A Comprehensive Guide to Combining R and Python with `reticulate`

Alejandro Leonardo García Navarro

2024-06-20

Contents

Introduction

What is `reticulate`?

The `reticulate` package is a tool that allows to combine R and Python. It allows users to call Python from R and R from Python, combining the strengths of both programming languages in a single workflow.

With this library, you can import any Python module and access its functions, classes, and objects from R, enabling a more versatile and flexible approach to data analysis, machine learning, and statistical computing.

Benefits of combining R and Python

Combining R and Python brings together the best of both worlds:

1. Choose the best tool for each task by leveraging R's statistical analysis and Python's programming and machine learning strengths.
2. Access more libraries and packages from both ecosystems.
3. Easy transfer of data between R and Python for flexible data handling in complex analysis pipelines.

Prerequisites and installation

Before using the library, make sure you have the following prerequisites:

1. R Installation: Make sure you have R installed on your system. You can download it from CRAN.
2. Python Installation: Install Python on your system.
3. RStudio (Optional but recommended): Using RStudio as your IDE can simplify the process of using `reticulate`. Download RStudio from here.

Once you have completed all the prerequisites, it is time to install the package. Use the following command in your R console:

```
install.packages("reticulate")
```

After installation, load the package using:

```
library(reticulate)
```

Basic Usage

Importing Python Modules

To import a Python module in R using the `reticulate` package, you use the `import` function. For example, to import the `numpy` library, you can use:

```
np <- import("numpy")
```

With this, you can use the `np` object to access `numpy` functions and methods just as you would in Python:

```
# Create a numpy array
array <- np$array(c(1, 2, 3, 4, 5))
print(array)
```

```
## [1] 1 2 3 4 5
```

Running Python Code in R

Sometimes, it might be useful to execute Python code directly within an R script, and this can be easily done using the `py_run_string` function. This function allows you to run Python code as a string:

```
py_run_string("print('Hello from Python')")
```

```
## Hello from Python
```

Alternatively, it may be more convenient to directly execute a Python script file. For this, you can use the `py_run_file` function:

```r
# py_run_file("path/to/your_script.py")
py_run_file("test.py")
```

```
## The sum of 4 and 6 is 10
```

Accessing Python Objects in R

In the same way, you can access and manipulate Python objects in R. For example, if you create a Python list, you can access it in R:

```r
# You can access a Python list
py_run_string("my_list = [1, 2, 3, 4, 5]")
my_list <- py$my_list
print(my_list)
```

```
## [1] 1 2 3 4 5
```

```r
# You can also manipulate the list
my_list[1] <- 4
print(my_list)
```

```
## [1] 4 2 3 4 5
```

You can also access Python functions and call them from R:

```r
py_run_string("
def greet(name):
  return 'Hello, ' + name + '!'
")

greet <- py$greet
print(greet("World"))
```

```
## [1] "Hello, World!"
```

```r
print(greet("James"))
```

```
## [1] "Hello, James!"
```

Converting Data Types Between R and Python