Integration: R and Python

Programming for Statistical Science

Shawn Santo

Supplementary materials

Full video lecture available in Zoom Cloud Recordings

Additional resources

- `reticulate` vignette

Package `reticulate`

- R and Python are both great languages.
- What you can do in one language (for the most part) you can do in the other language
- Why not leverage the best of Python and R in a seamless workflow?

R package `reticulate` facilitates this seamless integrated workflow.

Setup

You'll need package `reticulate` and Python installed on your machine. Python is already installed on `Rook`. To verify RStudio can find Python run `py_discover_config()`.

```
# For use on Rook
reticulate::use_python(python = "/usr/bin/python3", required = TRUE)
library(reticulate)
```

```
py_discover_config()
```

```
#> python:         /usr/bin/python3
#> libpython:      /usr/lib64/libpython3.7m.so
#> pythonhome:     //usr://usr
#> version:        3.7.5 (default, Oct 17 2019, 12:21:00)  [GCC 8.3.1 20190223 (Red Hat 8.3.1-2)]
#> numpy:          /home/fac/sms185/.local/lib/python3.7/site-packages/numpy
#> numpy_version:  1.17.4
#>
#> NOTE: Python version was forced by use_python function
```

On your own machine you may need to configure which version of Python to use and where that version is located. To do so, use function `use_python()`.

Integrate Python into your R workflow

1. Include Python engine chunks into your R Markdown document. You will have the full set of available chunk options.

2. Call (source) Python scripts with `source_python()`.

3. Import Python modules with `import()`. For example, `import("pandas")` imports the `pandas` module into R, provided `pandas` is installed.

4. Transform your R console with `repl_python()` so you can interactively run Python code. Type `exit` to return to your R console.

REPL: read - evaluate - print - loop

Mixing Python and R chunks

Python in R Markdown

To insert Python code chunks in R Markdown, click the dropdown arrow on insert and select Python. Going forward, I'll place a code comment indicating which type of code chunk the given code resides in.

```
# python chunk
message = "Hello from a Python code chunk!"
print(message)
```

```
#> Hello from a Python code chunk!
```

```
# python chunk
colors = ['red', 'white', 'blue', 'green', 'purple']
colors[1:3]
```

```
#> ['white', 'blue']
```

```
# python chunk
colors.sort()
colors
```

```
#> ['blue', 'green', 'purple', 'red', 'white']
```

```
# python chunk
type(colors)
```

```
#> <class 'list'>
```

```python
# python chunk
x = list(range(1, 10))
y = list(range(-10, -1))

result = []

for i in range(1, 10):
  result.append(round(x[i - 1] ** y[i - 1], 4))

print(result)
```

```
#> [1.0, 0.002, 0.0002, 0.0001, 0.0001, 0.0001, 0.0004, 0.002, 0.0123]
```

```python
# python chunk
z = (1, 1, 2, 2, 6, 6, 18, 18)
t = [1, 1, 2, 2, 6, 6, 18, 18]
[type(z), type(t)]
```

```
#> [<class 'tuple'>, <class 'list'>]
```

```python
# python chunk
z *= 2
z
```

```
#> (1, 1, 2, 2, 6, 6, 18, 18, 1, 1, 2, 2, 6, 6, 18, 18)
```

```python
# python chunk
t[0] += 199
t
```

```
#> [200, 1, 2, 2, 6, 6, 18, 18]
```

Let's try and use objects `z` and `t` in an R chunk to take advantage of R's vectorization functionality.

```
# r chunk
z + t
```

```
#> Error in eval(expr, envir, enclos): object 'z' not found
```

```
# r chunk
t
```

```
#> function (x)
#> UseMethod("t")
#> <bytecode: 0x55ea16f42278>
#> <environment: namespace:base>
```

Objects `z` and `t` in our Python chunks do not exist in our R environment. How can we interact with these objects in R?

