DS4B: Exercises

A. Fidalgo

Last update: *March 26, 20:22*

The following are a question sets to help master the concepts that we discussed.

3 Exercises on sub-setting

3.1 Sub-setting with indices

For this question, use the built-in vector `letters` to create the following vectors.

- a = (a),
- b = (a, b, c),
- c = (a, d, e, f, ...), all but (b, c),
- d = (a, b, c, ..., w), all but the last three,
- c = (a, d, g, j, ...), every third letter.

3.2 Sub-setting with conditions

The function `runif` creates random numbers from a uniform (0,1) distribution. The function `floor` gives the highest integer of a number.

i. Set a seed for your random numbers (`setseed`).
ii. Create a vector 'data' of 50 random numbers, uniformly distributed between 80 and 120.
iii. Take a subsample of 'data' for which the values are below 90. How many are there?
iv. Take a subsample of 'data' for which the values are below 90 or above 105.

v. Take a subsample of 'data' for which the values are even.
vi. Take a subsample of 'data' for which the values are above the sample average.
vii. What are the indexes of the even elements of 'data' (hint: use `which`)?

3.3 Sub-setting data frames

i. Load the built-in data set `iris` and inspect it.
ii. Calculate the maximum value of `Sepal.Length`.
iii. Create a data frame with every 5th observation of the original data set.
iv. Create a data frame with only the sepal length and width.
v. Take a sample of 10 observations from iris (hint: use `sample`).
vi. Create a data frame with the observations with the largest and the smallest petal width.
vii. With the data frame with the observations with the largest petal width, create a data frame with the smallest sepal width.

2 Exercises on simple vectors creation

The following exercises are intended to illustrate some basic functions in R, in particular when applied to the most common data structure, namely the vector.

Even if we do not aim at the most elegant answers, we should aim at efficient ways. Brute-force solutions are strictly forbidden.

2.1 Creation of simple vectors

Create the following simple vectors.

- $a = (1, 2, 3, ..., 15)$,
- $b = (15, 14, 13, ..., 0, -1, -2)$,
- $c = (15, 14, 13, ..., 10, 9, -9, -10, ..., -13, -14, -15)$

2.2 Empty vector

Create an empty vector with 30 numeric values. Check its type and its length.
(Hint: use `numeric()`.)

2.3 Character vector

Create the vector `participants` with the names of the participants in the class.

2.4 Simple manipulations

Consider the following scores at a test.

$$(60, 84, 65, 67, 75, 72, 80, 85, 63, 82, 70, 75)$$

Calculate the mean of the sample.
Recall the formula for the z-score of an observation:

$$z_i = \frac{x_i - \mu}{\sigma}$$

Create a vector with the z-score of each grade.

2.5 Simple calculations

Calculate the following expressions, where n is the number of elements in the series.

$$\frac{1}{n}\sum_{i=5}^{55}(i^2 - 3i)$$

$$\frac{1}{n}\sum_{i=1}^{10}\left(\frac{3^i}{3i} + \sqrt{3i}\right)$$

2.6 Create vectors with `rep`

For the following questions, you may want to check the help on the function `rep`.
Start by creating the vector

- dna = (A, C, G, T).

Then, create the following vectors:

- a = (A, C, G, T, A, C, G, T, A, C, G, T, ..., A, C, G, T) where there are 12 A's as shown.
- b = (A, C, G, T, A, C, G, T, A, C, G, T, ..., A, C) where there are 15 A's as shown; the last element of the vector being C.
- c = (A, ..., A, C, ..., C, G, ..., G, T, ..., T) where there are 12 A's, 15 C's, 30 G's and 4 T's, in the order shown.

2.7 Create vectors with `seq`

For the following questions, you may want to check the help on the function `seq`.
Create the following vectors:

- $a = (1, 2, 3, ..., 15)$,
- $b = (15, 14, 13, ..., 0, -1, -2)$,
- $c = (1, 1.2, 1.4, ..., 8)$.

Create the following vectors, potentially based on auxiliary vectors.

- $d = (0.5^3 1.3^1, 0.5^6 1.3^4, ..., 0.5^{36} 1.3^{34})$,
- $e = (5, \frac{5^2}{2}, \frac{5^3}{3}, ..., \frac{5^{25}}{25})$.

Use the function `all.equal` to check that the last elements of each vector are indeed 1.0889154×10^{-7} and 1.1920929×10^{16}, respectively.

2.8 Use `paste` in vector creation

For the following questions, you may want to check the help on the function `paste`.
Create the following vectors:

- a = (City 1, City 2, ..., City 6),
- b = (n1, n2, ..., n6)
- b = (Gender=M, Gender=F, Gender=M, Gender=F, Gender=M, Gender=F)

1 Exercises on `.Rmd` files

1.1 Global chunk options

We can use options, e.g., `echo`, to control the output and the format of the R chunks.
Suppose you want to use the same options for all the R chunks. It would be tedious to write it n times!
How would you set that option once for the whole document? (Hint: not a YAML issue.)
How would you make an exception for one specific R chunk?

1.2 Yet another output format for the same `.Rmd` file

The YAML of this `.Rmd` file contains the following code.

```
output:
  prettydoc::html_pretty:
    theme: architect
    highlight: github
  pdf_document:
    highlight: tango
```

```
    number_sections: no
    toc: no
```

If you `Knit` the document as such, it will almost certainly produce an error. Why?
Make sure you can `Knit` this `.Rmd` file with **all** the output options.

1.3 Dynamic document 1.0

There is a dynamic element in the title part of this document.
Guess how it works and modify it at your will.

1.4 Line break in `.Rmd` file

The text in the `Rmd` file looks exactly like copied below as code. Why doesn't the output file (whatever its type) reflect the line break?

```
The text in the `Rmd` file looks exactly like copied below as code.
Why doesn't the output file (whatever its type) reflect the line break?
```