HPBBM. Exercises R. Scripts, data structures, reading data.

Ramon Diaz-Uriarte*

2016-10-21 (Release: Rev: 1edb189)

Contents

A Recommendations

- Return the exercise as a single text file with commented code.
- Use comments if you think it is justified.

A.1 Working directory

All of your code should run. Create a directory called `R-ex-1`; place in this directory the data files and the script with code. I will run your scripts from this directory. So, when reading data, you do NOT have to specify paths. For instance, do not do

```
read.something("~/R-ex-1/somedata")
```

but do

```
read.something("somedata")
```

*Dept. of Biochemistry, Universidad Autónoma de Madrid, Spain, http://ligarto.org/rdiaz, rdiaz02@gmail.com

A.2 Grading and points

All exercises are worth as many points as sections or entries they have (for instance, exercise 2 is worth 4 points), **EXCEPT** exercise 7 that is worth 4 points.

1 Scripts

1.1 Creating the script

Create a script that will:

1.1.1. Make sure there are no objects in the workspace (i.e., it will remove everything in the workspace).

1.1.2. Create a variable `x` and assign to it a sequence of numbers from 123 to 297 in jumps of 3.5.

1.1.3. Obtain a summary of that variable x.

1.2 Running the script

1.2.1. How would you run the script from RStudio?

1.2.2. How would you run the script directly from an R session?

1.2.3. How would you run the script from the command line or console of your operating system?

2 Some read.table operations

Take the file `AnotherDataSet.txt` and create the following different data sets:

- `d1.txt`, which is like `AnotherDataSet` but with the 23.4 substituted by an NA (as it says: write "NA", without the quotes, where it used to say 23.4).
- `d2.txt`, which is like `d1.txt`, but leave a blank instead of 23.4 and a blank space instead of the 14 in the second row and column.
- `d3.txt`, which is the same as `d2.txt` but without the header.
- `d4.txt`, which is the same as `d1.txt` but with it has a first row that says `@Hello, I created this file` and a final row that says `@Good bye!`.

Of course, do the above with a text editor, not a word processor. Now, what are the commands you need to use to properly read the above files?

2.1. For `d1.txt`

2.2. For `d2.txt`

2.3. For `d3.txt`. For `d3.txt` do not worry about column names, just about the contents.

2.4. For `d4.txt`

3 Packages

Go to Bioconductor and install package `OncoSimulR`. Go to CRAN and install package `car`

3.1. How can you find the version of the currently installed OncoSimulR package? (Please, give the version number)

3.2. How can you find the version of the currently installed car package? (Please, give the version number)

3.3. Function `oncoSimulIndiv` is a function from package `OncoSimulR`. If you want to use that function, what do you need to do (i.e., is installing the package enough, or do you need to do anything else?)

4 The help

4.1. Start a new R session (or a new RStudio session) and type `apropos("scatter3d")`. What do you get?

4.2. Do `help.search("scatter3d")`. What do you get? (Not all of you need to get the same output).

4.3. Now, load package `car`. Repeat the above. What do you get?

4.4. Do you understand why you get different results with `apropos` before and after loading `car`? Look at the help of `apropos` to understand what happens.

4.5. Do you understand the difference between `apropos` and `help.search` for `scatter3d`?

4.6. Now (in this session where you have `car` loaded) do both `help.search("scatter")` and `apropos("scatter")`. Which is more helpful if you are trying to find a function that has `scatter` in its name? Can you miss a function with `scatter` in its name that you might have installed in your machine if you use `apropos`?

4.7. Is there a way to find (maybe using `help.search`? look at its help) all functions that have `scatter` in their name (such as `scatter3d`) even if the package is not loaded? To help you here: compare (in an R session without `OncoSimulR` loaded) these two: `help.search("oncoSimulPop", fields = "alias")` and `help.search("oncoSimulPop", fields = "name")`.

