6434862 0.6084627
##   Crew 1.9389102 0.5157536
```

```
LOR(Titanic2Way)
```

```
##       Survived
## Class          No        Yes
##   1st  -1.4998086  1.4998086
##   2nd  -0.4551536  0.4551536
##   3rd   0.4968197 -0.4968197
##   Crew  0.6621261 -0.6621261
```