Integrating Python and R

Mahdi Goudarzi

R Meetup

May 29, 2018

Outline

1. Why connection? Comparison between R and Python
   - Introducing R
   - Introducing Python

2. Packages
   - reticulate (calling Python from R)
   - RPy2(Calling R from Python)
   - pyRserve (Calling R from Python)

Introducing R

- Open source language for data analysis, statistics and graphical models.
- Huge community that provides support through mailing lists, user-contributed documentation and a very active Stack Overflow group.
- There is also CRAN, a huge repository of curated R packages to which users can easily contribute.
- R and visualization are a perfect match. Some must-see visualization packages are ggplot2, ggvis, googleVis and rCharts.
- R's learning curve is non-trivial

Introducing Python

- Python was created by Guido Van Rossem in 1991
- Emphasizes productivity and code readability.
- It's a flexible language that is great to do something novel, and given its focus on readability and simplicity.
- The IPython Notebook makes it easier to work with Python and data.
- Python is a general purpose language that is easy and intuitive.
- compared to R, visualizations are usually more convoluted, and the results are not always so pleasing to the eye.
- It does not offer an alternative to the hundreds of essential R packages.
- Python learning curve is almost flat.

Packages

reticulate

- Reticulate embeds a Python session within your R session.
- A comprehensive set of tools for switching between Python and R.
- Calling Python from R in a variety of ways including R Markdown, sourcing Python scripts, importing python modules.
- Translation between R and Python objects (for example , between R and Pandas data frames, R matrices and NumPy arrays).

Python in R Markdown

The **reticulate** package includes a Python engine for *RMarkdown* with the following features:

- Run Python chunks in a single Python session embedded within your R session
- Printing of Python output, including graphical output from matplotlib.
- Access to objects created within Python chunks from R using the py object.
- Access to objects created within R chunks from Python using the r object

R markdown

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

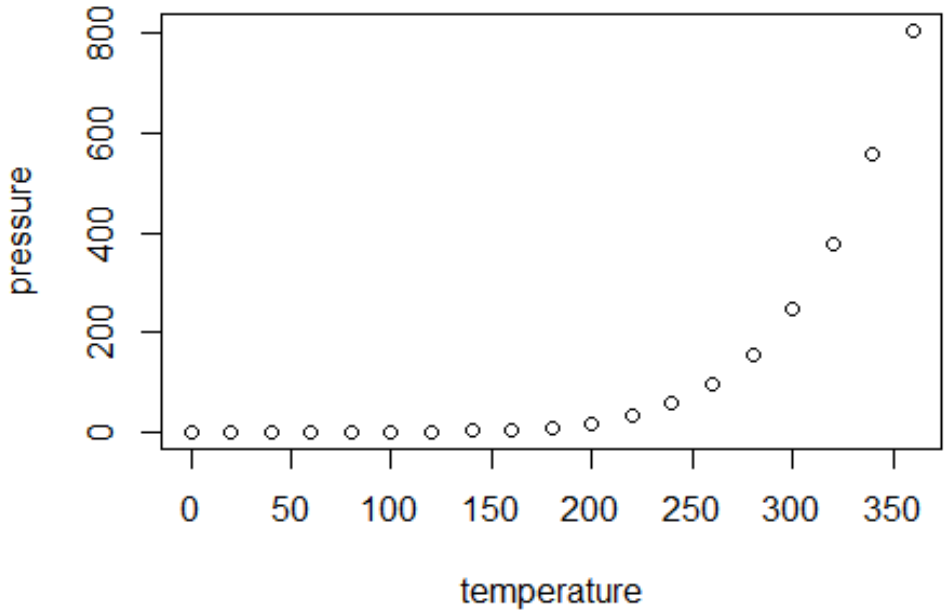
```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:

```
plot(pressure)
```

Rmarkdown contd....



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Example 1

```
#use_python(path_to_python)
knitr::knit_engines$set(python = reticulate::eng_python)
library(reticulate)
use_python('C:\\Python27')

import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
meow = np.array([[-1,1,1],[1,1,-1],[1,1,1]])
print(meow)
## [[-1  1  1]
##  [ 1  1 -1]
##  [ 1  1  1]]

from numpy import linalg
woof=linalg.inv(meow)
print(woof)

