Being Bilingual: Coding in Both R and Python

noRth 2020

Haema Nilakanta, PhD

July 14, 2020

Introduction

- The ongoing question: R or Python?

Introduction

- The ongoing question: R or Python?
- Why choose only one? Why not wrap one in the other?

Introduction

- The ongoing question: R or Python?
- Why choose only one? Why not wrap one in the other?
- Build more seamless data science pipelines & leverage strengths of both

Introduction

- The ongoing question: R or Python?
- Why choose only one? Why not wrap one in the other?
- Build more seamless data science pipelines & leverage strengths of both
- Objective: introduce how to work with R and Python in the R interface using the `reticulate` package
- I will assume minimal knowledge of Python

library(reticulate)

- First author and maintainer Kevin Ushey, RStudio
- Can use Python already on your system, a virtual environment, specific versions, or Miniconda
- Works with Python versions ≥ 2.7
- "The package enables you to *reticulate* Python code into R, creating a new breed of project that weaves together the two languages."

Setting up Python

- Some systems come downloaded with Python
- If not, many ways to download (one option: Anaconda)
- Anaconda loads Python and some well known packages vs. Miniconda which loads Python and tools to install more packages (lighter weight)
- First time installing and loading library, option to install Miniconda
  - Happens if you don't specify Python source explicitly to use
  - Creates an `r-reticulate` Conda environment with Python 3.6.10 with numpy version 1.18.15

Install package

```
# Install the package
install.packages("reticulate")
```

```
# Load the library
library(reticulate)

# and check the package version
packageVersion("reticulate")
```

```
## [1] '1.16'
```

Check python version

```
# Check what Python source it's using
py_config()
```

```
python:         /Users/haema/Library/r-miniconda/envs/r-reticulate/bin/python
libpython:      /Users/haema/Library/r-miniconda/envs/r-reticulate/lib/libpython3.6m.dylib
pythonhome:     /Users/haema/Library/r-miniconda/envs/r-reticulate:/Users/haema/Library/r-miniconda/envs/r
version:        3.6.10 | packaged by conda-forge | (default, Apr 24 2020, 16:27:41)  [GCC Clang 9.0.1 ]
numpy:          /Users/haema/Library/r-miniconda/envs/r-reticulate/lib/python3.6/site-packages/numpy
numpy_version:  1.18.5
```

Translation "dictionary"

Purpose	R	Python
Load packages	`library(name)`	`import name`
Data wrangling	`base` or `dplyr`	`pandas`
Computations	`base`	`numpy`
Graphics	`ggplot2`	`matplotlib`
...	...	...

Interacting with Python

1. Interactive Python (REPL)
2. Import Python packages
3. Load external Python scripts

Interacting with Python

1. Interactive Python (REPL)
2. Import Python packages
3. Load external Python scripts

Interactive Python

Can work with Python in the console itself (REPL = Read–Eval–Print Loop)

```
# Start an interactive session
repl_python()
Python 3.6.10
(/Users/haema/Library/r-miniconda/envs/r-reticulate/
    bin/python)
Reticulate 1.16 REPL -- A Python interpreter in R.
>>>
```

- The ">>>" indicates Python environment
- To exit session, type `exit` and hit enter
- Whatever is defined in this session will remain in Python session (coding version of Vegas)

Interacting with Python

1. Interactive Python (REPL)
2. Import Python packages
3. Load external Python scripts

Import packages

As with R, you may need functions available in other packages. Sometimes these packages are already installed (e.g., numpy and os)

```
# Load the os (operating system) package
os = import("os")

# print current working directory.
# In python keep () to run the function
os$getcwd()
```

```
## [1] "/Users/haema/Documents/noRth_reticulate_20200714"
```

```
# notice how it matches
getwd()
```

```
## [1] "/Users/haema/Documents/noRth_reticulate_20200714"
```

Import packages

For packages that are not pre-installed, specify the environment you want to install it to

```
# scipy popular Python scientific computing package
conda_install("r-reticulate", "scipy")

# another approach
# sklearn holds many machine learning functions
py_install('sklearn', pip = TRUE)

# tensorflow popular package for deep-learning modules
# reticulate designed to install package from CRAN
install.packages("tensorflow")
```

Then we can import the packages as before

```
scipy = import("scipy")
library(tensorflow)
```

Interacting with Python

1. Interactive Python (REPL)
2. Import Python packages
3. Load external Python scripts

Read in Python files

Similarly, we can read in a Python file (e.g., load a function). Consider the following function stored in logitfunc.py to compute $logit(x) = \frac{e^x}{1+e^x}$.

```
import numpy as np

def logit_py(x):
  return np.exp(x)/(1+np.exp(x))
```

Load file with

```
source_python("logitfunc.py")
logit_py(0.5)
```

```
## [1] 0.6224593
```

Interacting with Python

1. Interactive Python (REPL)
2. Import Python packages
3. Load external Python scripts

Example with Natural Language Processing (NLP)

Run through example if time permits

Conclusion

- Work with Python in R using library(reticulate)
- Build more seamless pipelines and leverage both systems
- Some more resources (clickable links):
  - Rstudio Reticulate
  - CRAN reticulate
  - Tutorial Rshiny + Python (virtual env)

Thank You!