The Easy Way to Install a Package in R (with 8 Code Examples)

Référence: https://www.dataquest.io/blog/install-package-r/

1- Introduction

R is a powerhouse programming language with many data science applications. But before we can put its packages to work, we need to install them. Here's how.

R is a programming language for statistical computing, especially efficient for performing data science tasks. This popularity is because R offers an impressive choice of data science-oriented packages, which are collections of methods for implementing specific functionality not included in basic R. To start working with the packages in R, we first need to install them.

2- Installing R Packages from the CRAN Repository

The Comprehensive R Archive Network (CRAN) repository stores thousands of stable R packages designed for a variety of data-related tasks. Most often, you'll use this repository to install various R packages.

To install an R package from CRAN, we can use the `install.packages()` function:

```
install.packages('readr')
```

Here, we've installed the *readr* R package used for reading data from the files of different types: comma-separated values (CSV), tab-separated values (TSV), fixed-width files, etc. Make sure that the name of the package is in quotation marks.

We can use the same function to install several R packages at once. In this case, we need to apply first the `c()` function to create a character vector containing all the desired packages as its items:

```
install.packages(c('readr', 'ggplot2', 'tidyr'))
```

Above, we've installed three R packages: the already-familiar *readr*, *ggplot2* (for data visualization), and *tidyr* (for data cleaning).

If we're unsure which R module to use for a certain data science task, we can visit the CRAN Task Views page. Here, we find a convenient list of task categories (topics) and a brief description of each. Selecting the necessary topic, we'll open the page with all the R packages relevant to that category, as well as an exhaustive description of usage for each of them.

In both examples above, we passed in only one argument to the `install.packages()` function, which is actually the value of the `pkgs` parameter. This is usually enough for the majority of cases. However, this function has many optional parameters that can be useful in some cases (and we'll see some examples soon). For the full information about the usage of the `install.packages()` function, including all its possible parameters and their detailed descriptions, check the R Documentation page.

3- Installing R Packages from the CRAN Repository: Alternative Method

If we work with R in an IDE, we can use the menu instead of the `install.packages()` function to install the necessary modules from the CRAN repository. For example, in RStudio, the most popular IDE for R, we need to complete the following steps:

- Click `Tools` → `Install Packages`
- Select `Repository (CRAN)` in the `Install from:` slot
- Type the package name (or several package names, separated with a white space or comma)
- Leave `Install dependencies` ticked as by default
- Click `Install`

In the other IDEs for working with R, the buttons and commands will be called differently, but the logic behind them is the same, and all the steps are usually intuitive.

4- Installing R Packages from Other Sources

4.1- Installing R packages from GitHub

Sometimes, we may want to install a specific package *not* from CRAN but from another repository. The most common example is when we need to use an R module available only on GitHub. In such cases, we should perform the following steps:

- Install RTools if it hasn't been already installed (*otherwise, skip this step*)
    - Open https://cran.r-project.org/
    - Select the necessary operating system for downloading the installer (e.g., `Download R for Windows`)
    - Select `RTools`
    - Select the latest version of RTools
    - Wait for the completion of downloading
    - Run the installer with all the options by default (here we may need to click `Run anyway` on the first pop-up window)
    - For the new versions of R (v4.0.0), add `PATH='${RTOOLS40_HOME}\usr\bin;${PATH}'` to the `.Renviron` file
- Install the *devtools* package from CRAN:

```
install.packages('devtools')
```

OR

```
install.packages('devtools', lib='~/R/lib')
```

- Call the `install_github()` function from the *devtools* package (no need to download the whole package) using the following syntax:

  ```
  devtools::install_github(username/repo_name[/subdir])
  ```

  For example:

```
devtools::install_github('rstudio/shiny')
```

The approach described above is applicable only for **public** GitHub repositories. To install an R package from a private repository, we need to set the `auth_token` optional parameter of the `install_github()` function with a token obtained from https://github.com/settings/tokens. For more information on the usage of the `install_github()` function, consult the R Documentation

4.2- Installing R Packages from Other External Repositories

If we need to install a package stored neither on CRAN nor GitHub but on another external repository, we have to use again the `install.packages()` function, this time with two more optional parameters: `repos` representing the URL of the necessary repository and `dependencies` set to `TRUE` or `FALSE` depending on whether we want to install them too or not. For example:

```
install.packages('furrr', repos='http://cran.us.r-project.org',
dependencies=TRUE)
```

Note: the code above uses one of the CRAN packages as an example, but we can use any other URL where the necessary module is stored.

The `repos` and `dependencies` arguments are used also when working with R from Jupyter Notebook to install packages that aren't available in the essentials.

4.3- Installing R Packages from Zip Files

If you have an R package downloaded on your local machine as a .zip or tar.gz file, you can install it using the `install.packages()` function passing in the path where the zip file is saved (this will actually be the `pkgs` parameter), setting `repos` to `NULL` and `type` to `source`. For example:

```
install.packages('C:/Users/User/Downloads/abc_2.1.zip', repos=NULL,
type='source')
```

This approach works fine also when the zip file is located not on our local computer but directly on a repository like CRAN. In this case, we just need to provide the corresponding URL instead of a local path.

Alternatively, if we work with R in an IDE, we can use the menu instead of the `install.packages()` function to install the necessary package from a .zip or tar.gz file saved on our local machine, just like we did earlier for the packages from the CRAN repository. For example, in RStudio, we need to complete the following steps:

- Click `Tools` → `Install Packages`
- Select `Package Archive File (.zip, .tar.gz)` in the `Install from:` slot
- Find the corresponding file on the local machine, and click `Open`
- Click `Install`

In the other IDEs for working with R, the process will be similar.

5- Loading R Packages

Once the necessary packages are installed on our computer, the next step is to load each package to start working with them. While we need to install each package only once (i.e., the first time using R on a particular computer), we have to load each package for every new R session. For this purpose, we use the `library()` function and pass in the package name, as follows:

```
library(readr)
```

Note that, in this case, we don't need to include the package name in quotation marks.

6- Conclusion

In this tutorial, we discussed the ways of installing R packages on our local machine from various sources, such as the official CRAN repository, public or private GitHub repos, any other external repository, and a .zip or tar.gz file. In addition, we covered how to select the necessary package for a particular data science problem and how to load a package after its installation to begin working with it.