## [[-0.5 -0.   0.5]
##  [ 0.5  0.5  0. ]
##  [ 0.  -0.5  0.5]]
```

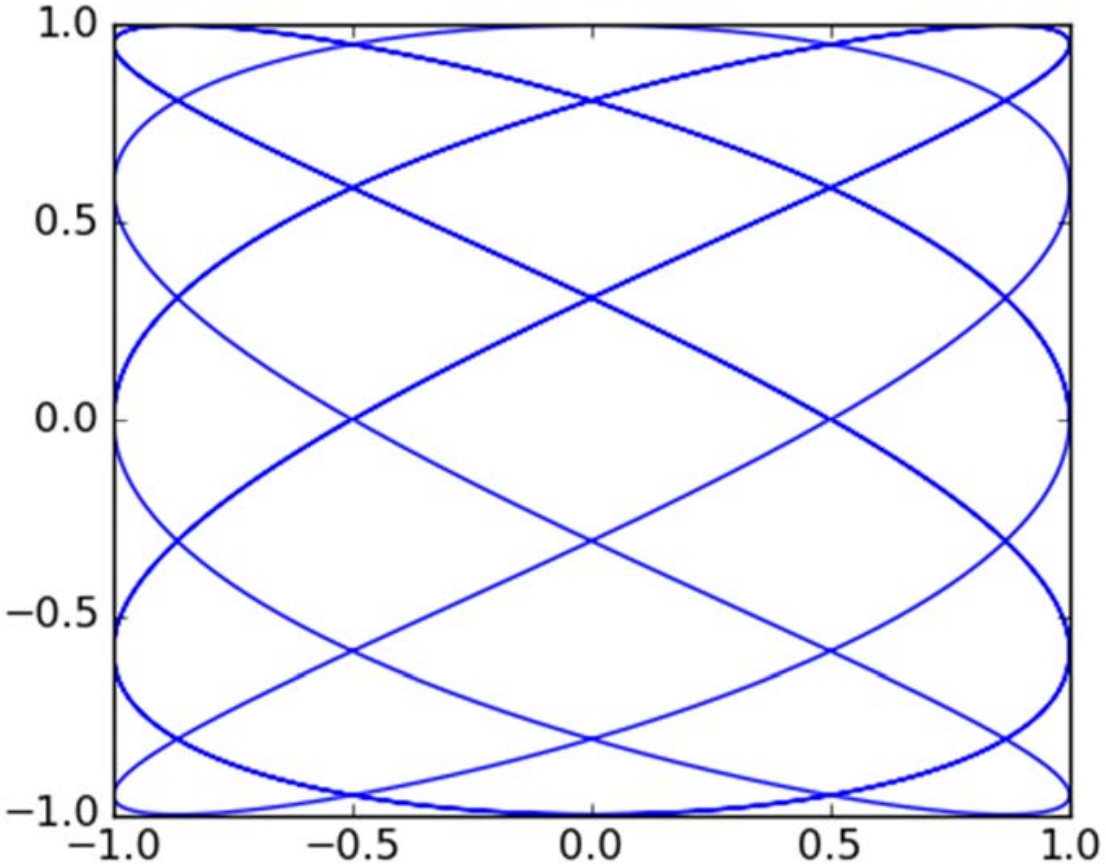
Example 2

```{python}

import numpy as np
import matplotlib.pyplot as plt
phi = np.arange(0,3*np.pi,0.0025)
x=np.cos(5*phi)
y=np.sin(3*phi)
plt.plot(x,y)
plt.show()

```

Example 2



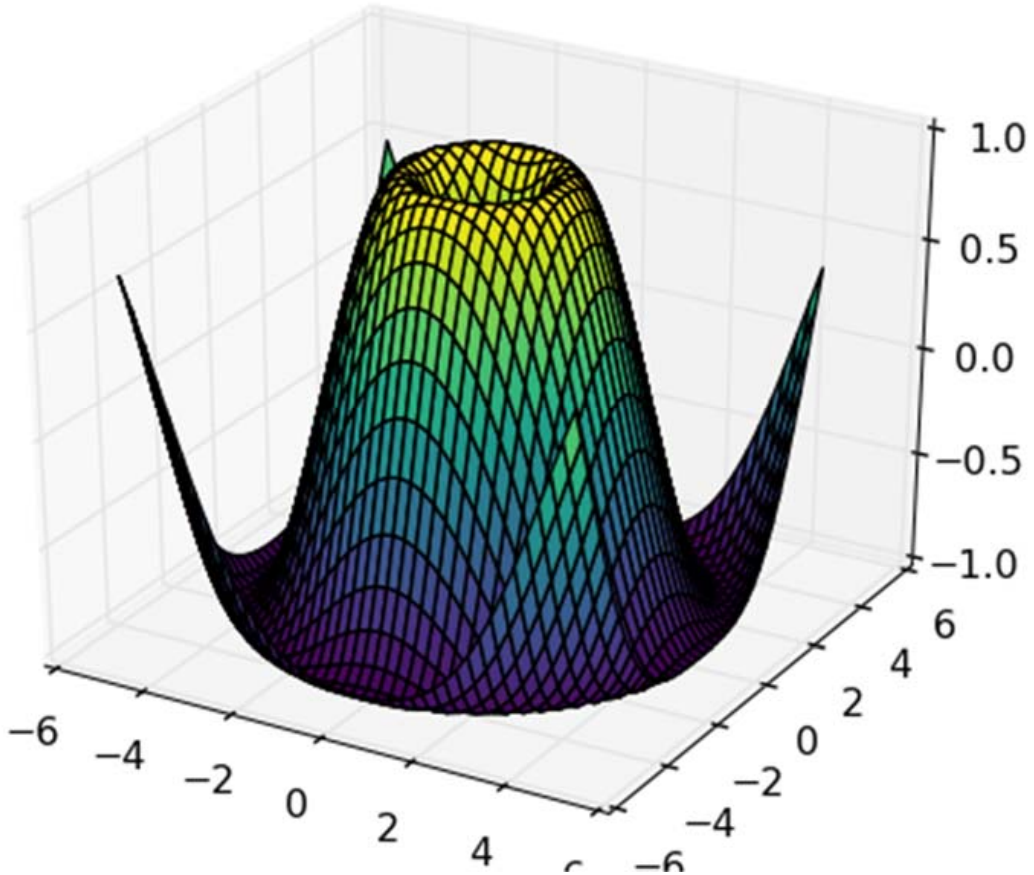
Example 3

```{python}

import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)
fig = plt.figure()
ax = Axes3D(fig)
ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.viridis)
plt.show()

```

Example 3



Sourcing Python Script

Scripting

We can source any Python script just as you would source an R script using the *source python*. Here **flights.py** is a Python script.

#Sourcing Scripts

```
source_python('add.py')
add(5,10)

## [1] 15

library(data.table)
library(curl)
mydat <-
fread('https://raw.githubusercontent.com/ismayc/pnwflights14/master/data/flig
hts.csv')
write.csv(mydat,file="MyData.csv")
head(mydat)

##    year month day dep_time dep_delay arr_time arr_delay carrier tailnum
## 1: 2014     1   1        1        96      235        70      AS  N508AS
## 2: 2014     1   1        4        -6      738       -23      US  N195UW
## 3: 2014     1   1        8        13      548        -4      UA  N37422
## 4: 2014     1   1       28        -2      800       -23      US  N547UW
## 5: 2014     1   1       34        44      325        43      AS  N762AS
## 6: 2014     1   1       37        82      