A practical guide for improving transparency and reproducibility in neuroimaging research

Krzysztof J. Gorgolewski and Russell A. Poldrack
Department of Psychology, Stanford University

Abstract

Recent years have seen an increase in alarming signals regarding the lack of replicability in neuroscience, psychology, and other related fields. To avoid a widespread crisis in neuroimaging research and consequent loss of credibility in the public eye, we need to improve how we do science. This article aims to be a practical guide for researchers at any stage of their careers that will help them make their research more reproducible and transparent while minimizing the additional effort that this might require. The guide covers three major topics in open science (data, code, and publications) and offers practical advice as well as highlighting advantages of adopting more open research practices that go beyond improved transparency and reproducibility.

Introduction

The question of how the brain creates the mind has captivated humankind for thousands of years. With recent advances in human *in vivo* brain imaging, we how have effective tools to peek into biological underpinnings of mind and behavior. Even though we are no longer constrained just to philosophical thought experiments and behavioral observations (which undoubtedly are extremely useful), the question at hand has not gotten any easier. These powerful new tools have largely demonstrated just how complex the biological bases of behavior actually are. Neuroimaging allows us to give more biologically grounded answers to burning questions about everyday human behavior (“why do we crave things?”, “how do we control learned responses?”, “how do we regulate emotions?” etc.), as well as influencing how we think about mental illnesses.

In addition to fantastic advances in terms of hardware we can use to study the human brain (function Magnetic Resonance Imaging, Magnetoencephalography, Electroencephalography etc.) we have also witnessed many new developments in terms of data processing and modelling. Many bright minds have contributed to a growing library of methods that derive different features from brain signals. Those methods have widened our perspective on brain processes, but also resulted in methodological plurality [1]. Saying that there is no single best way to analyze a neuroimaging dataset is an understatement; we can confidently say that there many thousands of ways to do that.

Having access to a plethora of denoising and modelling algorithms can be both good and bad. On one side there are many aspects of brain anatomy and function that we can extract and use as dependent variables, which maximizes the chances of finding the most appropriate and powerful measure to ask a particular question. On the other side, the incentive structure of the current scientific enterprise combined with methodological plurality can be a dangerous mix.

Scientists rarely approach a problem without a theory, hypothesis, or a set of assumptions, and the high number of "researcher degrees of freedom" [2] can implicitly drive researchers to choose analysis workflows that provide results that are most consistent with their hypotheses. As Richard Feynman said "The first principle is that you must not fool yourself — and you are the easiest person to fool.". Additionally, neuroimaging (like almost every other scientific field) suffers from publication bias, in which "null" results are rarely published, leading to overestimated effect sizes (for review of this and other biases see [3].

Recent years have seen an increase in alarming signals about the lack of replicability in neuroscience, psychology, and other related fields [4]. Neuroimaging studies generally have low statistical power (estimated at 8%) due to the high cost of data collection which results in an inflation of the number of positive results that are false [5], . To avoid a widespread crisis in our field and consequently losing credibility in the public eye, we need to improve how we do science. This article aims to complement existing literature on the topic [6–8] by compiling a practical guide for researchers at any stage of their careers that will help them make their research more reproducible and transparent while minimizing the additional effort that this might require.

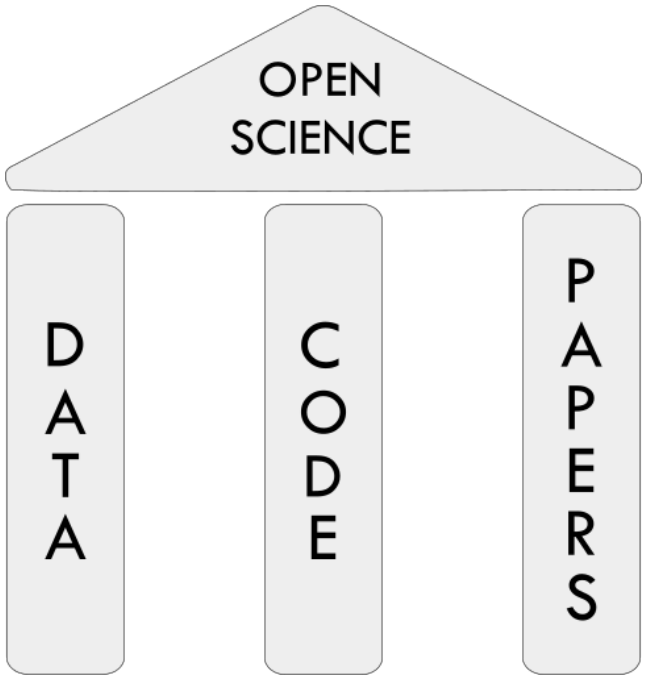


Figure 1. Three pillars of Open Science: data, code, and papers.

How to deal with data

Data are a central component of the scientific process. When data are made open accessible, they not only allow the scientific community to validate the accuracy of published findings, but also empower researchers to perform novel analyses or combine data from multiple sources. Papers accompanied by publicly available data are on average cited more often [9,10], while at

the same time exposing fewer statistical errors [11]. Data sharing has been mandated by some grant funding agencies, as well as journals. Some also argue that sharing data is an ethical obligation toward study participants, in order to maximize the benefits of their participation [12]. Neuroimaging has a substantial advantage in terms of ease of data capture since the data generation process is completely digital. In principle one could provide a digital record of the entire research process for the purpose of reproducibility. However, even though data sharing in neuroimaging has been extensively reviewed in [13] and [14] there is little practical advice on the topic.

Consent forms

Planning for data sharing should start at the ethical approval stage. Even though in the United States de-identified data can be freely shared without specific participant consent, the rules differ in other countries (and they may change in the upcoming revisions to the Common Rule, which governs research in the US). In addition it is only fair to inform your participants about your intention to maximize their generous gift by sharing their data, and to allow them to withdraw from research if they don't wish to have their data shared. However, consent form language needs to be carefully crafted. To streamline the creation of consent forms with data sharing clauses, we have prepared a set of templates that can be easily inserted into existing consent forms after minor adjustments[1]. Those templates have been derived from existing consent forms of leading data sharing projects (Nathan Kline Institute Enhanced sample [15] and Human Connectome Project [16]) followed by consultations with bioethics experts. The templates come in two flavors: one for normal populations and generic data and one for sensitive populations and/or data. The latter splits the data into two sets: a publicly available portion and a portion that requires approval of a data sharing committee (that would assess the ability of the applicant to protect sensitive data) in order to gain access to. We recommend using the restricted access version only for data and populations for which a) potential data re-identification is easy due to small sample and/or the level of detail of included variables (for example exact time and location of scanning) or b) re-identification would lead to negative consequences for the participants (for example in a study of HIV-positive subjects).

Data organization

To successfully share data one has to properly describe it and organize it. Even though some experimental details such as the MRI phase encoding direction may seem obvious for the researcher who obtained the data, they need to be clearly explained for external researchers. In addition, good data organization and description can reduce mistakes in analysis. While each experiment is different and may include unique measurements or procedures, most MRI datasets can be accurately described using one fairly simple scheme. Recently we have proposed such scheme - the Brain Imaging Data Structure (BIDS) [17]. It was inspired by the data organization used by OpenfMRI database, but has evolved through extensive consultations with the neuroimaging community. BIDS aims at being simple to adopt,and roughly follows existing practices common in the neuroimaging community. It is heavily based on a specific

[1] https://open-brain-consent.readthedocs.org/en/latest/ultimate.html

organization of files and folders and uses simple file formats such as NifTI, tab-separated text and JSON. It does not require a database or any external piece of software for processing. A browser-based validator has been developed that allows one to easily check whether a dataset accurately follows the BIDS standard[2].

An additional benefit of using a standardized data organization scheme is that it greatly streamlines the data curation that is necessary when submitting data to a data sharing repository. For example datasets formatted according to BIDS undergo a faster and more streamlined curation process when submitted to OpenfMRI database [18].

Publishing data

Data should be submitted to a repository before submitting the relevant paper. This allows the author to point the readers and reviewers to the location of the data in the manuscript. The manuscript can benefit from increased transparency due to shared data and the data itself can become a resource enabling additional future research.

The most appropriate places for depositing data are field-specific repositories. Currently in human neuroimaging there are two well recognized repositories accepting data from everyone: FCP/INDI [19] (for any datasets that include resting state fMRI and T1 weighted scans) and OpenfMRI [18] (for any datasets that include any MRI data). Field specific repositories have the advantage of more focused curation process that can greatly improve the value of your data. They also increase data discoverability since researchers search through them first when looking for datasets, and some (like OpenfMRI) are indexed by PubMed which allows the dataset to be directly linked to the paper via the LinkOut mechanism.

If for some reason field specific repositories are not an option we recommend using field agnostic repositories such as FigShare, Dryad, or DataVerse. When picking a repository one should think of long term data retention. No one can guarantee existence of a repository in the far future, but historical track record and the support of well established institutions can increase the chances that the data will be available in the decades to come. In addition a platform such as Open Science Framework (www.osf.io) can be used to link together datasets deposited in field agnostic repositories with code and preprints (see below). If one is concerned about losing competitive advantage by sharing data before the relevant manuscript will be accepted and published (so called “scooping”) one can consider setting an embargo period on the submitted dataset. OSF[3], figshare[4], and Dryad[5] support this functionality.

Since data-agnostic repositories do not impose any restriction on the form in which you deposit your data nor do they check completeness, it is essential to ensure that all of the necessary data and metadata are present. Using a data organization scheme designed for neuroimaging needs such as BIDS or XCEDE [20] can help ensure that data are represented accurately. In

[2] http://incf.github.io/bids-validator
[3] https://osf.io/faq/
[4] https://figshare.com/blog/The_future_of_figshare/166
[5] http://datadryad.org/pages/faq

addition, it is a good idea to ask a colleague who is unfamiliar with the data to evaluate the quality and completeness of the description.

If the data accompanying the paper is very large or particularly complex you should consider writing a separate data paper to describe the dataset [21]. A data paper is new type of publication dedicated purely to description of the data rather than its analysis. It can provide more space to describe the experimental procedures and data organization details, and also provides a mechanism for credit when the data are reused in the future. In addition, one often receives useful feedback about the dataset description through the peer review process. The list of journals that currently accept neuroimaging data papers includes but is not limited to: Scientific Data, Gigascience, Data in Brief, F1000Research, Neuroinformatics, and Frontiers in Neuroscience.

In addition to raw data we also encourage authors to share derivatives such as preprocessed volumes, statistical maps or tables of summary measures. Because other researchers are often interested in reusing the results rather than the raw data, this can further increase the impact of the data. For example, statistical maps can be used to perform image-based meta analysis or derive regions of interest for new studies. For sharing statistical maps we encourage authors to use the NeuroVault.org platform [22]. The UCLA Multimodal Connectivity Database [23] provides similar service but for connectivity matrices (derived from fMRI or DWI data).

