RPythonPlus : an interface between R and Python for datascientists

Author : Pierre GUYOMARD, PhD candidate in Computer Science
Contact : piguyomard@outlook.com

English Version

Introduction

Python[1] is an object-oriented, interpreted, cross-platform language. It is particularly suitable for use in the field of exact sciences or finance. For example, data scientists (*data scientists* and *data analysts*) make common use of it. The field of artificial intelligence, thanks to its specialized libraries, also makes great use of it. Finally, there are few fields that do not use Python and, thanks to its readability, it is a language of educational utility.

R[2] is a language particularly suited to descriptive and inferential statistics. It has natively, unlike *Python*, mathematical and graphical functionalities, making it easy to use for a user who is not an expert in programming, insofar as its syntax is different from that of object-oriented languages. Moreover, *R* is a vector language speeding calculations on elements made up of repetitions of identical objects.

The complementary use of the two languages according to the type of computer processing presented a certain interest reported by many scientists. Thus, *rpy2*[3] is a Python library allowing the use of R objects, functions, methods and libraries in Python while respecting the usual syntax of the latter. Despite this, it is impossible to set up computer processing requiring the consecutive use of the two languages via *rpy2* . It is for example impossible to open a *dataframe* with Pandas, to carry out easy processing using Python before analyzing it with the *R package* mixOmics[4]. Therefore, based on the *rpy2* module, a high-level interface for converting core *Python objects to and from their R* counterparts has been implemented.

Materials and methods

Conversion process

Insofar as complex objects (python-dictionaries, r-lists, python-dataframes...) are most of the time decomposable into successions of native objects (integers, floats, character strings ...) and that *rpy2* is based on the native Python types, an iterative structure would have sufficed to implement a functional conversion; However, for large objects, such a process would have taken a long time. This is why a parallelized implementation has been set up with the Python *multiprocessing library*[5]. Thus, the general operation of this interface consists in segmenting a complex object into several parts, in converting each of the sub-parts in parallel and then in assembling these sub-parts in order; the conversion methods then being specific to each object passed as input.

Conversions available

Each of the interface conversions is bidirectional . In this way, the *dataframes Pandas*[6], Python's library, maps to R *dataframes* while R lists map to Python dictionaries. Python lists and tuples (sometimes called vectors) correspond to R vectors. Finally, *Numpy arrays*[7] (Python library) have matrices as their R equivalent. All of this information is summarized in Table 1. The instantiation methods are also explained there in each of the languages in order to remove any ambiguity.

Python		R	
Object name	**Instantiation**	**Object name**	**Instantiation**
Pandas Dataframes	pandas.DataFrame ()	Dataframes	data.frame()
Dictionaries	dict()	Lists	list()
Lists, tuples	list(), tuple()	Vectors	c()
Numpy array	numpy.array()	Dies	matrix()

Table 1: correspondences between objects allowed by RPythonPlus

Interface implementation

The conversion interface was developed under Python 3.7.11 and R 4.0.5 using rpy2 3.4.5 under Ubuntu 20.04 64 bit. An R dependency, called rpythonplus_ *toolbox* , has been developed for the operation of the interface and contains functions necessary for the operability of these.

Commonly used in analysis and data science, the Pandas version 1.1.3 and NumPy 1.21.6 libraries have been included in the conversion interface. Other Python libraries such as *math* (version unavailable), *re* version 2.2.1 and *gc* (version unavailable) were used.

All of the RPythonPlus code and the corresponding dependencies are available at the following address: https://github.com/pierre-guyomard/RPythonPlus

Version Française

Introduction

Python[1] est un langage orienté objet, interprété et multiplateforme. Il est particulièrement adapté à une utilisation dans le domaine des sciences exactes ou financier. Par exemple, les scientifiques des données (*data scientists* et *data analysts*) en font un usage courant. Le domaine de l'intelligence artificielle, grâce à ses librairies spécialisées, en fait également un grand emploi. Il existe finalement peu de domaines qui n'utilisent pas Python et, grâce à sa lisibilité, c'est un langage d'utilité pédagogique.

R[2] est un langage particulièrement adapté aux statistiques descriptives et inférentielles. Il dispose nativement, à la différence de *Python*, de fonctionnalités mathématiques et graphiques, rendant l'emploi simple pour un utilisateur non expert en programmation, dans la mesure où sa syntaxe est différente de celle des langages orientés objet. De plus, *R* est un langage vectoriel accélérant les calculs sur des éléments constitués de répétitions d'objets identiques.

