

The Neuroconductor project

John Muschelli^{*,a}, 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*

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. **The Neuroconductor project offers an easy and unified framework in R for the preprocessing and advanced statistical analysis of imaging data.**

Introduction

Neuroimaging research has been increasing in popularity. There are **XX** number of studies in fMRI and **XX** in other structural imaging. **Talk about big projects like ADNI, ABIDE, 1000 Functional connectomes, etc.** The NeuroImage journal, one of the most prestigious (not right word) journals for neuroimaging is dominated by analyses that were in bash using third party software, NiPype, or MATLAB. Python reproducibility has increased with the introduction of Jupyter IPython notebooks. Much of this development has mirrored the tools in R, which have some of the most state-of-the-art reproducibility tools. Also, similar to PYthon, R is free and open source.

- Neuroimaging work more common
- Reproducibility problems
 - R has some of the best reproducible tools

R is increasing in popularity, with there being over 9,000 package now on the Comprehensive R Archive Network (CRAN). R has a strong package system, which has a large system of checks to ensure a large amount of operability. Although CRAN has a tested and stable system, that has been developed over the last 1000 years(?), there are some aspects of the CRAN checking system that does not work for neuroimaging packages.

Several of the statistical challenges encountered in the analysis of imaging data are shared with the field of genomics. An example of a statistical problem common to both imaging and genomic data is the presence of between-subject variability that is technical in nature, for instance the well-known batch effects (CITE) seen in genomics, due to different preprocessing dates, similar to between-scanner variability seen in large multi-site imaging studies (CITE). I want to say something like the Neuroconductor project will allow to integrate well-developed and robust statistical methods for handling such challenges, and it will make easy to adapt genomic tools to the world of brain imaging.

Talk about ComBat, SVA, and limma; maybe mention how popular they are by number of citations? Wondering if we could have one showcase section where we show that just using the Empirical Bayes function in limma automatically improves the statistical analysis of brain imaging data; that would sell Neuroconductor big deal.

*Corresponding Author

Email addresses: jmusche1@jhu.edu (John Muschelli), adig@jhu.edu (Adrian Gherman), bcaffo@jhsp.h.edu (Brian S. Caffo), ccraini1@jhu.edu (Ciprian M. Crainiceanu)

Other note: mention the cluster PNAS paper; we want to avoid such situations in the future; we need clear, reproducible and well-implemented statistical methods, and R is the perfect platform for that. We need to think: what can we do better in R that no other language/software can offer?

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 references.

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 conjunction 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")`.

Data packages and atlases

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

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.

List of Tables

| | | |
|---|--|---|
| 1 | Initial Neuroconductor packages. | 5 |
|---|--|---|

Table 1: **Initial Neuroconductor packages.**

| Package | Description | Popular functions and applications | Reference |
|--------------------------|---|------------------------------------|-----------|
| <code>oro.nifti</code> | | | |
| <code>freesurfer</code> | | | |
| <code>fslr</code> | | | |
| <code>brainR</code> | | | |
| <code>papayar</code> | | | |
| <code>ANTsR</code> | R Package for ANTS | | |
| <code>drammsr</code> | R Package for DRAMMS | DRAMMS registration, RAVENS maps | |
| <code>WhiteStripe</code> | Intensity normalization with White Stripe | | |
| <code>RAVEL</code> | Stratified quantile normalization | | |
| <code>oasisr</code> | | | |
| <code>shrinkR</code> | | | |