747        88      DL  N806DN
##    flight origin dest air_time distance hour minute
## 1:    145    PDX  ANC      194     1542    0      1
## 2:   1830    SEA  CLT      252     2279    0      4
## 3:   1609    PDX  IAH      201     1825    0      8
## 4:    466    PDX  CLT      251     2282    0     28
## 5:    121    SEA  ANC      201     1448    0     34
## 6:   1823    SEA  DTW      224     1927    0     37
```

Sourcing Python Script contd...

```
source_python('flights.py')
flights<-read_flights("MyData.csv")
head(flights)

##    carrier dep_delay arr_delay
## 6       UA       227       219
## 17      AA         0       -19
## 20      UA        -3         7
## 22      UA        -3        -2
## 51      AA        -4       -28
## 86      AA       -10       -11
```

Python REPL

REPL

If you want to work with Python interactively you can call the repl python() function, which provides a Python REPL embedded within your R session. Objects created within the Python REPL can be accessed from R using the py object exported from reticulate.

> repl python()
>>>your code
>>>quit
> Return to R

RPy2: Calling R from Python

Description

RPy2 is an interface to R running embedded in a Python process.
The project is mature, stable, and widely used.

Installing

pip install rpy2 OR conda install rpy2
Python Version : 3.4> (It is compatible with 2.7 as well)
, R version 3.2+

** Set all the packages from R accessible from Python environment.

Simple coding

