What is a R package and how to use it?

Unlike other programs, only fundamental functionalities come by default with R. You will thus often need to install some “extensions” to perform the analyses you want. These extensions which are are collections of functions and datasets developed and published by R users are called **packages**. They extend existing base R functionalities by adding new ones. R is open source so everyone can write code and publish it as a package, and everyone can install a package and start using the functions or datasets built inside the package, all this for free.

In order to use a package, it needs to be installed on your computer by running `install.packages("name_of_package")` (do not forget `""` around the name of the package, otherwise R will look for an object saved under that name!). Once the package is installed, you must load the package and only after it has been loaded you can use all the functions and datasets it contains. To load a package, run `library(name_of_package)` (this time `""` around the name of the package are optional, but can still be used if you wish).

Inefficient way to install and load R packages

Depending on how long you have been using R, you may use a limited amount of packages or, on the contrary, a large amount of them. As you use more and more packages you will soon start to have (too) many lines of code just for installing and loading them.

Here is a preview of the code from my PhD thesis showing how the installation and loading of R packages looked like when I started working on R (only a fraction of them are displayed to shorten the code):

```
# Installation of required packages
install.packages("tidyverse")
install.packages("ggplot2")
install.packages("readxl")
install.packages("dplyr")
install.packages("tidyr")
install.packages("ggfortify")
install.packages("DT")
install.packages("reshape2")
install.packages("knitr")
install.packages("lubridate")

# Load packages
library("tidyverse")
library("ggplot2")
library("readxl")
library("dplyr")
library("tidyr")
library("ggfortify")
library("DT")
library("reshape2")
library("knitr")
library("lubridate")
```

As you can guess the code became longer and longer as I needed more and more packages for my analyses. Moreover, I tended to reinstall all packages as I was working on 4 different computers and I could not remember which packages were already installed on which machine. Reinstalling all packages everytime I opened my script or R Markdown document was a waste of time.

More efficient way

Then one day, a colleague of mine shared some of his code with me. I am glad he did as he introduced me

to a much more efficient way to install and load R packages. He gave me the permission to share the tip, so here is the code I now use to perform the task of installing and loading R packages:

```
# Package names
packages <- c("ggplot2", "readxl", "dplyr", "tidyr", "ggfortify", "DT",
"reshape2", "knitr", "lubridate", "pwr", "psy", "car", "doBy", "imputeMissings",
"RcmdrMisc", "questionr", "vcd", "multcomp", "KappaGUI", "rcompanion",
"FactoMineR", "factoextra", "corrplot", "ltm", "goeveg", "corrplot", "FSA",
"MASS", "scales", "nlme", "psych", "ordinal", "lmtest", "ggpubr", "dslabs",
"stringr", "assist", "ggstatsplot", "forcats", "styler", "remedy", "snakecaser",
"addinslist", "esquisse", "here", "summarytools", "magrittr", "tidyverse",
"funModeling", "pander", "cluster")

# Install packages not yet installed
installed_packages <- packages %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  install.packages(packages[!installed_packages])
}

# Packages loading
lapply(packages, library, character.only = TRUE) %>%
  invisible()
```

This code for installing and loading R packages is more efficient in several ways:

1. The function `install.packages()` accepts a vector as argument, so one line of code for each package in the past is now one line including all packages
2. In the second part of the code, it checks whether a package is already installed or not, and then install only the missing ones
3. Regarding the packages loading (the last part of the code), the `lapply()` function is used to call the `library()` function on all packages at once, which makes the code more condense.
4. The output when loading a package is rarely useful. The `invisible()` function removes this output.

From that day on, every time I need to use a new package, I simply add it to the vector `packages` at the top of the code, which is located at the top of my scripts and R Markdown documents. No matter on which computer I am working on, running the entire code will install only the missing packages and will load all of them. This greatly reduced the running time for the installation and loading of my R packages.

Thanks for reading. I hope the article helped you to install and load R packages in a more efficient way.…