**2 Steps to Python**

Yeah, you heard me right. With only 2 steps, we are able to use Python in R!

**Step 1 – Reticulate Setup**

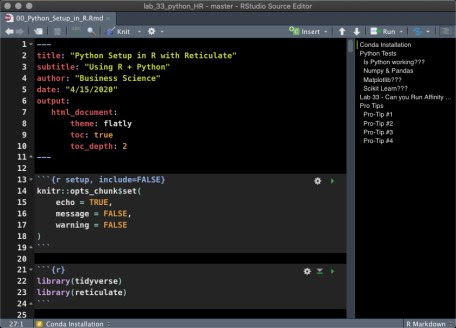
Fire up an R Markdown document and load tidyverse and reticulate:

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

library(tidyverse)

library(reticulate)

Your R Markdown should have something that looks like this (possibly without the outline, but that’s where we are headed).



R Markdown (Rmd) File with reticulate

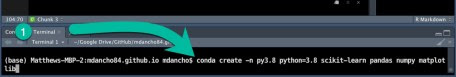
**Step 2 – Conda Installation**

Next, we need to make sure we have the Python Environment setup that we want to use. For Python Environments, we will use Anaconda (Conda), a python environment management tool specifically developed for data scientists.

**Download Conda**

* Anaconda Distribution – [Installation Instructions](https://docs.anaconda.com/anaconda/install/)

**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

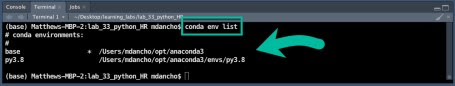
This code does the following:

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

In the future you can always add more python packages

**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.



**List your Conda Enviromnents (in R Markdown)**

Back in R Markdown, we can do the same thing using retculate::conda\_list().

conda\_list()

## name python

## 1 anaconda3 /Users/mdancho/opt/anaconda3/bin/python

## 2 py3.8 /Users/mdancho/opt/anaconda3/envs/py3.8/bin/python

**Set Your Conda Environment (in R Markdown)**

Make sure your R Markdown document activates the “py3.8” environment using use\_condaenv().

use\_condaenv("py3.8", required = TRUE)

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

py\_config()

## python: /Users/mdancho/opt/anaconda3/envs/py3.8/bin/python

## libpython: /Users/mdancho/opt/anaconda3/envs/py3.8/lib/libpython3.8.dylib

## pythonhome: /Users/mdancho/opt/anaconda3/envs/py3.8:/Users/mdancho/opt/anaconda3/envs/py3.8

## version: 3.8.2 (default, Mar 26 2020, 10:43:30) [Clang 4.0.1 (tags/RELEASE\_401/final)]

## numpy: /Users/mdancho/opt/anaconda3/envs/py3.8/lib/python3.8/site-packages/numpy

## numpy\_version: 1.18.1

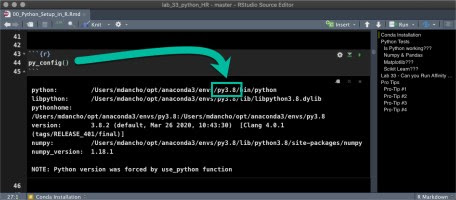
##

## NOTE: Python version was forced by use\_python function

You should see something like this where the python path is:

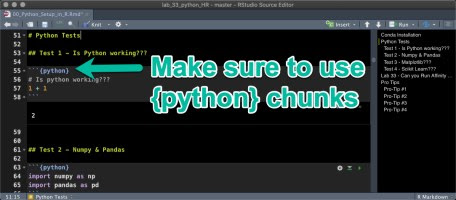
* python: /Users/mdancho/opt/anaconda3/envs/py3.8/bin/python.

It may not be exact, but you should see “py3.8” in the file path.



**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.



**Test 1 – Is Python working???**

* Let’s add 1 + 1
* You should see 2

# Is python working???

1 + 1

## 2

**Test 2 – 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

**Numpy**

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])

**Pandas**

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

##

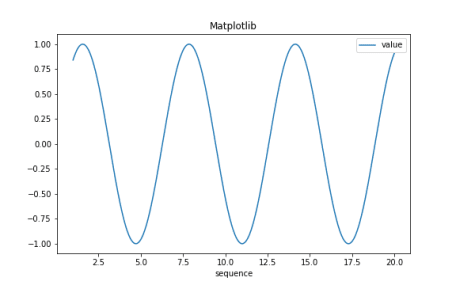
## [1900 rows x 2 columns]

**Test 3 – 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")



**Test 4 – Scikit Learn**

Run a test Random Forest using RandomForestClassifier from the sklearn.ensemble module of Scikit Learn.

