Using Python and R

Diego López Tamayo [*]

Contents

Data scientists that learn to use the strengths of both languages are valuable because they have NO LIMITS.

[*] El Colegio de México, diego.lopez@colmex.mx

Connecting R and Python

Why would you want to do this?

The 2 most popular data science languages - Python and R - are often pitted as rivals. This couldn't be further from the truth. Data scientists that learn to use the strengths of both languages are valuable because they have NO LIMITS.

- Machine Learning: They can switch to Python to leverage scikit learn and tensorflow.
- Data Wrangling, Visualization, Apps & Reporting: They can quickly change to R to use tidyverse, shiny and rmarkdown.

The bottom line is that knowing both R and Python makes you SUPER PRODUCTIVE.

Thanks to the R reticulate package, you can run Python code right within an R script—and pass data back and forth between Python and R. In addition to reticulate, you need Python installed on your system. Well cover all this in the following 2 steps:

Step 1: Conda Installation

For Python Environments, we will use Anaconda (Conda), a python environment management tool specifically developed for data scientists.

1. Download Anaconda Distribution
2. Create a New Python Environment: Run the following code in your terminal:

```
"""
conda create -n py3.8 python=3.8 scikit-learn pandas numpy matplotlib
"""
```

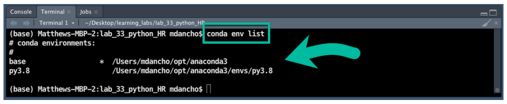
```
## '\nconda create -n py3.8 python=3.8 scikit-learn pandas numpy matplotlib\n'
```

This code does the following:

- Creates a new Python environment called "py3.8"
- Installs python version 3.8
- Installs the latest versions of scikit-learn, pandas, numpy, and matplotlib.

3. List your Conda Environments (in the Terminal)

- Use conda list env to list your Conda Environments in the Terminal.
- If you see py3.8, you are good to go.



```
(base) Matthews-MBP-2:lab_33_python_HR mdancho$ conda env list
# conda environments:
#
base                  *  /Users/mdancho/opt/anaconda3
py3.8                    /Users/mdancho/opt/anaconda3/envs/py3.8

(base) Matthews-MBP-2:lab_33_python_HR mdancho$ 
```

Step 2: Reticulate Setup

Fire up an R Markdown document and (install) load **tidyverse** and **reticulate**:

tidyverse - Loads the core data wrangling and visualization packages needed to work in R. reticulate - The key link between R and Python.

1. List your Conda Enviromnents typing in console **reticulate::conda_list()**

```
reticulate::conda_list()
```

```
##          name                                           python
## 1  r-miniconda                   /Users/Soporte/Library/r-miniconda/bin/python
## 2 r-reticulate /Users/Soporte/Library/r-miniconda/envs/r-reticulate/bin/python
## 3        py3.8                            /opt/anaconda3/envs/py3.8/bin/python
```

2. If you see py3.8, you are good to go.
3. Make sure your R Markdown document activates the “py3.8” environment using use_condaenv().

```
# use_condaenv("py3.8", required = TRUE)
```

4. Double check that reticulate is actually using your new conda env.

```
py_config()
```

```
## python:         /opt/anaconda3/envs/py3.8/bin/python
## libpython:      /opt/anaconda3/envs/py3.8/lib/libpython3.8.dylib
## pythonhome:     /opt/anaconda3/envs/py3.8:/opt/anaconda3/envs/py3.8
## version:        3.8.3 (default, May 19 2020, 13:54:14)  [Clang 10.0.0 ]
## numpy:          /opt/anaconda3/envs/py3.8/lib/python3.8/site-packages/numpy
## numpy_version:  1.18.1
##
## NOTE: Python version was forced by use_python function
```

You should see something like this where the python path is: python: **/Users/mdancho/opt/anaconda3/envs/py3.8/bin/** It may not be exact, but you should see “py3.8” in the file path.

Basic Python Tests

All of the code in this section uses python code chunks. This means you need to use {python} instead of {r} code chunks. **Pro Tip** Set up a Keyboard shortcut for Python Code Chunks. This is a massive productivity booster for Rmarkdown documents. My recommendation: Ctrl + Alt + P

- Test a sum

```
# Is python working???
1 + 1
```

```
## 2
```

Test Numpy & Pandas

Import numpy and pandas using the import shorthand np and pd respectively: numpy - Math Calculations pandas - Data Wrangling

```
import numpy as np
import pandas as pd
```

