Neuroconductor: A Framework for Reproducible Neuroimaging Analysis in R

John Muschelli*[,a], Jean-Philippe Fortin[b], Adrian Gherman[a], Brian S. Caffo[a], Ciprian M. Crainiceanu[a]

[a]*Johns Hopkins Bloomberg School of Public Health, Department of Biostatistics, 615 N Wolfe St, Baltimore, MD, 21205*

[b]*Perelman School of Medicine, University of Pennsylvania, Department of Biostatistics and Epidemiology, 423 Guardian Drive, Philadelphia, PA 19104*

Abstract

The Neuroconductor project is an initiative for the collaborative creation of extensible software for computational imaging analysis, with a focus on neuroimaging. The goals of the project include: integrate fast and collaborative development of tested software, increasing reproducibility in analyses of imaging data, and promoting the achievement of remote reproducibility of research results. We describe details of our goals, identify the current implementation and current challenges, and provide examples of existing software and how they would integrate into the Neuroconductor system.

Introduction

Neuroimaging research has been increasing in popularity. There are X number of studies in fMRI and XX in other structural imaging. Both NeuroImage and Human Brain Mapping journals, two key journals for neuroimaging methodology and analysis, are dominated by analyses that used (typically *nix) command-line software, such as FSL or AFNI or custom C, C++ or python software, ran in Matlab via SPM or custom Matlab scripts, or used pipelining software, such as NiPype or LONI (Carp 2012). The heterogeneous nature of neuroimaging software has led to a difficult circumstance for reproducing analyses across and often hinders entrance into the community. Efforts such as NiPype, SPM and related tools, have made great strides, since python and Matlab are such popular scripting languages for analysis, and have robust implementations across all popular opperating systems and environments. Creating reproducible python analyses and processing has notably improved, in particular, with the introduction of Jupyter IPython notebooks.

The R scripting language has much to offer as a potential environment for neuroimaging data analysis *CITE ANI'S ARTICLE AND THE NEUROIMAGE ARTICLES ABOUT R*. Like python, R is open source, free and robustly implemented across every major computing environment and OS. With the growth of complex neuroimaging data, sophisticated statistical methods and machine learning methods are necessary, which R can offer. It also has a very large and growing user base *NUMBERS* as well a highly developed and motivated open source developer base, creating an enormous collection of statistics and machine learning packages. The Comprehensive R Archive Network (CRAN) has over 9,000 packages on CRAN and Bioconductor has 1295 software packages. Recent advances in R have made multi-threading, GPU processing and parallel processing trivial. Moreover, with the introduction of the knitr package and RMarkdown format, R has one of the most robust tools for reproducible research. In addition, interactive web application development is now possible (and easy) with Shiny. Finally, the RStudio environment gives a powerful and popular IDE for R. (http://r4stats.com/articles/popularity/)

R has a strong package authoring and management system, which has a large system of checks to ensure interoperability inspiring. The Comprehensive R Archive Network (CRAN) is a primary repository for R packages. In addition, domain specific repositories, notably Bioconductor, also host a large number of packages. A legion of developers, coupled with dedicated repository maintainers and mirrors and the amazing growth of packages with

*Corresponding Author
Email addresses: jmusche1@jhu.edu (John Muschelli), fortin946@gmail.com (Jean-Philippe Fortin), adig@jhu.edu (Adrian Gherman), bcaffo@jhsph.edu (Brian S. Caffo), ccraini1@jhu.edu (Ciprian M. Crainiceanu)

over 9,000 package at last count on CRAN alone. CRAN maintains “task views” to list packages relevant for specific domains. The medical imaging task view lists 27 packages for medical image analysis.

CRAN has numerous limitations as a community repository for a domain, such as medical or neuro imaging. Some of the key shortcomings to be addressed include: stricter package requirements in the form of documentation and style, version control integration, development versions of packages, continuous integration and lower repository maintainer requirements / involvement. In computational biology, the Bioconductor repository was created to address some of these shortcomings. However, since the creation of CRAN and Bioconductor, several new developments have occurred to inspire the creation of a package repository specific to Neuroimaging. An influential development is the overwhelming use of git and github for version control and package development. In addition, continuous integration tools, like travis.ci allow package building in the cloud.

Goals of Neuroconductor

1. Provide a unified neuroimaging framework for R packages
2. Provide developers an infrastructure to test package with external dependencies
3. Expand the statistical tools available to neuroimaging analysis
4. Make neuroimaging analysis accessible by providing clear workflows for a broad audience of users

Why R?

3. Leverage R resources/plotting/reproducibility/package development
4. Leverage Bioconductor resources - they have similar problems, years of testing, and code
5. Bayesian Statistics

Additional Checks

Third Party Software

- FSL
- AFNI
- FREESURFER
- SPM

We will refer to R-Forge, OmegaHat, Bioconductor, and CRAN as standard repositories.

Devtools

In YEAR, Hadley and RStudio had published the `devtools` package. The `devtools` package provided the tools to install R packages from a multitude of sources. The Neuroconductor relies on the installation script for R packages on GitHub. Moreover, it allowed for the introduction of a flag in the installation of an R package (the `Remotes:` field) that allowed users specify a dependency for the package that can be located on a source that is not a standard repository. Previous to this, if a package depended on a package that was not in a standard repository, the user would have to manually install that dependency before installing the package in question.

In addition to installing packages from GitHub, there are additional options to the installer scripts, which allow users to install specific snapshots of these packages. These snapshots can be based on GitHub commit identifiers (IDs), tags, or referrences.

The `remotes` package provides a lightweight version of the `devtools` package for installing from non-standard and standard sources.

In addition to the install functions, the `devtools` package allows for a up-to-date R package development system. The RStudio IDE integrates the `devtools` package so that R package development can be done in a more standardized way.

GitHub

GitHub APi. Git vs. svn.

TRAVIS

Seperately from any development in the R community, continuous integration (CI) services have become largely available that allows for automated building and checking of software. The Travis CI is a "hosted, distributed continuous integration service used to build and test software projects hosted at GitHub". In conjuction with the R community, Travis CI has configured the ability to seamlessly check R packages on multiple systems. In addition to the GitHub API described above, the Travis CI API provides an automated system for checking R package installation.

DRAT

The `drat` (Drat R Archive Template) package has provided a template to set up a repository similar to CRAN mirrors and other standard sources. One of the large benefits of using a drat repository is that users can use the default way of installing packages in R (`install.packages`) versus that from `devtools` (e.g. `install_github`) and requires no additional dependencies such as `devtools` or `remotes`.

One good example of a drat repository is ROpenSci, which has a series of packages that are based on GitHub. The only additional step for installing from a drat repository is to specify the `repos` argument in `install.packages`: `install.packages("package_name", repos="http://path/to/drat/repo")`.

Bioconductor

In 2004, the Bioconductor system enabled the bioinformatics and genomics work of R users to be more integrated and systemized (Gentleman et al. 2004).

Data Packages

Like Bioconductor, we need data packages that allow users to test software and examples on.

References

Carp, Joshua. 2012. "The Secret Lives of Experiments: Methods Reporting in the fMRI Literature." *NeuroImage* 63 (1): 289–300. doi:10.1016/j.neuroimage.2012.07.004.

Gentleman, Robert C, Vincent J Carey, Douglas M Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, et al. 2004. "Bioconductor: Open Software Development for Computational Biology and Bioinformatics." *Genome Biology* 5 (10). BioMed Central: 1.