`Pybind11/reticulate` comme alternative à `Rcpp`

François-David Collin[1]

[1]IMAG Montpellier

Résumé

Selon des enquêtes menées auprès des développeurs, le langage C++ serait le langage « bas-niveau » et haute performance le plus utilisé actuellement. Il est donc naturel de vouloir utiliser ce langage pour développer en `R` en l'optimisant avec des fonctions en C++, et l'objectif d'un package comme `Rcpp` est d'en faciliter l'intégration. Dans l'écosystème Python, une démarche similaire est à l'origine de `pybind11` pour créer des modules Python à partir de code C++. Pour la même base de code C++, maintenir deux interfaces différentes pour `Rcpp` et `pybind11` constitue une charge de travail importante, nous proposons une alternative fonctionnelle qui partirait uniquement de Python/`pybind11` grâce à l'utilisation `reticulate`, qui permet d'intégrer directement des modules Python depuis R.

Introduction

Dans le cadre d'un développement méthodologique précis, `abcranger` [1], exclusivement développé en C++, un module Python a été développé grâce à `pybind11` [2]. Ensuite, pour des besoins spécifiques, une interface en `R` avec `Rcpp` [3] a été ébauchée, mais le développement a été abandonné en raison de problèmes variés :

- système de *build* hétérogène entre `Rcpp` et `pybind11` (`Rcpp` basé sur `makefile`, `pybind11` sur `cmake`)
- la partie C++ utilisant extensivement `Eigen` [4], gestion conflictuelle des dépendances C++ avec `RcppEigen` imposant une version obsolète d'`Eigen`
- Impossibilité d'éviter une copie mémoire des données entre `R` et C++ (toujours à cause d'`Eigen`)

Exemple minimal Python/pybind11

Nous illustrerons notre propos avec un exemple simple avec trois composantes essentielles :

- un `CMakelist.txt` pour la compilation du module Python utilisant `pybind11`
- un fichier C++ contenant la fonction à exposer
- un fichier Python utilisant le module Python

Nous toucherons aussi un mot de la prise en charge poussée du C++ dit « moderne » par `pybind11`, qui permet de bénéficier de l'utilisation de librairies écrites en C++ moderne, comme `Eigen` [4], avec quelques considérations générales sur l'évolution du C++ et de sa communauté [5], [6]

Utilisation avec reticulate

Nous présenterons brièvement le package reticulate, et l'utilisation directe de modules Python depuis R comme `numpy` [7]. Ensuite, nous montrerons comment importer le module Python créé précédemment, et l'utiliser depuis R.

Conclusion

Sans prétendre à l'exhaustivité, l'utilisation de `pybind11` pour développer des modules Python permet de créer une interface R à partir de code C++ et d'une interface Python/`pybind11` déjà existant, et ce, sans avoir à maintenir deux interfaces R et Python.

Références

[1] F.-D. Collin, A. Estoup, J.-M. Marin, et L. Raynal, « Bringing ABC inference to the machine learning realm: AbcRanger, an optimized random forests library for ABC », in *JOBIM 2020*, 2020, vol. 2020, p. 66.

[2] W. Jakob, J. Rhinelander, et D. Moldovan, « pybind11–Seamless operability between C++ 11 and Python », *URL: https://github.com/pybind/pybind11*, 2017.

[3] D. Eddelbuettel et R. François, « Rcpp: Seamless R and C++ Integration », *Journal of Statistical Software*, vol. 40, n 8, p. 1-18, 2011, doi: 10.18637/jss.v040.i08.

[4] G. Guennebaud, B. Jacob, *et al.*, « Eigen v3 ». http://eigen.tuxfamily.org, 2010.

[5] « Stack Overflow Developer Survey 2022 », *Stack Overflow*. https://survey.stackoverflow.co/2022/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2022 (consulté le 23 mars 2023).

[6] « GitHub Language Stats ». https://madnight.github.io/githut/ (consulté le 23 mars 2023).

[7] C. R. Harris *et al.*, « Array programming with NumPy », *Nature*, vol. 585, n 7825, p. 357-362, sept. 2020, doi: 10.1038/s41586-020-2649-2.