R packages

Sangbaptist

2023-10-01

Table of contents

What is a package in R language?

An R package is a collection of functions, data and compiled code that is bundled together in a standardized way to extend the functionality of R. In R, packages allow users to document, share and reuse codes.

I would recommend these books on R packages: Hadley Wickham & Jennifer Bryian and Hadley Wickham.

Examples packages in R are:

- **tidyverse**: This is a collection of some R packages designed to make data analysis easier, efficient and more intuitive.
- **caret**: Short for **Classification and REgression Training** provides uniform interfaces for creating predictive models in R, with tools for data *splitting, pre-processing, feature selection, model tuning* and many more.
- **ggplot2**: It is primarily for data visualization, offering a flexible and structured approach to creating static and dynamic plots.Rnews
- **shiny**: Shiny is designed for building **interactive web applications** that allow users to interact with data in real-time, without the need for advanced web programming knowledge.

Key features of an R package:

1. **Function**: In an R package, a function is a piece of code that performs a specific task. Functions are the main building blocks of the package, allowing users to perform calculations, transformations, visualizations, or analyses. These functions reside in the **R/ directory** and are what users call directly when they use the package. For example, if a package provides a function `plot_data()`, the function would be defined here. Functions are stored in .R files within the **R/ directory** but their usage and behavior are also documented in the **man/ directory** (via the `.Rd` files).
2. **Data**: Many R packages include datasets, especially if they are intended for teaching, demonstration, or testing. These datasets are stored in the **data/ directory** and are often provided as `.rda` or `.csv` files. The data/ directory holds these datasets, while the man/ directory will include documentation on how to use them. The vignettes/ directory may also contain tutorials demonstrating how to work with the datasets.
3. **Documentation**: Documentation in an R package describes the functions, data, and overall package usage. It includes user-friendly descriptions, examples, and details about the parameters, input/output, and purpose of each function or dataset. This documentation is stored in the **man/ directory** as `.Rd` files and can be accessed by users using `?function_name` or `help()` in R. Documentation is created using `.Rd` files in the **man/ directory**, and long-form tutorials or explanations can also be found in the **vignettes/ directory**.

Why use R package?

While R packages can greatly enhance productivity, simplify complex tasks, and provide access to advanced techniques, they are not always necessary for every task. The decision to use

an R package should depend on the task's complexity, performance needs, and your familiarity with the package. R packages extend the base capabilities of R by providing additional functions, tools, and datasets that are not included in the standard R installation. This allows users to perform more advanced tasks, such as machine learning, data visualization, or web development, without needing to reinvent the wheel.

Here are some few importance or advantages of using R packages:

1. **Reusability**: Packages in R provide ready-made solutions that save you from writing code from scratch.
2. **Community Support**. There are many R communities that you can get support from including **networking, learning opportunities** and **collaboration**, at all level. Examples are CRAN, R Consortium, Github and many more.
3. **Specialised tools**: Packages often have specific focus area like **genomics**, **machine learning**, **finance** or **visualisation**, providing tools tailored for those domains.
4. **Efficiency**: Some R packages are built with highly optimized algorithms, often implemented in C, C++, or Fortran, which are faster than pure R code. This makes certain packages more efficient, especially for large datasets or complex operations.

Sources of R packages

When R is installed on your computer, it comes with a set of **base packages**(e.g. **utils**, **graphics**, **datasets**, etc) and **recommended packages** (**MASS**, **survival**, **rpart**, etc) by default. The number of these packages (combined) depends on the R version your are using. These packages are essential for performing basic tasks and provide foundational functions for R. However, there are many other available packages from online.

R packages come from several primary sources, each contributing to the growth and accessibility of the R ecosystem.

Two of the biggest sources of R packages are CRAN (Comprehensive R Archive Network) and Bioconductor(an open source project for building tools to analyse genomic data) repositories. For more about repositories, check my other post.

Common package listing in R

How to install a package in R:

```
install.packages("package_name") #write the name of the package you
                                  #are intending to install (e.g. rio)
```

How to list packages installed with R:

```
installed.packages() #Do not mistake "installed" with "install"
```

List currently loaded packages

```
.packages()
```

List all installed packages

```
.packages(all.available = TRUE)
```

Listing available packages in R

```
library()
```

Loading an installed package into R

```
library(package_name) #Loading a single package
library(c(package_1, package_2)) # Loading two or more packages
require(package_name) #another way to load a package
```

Listing packages loaded by default

```
getOption("defaultPackages")
```