Finally published data should be accompanied by an appropriate license. Data are treated differently by the legal system than creative works (i.e. papers, figures) and software and thus require special licenses. Following the lead of major scientific institutions such as BioMed Central, CERN, or The British Library we recommend using an unrestricted Public Domain license (such as CC0 or PDDL) for data[6]. Using such license would maximize the impact of the shared data, by not imposing any restriction on how it can be used and combined with other data. The appropriate legal language that needs to accompany your data can be obtained from https://creativecommons.org/publicdomain/zero/1.0 or http://opendatacommons.org/licenses/pddl//. There are also other more restrictive license options (see http://www.dcc.ac.uk/resources/how-guides/license-research-data). However, additional restrictions can have unintended consequences. For example, including a Non-Commercial clause, while seemingly innocuous, could in its broadest interpretation prevent your data from being used for teaching or research at a private university. Similarly, a No-Derivatives clause can prevent your data from being combined in any form with other data (for example a brain template released under No-Derivatives license cannot be used as a coregistration target).

How to deal with code

Neuroimaging data analysis has required computers since its inception. A combination of compiled or script code is involved in every PET, MRI, or EEG study, as in most other fields of science. The code we write to analyze data is a vital part of the scientific process, and similar to

[6] https://wiki.creativecommons.org/wiki/CC0_use_for_data

data, is not only necessary to interpret and validate results, but can be also used to address new research questions. Therefore the sharing of code is as important as the sharing of data for scientific transparency and reproducibility.

Because most researchers are not trained in software engineering, the code that is written to analyze neuroimaging data (as in other areas of science) is often undocumented and lacks the formal tests that professional programmers use to ensure accuracy. In addition to the lack of training, there are few incentives to spend the time necessary to generate high-quality and well-documented code. Changes in the incentive structure of science will take years, but in the meantime, perceived poor quality of code and lack of thorough documentation should not prevent scientists from publishing it [24]. Sharing undocumented code is a much better than not sharing code at all and can still provide benefits to the author. Perhaps the most compelling motivation for sharing code comes from citation rates. Papers accompanied by usable code are on average cited more often than their counterparts without the code [25].

An additional concern that stops researchers from sharing code is fear that they will have to provide user support and answer a flood of emails from other researchers who may have problems understanding the codebase. However, sharing code does not oblige a researcher to provide user support. One useful solution to this problem is to set up a mailing list (for example with Google) and point all users to ask questions through it; in this way, answers are searchable, so that future users with the same questions can find them via a web search. Alternatively one can point user to a community driven user support forum for neuroinformatics (such as NeuroStars.org) and ask them to tag their questions with a label uniquely identifying the software or script in question; we have found this to be a useful support solution for the OpenfMRI project. Both solutions foster a community that can lead to users helping each other with problems, thus relieving some of the burden from the author of the software. In addition, since the user support happens through a dedicated platform there is less pressure on the author to immediately address issues than there would be with user requests send directly by email.

Many of the issues with code quality and ease of sharing can be addressed by careful planning. One tool that all research programmers should incorporate into their toolbox is the use of a Version Control System (VCS) such as git. VCS provides a mechanism for taking snapshots of evolving codebase that allow tracking of changes and reverting them if there is a need (e.g., after making a change that ends up breaking things). Adopting a VCS leads a to cleaner code base that is not cluttered by manual copies of different versions of a particular script (e.g, “script_version3_good_Jan31_try3.py”). VCS also allows one to quickly switch between branches - alternative and parallel versions of the codebase - to test a new approach or method without having to alter a tried and tested codebase. For a great introduction to git we refer the reader to [26]. We encourage scientists to use git rather than other VCS due to a passionate and rapidly growing community of scientists who use the GitHub.com platform, which is a freely available implementation of the git VCS system. In the simplest use case GitHub is a platform for sharing code (which is extremely simple for those who already use git as their VCS), but it also includes other features which make contributing to collaborative projects, reviewing, and

testing code simple and efficient. The Open Science Framework mentioned above can be used to link together data and code related to a single project. It can also be used to set embargo period on the code so it could be submitted with the paper while minimising the risk of "scooping".

Striving for automation whenever possible is another strategy that will not only result in more reproducible research, but can also save a lot of time. Some analysis steps seem to be easy to perform manually, but that remains true only when they need to be performed just once. Quite often in the course of a project parameters are modified, list of subjects are changed, and processing steps need to be rerun. This is a situation in which having a set of scripts that can perform all of the processing steps automatically instead of relying on manual interventions can really pay off. There are many frameworks that help design and efficiently run neuroimaging analyses in automated fashion. Those include, but are not limited to: Nipype [27], PSOM [28], aa [29], and make [30]. As an example, for our recent work on the MyConnectome project[31] we created a fully automated analysis pipeline, which we implemented using a virtual machine[7].

While automation can be very useful for reproducibility, the scientific process often involves interactive interrogation of data interleaved with notes and plots. Fortunately there is a growing set of tools that facilitate this interactive style of work while preserving a trace of all the computational steps, which increases reproducibility. This philosophy is also known as "literate programming" [32] and combines analysis code, plots, and text narrative. The list of tools supporting this style of work includes, but is not limited to: Jupyter (for R, Python and Julia)[8], R Markdown (for R)[9] and matlabweb (for MATLAB)[10]. Using one of those tools not only provides the ability to revisit an interactive analysis performed in the past, but also to share an analysis accompanied by plots and narrative text with collaborators. Files created by one of such systems (in case of Jupyter they are called Notebooks) can be shared together with the rest of the code on GitHub, which will automatically render included plots so they can be viewed directly from the browser without requiring installation of any additional software.

As with data, it is important to accompany shared code with an appropriate license. Following [6] we recommend choosing a license that is compatible with the open source definition such as Apache 2.0, MIT, or GNU General Public License (GPL)[11]. The most important concept to understand when choosing a license is "copyleft". A license with a "copyleft" property (such as GPL) allows derivatives of your software to be published, but only if done under the same license. This property limits the range of code your software can be combined with (due to license incompatibility) and thus can restrict the reusability of your code; for this reason, we generally employ minimally restrictive licenses such as the MIT license. Choosing an open

[7] https://github.com/poldrack/myconnectome-vm
[8] http://jupyter.org
[9] http://rmarkdown.rstudio.com
[10] https://www.ctan.org/pkg/matlabweb
[11] https://opensource.org/licenses

source license and applying it to your code can be greatly simplified by using a service such as choosealicense.com.

How to deal with publications

Finally, the most important step in dissemination of results is publishing a paper. An essential key to increasing transparency and reproducibility of scientific outputs is accurate description of methods and data. This not only means that the manuscript should include links to data and code mentioned before (which entails that both data and code should be deposited before submitting the manuscript), but also thorough and detailed description of methods used to come to a given conclusion. As an author one often struggles with a fine balance between detailed description of different analyses performed during the project and and the need to explain the scientific finding in the most clear way. It is not unheard of that for the sake of a better narrative some results are omitted[12]. At the same time there is a clear need to present results in a coherent narrative with a clear interpretation that binds the new results with an existing pool of knowledge[13]. We submit that one does not have exclude the other. A clear narrative can be provided in the main body of the manuscript and the details of methods used together with null results and other analyses performed on the dataset can be included in the supplementary materials, as well as in the documentation of the shared code. In this way, the main narrative of the paper is not obfuscated too many details and auxiliary analyses, but all of the results (even null ones) are available for the interested parties. Such results from extra analyses could include for example all of the additional contrasts that were not significant and thus not reported in the main body of the manuscript (of which unthresholded statistical maps should be shared for example using a platform such as NeuroVault). Often these extra analyses and null results may seem uninteresting from the author's point of view, but one cannot truly predict what other scientists can be interested in. In particular, the null results (which are difficult to publish independently) can contribute to growing body of evidence that can be used in the future to perform meta analyses.For more extensive set of recommendation for reporting neuroimaging studies, see the recent report from the Organization for Human Brain Mapping's Committee on Best Practices in Data Analysis and Sharing (COBIDAS) report[14].

The last important topic to cover is accessibility of the manuscript. To maximize the impact of published research one should consider making the manuscript publicly available. In fact many funding bodies (NIH, Wellcome Trust) require this for all manuscripts describing research that they have funded. Many journals provide an option to make papers open access, albeit sometimes at prohibitively high price (for example the leading specialist neuroimaging journal - NeuroImage - requires a fee of $3000). Unfortunately the most prestigious journals (Nature and Science) do not provide such option despite many requests from the scientific community. Papers published in those journals remain "paywalled" - available only through institutions which pay subscription fees, or through public repositories (such as PubMed Central) after a sometimes lengthy embargo period. The scientific publishing landscape is changing [33,34], and

[12] http://sometimesimwrong.typepad.com/wrong/2015/11/guest-post-a-tale-of-two-papers.html
[13] http://www.russpoldrack.org/2015/11/are-good-science-and-great-storytelling.html
[14] www.humanbrainmapping.org/**cobidas**/

we hope it will evolve in a way that will give everyone access to published work as well as to the means of publication. In the meantime we recommend ensuring open access by publishing preprints at Bioxiv or arXiv before submitting the paper to a designated journal. In addition to making the manuscript publicly available without any cost, this solution has other advantages. Firstly it allows the wider community to give feedback to the authors about the manuscript and potentially improve it which is beneficial for both the authors as well as the journal the paper will be submitted to; for example, the present paper received useful comments from three individuals in addition to the appointed peer reviewers. Secondly, in case of hot topics publishing a preprint establishes precedence on being the first one to describe a particular finding. Finally since preprints have assigned DOIs other researchers can reference them even before they will be published in a journal. Preprints are increasingly popular and vast majority of journals accept manuscripts that have been previously published as preprints. We are not aware of any neuroscience journals that do not allow authors to deposit preprints before submission, although some journals such as *Neuron* and *Current Biology* consider each submission independently and thus one should contact the editor prior to submission.

To further improve accessibility and impact of research outputs one can also consider sharing papers that have already been published in subscription based journals. Unfortunately this can be difficult due to copyright transfer agreements many journals require from authors. Such agreement give the journal exclusive right to the content of the paper. However, each publisher uses a different set of rules and some of them allow limited sharing of your work you have surrender your rights to. For example Elsevier (publisher of NeuroImage) allows authors to publish their accepted manuscripts (without the journal formatting) on a non-commercial website, a blog or a preprint repository[15]. Wiley (publisher of Human Brain Mapping) has a similar policy for submitted manuscripts (before the paper gets accepted), but requires an embargo of 12 months before authors can share the accepted manuscript[16]. Policies for other journals might vary. SherPa/ROMEO (http://www.sherpa.ac.uk/romeo) is a databaset that allows authors to quickly check what the journal they published with allows to share and when.