Calling Python from R

```
# python chunk
news = {
  'title': "Billion-Dollar Art Heist: Thieves" +
           "Cut Alarms With Fire at Dresden's Green Vault Palace",
  'author': None,
  'name': "Google News",
  'id': "google-news"
}

type(news)
```

```
#> <class 'dict'>
```

```
# python chunk
news
```

```
#> {'title': "Billion-Dollar Art Heist: ThievesCut Alarms With Fire at Dresden's
```

Python code is executed by default in the main module. You can then access any objects created using the `py` object exported by reticulate.

```
# r chunk
py$news
```

```
#> $title
#> [1] "Billion-Dollar Art Heist: ThievesCut Alarms With Fire at Dresden's Green V
#>
#> $author
#> NULL
#>
#> $name
#> [1] "Google News"
#>
#> $id
#> [1] "google-news"
```

Object `py$news` is an R list. Package reticulate translated the Python dictionary to an R list object.

```
# r chunk
py$news[["title"]]
```

```
#> [1] "Billion-Dollar Art Heist: ThievesCut Alarms With Fire at Dresden's Green V
```

```r
# r chunk
py$news$name
```

```
#> [1] "Google News"
```

```r
# r chunk
news_header <- py$news[1:2]
news_header
```

```
#> $title
#> [1] "Billion-Dollar Art Heist: ThievesCut Alarms With Fire at Dresden's Green V
#>
#> $author
#> NULL
```

Use `py$_<obj_name>` to work with a Python object in an R chunk.

Another example

```
# python chunk
nums = [1, 2, 3, 4, 5]
stuff = [4, 1.0, "cat", "dog", [3, 2, 1, 0], (2, 3)]
```

What types of objects will `nums` and `stuff` be in R?

```
# r chunk
str(py$nums)
```

```
#>  int [1:5] 1 2 3 4 5
```

```
# r chunk
str(py$stuff)
```

```
#> List of 6
#>  $ : int 4
#>  $ : num 1
#>  $ : chr "cat"
#>  $ : chr "dog"
#>  $ : int [1:4] 3 2 1 0
#>  $ :List of 2
#>   ..$ : int 2
#>   ..$ : int 3
```

Type conversions

R	Python	Examples
Single-element vector	Scalar	`1`, `1L`, `TRUE`, `"abcde"`
Multi-element vector	List	`c(1.0, 2.0, 3.0)`, `c(1L, 2L, 3L)`
List of multiple types	Tuple	`list(1L, TRUE, "foo")`, `tuple(3, 4, 5)`
Named list	Dictionary	`list(a = 1L, b = 2.0)`, `dict(x = x_data)`
Matrix/Array	NumPy ndarray	`matrix(c(1,2,3,4), nrow = 2, ncol = 2)`
Data Frame	Pandas DataFrame	`data.frame(x = c(1,2,3), y = c("a", "b", "c"))`
Function	Python function	`function(x) x + 1`
NULL, TRUE, FALSE	None, True, False	`NULL`, `TRUE`, `FALSE`

Calling R from Python

We can easily go the other way in terms of object conversion: R objects that we want to use in a Python code chunk.

```r
# r chunk
mtcars_small <- mtcars %>%
  select(mpg, cyl, wt) %>%
  sample_n(4)
```

```python
# python chunk
import pandas
r.mtcars_small.mean()
```

```
#> mpg    20.3000
#> cyl     6.0000
#> wt      3.4875
#> dtype: float64
```

Use `r._<obj_name>` to work with an R object in a Python chunk.

Exercises

1. Use Python to read in data from the Montgomery County of Maryland Adoption center - https://data.montgomerycountymd.gov/api/views/e54u-qx42/rows.csv?accessType=DOWNLOAD. In a Python code chunk, clean up the variable names so they are all lowercase and every space is replaced with a _. Subset the data frame so it only contains columns `'animal_id':'sex'`; save it as a data frame object named `pets`.

   In an R chunk, get the counts for each breed. Create a bar plot that shows the counts of the animal breeds where there are at least 4 adoptable pets of said breed. Color the bars according to the animal's type.

2. Diagnose the error in the below set of code.

```
# r chunk
x <- seq(1, 11, by = 2)
```

```
# python chunk
y = list(range(-20, 21))
y[r.x[5]]
```

```
#> Error in py_call_impl(callable, dots$args, dots$keywords): TypeError: list indices must be integer
#>
#> Detailed traceback:
#>   File "<string>", line 1, in <module>
```

Exercise 1 hints

Python code chunk starter code:

```
# python chunk
import pandas as pd
pets = pd.read_csv("https://data.montgomerycountymd.gov/api/views/e54u-qx
```

See also `columns`, `str.replace()`, and `str.lower()`.

