Interface entre Python et R

Zhentao Li

13 avril 2016

Le langage R

- Le langage R est un langage beaucoup utlisé pour le calcul statistiques.

Le langage R

- Le langage R est un langage beaucoup utlisé pour le calcul statistiques.
- Comme `scipy`, ce langage manipule les valeurs vectorielles.

Le langage R

- Le langage R est un langage beaucoup utlisé pour le calcul statistiques.
- Comme `scipy`, ce langage manipule les valeurs vectorielles.
- Les commandes vus dans le cours de calcul scientifique peuvent être considérés comme des analogues de commandes dans R.

Opération	R	python (avec `import scipy as sp` en entête)
assignation	`<-` (ou `=`)	`=`
création de vecteur	`c(3, 4, 5)`	`sp.array([3, 4, 5])`
concaténation	`c(v, 6, 7)`	`sp.append(v, [6, 7])`
indexation	`v[2]`	`v[1]` (les indices commencent à 0 au lieu de 1)
suite	`1:10`	`sp.arange(1, 11)`
longueur	`length(v)`	`len(v)`
addition élément par élément	`v1 + v2`	`v1 + v2`
multiplication	`v1 * v2`	`v1 * v2`

Correspondance

création de matrices	`rbind(1:2, c(5,8))`	`sp.array([range(1, 3), sp.array([5, 8])])`
multiplication matricielle	`m1 %*% m2`	`m1.dot(m2)`
modulo (élément par élément)	`v %% 2`	`v % 2`
sélection par vecteur d'indices	`v1[v2]`	`v1[v2]`
comparisons	`v > 2`	`v > 2`
booléens	`TRUE; FALSE` (ou `T; F`)	`True; False`
valeur vide	`NULL`	`None`
moyenne	`mean(v)`	`sp.mean(v)`
déviation	`sd(v)`	`sp.std(v)`
liste R/dictionnaire python	`list(a=3, b=4)`	`dict(a=3, b=4)` (pas exactement équivalent)
élément d'une liste R	`l$a`	`l['a']`
élément d'une liste R	`l[[1]]`	`l.values()[0]` (ordre non préservé)

Correspondance de fonctions

R

```
compte_impairs <- function(v) {
    length(v[v %% 2 == 1])
}
```

Python

```
def compte_impairs(v):
    return len(v[v % 2 == 1])
```

Remarque (avancée): En fait, il y a une syntaxe que resemble encore plus à celui de R en Python: l'instruction `lambda` qui crée une fonction sans nom.

```
compte_impairs = lambda v: len(v[v % 2 == 1])
```

Interface entre R et Python

Il existe plusieurs modules pour pouvoir appeller les fonctions et échanger des données entre les deux langages. Les plus populaires sont

- rpy2
- pyRserver (avec Rserve du côté R)
- PypeR

Nous allons voir l'utilisation de `rpy2` qui est relativement simple.

rpy2

Il se trouve que la plupart des fonctionalités de `rpy2` sont dans le sous-module `robjects` et l'objet `r` de `rpy2.robjects`.

```
from rpy2 import robjects
from rpy2.robjects import r
```

Remarque: Les **objets** sont une notion que nous avons déjà vu lorsque nous avons parlé de `Class`.

Un exemple rpy2

Les constantes dans R sont accessible avec la syntaxe suivante.

```
>>> from rpy2.robjects import r
>>> r.pi
<FloatVector - Python:0x29d9998 / R:0x22e30f8>
[3.141593]
>>> r.pi[0]
3.141592653589793
>>> v = r.pi + 2
>>> v
<FloatVector - Python:0x29d9c68 / R:0x1657398>
[3.141593, 2.000000]
>>> r.sum(v)
<FloatVector - Python:0x29dba70 / R:0x22ed868>
[5.141593]
>>> # On peut aussi appeler des fonctions python sur les vecteurs R
>>> sum(v)
5.141592653589793
```

Création d'objets R

Pour créer un vecteur, nous pouvons utilier `c()` de R.

```
>>> r.c(1, 2, 3)
<IntVector - Python:0x29dccf8 / R:0x1657520>
[       1,        2,        3]
>>> r.c(1.0, 2, 3)
<FloatVector - Python:0x29dcbd8 / R:0x27c8ac0>
[1.000000, 2.000000, 3.000000]
```

Ou en faissant appel à `robjects.IntVector`, `robjects.FloatVector`, `robjects.StrVector`, etc.

```
>>> robjects.IntVector([1,2,3])
<IntVector - Python:0x29dd9e0 / R:0x1657638>
[       1,        2,        3]
>>> robjects.FloatVector([1,2,3])
<FloatVector - Python:0x29dd488 / R:0x27c8a30>
[1.000000, 2.000000, 3.000000]
>>> robjects.StrVector([1,2,3])
<StrVector - Python:0x29dd170 / R:0x1f0dda8>
['1', '2', '3']
```

Interface avec scipy (ou numpy)

Il est possible de transformer les vecteurs `numpy/scipy` en vecteurs R et vice versa.

```
>>> v1 = r.c(3,5,7)
>>> v2 = sp.array(v1)
>>> v2
array([3, 5, 7], dtype=int32)
>>> v3 = v2 * 2
>>> v4 = robjects.IntVector(v3)
>>> v4
<IntVector - Python:0x2ea8440 / R:0x16578d8>
[       6,       10,       14]
```

Exécution directe

Il est possible d'exécuter une (ou plusieurs) ligne(s) de R directement.

```
>>> v = r('1:10')
>>> v
<IntVector - Python:0x2979830 / R:0x1d49fe0>
[       1,        2,        3, ...,        8,        9,
>>> r('sum(1:10)')
<IntVector - Python:0x29dda70 / R:0x22f0448>
[      55]
```

Nous appelons `rpy2.robjects.r` comme une fonction avec un seul paramètre: une chaîne de caractères. Cette chaîne a comme valeur le code R que nous voulons exécuter.

Importation de paquets R

```
>>> from rpy2.robjects.packages import importr
>>> datasets = importr('datasets')
>>> datasets.occupationalStatus
<Matrix - Python:0x3d78128 / R:0x3e10060>
[      50,       16,       12, ...,      177,       71,
```