There are multiple options when it comes to choosing a repository to share manuscripts published in subscription-based journals. Private websites, institutional repositories, and preprint servers seems to be well within the legal restrictions of most journals. Commercial websites such as researchgate.com and academia.edu remain a legal grey zone (with some reports of Elsevier taking legal actions to remove papers from one of them[17]). If the research has been at least partially funded by NIH one can deposit the manuscript in PubMed Central (respecting appropriate embargos)[18].

[15] https://www.elsevier.com/about/company-information/policies/sharing
[16] http://olabout.wiley.com/WileyCDA/Section/id-826716.html
[17] http://svpow.com/2013/12/06/elsevier-is-taking-down-papers-from-academia-edu/
[18] https://nihms.nih.gov

Discussion

In this guide we have carefully selected a list of enhancements that every neuroimaging researcher can make to their scientific workflow that will improve the impact of their research, benefiting not only them individually the community as a whole. We have limited the list to mechanisms that have been tested and discussed in the community for number of years and which have clear benefits to the individual researcher. However, the way science is conducted is evolving constantly and there are many more visions that could be implemented. In the following section we discuss some of the emerging trends that may become commonplace in the future.

Pre-registration

We have mentioned in the introduction that the field of neuroimaging is both blessed and cursed with plurality of analysis choices which can lead to biases in published results (since many decisions about statistical treatment of data are made after seeing the data). We recommended taking advantage of supplementary materials to elaborate on all performed analyses and sharing statistical maps of null effect contrasts as a partial remedy of this problem. However, further reduction of publication bias can be achieved even more effectively by adopting the pre-registration mechanism [35]. This way of doing research, originally adopted from clinical trials, involves writing and registering (in a third party repository) a study plan outlining details of data acquisition, subject exclusion criteria, and planned analyses even before that data have been acquired. This not only motivates researchers to formulate hypotheses before seeing data, but also allows for a clear distinction between results of hypothesis driven confirmatory analyses (included in the pre-registration) and exploratory analyses (added after seeing the data). It is worth mentioning that exploratory analyses are by no means inferior to confirmatory analyses; they are an important part of science, generating new hypotheses that can be tested by future studies. However exploratory analyses can suffer from bias (since their inception was influenced by the data itself) and thus require additional evidence. Unfortunately, confirmatory and exploratory analyses are often not properly distinguished in publications, a problem that could be remedied by preregistration. Preregistration also plays a vital role in highlighting hypotheses that turned out not to be confirmed by the data (“null effects”).

It is clear that preregistration can help in research transparency and reproducibility by reducing biases. It is also important to acknowledge that putting together and registering a binding research plan requires a significant time investment from the researcher and thus is not common a common practice (with exception to replication studies [4,36]). There are, however, additional incentives for individual researchers to preregister their studies. For example, the Center for Open Science spearheaded the Registered Reports[19] initiative in 2012. According to this mechanism, authors send their preregistration reports (Introduction, Methods parts of a future paper and optionally analysis of pilot data) for peer review to a journal for peer review. Validity of the experimental plan is assessed and if deemed sufficient receives “In-principle acceptance” (IPA), in which case the journal guarantees to publish the final version of the paper

[19] https://osf.io/8mpji

(after data collection and analysis) independently of the results (i.e. even if the hypothesized effect was not found).. Currently journals accepting in neuroimaging papers participating in the Registered Reports program include: AIMS Neuroscience [37], Attention, Perception, and Psychophysics, Cognition and Emotion [38], Cortex [39] and European Journal of Neuroscience. Additionally, The Center for Open Science started a Preregistration Challenge[20] providing $1000 reward for the first 1000 preregistered eligible studies. This initiative is independent of the Registered Reports and does not guarantee publication, but the list of eligible journals is much longer (includes such journals as PloS Biology, Hippocampus, or Stroke).

Peer review and giving feedback

An important part of the scientific method is peer review but with a few notable exceptions (eLife, GigaScience, ScienceOpen, and F1000Research), the review procedure happens behind closed doors and thus leaves the reader without any information on how a published paper was evaluated (other than the fact that it was accepted). In addition, at most journals reviewers do not get credit for their hard work, though some (such as the Frontiers journals) list the reviewers on each published paper. This situation can be remedied by publishing reviews performed for journals after the paper has been published. Several outlets exists that allow that. PubMed Commons allow registered and verified users of PubMed to provide comments under every paper indexed by PubMed. Those comments have to be signed so there is no option to remain anonymous (which is important for junior researchers afraid of a blowback after criticizing work from an established lab). Another option is PubPeer - a website that allow anyone to comment on any published paper or preprint. It supports both anonymous and signed comments so it's up to the reviewer to decide what is better for them. Finally there is Publons.com - a platform for tracking reviewers profiles and publishing reviews. Thanks to collaborations with many journals it is very easy to use and even allows you to get credit for publishing your reviews anonymously.

All of those platforms can be used not only to share reviews solicited from reviewers by journals, but also to share comments and give feedback about already published work or preprints shared by other researchers. Peer review expanded to the whole community can improve the quality of research, catch mistakes, or help with the clarity of both preprints and already published work. Giving feedback on preprints can be especially useful when it comes to highlighting already published work that authors might have missed (which considering the number of papers published every year is not unlikely).

Signing openly shared reviews can have some benefits when it comes to establishing one's reputation as an expert in the field. Well thought through and carefully worded reviews consisting of constructive criticism are hard to come by and extremely valuable. By sharing and signing reviews researchers can not only help their peers, but also boost their reputation which can potentially seen favourably by hiring committees and grant review boards. However, we feel

[20] https://cos.io/prereg/

that the option of anonymous reviews remains very important since on many occasions it will be the only way for researchers to express concerns about validity of some work.

Box 1. Simple steps towards open science

Data:

- Include a section about data sharing to your consent forms.
- Share your raw data upon paper submission using a repository dedicated for neuroimaging.
- Consider writing a separate data paper for more complex and interesting datasets.
- Remember that sharing your data improves the impact and citation rates of your research!

Code:

- Use version control system for all your projects.
- Share your code on GitHub.com even if it's not well documented.
- Set up a mailing list for user related questions.
- People reusing the code you shared will cite the relevant papers.

Papers:

- Include all extra analyses and null results in the supplementary materials without sacrificing the clarity of the message in the main body of the manuscript.
- Submit preprints to claim precedence, solicit feedback and give access to your research.

Summary

The scientific method is evolving towards a more transparent and collaborative endeavour. The age of digital communication allows us to go beyond printed summaries and dive deeper into underlying data and code. In this guide we hope to have shown that there are many improvements in scientific practice everyone can implement with relatively little added effort that will improve transparency, replicability and impact of their research. Even though the added transparency might in rare cases expose errors those are a natural part of the scientific process, as a community researchers should acknowledge their existence and try to learn from them instead of hiding them and antagonizing those who make them.

Acknowledgements

This work was supported by Laura and John Arnold Foundation, National Science Foundation, and National Institutes of Health. We would also like to thank our reviewers: Nikolas Kriegeskorte, Brian Nosek, Cyril Pernet, Yaroslav Halchenko, and Cameron Craddock for helpful comments and corrections as well as Michael Milham for stimulating discussion.

References

1. Carp J. On the plurality of (methodological) worlds: estimating the analytic flexibility of FMRI experiments. Front Neurosci. 2012;6: 149. doi:10.3389/fnins.2012.00149

2. Simmons JP, Nelson LD, Simonsohn U. False-positive psychology: undisclosed flexibility in data collection and analysis allows presenting anything as significant. Psychol Sci. 2011;22: 1359–1366. doi:10.1177/0956797611417632

3. Ioannidis JPA, Munafò MR, Fusar-Poli P, Nosek BA, David SP. Publication and other reporting biases in cognitive sciences: detection, prevalence, and prevention. Trends Cogn Sci. 2014;18: 235–241. doi:10.1016/j.tics.2014.02.010

4. Open Science Collaboration. Estimating the reproducibility of psychological science. Science. 2015;349. doi:10.1126/science.aac4716

5. Button KS, Ioannidis JPA, Mokrysz C, Nosek BA, Flint J, Robinson ESJ, et al. Power failure: why small sample size undermines the reliability of neuroscience. Nat Rev Neurosci. 2013;14: 365–376. doi:10.1038/nrn3475

6. Halchenko YO, Hanke M. Four aspects to make science open “by design” and not as an after-thought. Gigascience. 2015;4: 31. doi:10.1186/s13742-015-0072-7

7. Pernet C, Poline J-B. Improving functional magnetic resonance imaging reproducibility [Internet]. Gigascience. 2015. p. 15. doi:10.1186/s13742-015-0055-8

8. Eglen S, Marwick B, Halchenko Y, Hanke M, Sufi S, Gleeson P, et al. Towards standard practices for sharing computer code and programs in neuroscience [Internet]. bioRxiv. 2016. p. 045104. doi:10.1101/045104

9. Piwowar HA, Day RS, Fridsma DB. Sharing detailed research data is associated with increased citation rate. PLoS One. 2007;2: e308. doi:10.1371/journal.pone.0000308

10. Piwowar HA, Vision TJ. Data reuse and the open data citation advantage. PeerJ. 2013;1: e175. doi:10.7717/peerj.175

11. Wicherts JM, Bakker M, Molenaar D. Willingness to Share Research Data Is Related to the Strength of the Evidence and the Quality of Reporting of Statistical Results. Tractenberg RE, editor. PLoS One. 2011;6: e26828. doi:10.1371/journal.pone.0026828

12. Brakewood B, Poldrack RA. The ethics of secondary data analysis: considering the application of Belmont principles to the sharing of neuroimaging data. Neuroimage. 2013;82: 671–676. doi:10.1016/j.neuroimage.2013.02.040

13. Poline J-B, Breeze JL, Ghosh S, Gorgolewski K, Halchenko YO, Hanke M, et al. Data sharing in neuroimaging research. Front Neuroinform. 2012;6: 9. doi:10.3389/fninf.2012.00009

14. Poldrack RA, Gorgolewski KJ. Making big data open: data sharing in neuroimaging. Nat

Neurosci. Nature Publishing Group; 2014;17: 1510–1517. doi:10.1038/nn.3818

15. Nooner KB, Colcombe SJ, Tobe RH, Mennes M, Benedict MM, Moreno AL, et al. The NKI-Rockland Sample: A Model for Accelerating the Pace of Discovery Science in Psychiatry. Front Neurosci. 2012;6: 152. doi:10.3389/fnins.2012.00152

16. Van Essen DC, Smith SM, Barch DM, Behrens TEJ, Yacoub E, Ugurbil K, et al. The WU-Minn Human Connectome Project: an overview. Neuroimage. 2013;80: 62–79. doi:10.1016/j.neuroimage.2013.05.041