Consult https://pandas.pydata.org/pandas-docs/stable/getting_started/comparison/comparison_with_r.html for the translation from R to Python with regards to `dplyr` and `pandas`.

Cautious integration

In general, you need to know the rules of the less flexible language with regards to code integration.

Common gotchas:

- 1 in R is not 1 in Python with regards to the type
- R has 1-based indices, Python has 0-based indices
- Python list indices must be integers

For certain circumstances you may need to force conversion of R types to Python types. R functions `dict()` and `tuple()` allow manual conversion to Python dictionaries and tuples, respectively.

Exercise

Investigate the conversion from Python to R for a Python Set. How about for an object of class `range` in Python?

```
# python chunk
x = range(1, 5)
s = {1, 1, 3, 4, 5, 5, 10, 10}
```

Sourcing Python scripts

Read and evaluate a Python script

Consider the simple Python script

```
def add(x, y):
  return x + y
```

I'll save this as `add.py` in a directory named `python_scripts`. To read and evaluate this in R, use `source_python()`.

```
# r chunk
source_python("python_scripts/add.py")
```

What do you notice about your R environment?

```r
# r chunk
add(x = 1, y = 0)
```

```
#> [1] 1
```

```r
# r chunk
add(x = "Package reticulate is ", y = "great!")
```

```
#> [1] "Package reticulate is great!"
```

```r
# r chunk
z <- c(4, 5)
add(z[1], z[2])
```

```
#> [1] 9
```

```r
# r chunk
add(c(1, 2, 3), c(-3, -2, -1))
```

```
#> [1]  1  2  3 -3 -2 -1
```

Another example

Consider this Python script that returns a specific form of a matrix.

```
def mat_design(rows, cols, design = "I"):

    import numpy as np

    if design == "I":
        mat = np.eye(max(rows,cols))
    elif design == "zeros":
        mat = np.zeros((rows, cols))
    elif design == "ones":
        mat = np.ones((rows, cols))
    else:
        mat = "Invalid design"

    return mat
```

Use `source_python()` to bring it to your R environment.

```
# r chunk
source_python("python_scripts/mat_design.py")
```

```r
# r chunk
mat_design(3, 3, design = "I")
```

```
#> Error in py_call_impl(callable, dots$args, dots$keywords): TypeError: 'float' o
#>
#> Detailed traceback:
#>   File "<string>", line 6, in mat_design
#>   File "/home/fac/sms185/.local/lib/python3.7/site-packages/numpy/lib/twodim_ba
#>     m = zeros((N, M), dtype=dtype, order=order)
```

What happened?

```r
# r chunk
mat_design(3L, 5L, design = "I")
```

```
#>      [,1] [,2] [,3] [,4] [,5]
#> [1,]    1    0    0    0    0
#> [2,]    0    1    0    0    0
#> [3,]    0    0    1    0    0
#> [4,]    0    0    0    1    0
#> [5,]    0    0    0    0    1
```

```r
# r chunk
mat_design(2L, 3L, design = "ones")
```

```
#>      [,1] [,2] [,3]
#> [1,]    1    1    1
#> [2,]    1    1    1
```

```r
# r chunk
mat_design(2L, 3L, design = "zeros")
```

```
#>      [,1] [,2] [,3]
#> [1,]    0    0    0
#> [2,]    0    0    0
```

```r
# r chunk
mat_design(1000L, 1000L, design = "sparse")
```

```
#> [1] "Invalid design"
```

Integration beyond R and Python

R and other languages

- R and C++, `rcpp`, http://www.rcpp.org/
- R and MatLab, `R.matlab`, https://cran.r-project.org/web/packages/R.matlab/R.matlab.pdf
- R and Julia, `JuliaCall`, https://non-contradiction.github.io/JuliaCall/
- R and Java, `rJava`, http://www.rforge.net/rJava/

The Thesaurus of Mathematical Languages is a useful resource to consult as you integrate other languages with R.

References

1. Interface to Python. (2020). https://rstudio.github.io/reticulate/.

2. Mathesaurus. (2020). http://mathesaurus.sourceforge.net/.