```
ro.r('x=c()')
ro.r('x[1]=22')
ro.r('x[2]=44')
print(ro.r('x'))
type(ro.r('x'))
```

```
[1] 22 44
rpy2.robjects.vectors.FloatVector
```

Example

```
import rpy2.rinterface
rpy2.rinterface.set_initoptions(('rpy2', '--verbose',

from rpy2.robjects.packages import importr
    ...: base = importr('base')
    ...: print(base._libPaths())

import rpy2.robjects as ro
ro.r(''' .libPaths('C:/Users/Mahdi/Anaconda2/Lib/R/libr
```

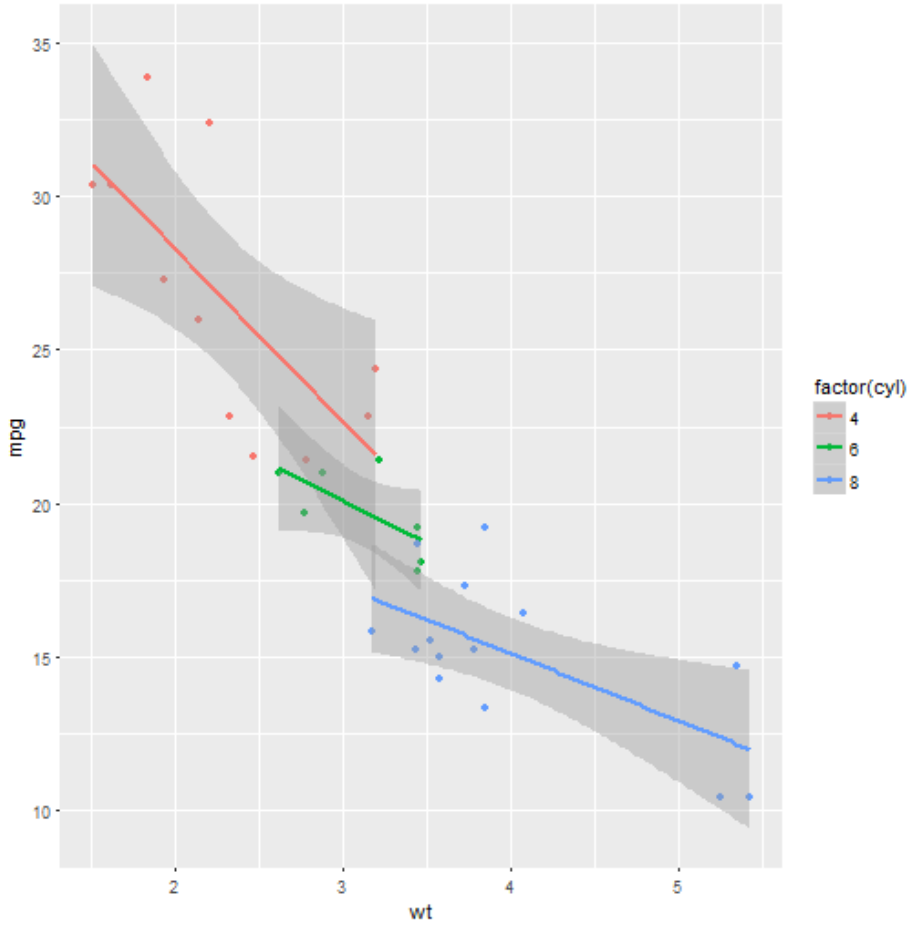
Example Contd

```
from numpy import *
import scipy as sp
from pandas import *
import rpy2.robjects as ro
import pandas.rpy.common as com
%matplotlib inline
import math, datetime
import rpy2.robjects.lib.ggplot2 as ggplot2
import rpy2.robjects as ro
from rpy2.robjects.packages import importr
base = importr('base')
datasets = importr('datasets')
```

Example Contd

