library() vs require()

2019-05-28

The use of library() or require() is interesting. Each have is own pros and cons. If we’ll use the help for library (?library), we’ll see that “library and require load and attach add-on packages”. So, while practicality they do the same thing, they act differently.

In both cases both functions load the packages with name package and put it on the search list. However, When we call library(), if the package doesn’t exists a warning will be raised. In addition, the script will stop running.

library(somepackage)

## Warning: Error in library(somepackage) : there is no package called  
## ‘somepackage’)

On the other hand, when we call require() the function will return TRUE if the package exists, and FALSE if the package doesn’t exists. furthermore, unlike library(), the script will not stop running.

print(require(somepackage))

## Loading required package: somepackage

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,  
## logical.return = TRUE, : there is no package called 'somepackage'

## [1] FALSE

So, why do we care about these differences? Beside knowing what each function does, we can use them to our own advantage.

It’s considered better practice to use library() when we want to load packages outside of functions. This is because we would like to know if and where we have a bug as soon as possible. Calling a set of function which doesn’t exists is certainly a bug. We’ll use require() inside functions, as we can manipulate it’s TRUE/FALSE output. For example, in the beginning of our script, we can write a function that will install all the necessary function if they aren’t already installed [look here](https://amirdjv.netlify.com/post/r-and-rstudio-tricks-and-shortcuts/). We could also skip parts of a script if the packages aren’t already installed, etc.

Thank you for reading!