Test numpy using the np.arange() function to create a sequence of numbers in an array.

```
np.arange(1, 10)
```

```
## array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Next, test pandas by creating a data frame df using pd.DataFrame().

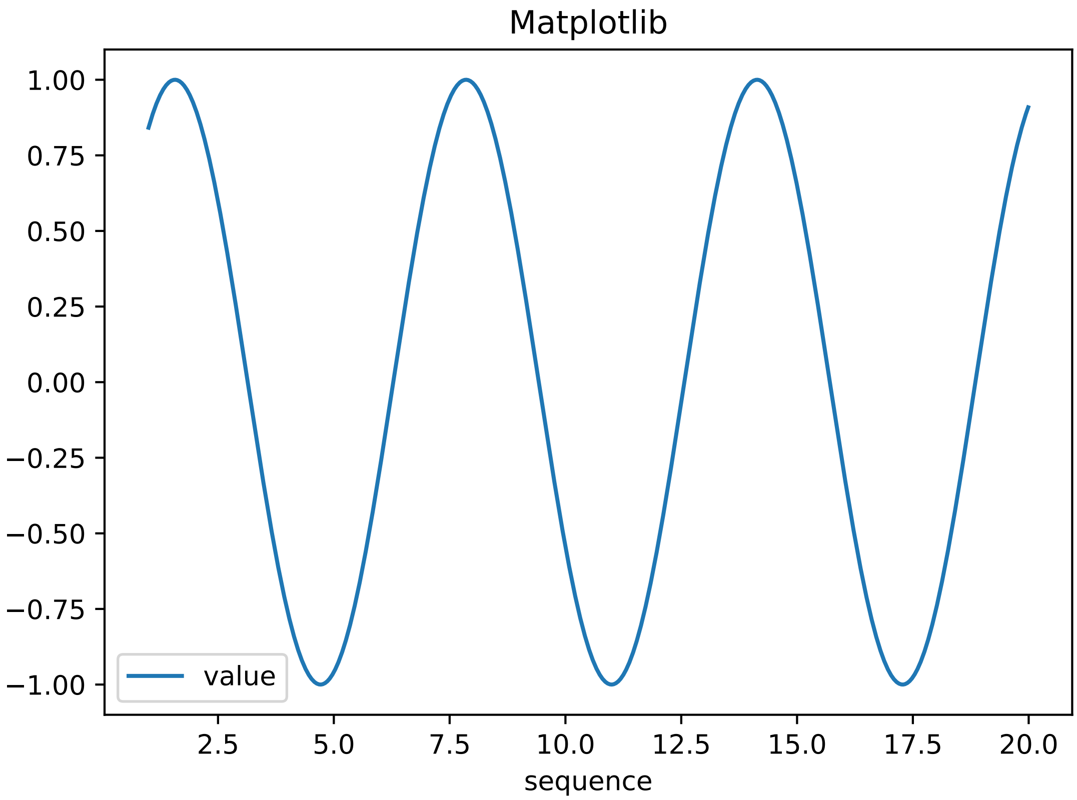
```
# Make a sequence in a data frame using dict format
df = pd.DataFrame(data = {"sequence":np.arange(1,20,.01)})
# Use assign (mutate) equivalent to calculate the np.sin() of the series
df = df.assign(value=np.sin(df["sequence"]))
df
```

```
##       sequence     value
## 0         1.00  0.841471
## 1          1.01  0.846832
## 2          1.02  0.852108
## 3          1.03  0.857299
## 4          1.04  0.862404
## ...         ...       ...
## 1895     19.95  0.891409
## 1896     19.96  0.895896
## 1897     19.97  0.900294
## 1898     19.98  0.904602
## 1899     19.99  0.908819
##
## [1900 rows x 2 columns]
```

Test Matplotlib

Matplotlib. Run the following pandas plotting code. If the visualization appears, matplotlib is installed.

```
import matplotlib as plt
df.plot(x="sequence", y = "value", title = "Matplotlib")
```



Pro-Tips

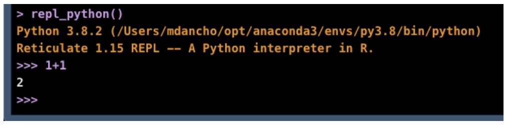
Python Chunks shortcut

Set up a Keyboard shortcut for Python Code Chunks. This is a massive productivity booster for Rmarkdown documents. My recommendation: Ctrl + Alt + P

Use Python Interactively

For debugging Python Code Chunks in R Markdown, it can help to use the repl_python() to convert your Console to a Python Code Console. To do so:

- In R Console, you can run python interactively using repl_python(). You will see >>> indicating you are in Python Mode.
- Make sure the correct Python / Conda Environment is selected.
- To escape Python in the console, just hit escape.



```
> repl_python()
Python 3.8.2 (/Users/mdancho/opt/anaconda3/envs/py3.8/bin/python)
Reticulate 1.15 REPL -- A Python interpreter in R.
>>> 1+1
2
>>>
```

Conda Terminal Commands

At some point you will need to create, modify, add more packages to your Conda Environment(s). Here are 4 useful commands:

1. Run conda env list to list the available conda environments
2. Run conda activate to activate a conda environment
3. Run conda update –all to update all python packages in a conda environment.
4. Run conda install to install a new package

Using Python in R

To keep things simple, let's start with just two lines of Python code to import the NumPy package for basic scientific computing and create an array of four numbers. The Python code looks like this:

```
import numpy as np
my_python_array = np.array([2,4,6,8])
for item in my_python_array:
    print(item)
```

```
## 2
## 4
## 6
## 8
```

Here's the cool part: You can use that array in R.In this next code chunk, I store that Python array in an R variable called my__r__array. And then I check the class of that array.

```
my_r_array <- py$my_python_array
class(my_r_array)
```

```
## [1] "array"
```

It's a class "array," which isn't exactly what you'd expect for an R object like this. But I can turn it into a regular vector with as.vector(my__r__array) and run whatever R operations I'd like on it, such as multiplying each item by 2.

```
my_r_vector <- as.vector(py$my_python_array)
class(my_r_vector)
```

```
## [1] "numeric"
```

```
my_r_vector_2 <- my_r_vector * 2
```

Next cool part: I can use that R variable back in Python, as r.my__r__array (more generally, r.variablename), such as

```
my_python_array2 = r.my_r_vector_2
print(my_python_array2)
```

```
## [4.0, 8.0, 12.0, 16.0]
```