L'utilisation complémentaire des deux langages selon le type de traitement informatique présenté un intérêt certain rapporté par de nombreux scientifiques. Ainsi, *rpy2*[3] est une librairie Python permettant d'utiliser les objets, fonctions, méthodes et librairies de R dans Python tout en respectant la syntaxe usuelle de ce dernier. Malgré ceci, il est impossible de mettre en place un traitement informatique nécessitant l'utilisation consécutive des deux langages via *rpy2*. Il est par exemple impossible d'ouvrir un *dataframe* avec Pandas, de réaliser des traitements aisés à l'aide de Python avant de l'analyser avec le package *R* mix*Omics*[4]. C'est pourquoi, basée sur le module *rpy2*, une interface haut niveau de conversion des principaux objets *Python* vers et depuis leurs équivalents *R* a été implémentée.

---

[1] VAN ROSSUM, Guido et DRAKE JR, Fred L. The python language reference. *Python software foundation*, 2014.
[2] Ross IHAKA et Robert GENTLEMAN, « R: A Language for Data Analysis and Graphics », *Journal of Computational and Graphical Statistics* 5, nº 3 (septembre 1996): 299-314, https://doi.org/10.1080/10618600.1996.10474713.
[3] https://rpy2.github.io/doc/v3.5.x/html/index.html - 15.08.2022
[4] Florian ROHART *et al.*, « MixOmics: An R Package for 'omics Feature Selection and Multiple Data Integration », éd. par Dina Schneidman, *PLOS Computational Biology* 13, nº 11 (3 novembre 2017): e1005752, https://doi.org/10.1371/journal.pcbi.1005752.

Matériels et méthodes

Processus de conversion

Dans la mesure où les objets complexes (python-dictionnaires, r-listes, python-dataframes…) sont la plupart du temps décomposables en successions d'objets natifs (entiers, flottants, chaînes de caractères…) et que *rpy2* se base sur les types natifs de Python, une structure itérative aurait suffi à implémenter une conversion fonctionnelle ; Toutefois, pour des objets de grande taille, un tel processus aurait été long à réaliser. C'est pourquoi, une implémentation parallélisée a été mise en place avec la librairie Python *multiprocessing*[5]. Ainsi, le fonctionnement général de la présente interface consiste à segmenter un objet complexe en plusieurs parties, à convertir en parallèle chacune des sous-parties puis à rassembler dans l'ordre ces sous-parties ; les modalités de conversions étant ensuite propre à chaque objet passé en entrée.

Conversions disponibles

Chacune des conversions de l'interface est bidirectionnelle. De cette façon, les *dataframes Pandas*[6], librairie de Python, entrent en correspondance avec les *dataframes* R tandis que les listes R correspondent aux dictionnaires Python. Les listes et les tuples (parfois appelés vecteurs) Python correspondent quant à aux vecteurs R. Enfin, les tableaux *Numpy*[7] (librairie Python) ont pour équivalent R les matrices. L'ensemble des ces informations est résumé dans le tableau 1. Les méthodes d'instanciation y sont également explicitées dans chacun des langages afin de lever toute ambiguïté.

Python		**R**	
Nom de l'objet	**Instanciation**	**Nom de l'objet**	**Instanciation**
Dataframes Pandas	pandas.DataFrame()	Dataframes	data.frame()
Dictionnaires	dict()	Listes	list()
Listes, tuples	list(), tuple()	Vecteurs	c()
Tableau Numpy	numpy.array()	Matrices	matrix()

Tableau 1 : correspondances entre objets permises par RPythonPlus

Implémentation de l'interface

L'interface de conversion a été développée sous Python 3.7.11 et R 4.0.5 à l'aide de rpy2 3.4.5 sous Ubuntu 20.04 64 bits. Une dépendance R, appelée rpythonplus_*toolbox*, a été développée pour le fonctionnement de l'interface et contient des fonctions nécessaires à l'opérabilité de celles-ci.

Couramment utilisées en analyse et science des données, les librairies Pandas version 1.1.3 et NumPy 1.21.6 ont été incluses dans l'interface de conversion. D'autres librairies Python telles que *math* (version indisponible), *re* version 2.2.1 et *gc* (version indisponible) ont été utilisées.

L'ensemble du code de RPythonPlus ainsi que les dépendances correspondantes sont disponibles à l'adresse suivante : https://github.com/pierre-guyomard/RPythonPlus

[5] https://github.com/python/cpython/tree/3.10/Lib/multiprocessing/ - 15.08.2022
[6] Wes MCKINNEY, « Pandas: A Foundational Python Library for Data Analysis and Statistics », *PyHPC 2011 : Python for High Performance and Scientific Computing*, 2011, 9.
[7] Charles R. HARRIS *et al.*, « Array Programming with NumPy », *Nature* 585, nº 7825 (17 septembre 2020): 357-62, https://doi.org/10.1038/s41586-020-2649-2.