5 Saving objects

5.1. Start a new, fresh, R session. Write code that will do this:

- Create an object called "x" with value 97.
- Create an object called "y" with value 95.
- Save a binary R object, called "`oneObject.RData`" that will contain **only** object x.

5.2. What would have been the difference of doing `save.image()`?

6 Vectors, data frames, etc

- Enter in R the following hear rates (in bit per minutes):

```
hr <- c(87, 78, 86, 62, 69, 69, 68, 67, 75, 76)
```

6.1. You are told the first three observations are from individuals of age 11, then you have two observations from individuals of age 63, then you have four observations from individuals of age 40, and you have an observation from an individual of age 47. Create a vector called `age`, but do it intelligently (use `rep` and similar). Of course, show the code.

6.2. Show the code to obtain the hear rates of all individuals who are less than 45 years old. Use the `hr` and `age` vectors to do it.

6.3. Create a new vector that contains the observations of the individuals who are 63 and 47 years old; call this "hr2". The names of these individuals are `Juan, Ana, Carmen`; place the names of the individuals in the vector.

6.4. Using the names, obtain the heart rates of Juan and Ana.

6.5. Create a matrix for Juan, Ana, and Carmen that contains, as columns, their hear rate and their age, and use their names as row names.

6.6. Repeat the above, but create a data frame, not a matrix.

6.7. Repeat the above, but do not use their names as row names, but just as another column. Could you do this with a matrix?

6.8. Using the matrix, obtain the heart rate of Juan. Do it using indices and using row names.

6.9. Using the matrix, obtain all the values associated to Juan (heart rate and age).

6.10. Using the data frame with names as row names, obtain the heart rate of Ana (there might be multiple ways of doing this).

6.11. Using the data frame with names as another column, obtain the heart rate of Ana (there might be multiple ways of doing this).

6.12. Using the data frame with names as another column, obtain all the values associate to Ana (her heart rate, her age, and her name).

6.13. Using the data frame with names as another column, obtain all the values (i.e. another data frame) for all individuals older than 60.

6.14. Using the data frame with names as row names, obtain all the values (i.e. another data frame) for all individuals older than 60.

6.15. Create a matrix of heart rates and ages for all the original measures for individuals who are older than 15. The simplest thing might be to create a matrix for all, and then create another matrix by subsetting this one.

6.16. It turns out we know the sexes of the above individuals. You find the following code: `sex <- c("M", "F")[c(1, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1)]` Do you think those sexes can correspond to the individuals above? Look in particular to the individuals of ages 63 and 47, for which I gave you the names (and, thus, implicitly their sexes) above.

6.17. Using the previous vector of sex (with, maybe, a correction for the value(s) of sex that might be wrong), add it to the matrix of heart rates and ages for the individuals that are older than 15 (see 6.15). I want you to combine (or something of that kind); I do not want you to create a new matrix from scratch.

6.18. Do the same thing with the data frame in 6.6.

7 Sort and order

For this exercise, you will need to look at the help of `sort` and `order`. And to get you going, do the following (and make sure you understand what is happening)

```
sort(age)

##  [1] 11 11 11 40 40 40 40 47 63 63

order(age)
```

```
##  [1]  1  2  3  6  7  8  9 10  4  5
```

```
sort(hr)
```

```
##  [1] 62 67 68 69 69 75 76 78 86 87
```

```
order(hr)
```

```
##  [1]  4  8  7  5  6  9 10  2  3  1
```

7.1. Using the matrix created in exercise 6, 6.15, create a new matrix that contains the exact same data but such that the values are order in such a way that you first order by age and then by heart rate. So you want the youngest individuals on top, and within age, you want the slowest heart rates on top. So we would get this:

```
##      hr age
## [1,] 67  40
## [2,] 68  40
## [3,] 69  40
## [4,] 75  40
## [5,] 76  47
## [6,] 62  63
## [7,] 69  63
```