17. Gorgolewski KJ, Auer T, Calhoun VD, Cameron Craddock R, Das S, Duff EP, et al. The Brain Imaging Data Structure: a standard for organizing and describing outputs of neuroimaging experiments [Internet]. bioRxiv. 2015. p. 034561. doi:10.1101/034561

18. Poldrack RA, Barch DM, Mitchell JP, Wager TD, Wagner AD, Devlin JT, et al. Toward open sharing of task-based fMRI data: the OpenfMRI project. Front Neuroinform. 2013;7: 1–12. doi:10.3389/fninf.2013.00012

19. Mennes M, Biswal BB, Castellanos FX, Milham MP. Making data sharing work: The FCP/INDI experience. Neuroimage. Elsevier Inc.; 2012; doi:10.1016/j.neuroimage.2012.10.064

20. Gadde S, Aucoin N, Grethe JS, Keator DB, Marcus DS, Pieper S. XCEDE: An Extensible Schema for Biomedical Data. Neuroinformatics. 2012;10: 19–32. doi:10.1007/s12021-011-9119-9

21. Gorgolewski KJ, Margulies DS, Milham MP. Making data sharing count: a publication-based solution. Front Neurosci. Frontiers; 2013;7: 9. doi:10.3389/fnins.2013.00009

22. Gorgolewski KJ, Varoquaux G, Rivera G, Schwarz Y, Ghosh SS, Maumet C, et al. NeuroVault.org: a web-based repository for collecting and sharing unthresholded statistical maps of the human brain. Front Neuroinform. Frontiers; 2015;9. doi:10.3389/fninf.2015.00008

23. Brown JA, Rudie JD, Bandrowski A, Van Horn JD, Bookheimer SY. The UCLA multimodal connectivity database: a web-based platform for brain connectivity matrix sharing and analysis. Front Neuroinform. 2012;6: 28. doi:10.3389/fninf.2012.00028

24. Barnes N. Publish your computer code: it is good enough. Nature. 2010;467: 753. doi:10.1038/467753a

25. Vandewalle P. Code Sharing Is Associated with Research Impact in Image Processing. Comput Sci Eng. AIP Publishing; 2012;14: 42–47. doi:10.1109/MCSE.2012.63

26. Blischak JD, Davenport ER, Wilson G. A Quick Introduction to Version Control with Git and GitHub. PLoS Comput Biol. 2016;12: e1004668. doi:10.1371/journal.pcbi.1004668

27. Gorgolewski K, Burns CD, Madison C, Clark D, Halchenko YO, Waskom ML, et al. Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. Front Neuroinform. 2011;5: 13. doi:10.3389/fninf.2011.00013

28. Bellec P, Courchesne SL, Dickinson P, Lerch J, Zijdenbos A, Evans AC. The pipeline system for Octave and Matlab (PSOM): a lightweight scripting framework and execution engine for scientific workflows. Front Neuroinform. 2012;6. doi:10.3389/fninf.2012.00007

29. Cusack R, Vicente-Grabovetsky A, Mitchell DJ, Wild CJ, Auer T, Linke AC, et al. Automatic analysis (aa): efficient neuroimaging workflows and parallel processing using Matlab and XML. Front Neuroinform. 2014;8: 90. doi:10.3389/fninf.2014.00090

30. Askren MK, McAllister-Day TK, Koh N, Mestre Z, Dines JN, Korman BA, et al. Using Make for Reproducible and Parallel Neuroimaging Workflow and Quality-Assurance. Front Neuroinform. Frontiers; 2016;10. doi:10.3389/fninf.2016.00002

31. Poldrack RA, Laumann TO, Koyejo O, Gregory B, Hover A, Chen M-Y, et al. Long-term neural and physiological phenotyping of a single human. Nat Commun. 2015;6: 8885. doi:10.1038/ncomms9885

32. Knuth DE. Literate programming. CSLI Lecture Notes, Stanford, CA: Center for the Study of Language and Information (CSLI), 1992. 1992; Available: http://adsabs.harvard.edu/abs/1992lipr.book.....K

33. The future of publishing: A new page. Nature. 2013;495: 425. doi:10.1038/495425a

34. Popova K. Evolution of Open Access Policies and Business Models: Which Way Leads to The Future? In: Scicasts [Internet]. [cited 5 Feb 2016]. Available: https://scicasts.com/insights/2123-open-science/10333-evolution-of-open-access-policies-and-business-models-which-way-to-the-future/

35. Aarts AA, Bosco F, Carp JM, Field JG, IJzerman H, Lewis M, et al. Maximizing the reproducibility of your research. In: Lilienfeld SO, Waldman ID, editors. Psychological Science Under Scrutiny: Recent Challenges and Proposed Solutions. 2014. Available: http://ar.ascb.org/publicpolicy/Articles/OSC2014final.pdf

36. Boekel W, Wagenmakers E-J, Belay L, Verhagen J, Brown S, Forstmann BU. A purely confirmatory replication study of structural brain-behavior correlations. Cortex. 2015;66: 115–133. doi:10.1016/j.cortex.2014.11.019

37. Chambers CD, Feredoes E, Muthukumaraswamy SD, Etchells PJ. Instead of “playing the game” it is time to change the rules: Registered Reports at AIMS Neuroscience and beyond. doi:10.3934/Neuroscience

38. Fischer A, van Reekum C. Editorial. Cognition and Emotion. 2015;29: 765–766. doi:10.1080/02699931.2015.1026223

39. Chambers CD. Registered reports: a new publishing initiative at Cortex. Cortex. 2013;49: 609–610. doi:10.1016/j.cortex.2012.12.016

Registered Reports: A step change in scientific publishing

www.elsevier.com/reviewers-update/story/innovation-in-publishing/registered-reports-a-step-change-in-scientific-publishing

Professor Chris Chambers, Registered Reports Editor of the Elsevier journal Cortex and one of the concept's founders, on how the initiative combats publication bias

By Professor Chris Chambers Posted on 13 November 2014

Share story:

Last year, *Cortex* launched an exciting initiative called Registered Reports – a format of empirical article that places study pre-registration at the center of peer review.

Our aim with Registered Reports is to enhance the transparency and reproducibility of science by reviewing study protocols before experiments are conducted. If we think the protocol has merit we will commit, in advance, to publishing the outcomes. Armed with this provisional acceptance of their work, authors can perform the research safe in the knowledge that the results themselves will not determine the article's publication. At the same time, readers of the final paper can feel more confident that the work is reproducible because the initial study predictions and analysis plans were independently reviewed.

Gauging the community's reaction

Registered Reports represents a major departure from standard peer review, and at the time of the *Cortex* launch there was much uncertainty about how the shift would be regarded. Would the scientific community take notice? Would the format be popular? What kinds of submissions would we receive? To raise the profile of the initiative, we wrote an open letter to *The Guardian*, signed by more than 80 scientists and members of journal editorial boards. Together, we called for Registered Reports to be offered across the Life Sciences as a way to liberate academia from the grip of managerial incentives that favor the production of publications over genuine discovery.

Since then, the response from the scientific community suggests that Registered Reports are poised to transform the publishing landscape. In addition to *Cortex*, where the first completed articles will soon be published, we've seen the format taken up by more than a dozen journals across Neuroscience, Psychology, Psychiatry, Biology, Nutrition, and Medicine, with many more in the pipeline. I'd like to highlight just a few examples:

- Earlier this year, the journal *Social Psychology* published a high-profile special issue of Registered Reports that tested the reproducibility of classic psychological phenomena.
- A new journal, *Comprehensive Results in Social Psychology*, is dedicated entirely to the Registered Reports.
- *eLife* is hosting a special issue of Registered Reports to assess the reproducibility of cancer biology research.
- Another major journal, to be named in the coming weeks, is poised to unveil them across the full spectrum of sciences ranging from Physics to Psychology.

This is just a snapshot of the progress we've seen since last summer (see here for a complete and regularly updated list of journals, frequently asked questions, and more).

A range of problems threaten the integrity of the scientific method. Such practices have been documented most thoroughly in Psychology, which is characterized by a paucity of replication studies, insufficient statistical power, a high prevalence of cherry picking (also known as *p*-hacking), *post hoc* hypothesising, lack of data sharing, and a journal culture marked by publication bias. Registered Reports are designed to counteract all of these problems.

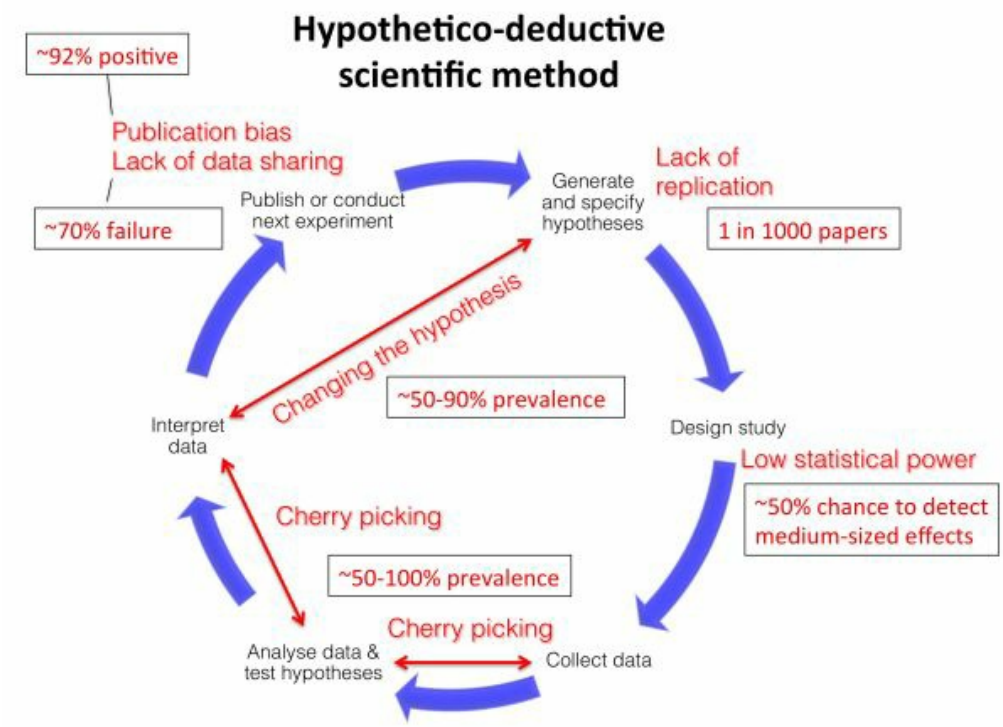


Figure 1: Hypothetico-deductive scientific method

Understanding the appeal of Registered Reports

So why is this proving such an attractive reform? First and foremost, the idea of accepting papers before results are known moves us beyond the assumption that the visibility of a scientific study should depend on its outcome. A number of Social Sciences and Life Sciences journals are locked in the grip of this powerful bias, prioritizing the publication of positive, novel findings while rejecting those that are negative. Even in medical research, where trial protocols have been pre-registered for decades, null or negative findings are far less likely to be published. By selectively reporting positive results we distort the literature, needlessly populating journals with false conclusions and sabotaging the ability of science to self-correct.

