

Statistics 360: Advanced R for Data Science

Lecture 11

Becky Lin

R and Python: References

- ▶ The reticulate website: <https://rstudio.github.io/reticulate/>
- ▶ Python setup: <https://docs.python.org/3/using/index.html>
- ▶ Python tutorial: <https://docs.python.org/3/tutorial/>

Calling Python from R

- ▶ Use cases:
 - ▶ Tap into a growing set of tools for data science, such as scikit-learn <https://scikit-learn.org/stable/> , keras <https://keras.io/> , ...
 - ▶ Workflow requires substantial computations in **both** languages; e.g., fit a neural network in Python, plot results using ggplot2
- ▶ Not recommended:
 - ▶ Using R/RStudio as a development environment for Python.
 - ▶ Better to go all-in with Python and use Jupyter Notebooks, Spyder, etc. as your IDE.

Prerequisites

- ▶ Install and load the reticulate package

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

- ▶ Install Python
 - ▶ See the Python setup link on the references slide, or the Python distributions on the next slide
 - ▶ Warning for Mac users: Macs with the Apple M1 processor can run both M1 and x86 (Intel processor) binaries. Make sure your R and Python installations are for the same processor, or you will get errors when you try to start Python from R.
- ▶ Install the Python packages that you need.
 - ▶ More on this below.

Python packages

- ▶ Python packages are like R packages.
- ▶ Install with `pip` or `conda`.
 - ▶ So far I've only been able to find a miniconda installer to work with my M1 version of R, so I'll discuss conda-based installs.
- ▶ conda comes with the Python distributions anaconda and miniconda <https://docs.conda.io/projects/conda/en/latest/user-guide/install>
 - ▶ miniconda is smaller and faster to install, but you will need to install packages yourself
 - ▶ anaconda includes all the python packages you will need for data science but takes more disk space and probably includes stuff you will never use
- ▶ Once you have conda install packages from the command line with `conda install <package>`.

Python environments

- ▶ Python environments, like RStudio projects, are used to compartmentalize your work with Python.
 - ▶ A complete Python installation, including its own Python executable and packages.
 - ▶ Create from the command line with `conda create --name <env_name>`
 - ▶ Then “activate” with `conda activate <env_name>` and “de-activate” with `conda deactivate`.

Installing packages with reticulate, Part I

- ▶ See https://rstudio.github.io/reticulate/articles/python_packages.html
- ▶ We'll first install into the "base" Python environment

```
library(reticulate)  
#conda_install(packages=c("pandas","scikit-learn")) # commented out to
```


Installing packages with reticulate, Part II

- ▶ Now install into another Python environment
- ▶ Note: Not currently working on my system, but it used to!

```
library(reticulate)
```

```
#conda_create("r-reticulate") # commented out to avoid re-doing every t  
#conda_install("r-reticulate", c("pandas","scikit-learn")) # commented
```

Using a conda environment

- ▶ For each R session in which you want to use your conda environment:

```
library(reticulate)
use_condaenv("r-reticulate",required=TRUE)
# Can then call py_config() to see which Python version is being
```

Python embedded in RMarkdown

- ▶ Example from https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html

```
# code chunk header is ```{python} rather than ```{r}  
import numpy as np  
import pandas as pd  
df = pd.DataFrame(np.random.randn(3,4), columns=['A', 'B', 'C', 'D'])  
df
```

```
##           A           B           C           D  
## 0  0.116952 -0.644985 -0.884050 -0.649658  
## 1 -0.221388  0.596401  0.335031  0.087408  
## 2 -0.473363 -1.736616 -0.293679  1.929203
```

Importing Python packages (modules)

- ▶ You can also import Python packages into R and call their functions directly.

```
npr <- import("numpy.random")
pd <- import("pandas") # import is from reticulate
df <- pd$DataFrame(npr$randn(3L,4L), columns=c('A','B','C','D'))
df
```

		A	B	C	D
## 1	0.3965882	1.32744536	-0.1481296	-0.2214696	
## 2	-1.3726563	-1.46410869	-0.6774934	-0.4406373	
## 3	-1.3183013	-0.07735525	-0.1054971	-0.7835564	

Notes

- ▶ Access Python functions from an imported package with \$.
- ▶ The `randn()` function requires integer arguments – have to use 3L and 4L to pass integers.
 - ▶ reticulate converts R vectors of length 1 to Python scalars.
 - ▶ In general reticulate will try to convert to/from appropriate data types. See the list at <https://rstudio.github.io/reticulate/index.html#type-conversions>
- ▶ I used the numpy random number generator, but passed column names as an R character vector.
 - ▶ reticulate converts this to a python list.

Sourcing Python scripts

- ▶ Source with `source_python()`. By default, objects created by the Python script are made available as R objects in your workspace, and also in the hidden environment `py`.
- ▶ Pass an environment if you want to store somewhere else.

```
# source_python("lec11_1.py") # default
my_py_env <- new.env()
source_python("lec11_1.py", envir=my_py_env)
ls(my_py_env)
```

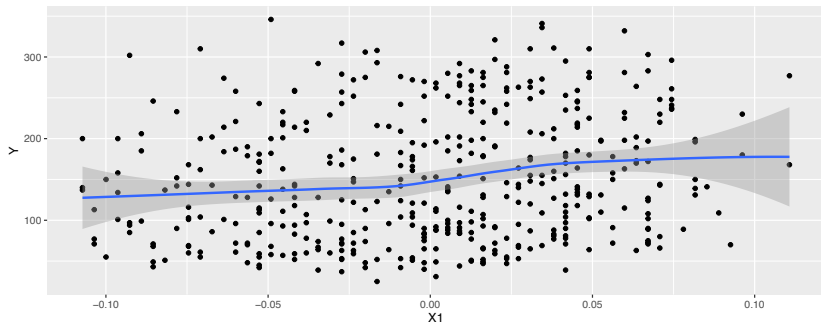
```
## [1] "df" "r"
my_py_env$df
```

```
##           A           B           C           D
## 1 -0.7867444  0.4330144  1.03313035 -1.415685
## 2  1.6289353 -0.5534447 -0.12533629 -1.269741
## 3 -0.9566918 -1.6008279  0.03205155  1.076558
```

Another example

```
source_python("lec11_2.py")  
MSE
```

```
## [1] 2859.69  
ddat <- data.frame(Y=diabetes_y,diabetes_X)  
library(ggplot2)  
ggplot(ddat,aes(x=X1,y=Y)) + geom_point() + geom_smooth()
```



Python REPL

- ▶ You can also start the Python interpreter and compute interactively.
 - ▶ Useful for debugging your Python scripts

```
# repl_python()  
# type your Python commands  
# objects will be available in R through py object  
# exit to quit
```