Statistical Computing: R - Homework 2024

Koen Plevoets

16-10-2024

This homework for Statistical Computing R involves some attachments which you find on Ufora (of the course page *C004077A - Statistical Computing*). Go either to the menu *Ufora-tools > Assignments* and open the assignment *R HW 2024*, or go to the menu *Content > R > Homework*. The attachments are the data file *mobility.csv* and/or *mobility.rda* (in case you cannot solve Question 1). Download them in a folder on your computer which you can easily access and solve the homework.

How to solve the assignment

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 Question 2 also asks 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 `RHW2024_`*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 2024* under the menu *Ufora-tools > Assignments*).

Data

You will work with the file *mobility.csv* containing an historical data set on the occupational mobility between fathers and sons in the UK, analyzed by Karl Pearson in 1904. It cross-tabulates 14 occupational categories of the fathers and sons in the rows and columns, respectively:

- `Arm` or `ARM`: Army

- `Art` or `ART`: Art
- `Tcc` or `TCC`: Teaching, clerical work and civil service
- `Cra` or `CRA`: Crafts
- `Div` or `DIV`: Divinity
- `Agr` or `AGR`: Agriculture
- `Lan` or `LAN`: Landownership
- `Law` or `LAW`: Law
- `Lit` or `LIT`: Literature
- `Com` or `COM`: Commerce
- `Med` or `MED`: Medicine
- `Nav` or `NAV`: Navy
- `Pol` or `POL`: Politics and court
- `Sch` or `SCH`: Scholarship and sciences

Each data entry is a frequency count.

Questions

1.

Read in the data file *mobility.csv* using R code. Since all functions for reading data in R create a data frame (or tibble or data.table or…), convert the data subsequently to a matrix, called `mobility`.

2.

The `mobility` matrix contains many 0 entries, which is why it is actually a *sparse matrix*. Sparse matrices can be efficiently stored with the R package **Matrix**. Use its function `Matrix(., sparse = TRUE)` to convert `mobility` to a sparse matrix, called `sparse_mobility`. Compare the (surprising) byte sizes of both matrices by including the following sentence in (the main text of) your RMarkdown report, in which you replace the '...' by an appropriate R computation:

The size of the *dense* matrix `mobility` is ... bytes and the size of the *sparse* matrix `sparse_mobility` is ... bytes.

Note: If you did not manage to solve Question 1, then you can load the file *mobility.rda* using the command `load("mobility.rda")`. This contains the `mobility` matrix with which you can solve this question.

3.

Write a function `sparseProp(x)` which returns the proportion (i.e. relative frequency) of 0's in an R object. This function should equally work on an ordinary vector as on a matrix (or any higher-dimensional array, for that matter), e.g.:

```r
y <- c(1, 0, 2, 8, 6, 0)
sparseProp(y)
```

```
## [1] 0.3333333
```

```r
sparseProp(mobility)
```

```
## [1] 0.2653061
```

4.

The three largest values in `mobility` are 54 (in the cell `Div`, `DIV`), 51 (in the cell `Art`, `ART`) and 28 (in the cell `Arm`, `ARM`). Use an *index matrix* containing only *character values* to extract these three values from `mobility`.

5.

R's base function `table()` tabulates the frequency of each value in a vector:

```r
table(y)
```

```
## y
## 0 1 2 6 8
## 2 1 1 1 1
```

By specifying the argument `dnn = NULL` you omit the name of the input object:

```r
table(y, dnn = NULL)
```

```
## 0 1 2 6 8
## 2 1 1 1 1
```

Write a function `freqOfFreq(x)` which tabulates the frequencies of the values in a matrix. If the input object `x` is not a matrix, however, then the function should raise an error. You can make use of base R's function `stopifnot()` to raise this error.

6.

Sparse frequency tables, such as `mobility`, are suitably analyzed by what the statistician I.J. Good calls their "frequency of frequencies" distribution. This involves tabulating the frequency of each frequency count, i.e. precisely what the function `freqOfFreq()` gives us. Write two functions `meanFOF()` and `varFOF()` which both take the output of `freqOfFreq()` as their input and which respectively compute the mean and variance on the basis of it. In other words, the functions `meanFOF()` and `varFOF()` should work as follows:

```r
meanFOF(freqOfFreq(mobility))
```

```
## [1] 3.954082
```

```r
varFOF(freqOfFreq(mobility))
```

```
## [1] 46.22865
```

Compare:

```r
mean(as.vector(mobility))
```

```
## [1] 3.954082
```

```r
var(as.vector(mobility))
```

```
## [1] 46.22865
```

Note: If you did not manage to solve Question 6, then just provide the *skeleton* of the `meanFOF()` and `varFOF()` functions. Supply the necessary comments to clarify your reasoning.

7.

If we wish to perform a *chi-squared test* for independence between the rows and columns of a frequency table, then we (first) need to compute the *expected frequency* for each cell in the table. The expected frequency of cell (i, j) in the table is denoted as E_{ij} and is computed on the basis of the *row totals* and *column totals* of the table according to the following formula:

$$E_{ij} = \frac{O_{i+} * O_{+j}}{O_{++}}$$

where:

- O_{i+} is the *row total* (i.e. total frequency) of row i of the table
- O_{+j} is the *column total* (i.e. total frequency) of column j of the table
- O_{++} is the *grand total* (i.e. total sum) of the table

Write a function `expected()` which:

- takes a matrix as its argument `x` and raises an error when it is not a matrix
- computes the expected frequencies *using only some of base R's meta-functions* (i.e. no `for` loops or other packages are required)
- returns the expected frequencies in a matrix of the same number of rows and columns as in the input matrix `x`

8.

The *Pearson residuals* for a chi-squared test, denoted as R_{ij}, are computed as follows:

$$R_{ij} = \frac{O_{ij} - E_{ij}}{\sqrt{E_{ij}}}$$

where O_{ij} is the *observed* frequency on row i and column j of the frequency table/matrix.

Write a function `pearson()` which:

- takes a matrix as its argument `x` and raises an error if `x` is not a matrix
- raises this error *efficiently*: it does not execute a call if it is executed by another function
- computes the Pearson residuals in a vectorized way and returns them in a matrix with the same number of rows and columns as `x`

Note: If you did not manage to solve Question 7, then you can generate the expected values (E_{ij}) with the base R command `chisq.test(mobility, correct = FALSE)$expected`. This will return a matrix with which you can start solving this question.

9.

Pearson's X^2 statistic is a single numerical value which is computed on the basis of the Pearson residuals as follows:

$$X^2 = \sum_{i=1}^{I} \sum_{j=1}^{J} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

where I and J are the total number of rows and columns, respectively, of the frequency table.

Write a function `chisquared()` which:

- takes a matrix as its argument `x` and raises an error when `x` is not a matrix (again, efficiently)
- returns the X^2 value as an instance of the S3 class `chisquared`

Note: If you did not manage to solve Question 8, then you can generate the Pearson residuals (R_{ij}) with the base R command `chisq.test(mobility, correct = FALSE)$residuals`. This will return a matrix with which you can start solving this question.

Since the `mobility` table is sparse, the chi-squared approximation may be poor so we will not pursue computing any p-value. Good luck!