The reason for this publication bias is simple human nature: in judging whether a manuscript is worthy of publication, editors and reviewers are guided not only by the robustness of the method but by their impressions of what the results contribute to knowledge. Do the outcomes constitute a major advance, worthy of space within a journal that rejects the majority of submissions? Results that are novel and eye-catching are naturally seen as more attractive and competitive than those that are null or ambiguous, even when the methodologies that produce them are the same. This bias, in turn, creates perverse incentives for individuals. When we reward scientists for getting "publishable results", we encourage a host of questionable practices to produce them (see Figure 1).

The unique selling point of Registered Reports is that they eliminate the need for scientists to strive for "publishable results". Registered Reports enshrine the ethos that science earns its stripes from the value of the research question and the rigor of the method, and never from whether the data sing a good tune. This idea is as old as the scientific method itself; in fact, it almost feels wrong to call Registered Reports an innovation in publishing when it is closer to being a *restoration* – a reinvention of publishing and the peer-review process as it was meant to be.

Some scientists have expressed fears that Registered Reports could restrict creativity by requiring authors to adhere to a fixed research methodology. In fact – and this is important to emphasize – the Registered Reports initiative places no restrictions whatsoever on creativity, flexibility or the reporting of serendipitous findings. While it is true that the pre-specified methods in a Registered Report must be followed, there are no bounds on the reporting of additional unregistered analyses. The only requirement is that such additional material is labelled transparently so that readers know which analyses were pre-registered and which were exploratory.

Where next for Registered Reports?

Ultimately, it is up to all of us to determine the future of any reform, and if the community continues to support Registered Reports then that future looks promising. Each field that adopts this initiative will be helping to create a scientific literature that is free from publication bias, that celebrates transparency, that welcomes replication as well as novelty, and in which the reported science will be more reproducible. Registered Reports isn't a one-shot cure for scientific publishing, but with every new journal that offers the format, and with every new article published, we strengthen the scientific record and offer scientists a positive incentive to embrace best practice.

Figure 2 illustrates the editorial pipeline for Registered Reports at *Cortex*, which has also been adopted by several other journals, including *Drug and Alcohol Dependence*. Details on review criteria and answers to frequently asked questions are available online.

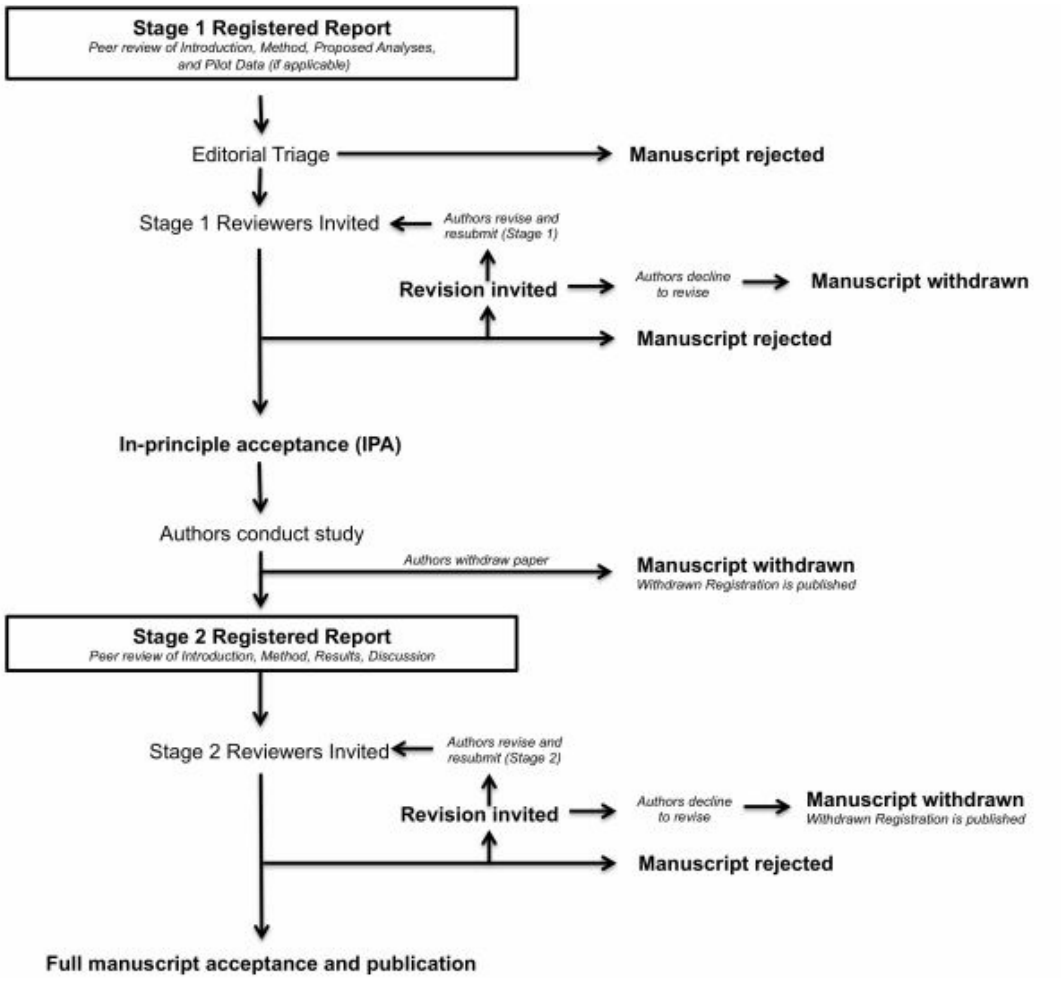


Figure 2: The editorial pipeline for Registered Reports at Cortex

Author biography

Chris Chambers (@chrisdc77) is a Professor of Cognitive Neuroscience at Cardiff University, Section Editor for Registered Reports at *Cortex* and *AIMS Neuroscience*, and Chair of the Registered Reports Committee at the Center for Open Science. His main research interests include the psychology and neuroscience of human impulse control, the interaction between science and the media, and evidence-based public policy.

OPTICON MDCCCXXVI

Hardwicke, T E et al 2014 Only Human: Scientists, Systems, and Suspect Statistics. *Opticon1826,* (16): 25, pp. 1-12, DOI: http://dx.doi.org/10.5334/opt.ch

REVIEW

Only Human: Scientists, Systems, and Suspect Statistics

A review of: Improving Scientific Practice: Dealing With The Human Factors, University of Amsterdam, Amsterdam, September 11, 2014

Tom E. Hardwicke*, Leila Jameel*, Matthew Jones*, Eryk J. Walczak* and Lucia Magis-Weinberg*

It is becoming increasingly clear that science has sailed into troubled waters. Recent revelations about cases of serious research fraud and widespread 'questionable research practices' have initiated a period of critical self-reflection in the scientific community and there is growing concern that several common research practices fall far short of the principles of robust scientific inquiry. At a recent symposium, 'Improving Scientific Practice: Dealing with the Human Factors' held at The University of Amsterdam, the notion of the objective, infallible, and dispassionate scientist was firmly challenged. The symposium was guided by the acknowledgement that scientists are only human, and thus subject to the desires, needs, biases, and limitations inherent to the human condition. In this article, five post-graduate students from University College London describe the issues addressed at the symposium and evaluate proposed solutions to the scientific integrity crisis.

Introduction

The success of science is often attributed to its objectivity: surely science is an impartial, transparent, and dispassionate method for obtaining the truth? In fact, there is growing concern that several aspects of typical scientific practice conflict with these principles and that the integrity of the scientific enterprise has been deeply compromised. The diverse range of issues include cases of serious researcher fraud (e.g., Levelt-Noort-Drenth Committee, 2012; RIKEN Research Paper Investigative Committee, 2014), substantial publication bias towards positive findings (Rosenthal, 1979; Fanelli, 2012), a preponderance of statistically underpowered studies that produce inflated and/or unreliable effects (Button, Ioannidis, Mokrysz, Nosek, Flint, Robinson, & Munafo, 2013), incomplete or erroneous reporting of study methodology (Carp, 2012; Vasilevsky, Brush, Paddock, Ponting, Tripathy et al., 2013), failure to comply with data access requests (Wicherts, Borsboom, Kats, & Molenaar, 2006), and the widespread prevalence of 'questionable research practices' that can insidiously generate false-positive findings

* Department of Experimental Psychology, University College London, UK
t.hardwicke.12@ucl.ac.uk, l.jameel@ucl.ac.uk, m.jones.12@ucl.ac.uk, eryk.walczak.11@ucl.ac.uk, lucia.magis-weinberg.12@ucl.ac.uk

Corresponding author: Tom E. Hardwicke

(Simmons, Nelson & Simonsohn, 2011; John, Loewenstein, & Prelec, 2012). In 2005, a devastating statistical proof was published, which claimed that 'most published research findings are false' (Ioannidis, 2005) and subsequent efforts to replicate existing findings have suggested that the suspected 'reproducibility crisis' is not just theoretically plausible; it is an empirical reality (Begley & Ellis, 2012; Prinz, Schlange, & Asadullah, 2011; Gilbert, 2014).

How is the scientific community to begin addressing these issues? For the organisers of a recent symposium, 'Improving Scientific Practice: Dealing with the Human Factors' hosted by The University of Amsterdam, the first step is to recognise that science is fundamentally a human endeavour, and thus subject to the limitations and biases that underlie human behaviour. Can we design a scientific ecosystem that acknowledges scientists are only human?

1. The damaged scientific ecosystem

The utopian idea of a 'pure' scientist is that of an individual motivated solely by the acquisition of knowledge. However, in reality scientists have human needs, desires, and motivations just like non-scientists (Mahoney, 1976). The scientific ecosystem which researchers inhabit is built and maintained by several organisations including universities, industry stakeholders, funding bodies, and publishers who also have interests that diverge from pure knowledge acquisition. Unfortunately, the present system does not adequately account for these human factors in science, and rewards individuals who are lucky or willing to 'play the game' (Bakker, van Dijk, & Wicherts, 2012). In this first section, we examine how the scientific ecosystem not only fails to guard against the inherent fallibilities of human behaviour, but actively perpetuates them.