```
grdevices = importr('grDevices')
grdevices.png(file="Rpy2ggplot2.png", width=512, heigh
#mtcars = datasets.data.fetch('mtcars')['mtcars']
mtcars=com.load_data('mtcars')
mtcars=com.convert_to_r_dataframe(mtcars)
pp = ggplot2.ggplot(mtcars) + \
     ggplot2.aes_string(x='wt', y='mpg', col='factor(c
     ggplot2.geom_point() + \
     ggplot2.geom_smooth(ggplot2.aes_string(group = 'c
pp.plot()
grdevices.dev_off()
```

Result



pyRserve

- pyRserve is a library for connecting Python to a R Process.
- In contract t rpy2 the R process does not have to run on the same machine.
- All data structures will automatically be converted from native R to native Python and numpy types and back.

Installation

library('Rserve') on R
pip install pyRserve on Python

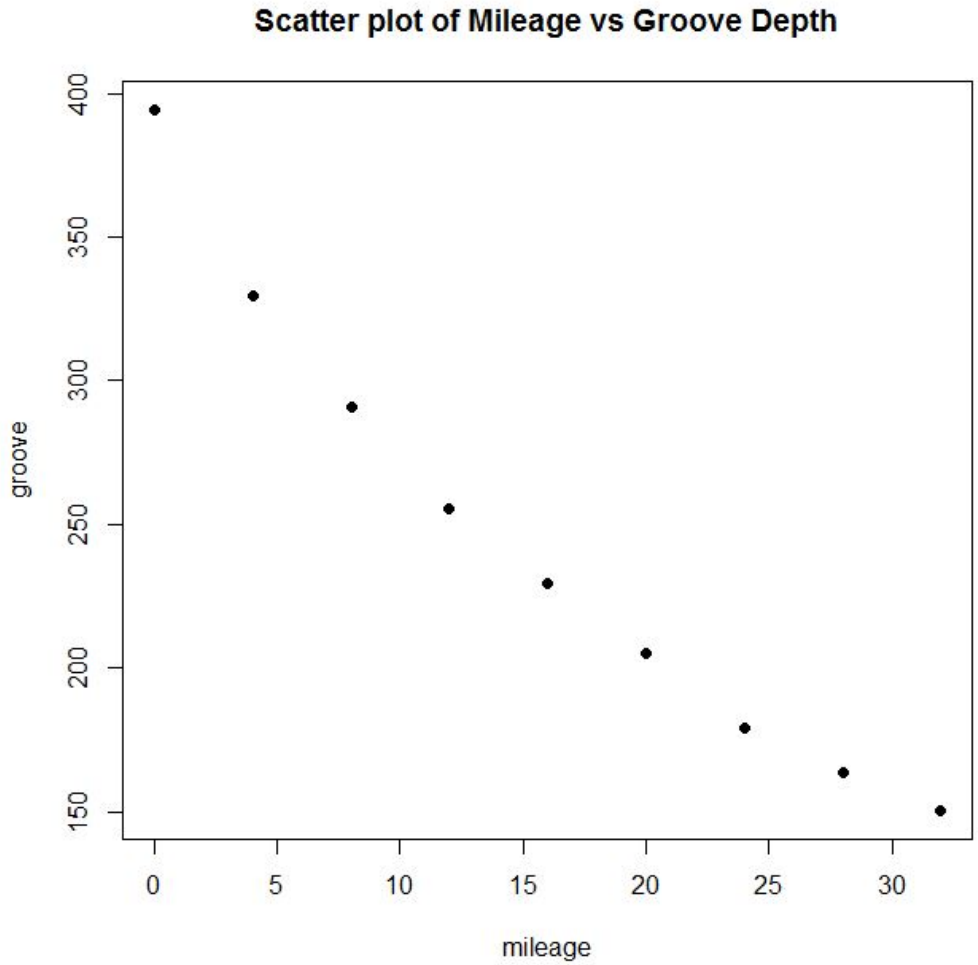
Example

```
import pyRserve
conn = pyRserve.connect(host='localhost')
       <Handle to Rserve on localhost:6311>
conn.voidEval('doubleit <- function(x) { x*2 }')
conn.eval('doubleit(2)')

my_r_script = '''
squareit <- function(x)
  { x**2 }
squareit(4)
'''
conn.eval(my_r_script)
```

Example

```
#import pyRserve
import pyRserve
#open pyRserve connection
conn = pyRserve.connect(host='localhost', port=6311)
#load your rscript into a variable (you can even write functions)
test_r_script = '''
mileage<-c(0,4,8,12,16,20,24,28,32)
groove<-c(394.33,329.5,291,255.17,229.33,204.83,179,163.83,150.33)
Tread<-data.frame(mileage,groove)
plot(Tread, pch=16 , main = "Scatter plot of Mileage vs Groove Depth")
                '''
#do the connection eval
conn.eval(test_r_script)
# closing the pyRserve connection
#conn.close()
```


Scatter plot of Mileage vs Groove Depth
groove
150
200
250
300
350
400
0
5
10
15
20
25
30
mileage

Example

Thank You