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(random\_state=0)

X = [[ 1, 2, 3], # 2 samples, 3 features

[11, 12, 13]]

y = [0, 1] # classes of each sample

clf.fit(X, y)

## RandomForestClassifier(bootstrap=True, ccp\_alpha=0.0, class\_weight=None,

## criterion='gini', max\_depth=None, max\_features='auto',

## max\_leaf\_nodes=None, max\_samples=None,

## min\_impurity\_decrease=0.0, min\_impurity\_split=None,

## min\_samples\_leaf=1, min\_samples\_split=2,

## min\_weight\_fraction\_leaf=0.0, n\_estimators=100,

## n\_jobs=None, oob\_score=False, random\_state=0, verbose=0,

## warm\_start=False)

Use the predict() method to make a prediction on the training data set.

clf.predict(X) # predict classes of the training data

## array([0, 1])

**Can you Run Affinity Progagation???**

A simple test is to run the AffinityPropagation test from [Scikit Learn’s website](https://scikit-learn.org/stable/auto_examples/cluster/plot_affinity_propagation.html#sphx-glr-auto-examples-cluster-plot-affinity-propagation-py).

from sklearn.cluster import AffinityPropagation

from sklearn.datasets import make\_blobs

# #############################################################################

# Generate sample data

centers = [[1, 1], [-1, -1], [1, -1]]

X, labels\_true = make\_blobs(n\_samples=300, centers=centers, cluster\_std=0.5,

random\_state=0)

# Compute Affinity Propagation

af = AffinityPropagation(preference=-50).fit(X)

cluster\_centers\_indices = af.cluster\_centers\_indices\_

labels = af.labels\_

n\_clusters\_ = len(cluster\_centers\_indices)

# #############################################################################

# Plot result

import matplotlib.pyplot as plt

from itertools import cycle

plt.close('all')

plt.figure(1)

plt.clf()

colors = cycle('bgrcmykbgrcmykbgrcmykbgrcmyk')

for k, col in zip(range(n\_clusters\_), colors):

class\_members = labels == k

cluster\_center = X[cluster\_centers\_indices[k]]

plt.plot(X[class\_members, 0], X[class\_members, 1], col + '.')

plt.plot(cluster\_center[0], cluster\_center[1], 'o', markerfacecolor=col,

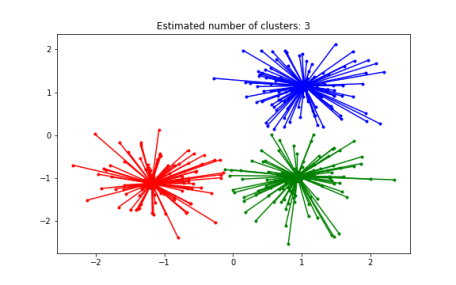
markeredgecolor='k', markersize=14)

for x in X[class\_members]:

plt.plot([cluster\_center[0], x[0]], [cluster\_center[1], x[1]], col)

plt.title('Estimated number of clusters: %d' % n\_clusters\_)

plt.show()



**Become Great at Shiny**

Up until now we haven’t talked about Shiny! It’s web application framework that is used to take your python and R machine learning models into ***Production.***

**If you need to learn R Shiny as fast as possible**, I have the perfect program for you. It will accelerate your career.

**Pro Tips (Python in R)**

Now that you have python running in R, use these pro-tips to make your experience way more enjoyable.

**Pro-Tip #1 – Python Chunk Keyboard Shortcut**

I can’t stress this one enough – **Set up a Keyboard shortcut for Python Code Chunks.** This is a massive productivity booster for Rmarkdown documents.

* My preference: Ctrl + Alt + P

When you hit Ctrl + Alt + P, a {python} code chunk will appear in your R Markdown document.

**Pro-Tip #2 – 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.

**Pro-Tip #3 – 4 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