1.1 Pressure to publish

Much of the scientific ecosystem revolves around a *de facto* principal commodity: the published research paper (Young, Ioannidis, & Al-Ubaydli, 2008). A metric called the *h*-index is sometimes used to evaluate scientists for hiring, promoting, and funding decisions (Hirsch, 2005). The *h*-index attempts to measure both productivity and research impact by combining number of publications and number of citations to these publications. However, citation rates are not necessarily indicative of quality or reliability: articles are also cited when they are critiqued, or when other researchers are unable to replicate the original finding. There was concern at the symposium that a single-minded drive for productivity is not conducive to the production of reliable research findings.

Part of the problem is that the emphasis on productivity adversely interacts with the personal career goals of individual scientists. For example, short-term contracts are common in academia and it has been suggested that, 'the research system may be exploiting the work of millions of young scientists for a number of years without being able to offer continuous, long-term stable investigative careers to the majority of them' (Ioannidis, Boyack, & Klavans, 2014). Consequently, there is a climate of fierce competition for increasingly limited funding (Anderson, Ronning, Vries, & Martinson, 2007b). This 'publish or perish' culture places inappropriate demands on a research process that should ideally be impartial and puts the integrity of the scientific enterprise in significant jeopardy (Fanelli, 2010).

1.2 Great expectations

Why has productivity become such an important factor in the scientific ecosystem? At the symposium, John Ioannidis, Professor of Medicine at Stanford University, argued that science's great success stories have led to unrealistic expectations. There is a pressure for research papers to be coherent, flawless narratives, but this only masks the scientific process in a veneer of perfection; the findings of most scientific studies are in reality highly nuanced (Giner-Sorolla, 2012). The problem is compounded by an increasing

trend towards publication of short empirical reports. This mode of publication might facilitate the rapid dissemination of new findings, but it could also incur the cost of inflated false-positive rates, reduced integration of new studies with the existing literature, and promote an unhealthy focus on seeking eye-catching 'newsworthy' effects over rigorous theory-driven experimentation (Ledgerwood & Sherman, 2012). The constant call for new discoveries, life changing innovations, and a publishing system that strongly favours positive results, puts unreasonable pressures on scientists that may encourage or coerce them to engage in behaviours that benefit their careers, but are inconsistent with good scientific practice (Fanelli, 2010).

1.3 Widespread questionable research practices

On a spectrum of scientific behaviours ranging from intentional misconduct (e.g., falsification, fabrication, and plagiarism) to flawless research conduct, the remaining 'grey area' is populated by a variety of questionable research practices (QRPs). QRPs describe a range of activities that intentionally or unintentionally distort data in favour of a researcher's own hypotheses (John et al., 2012; Simmons et al., 2011). These include 'cherry picking': omitting outcomes, variables, or conditions that do not support the author's own beliefs (Chan, Hrobjartsson, Haahr, Gotzsche, Altman, 2004); 'HARKing': hypothesising after the results are known to give the more compelling impression that findings were predicted *a priori* (Kerr, 1998); and 'p-hacking': prematurely examining data and exploiting techniques that may artificially increase the likelihood of meeting the standard statistical significance criterion (typically $\alpha = .05$), for example, making stop/continue data collection decisions (Armitage, McPherson, & Rowe, 1969), or engaging in post-hoc exclusion of outlier values (Bakker & Wicherts, 2014).

It seems clear that the damaged scientific ecosystem is partly to blame for the widespread engagement in QRPs (Fanelli, 2010; Bakker et al., 2012). However, they could also be an inevitable consequence of biases inherent to human cognition and thus difficult to overcome. For example, confirmation bias describes an effect whereby an individual preferentially seeks, interprets, and remembers information in a way that is consistent with their pre-existing beliefs (Nickerson, 1998). Although confirmation bias has only been sparsely investigated in scientific situations, some evidence indicates that scientists tend to discount evidence that might disconfirm their theoretical preferences (Brewer & Chin, 1994).

It is of great concern that QRPs are not just confined to a small subsection of the scientific community, but rather widespread and considered by many to be 'defensible' (John et al., 2012; Martinson, Anderson, de Vries, 2005). When used in a single study, these QRPs increase the likelihood of making a false-positive finding (Simmons et al., 2011). When employed on a large scale, such practices could have a devastating impact on the validity of the entire field of scientific inquiry (Ioannidis, 2005).

1.4 Unwillingness to share data

Even when it is accepted that false-positive findings are an inevitable by-product of the research process, much faith is placed in the notion of science as a 'self-correcting' enterprise (Merton, 1942). The idea is that spurious findings will eventually be exposed and purged whilst accurate findings will prevail. In order to facilitate self-correction, it is essential that scientists are open about their work so that it can be checked and repeated by their peers. Transparency is often considered to be a fundamental tenet of scientific investigation and many scientists subscribe to the norm of *communality*, which entails 'common ownership of scientific results and methods and the consequent imperative to share both freely' (Anderson, Ronning, De Vries & Martinson 2010; Merton, 1942). Unfortunately, at the symposium, Dr Jelte Wicherts (Tilburg University) depicted an aspect of the scientific ecosystem that contrasts vividly with this norm.

Data sharing is an important aspect of self-correcting science because it allows scientists to verify original analyses, conduct novel analyses, or carry out meta-analyses that can establish the reliability and magnitude of reported effects (Sieber, 1991; Tenopir, Allard, Douglass, Aydinoglu, Wu, Read, Manoff, & Frame, 2011). Wicherts described a 2006 paper in which attempts were made to access the data of 141 articles published in prominent psychology journals (Wicherts et al., 2006). Despite guidelines from the American Psychological Association (APA, 2001: 396) that compelled them to do so, 73% of authors did not share their data (for a similar finding in the biological sciences see Vines, Albert, Andrew, Débarre, Bock et al., 2014). Another concerning finding emerged when a subset of these papers was examined in greater detail: unwillingness to share data was associated with a higher prevalence of statistical reporting errors, particularly when those errors favoured an interpretation of the study's findings as statistically significant (Wicherts, Bakker, & Molenaar, 2011).

More generally, Wicherts and colleagues have found that statistical reporting errors are commonplace in the psychological literature (Bakker & Wicherts, 2011). Based on a reanalysis of 281 articles, the researchers estimated that around 18% of statistical results in the psychological literature are incorrectly reported. Similar to the findings outlined above, the majority of these errors support an interpretation of the study's results as statistically significant even though they were not. This is troubling as it suggests that researcher errors do not simply add noise to the research process, they introduce a systematic bias towards positive findings (Sarewitz, 2012). A field riddled with suspect statistics, QRPs, and a concomitant unwillingness to share data, is in danger of perpetuating falsehoods rather than establishing truths (Ioannidis, 2012).

1.5 Bad apples and a rotten barrel

At the symposium, Melissa Anderson, Professor of Higher Education at the University of Minnesota, argued that historically research governance has largely relied upon the self-regulation of scientists. This tendency has been motivated by faith in the scientific process to recruit individuals who are fit for the job and to weed out any 'funny business'. Underpinning this is a set of assumptions about the integrity and infallibility of scientists. Firstly, there is an implicit supposition that scientists are 'good people', motivated largely by the pursuit of knowledge. Scientists are also considered to be highly trained professionals who have undergone rigorous examinations and interviews. It is often assumed that rare cases of misconduct will be addressed by science's various mechanisms of self-correction: procedures such as peer-review, ethics committees, and study replication are all expected to filter 'bad science' from the system (Anderson et al., 2010; Merton, 1942).

Scientists are also subject to various legal and ethical protocols intended to promote research integrity. However, a recent examination of these protocols, presented during a symposium poster session, suggested that in Europe there is a complex system of overlapping regulatory bodies providing guidelines that vary considerably between countries and institutions (also see Godecharle, Nemery, & Dierickx, 2013). For example, there was considerable heterogeneity in the definition of 'misconduct' and the proposed mechanisms for dealing with it. It is hard to see how regulations characterised by such disunity and incoherence can provide effective oversight of integrity in the day-to-day workings of science.

It is also noteworthy that regulatory regimes are largely focused on dealing with researchers who engage in intentional misconduct. Anderson outlined how regulation is geared towards protecting scientific integrity from these 'bad apples'. However, she also highlighted the critical difference between 'misconduct' and 'misbehaviour'. According to US federal law, research misconduct is defined as fabrication, falsification, or plagiarism (Office of Science and Technology Policy, 2000). It must represent

a 'significant' departure from 'proper practice' and be 'intentional'. Research misbehaviour on the other hand, comprises more ambiguous activities, such as the QRPs highlighted earlier in this article (see Section 1.3). Throughout the symposium there was a general consensus that the scientific establishment should not only be concerned with the 'bad apples' that propagate full-blown research misconduct, but apply greater focus to the 'rotten barrel' that leads scientists to (perhaps unwittingly) engage in research misbehaviour.

2. Rehabilitating the scientific ecosystem

Whilst the symposium began by outlining threats to scientific integrity and possible causes, a variety of solutions were also proposed, some with fairly broad aims and others targeting specific issues. Many of the speakers stated that no single solution would provide a panacea, and suggested that multiple initiatives would be required. Several speakers and delegates proposed that funding should be invested in an empirical examination of research practices and potential solutions in order to ensure their effectiveness. Perhaps it is time for scientists to turn their microscopes upon themselves and examine how their own behaviour, intentional or otherwise, distorts the scientific process? Other attendees of the symposium were keen to seize upon the current momentum for change and begin repairing the scientific ecosystem as soon as possible. In practice, many of the solutions outlined below are already being implemented, but in an incremental and voluntary fashion. In this section, we evaluate the solutions proposed at the symposium and examine the idea that, ultimately, rehabilitation of the scientific ecosystem will require considerable cultural change.

2.1 Changing incentives

Section 1.1 commented on how the scientific ecosystem's incentive structure is grossly misaligned with the principles of good science. At the symposium Professor Ioannidis proposed an ambitious scheme for appraising and rewarding research: a new metric that captures productivity, quality, reproducibility, shareability, and translatability (PQRST; Ioannidis & Khoury, 2014). The idea is to diversify the types of scientific activity that are rewarded in order to prevent productivity becoming scientists' principal goal. Practically speaking, it should be reasonably straightforward to estimate productivity using existing measures (for example, the proportion of registered clinical trials on ClinicalTrials.gov published two years after study completion), but the remaining parameters would require adding new features to scientific databases. For example, to calculate a 'shareability' index databases would need to monitor whether authors have uploaded their data to a public repository. Given the conflicting interests that influence the scientific ecosystem, it seems that reaching agreement on which quality standards are appropriate to use will be a more considerable barrier to change. Ioannidis hopes that realigning incentive structures with principles of good science will reduce the prevalence of scientific misbehaviours like QRPs and unwillingness to share data.

2.2 Scientific integrity training

Changing incentive structures may help to address intentional engagement in QRPs; however, it is also plausible that many QRPs are employed unwittingly simply because researchers are not fully aware of the extent to which these practices are problematic. The issue of integrity training was raised repeatedly at the symposium, but interestingly these proposals were largely directed at educating junior scientists. The poster on European research misconduct regulations, presented by Godecharle and colleagues, also reflected this: only Irish guidelines mentioned providing training to senior scientists (see Godecharle et al., 2013).

