Reticulate

Victor Lopez

2023-01-11

Reticulate

- Los vectores de R se convierten a listas de Python
- Las listas de R se convierten a tuplas de Python
- Las name list de R se convierten a diccionarios de Python
- Matrices o arrays de R se convierten a numpy de Python
- Los df de R son df de Python
- Los factores se convierten a variables categoricas
- Lo demas se conserva

```
library(reticulate)
# use_python("/Users/VICTORWKEY/anaconda3/python.exe", required = TRUE)
np = import("numpy", convert = FALSE)
x = np$array(c(1, 2, 3))
sum = x$cumsum()

# En este caso lo va imprimir como array de numpy debido a que usamos convert = FALSE
sum
```

```
## array([1., 3., 6.])
```

```
# Convertimos de objeto Python a R
py_to_r(sum)
```

```
## [1] 1 3 6
```

Cargar funciones de un .py

```
# source_python("sum.py")
# add(4, 6)
```

Ayuda de Python en R

```
# Python
py_help(np$abs)
```

Mas metodos de Reticulate

```
a = np_array(c(1:10), dtype = "float16")
a
```

```
## array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.], dtype=float16)
```

R y Python combinados

```
# CODIGO DE R
dataframe = iris
head(dataframe)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
dataframe_py = r_to_py(dataframe)
```

```
# CODIGO DE PYTHON
import numpy as np
import pandas as pd

r.dataframe_py.head() # Accedemos a las variables que hayamos transformado en R
```

```
##    Sepal.Length  Sepal.Width  Petal.Length  Petal.Width Species
## 0           5.1          3.5           1.4          0.2  setosa
## 1           4.9          3.0           1.4          0.2  setosa
## 2           4.7          3.2           1.3          0.2  setosa
## 3           4.6          3.1           1.5          0.2  setosa
## 4           5.0          3.6           1.4          0.2  setosa
```

Otro ejemplo:

```
# CODIGO DE R
library(Matrix)
N = 6
```

```r
set.seed(123)
sparse_mat = sparseMatrix(
  i = sample(N, N, replace = F),
  j = sample(N, N, replace = F),
  x = runif(N),
  dims = c(N, N)
)
sparse_mat
```

```
## 6 x 6 sparse Matrix of class "dgCMatrix"
##
## [1,] .         .          0.8895393 .         .        .
## [2,] .         0.04205953 .         .         .        .
## [3,] .         .          .         .         0.899825 .
## [4,] .         .          .         .         .        0.3279207
## [5,] 0.9545036 .          .         .         .        .
## [6,] .         .          .         0.2460877 .        .
```

```r
sparse_mat_py = r_to_py(sparse_mat)
```

```r
# CODIGO DE PYTHON
r.sparse_mat_py
```

```
## <6x6 sparse matrix of type '<class 'numpy.float64'>'
##  with 6 stored elements in Compressed Sparse Column format>
```

```r
print(r.sparse_mat_py)

# Para mandar una variable de python a R no es necesario convertirla en el bloque de Python, solo en el
```

```
##   (4, 0) 0.9545036491472274
##   (1, 1) 0.04205953353084624
##   (0, 2) 0.8895393160637468
##   (5, 3) 0.24608773435465991
##   (2, 4) 0.8998249704018235
##   (3, 5) 0.3279207192827016
```

```r
# CODIGO DE R
py_to_r(sparse_mat_py) # Y de esta manera traemos una variable de python que anteriormente fue exportad
```

```
## 6 x 6 sparse Matrix of class "dgCMatrix"
##
## [1,] .         .          0.8895393 .         .        .
## [2,] .         0.04205953 .         .         .        .
## [3,] .         .          .         .         0.899825 .
## [4,] .         .          .         .         .        0.3279207
## [5,] 0.9545036 .          .         .         .        .
## [6,] .         .          .         0.2460877 .        .
```