A11

Nipype interfaces in CBRAIN

Tristan Glatard[1,2], Samir Das[1], Reza Adalat[1], Natacha Beck[1], Rémi Bernard[1], Najmeh Khalili-Mahani[1], Pierre Rioux[1], Marc-Étienne Rousseau[1], Alan C. Evans[1]
[1]McGill Centre for Integrative Neuroscience (MCIN), Ludmer Centre for Neuroinformatics and Mental Health, Montreal Neurological Institute (MNI), McGill University, Montréal, Québec, Canada; [2]University of Lyon, CNRS, INSERM, CREATIS., Villeurbanne, France
Correspondence: Tristan Glatard (tristan.glatard@mcgill.ca) – McGill Centre for Integrative Neuroscience (MCIN), Ludmer Centre for Neuroinformatics and Mental Health, Montreal Neurological Institute (MNI), McGill University, Montréal, Québec, Canada
GigaScience 2016, **5(Suppl 1):**A11

Introduction

We aim at the large-scale, automatic sharing of software tools between neuroimaging processing platforms, which will increase the relevance of such platforms by providing them with richer repositories of higher-quality tools. Currently, efforts are hampered by the repetitive porting of the same few tools in different platforms. During the HBM 2015 Hackathon, we worked on the export of software tools from the Nipype workflow engine [1] to the CBRAIN web platform for distributed computing [2]. Nipype includes a large number of tools that would be useful to CBRAIN users.

Approach

We developed nipype2boutiques, a tool to export Nipype interfaces to the "Boutiques" tool description format (step 1. on Fig. 10.). Boutiques descriptions are importable by CBRAIN and other platforms (Virtual Imaging Platform [3] and the Pegasus workflow engine [4]). They point to a Docker image containing the implementation of the tool. nipype2boutiques relies on nipype_cmd a tool to run Nipype Interfaces as Linux command lines. nipype2boutiques parses the inputs and outputs of a Nipype interface and extracts their name, type, description and position on the nipype_cmd command line. nipype2boutiques then generates a Boutiques descriptor pointing to a Docker image where the Nipype interface is available. Once a Nipype interface is exported using nipype2boutiques it can be imported to CBRAIN.

Results

We tested nipype2boutiques on a few Nipype interfaces from the FSL Nipype module. We exported 64 FSL tools automatically from Nipype to CBRAIN, and made them available [https://github.com/glatard/boutiques-nipype-fsl]. Limitations remain on the type of Nipype interface that can be exported by nipype2boutiques: in particular, InputMultiPath is? currently not supported, and output files have to be written in the execution directory of the Nipype Interface.

Conclusions

We prototyped a software tool to export Nipype Interfaces as Boutiques descriptors, which can be imported by CBRAIN and other platforms. Although the solution is still limited to simple interfaces, we believe that it has the potential to enable fully automatic tool sharing between Nipype and CBRAIN. Future extensions of nipype2boutiques will be published in the Nipype Github repository [https://github.com/nipy/nipype]. We also plan on a tighter integration of Nipype workflows in CBRAIN, following the model adopted in [5].

Availability of Supporting Data

More information about this project can be found at: http://cbrain.mcgill.ca.

Competing interests

None.

Author's contributions

TG wrote the software and the report; SD contributed to the concept elaboration at the OHBM event, RA, NB, PR and MER provided support on the CBRAIN framework, RB implemented Boutiques in CBRAIN, NKM provided background information on fMRI packages, ACE spearheaded the project.

Acknowledgements

Report from 2015 OHBM Hackathon (HI). The authors would like to thank the organizers and attendees of the 2015 OHBM Hackathon.

References

1. Gorgolewski Krzysztof, Burns Christopher D, Madison Cindee, Clark Dav, Halchenko Yaroslav O, Waskom Michael L, Ghosh Satrajit S. Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in Python. Frontiers in Neuroinformatics. 2011; 5.
2. Sherif Tarek, Rioux Pierre, Rousseau Marc-Etienne, Kassis Nicolas, Beck Natacha, Adalat Reza, Das Samir, Glatard Tristan, Evans Alan C. CBRAIN: a web-based, distributed computing platform for collaborative neuroimaging research. Frontiers in neuroinformatics. 2014; 8.
3. Glatard T, Lartizien C, BGibaud, Ferreira da Silva R, Forestier G, Cervenansky F, Aless,rini M, Benoit-Cattin H, Bernard O, Camarasu-Pop S, Cerezo N, Clarysse P, Gaignard A, Hugonnard P, Liebgott H, Marache S, Marion A, Montagnat J, Tabary J, Friboulet D. A Virtual Imaging Platform for multi-modality medical image simulation. IEEE Transactions on Medical Imaging. 2013; 32: 110–118.
4. Deelman E, Vahi K, Rynge M, Juve G, Mayani R, da Silva RF. Pegasus in the Cloud: Science Automation through Workflow Technologies. Internet Computing, IEEE. 2016; 20: 70–76.
5. Glatard T, Quirion PO, Adalat R, Beck N, Bernard R, Caron BL, Nguyen Q, Rioux P, Rousseau M-E, Evans AC, Bellec P. Integration between PSOM and CBRAIN for distributed execution of neuroimaging pipelines. In: Meeting of the Organization for Human Brain Mapping, Geneva, Switzerland, OHBM 2016, Geneva, 2016.

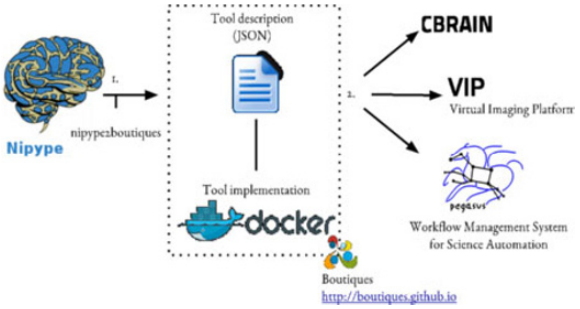

Fig. 10 (abstract A11). System architecture

A12

DueCredit: automated collection of citations for software, methods, and data

Yaroslav O. Halchenko, Matteo Visconti di Oleggio Castello
Department of Pscyhological & Brain Sciences, Dartmouth College, Hanover, NH, USA
Correspondence: Yaroslav O. Halchenko (yoh@onerussian.com) – Department of Pscyhological & Brain Sciences, Dartmouth College, Hanover, NH, USA
GigaScience 2016, **5(Suppl 1):**A12

Introduction

Data analysis software and canonical datasets are the driving force behind many fields of empirical sciences. Despite being of paramount importance, those resources are most often not adequately cited. Although some can consider this a "social" problem, its roots are technical: Users of those resources often are simply not aware of the underlying computational libraries and methods they have been using in their research projects. This in-turn fosters inefficient practices that encourage the development of new projects, instead of contributing to existing established ones. Some projects (e.g. FSL [1]) facilitate citation of the utilized methods, but such efforts are not uniform, and the output is rarely in commonly used citation formats (e.g. BibTeX). DueCredit is a simple framework to embed information about publications or other references within the original code or dataset descriptors. References are automatically reported to the user whenever a given functionality or dataset is being used.

Approach

DueCredit is currently available for Python, but we envision extending support to other frameworks (e.g., Matlab, R). Until DueCredit