Some research suggests that the effectiveness of formal ethical training might be limited in comparison to the influence of lab culture or mentoring (e.g., Anderson,

Horn, Risbey, Ronning, DeVries, & Martinson, 2007a). At the symposium, Anderson proposed that principal investigators should improve awareness of research integrity amongst junior researchers through lab-based discussions, and should seek to engage students by employing relevant real-life examples. For instance, mentoring sessions could utilise role-play in which researchers confront ambiguous research scenarios they might actually find themselves in. This would constitute a shift away from simply briefing students on the regulations and protocols that they are expected to follow as researchers. Instead it would concentrate on highlighting the difficulties of conducting research and show them how to solve problems in a realistic environment. However, we note that focusing training efforts solely on junior scientists may not be sufficient to address the present threats to scientific integrity whilst engagement in QRPs is widespread amongst senior scientists (John et al., 2012).

2.3 Preregistration of study protocols

Even if changing incentives and introducing training schemes are effective in improving scientific integrity, they may not be sufficient to eliminate the influence of QRPs that could arise as a consequence of biases inherent in human cognition (see Section 1.3). A potential solution to this problem, *preregistration*, was introduced to the symposium by Eric-Jan Wagenmakers, Professor of Cognitive Science at the University of Amsterdam. The central premise of preregistration is that researchers specify a methodology, sample size, and data analysis plan prior to conducting a study. This preregistration document can be uploaded to a public repository, such as The Open Science Framework (OSF) and referred to in any subsequent paper that reports the study. A stronger version of preregistration involves the submission of the preregistration document to a journal where, assuming the study is of satisfactory methodological quality, it will be accepted on the basis of the preregistration alone, and the journal would be committed to publishing the study *regardless of the results*.

The anticipated benefits of preregistration are two-fold. Primarily, it would prevent researchers from engaging in many QRPs because they are held to account by their own preregistration document. For example, it would be impossible for a researcher to engage in 'cherry picking', inappropriate post-hoc outlier exclusion, data 'peeking', or HARKing (see Section 1.3; John et al., 2012), when the relevant parameters have been specified prior to data collection. Furthermore, journal-based preregistration would help to address publication bias by ensuring that publication is dependent primarily upon methodological quality rather than the nature of the results (Chambers, 2013). This would help to reduce the 'file-drawer' problem (Rosenthal, 1979) whereby findings that do not achieve statistical significance are considerably less likely to be published—a state of affairs that drives the current publication bias towards positive findings (Fanelli, 2012) and undermines the validity of the academic literature (Ioannidis, 2005).

Several members of the symposium audience pointed out potential problems with preregistration. For example, it was suggested that there would be nothing stopping a scientist from engaging in QRPs and then 'preregistering' a study that they had in fact already completed. This is true, countered Wagenmakers, but in a preregistration scheme, such practices would clearly be fraud, and thus only likely to be committed by a small minority. A more practical criticism was that preregistration could increase workload because it involves two stages of peer review: prior to data collection to evaluate methodology and after data collection to evaluate adherence to the preregistration plan. However, Chambers et al. (2014) argue that journal-based preregistration could in fact save time. In the current publication system it is common for a manuscript to be submitted and reviewed at multiple journals, often being rejected

several times based on either methodological problems or because the results are not deemed 'interesting'. However, in a preregistration scheme, studies are primarily judged on their methodological quality, which is established prior to the study being run. Thus, a more thorough reviewer-author interaction at the pre-data collection stage will ultimately reduce the likelihood that the research has to undergo several rounds of submission and review at multiple journals. A system that scrutinises research protocols and methods prior to commencing data collection could also be helpful for authors. Under the current system, irreparable methodological issues may only come to light when authors have already invested time and money in running the study. Whereas, in the new system authors would receive feedback about their proposals before commencing the study, allowing for improvements to be made. Overall then, the time-cost for authors, reviewers and editors could be negligible or even an improvement compared to the present system.

Other delegates objected on the grounds that preregistration may shackle science by outlawing creative post-hoc explorations of data or restricting observational research (see also Scott, 2013). But Wagenmakers argued that preregistered studies could still include post-hoc exploratory analyses that the authors and reviewers believe to be appropriate. By using preregistration, a clear distinction would be made between confirmatory analyses specified in the preregistration, and exploratory analyses inspired by the data (see Wagenmakers, Wetzels, Borsboom, van der Maas, & Kievit, 2012). Readers could then treat author claims with the appropriate degree of skepticism depending on the status of their analysis. Furthermore, fraudulent preregistration could backfire, as editors are likely to require revisions to the proposed protocol (Chambers, Feredoes, Muthukumaraswamy, & Etchells, 2014). Thus, even relatively minor changes to the experimental procedure would be impossible if the study had already been completed.

2.4 Transparency through data sharing

Preregistration may address integrity issues prior to and during data collection, but the studies described earlier by Jelte Wicherts and colleagues suggest a widespread unwillingness to share data with fellow scientists *after* findings have been published (Wicherts et al., 2006). Wicherts believes that his work describes a culture of secrecy in which misconduct can flourish, and he has built a strong case for obligatory data sharing in the scientific community (Wicherts, 2011).

However, there are practical and ethical issues to overcome. Martin Bobrow (2013) for example, agrees that there is, 'an ethical imperative…to maximize the value of research data' but also acknowledges the need to be cautious, as there is a risk of individuals being identified in sensitive datasets. Bobrow suggests that as more research is shared it is important to assess how these data are being used, to examine the risks, and to devise appropriate governance that balances privacy with public benefit.

In the neuroimaging community, concerns have been raised about the various technical issues involved in sharing large and complex brain imaging data (Nature Neuroscience Editorial, 2000). A more general issue that has arisen from this debate is that many researchers fear being 'scooped' if discoveries are made using their dataset before they have been able to finish analysing the data themselves. These concerns appear to be an unfortunate consequence of a scientific ecosystem that incentivises productivity in terms of publications and fails to account for other activities that contribute to credible scientific inquiry (see Section 1). The PQRST metric proposed by Ioannidis and Khoury (2014; see Section 2.1) explicitly incorporates 'shareability' as an index of scientific quality.

Generally speaking, professional guidelines, for example those provided by the American Psychological Association, do not appear

to offer sufficient compulsion for authors to share their data. At present, data sharing policies vary substantially across journals (Alsheikh-Ali, Qureshi, Al-Mallah, & Ioannidis, 2011) and Wicherts recommends that journals require from authors to upload their data to a public repository (e.g., The OSF) along with a relevant codebook so that other researchers can navigate the dataset. Although this may generate additional work for the original author in terms of preparing the dataset for other users, Wicherts argues that data sharing in this manner is an essential component of transparent scientific practice.

2.5 Cultural change

Some of the solutions outlined above have either been met with resistance, or at least not fully embraced by the scientific community (e.g., Scott, 2013). There is some evidence suggesting that scientists are generally open to change, but wary of new schemes and regulations that might impose rigidity on their practice (Fuchs, Jenny & Fiedler, 2012). A more comprehensive solution to the current problems faced by science would comprise a wholesale rehabilitation of scientific culture, in tandem with some of the more practical initiatives proposed above.

Individual scientists rarely work in isolation, typically operating in teams, situated within departments and institutions, and interacting with colleagues in their disciplinary field through publications, attendance at conferences, and informal communications both public (e.g., social media) and private (e.g., e-mail). These different communities each have a cultural identity and establish proximal norms that influence the behaviour of community members. In order for any of the previously proposed procedures or regulations to be effective, the culture of science may need to shift so that individuals are supported by their colleagues to make the right decisions.

In a culture where scientists have to 'play the game' to survive (Bakker et al., 2012), it is hard for an individual scientist to prioritise the integrity of their research. Martinson et al. (2005) found a significant association between self-reported scientific misbehaviour and perceived inequities in the funding allocation process. These findings suggest that when people feel 'wronged' or are working in a climate they believe to be rife with competition and power games, they are more likely to prioritise the success of their own careers over behaviours that support credible scientific inquiry. Anderson also described a study (unpublished data) in which 7,000 mid-career and early-career researchers were asked whether they had ever engaged in either research misconduct or misbehaviour. A very modest number reported misconduct, but many reported misbehaviour. Researchers were also asked to report what they thought about other researchers' engagement in these practices. Interestingly, a positive correlation was identified between those who self-reported increased levels of research misconduct or misbehaviour, and the extent to which they perceived others were engaged in such practices. This depicts a scientific ecosystem in which individuals are more likely to engage in misconduct and misbehaviour if they think others around them are too.

At the symposium, Anderson proposed a number of initiatives that sought to challenge the current scientific culture. There is some evidence to suggest that signing an institutional research integrity oath or honour code, and receiving reminders of these agreements, could reduce research misbehaviour. In an experiment with students at MIT and Yale, Mazar, Amir and Ariely (2008) found that simply printing the statement 'I understand that this…falls under [MIT's/Yale's] honor system' on test papers significantly reduced cheating regardless of the incentive offered and despite no real honor code existing at these institutions. Whilst this is a promising finding, the idea remains to be investigated in research settings involving real research misconduct or misbehaviour, where the stakes are higher, and the factors influencing engagement in QRPs are diverse. Perhaps journal submission portals

or PhD vivas could require researchers to sign a research integrity code when submitting a manuscript or thesis? It would also be important to consider how an honor code could be applied to complex 'grey area' behaviours, since the usual mechanisms are clearly insufficient for regulating research misbehaviour.

3. Conclusion

Whilst the issues faced by the scientific disciplines are alarming, it is exciting to be part of a community that is reflecting critically on an unsustainable status quo. Many of the current issues have been raised previously but change has not been forthcoming. The main differences this time are an increased awareness about these issues within the scientific community and widespread access to technological apparatus that can support inventive and accessible solutions.

However these are also unsettling times for young researchers finding their feet in a scientific system that appears to have drifted far from its principal goal of truth-seeking. In his book *Advice For A Young Investigator*, the neuroscientist Ramón y Cajal suggests that 'two emotions must be unusually strong in the great scientific scholar: a devotion to truth and a passion for reputation' (Ramón y Cajal,1897/1999: 40). Yet in a scientific ecosystem that rewards researchers for their productivity more than for their methodological rigor, a young investigator who is fully devoted to the truth cannot afford to be passionate about their reputation, and a young investigator passionate about their reputation cannot afford to be fully devoted to the truth. It is time to rehabilitate the scientific ecosystem, and the first step is to acknowledge that scientists are only human.

Acknowledgements

The authors would like to thank the Research Department of Experimental Psychology, University College London, for the provision of travel subsidies that enabled attendance at this event. We are grateful to David Shanks and Kari Vyas for their comments on an earlier draft of this article. We would also like to thank the organisers of the symposium addressed in this paper for arranging such an enjoyable and thought-provoking array of speakers on the topic 'Human Factors in Science'.

References

Alsheikh-Ali, A A, Qureshi, W, Al-Mallah, M H, and **Ioannidis, J P A** 2011 Public availability of published research data in high-impact journals. *PLoS ONE*, 6(9): e24357. DOI: http://dx.doi.org/10.1371/journal.pone.0024357

American Psychological Association 2001 Ethical standards for the reporting and publishing of scientific information. In *Publication Manual of the American Psychological Association* (5th ed., pp. 387–396). Washington, DC.

Anderson, M S, Horn, A S, Risbey, K R, Ronning, E A, De Vries, R, and **Martinson, B C** 2007a What do mentoring and training in the responsible conduct of research have to do with scientists' misbehavior? Findings from a national survey of NIH-funded scientists. *Academic Medicine: Journal of the Association of American Medical Colleges,* 82(9), 853–60. DOI: http://dx.doi.org/10.1097/ACM.0b013e31812f764c

Anderson, M S, Ronning, E A, De Vries, R, and **Martinson, B C** 2007b The perverse effects of competition on scientists' work and relationships. *Science and Engineering Ethics*, 13: 437–461. DOI: http://dx.doi.org/10.1007/s11948-007-9042-5

Anderson, M S, Ronning, E A, De Vries, R, and **Martinson, B C** 2010 Extending the Mertonian norms: Scientists' subscription to norms of research. *The Journal of Higher Education,* 81(3): 366-393. DOI: http://dx.doi.org/10.1353/jhe.0.0095

Armitage, P, McPherson, C K, and **Rowe, B C** 1969 Repeated significance tests on accumulating data. *Journal of the Royal Statistical Society Series A (General)*, 132(2): 235–244. DOI: http://dx.doi.org/10.2307/2343787

Bakker, M, van Dijk, A, and **Wicherts, J M** 2012 The rules of the game called psychological science. *Perspectives on Psycho-*

logical Science, 7: 543–554. DOI: http://dx.doi.org/10.1177/1745691612459060

Begley, C G, and **Ellis, L M** 2012 Drug development: Raise standards for preclinical cancer research. *Nature* 483(7391): 531–533. DOI: http://dx.doi.org/10.1038/483531a

Bobrow, M 2013 Balancing privacy with public benefit, *Nature,* 500 (7461): 123. DOI: http://dx.doi.org/10.1038/500123a

Brewer, W F, and **Chinn, C A** 1994 Scientists' responses to anomalous data: Evidence from psychology, history, and philosophy of science. *Proceedings of the Biennial Meeting of the Philosophy of Science Association*, 1: 304–313.

Button, K S, Ioannidis, J P A, Mokrysz, C, Nosek, B A, Flint, J, Robinson, E S J, and **Munafo, M R** 2013 Power failure: why small sample size undermines the reliability of neuroscience. *Nature Reviews Neuroscience*, 14: 365–376. DOI: http://dx.doi.org/10.1038/nrn3475

Carp, J 2012 The secret lives of experiments: Methods reporting in the fMRI literature. *NeuroImage*, 63: 289–300. DOI: http://dx.doi.org/10.1016/j.neuroimage.2012.07.004

Chambers, C D 2013 Registered Reports: A new publishing initiative at Cortex. *Cortex*, 49(3), 609–610. DOI: http://dx.doi.org/10.1016/j.cortex.2012.12.016

Chambers, C D, Feredoes, E, Muthukumaraswamy, S D, and **Etchells, P J** 2014 Instead of "playing the game" it is time to change the rules: Registered Reports at AIMS Neuroscience and beyond. *AIMS Neuroscience*, 1(1) 4–17. DOI: http://dx.doi.org/10.3934/Neuroscience2014.1.4

Chan, A W, Hrobjartsson, A, Haahr, M T, Gotzsche, P C, and **Altman, D G** 2004 Empirical evidence for selective reporting of outcomes in randomized trials: Comparison of protocols to published articles. *JAMA*, 291: 2457–2465. DOI: http://dx.doi.org/10.1001/jama.291.20.2457.

[Editorial] 2000 A debate over fMRI data sharing. *Nature Neuroscience*, 3(9): 845-846. DOI: http://dx.doi.org/10.1038/78728

Fanelli, D 2010 Do pressures to publish increase scientists' bias? An empirical support from US states data. *PLoS ONE*, 5: e10271. DOI: http://dx.doi.org/10.1371/journal.pone.0010271

Fanelli, D 2012 Negative results are disappearing from most disciplines and countries. *Scientometrics*, 90, 891–904. DOI: http://dx.doi.org/10.1007/s11192-011-0494-7

Fuchs, H M, Jenny, M, and **Fiedler, S** 2012 Psychologists are open to change, yet wary of rules. *Perspectives on Psychological Science*, 7(6): 639–642. DOI: http://dx.doi.org/10.1177/1745691612459521

Gilbert, E A 2014 Reproducibility project: Psychology (preliminary) results. Retrieved from: https://osf.io/vtksf/

Giner-Sorolla, R 2012 Science or art? How aesthetic standards grease the way through the publication bottleneck but undermine science. *Perspectives on Psychological Science*, 7(6), 562–571. DOI: http://dx.doi.org/10.1177/1745691612457576

Godecharle, S, Nemery, B, and **Dierickx, K** 2013 Guidance on research integrity: no union in Europe. *The Lancet*, 381(9872): 1097–1098. DOI: http://dx.doi.org/10.1016/S0140-6736(13)60759-X

Hirsch, J E 2005 An index to quantify an individual's scientific research output. *PNAS*, 102: 16569–16572. DOI: http://dx.doi.org/10.1073/pnas.0507655102

Ioannidis, J P A 2005 Why most published research findings are false. *Perspectives in Psychological Science,* 7(6): 645–654. DOI: http://dx.doi.org/10.1371/journal.pmed.0020124

Ioannidis, J P A 2012 Why science is not necessarily self-correcting. *PLoS Medicine* 2(8): e124. DOI: http://dx.doi.org/10.1177/1745691612464056

Ioannidis, J P A, Boyack, K W, and **Klavans, R** 2014 Estimates of the continuously publishing core in the scientific workforce. *PloS*

One, 9(7), e101698. DOI: http://dx.doi.org/10.1371/journal.pone.0101698

Ioannidis, J P A, and **Khoury, M J** 2014 Assessing value in biomedical research: the PQRST of appraisal and reward. *JAMA*, 312(5), 483–4. DOI: http://dx.doi.org/10.1001/jama.2014.6932

John, L, Loewenstein, G, and **Prelec, D** 2012 Measuring the prevalence of questionable research practices with incentives for truth-telling. *Psychological Science* 23(5), 524–32. DOI: http://dx.doi.org/10.1177/0956797611430953

Kerr, N L 1998 HARKing: Hypothesizing after the results are known. *Personality and Social Psychology Review*, 2(3): 196–217. DOI: http://dx.doi.org/10.1207/s15327957pspr0203_4

Ledgerwood, A, and **Sherman, J W** 2012 Short, sweet, and problematic? The rise of the short report in psychological science. *Perspectives on Psychological Science*, 7(1): 60–66. DOI: http://dx.doi.org/10.1177/1745691611427304

Levelt Committee, Noort Committee, and Drenth Committee 2012 Flawed science: The fraudulent research practices of social psychologist Diederik Stapel. https://www.commissielevelt.nl/wp-content/uploads_per_blog/commissielevelt/2013/01/finalreportLevelt1.pdf

Mahoney, M J 1976 *Scientist as subject: The psychological imperative*. Cambridge, MA: Ballinger Publishing Company.

Martinson, B C, Anderson, M S, and **de Vries, R** 2005 Scientists behaving badly. *Nature,* 435(7034): 737–738. DOI: http://dx.doi.org/10.1038/435737a

Mazar, N, Amir, O, and **Ariely, D** 2008 The dishonesty of honest people: A theory of self-concept maintenance. *Journal of Marketing Research,* 45(6): 633–644. DOI: http://dx.doi.org/10.1509/jmkr.45.6.633

Merton, R K 1942 The Normative Structure of Science. In: Storer, N W 1973 *The Sociology of Science*. Chicago: University of Chicago Press. pp 267–278.

Nickerson, R S 1998 Confirmation bias: A ubiquitous phenomenon in many guises. *Review of General Psychology*, 2: 175–220. DOI: http://dx.doi.org/10.1037/1089-2680.2.2.175

Office of Science and Technology Policy 2000 Federal Research Misconduct Policy. *Federal Register*, 65(235): 76260–76264.

Prinz, F, Schlange, T, and **Asadullah, K** 2011 Believe it or not: how much can we rely on published data on potential drug targets? *Nature Reviews Drug Discovery* 10: 712. DOI: http://dx.doi.org/10.1038/nrd3439-c1

Ramón y Cajal, S 1897/1999 Advice for a young investigator. Translated by Swanson N & Swanson L W. Cambridge, MA: MIT Press.

RIKEN Research Paper Investigative Committee 2014 Report on STAP Cell Research Paper Investigation. Retrieved from: http://www3.riken.jp/stap/e/f1document1.pdf

Rosenthal, R 1979 The "file drawer problem" and tolerance for null results. *Psychological Bulletin*, 86(3): 638–641. DOI: http://dx.doi.org/10.1037/0033-2909.86.3.638

Sarewitz, D 2012 Beware the creeping cracks of bias. *Nature*, 485: 149. DOI: http://dx.doi.org/10.1038/485149a

Scott, S 2013 Pre-registration would put science in chains. *Times Higher Education*. Retrieved from: http://www.timeshighereducation.co.uk/comment/opinion/science-in-chains/2005954.article

Sieber, J E 1991 Openness in the social sciences: Sharing data. *Ethics & Behavior*, 1: 69–86, DOI: http://dx.doi.org/10.1207/s15327019eb0102_1

Simmons, J P, Nelson, L D, and **Simonsohn, U** 2011 False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science*, 22(11): 1359–1366. DOI: http://dx.doi.org/10.1177/0956797611417632

Tenopir, C, Allard, S, Douglass, K, Aydinoglu, A U, Wu, L, Read, E, Manoff, M, and **Frame, M** 2011 Data sharing by scientists: Practices and perceptions. *PLoS ONE*, 6: e21101. DOI: http://dx.doi.org/10.1371/journal.pone.0021101

Vasilevsky, N A, Brush, M H, Paddock, H, Ponting, L, Tripathy, S J, LaRocca, G M, and **Haendel, M A** 2013 On the reproducibility of science: unique identification of research resources in the biomedical literature. *PeerJ* 1: e148. DOI: http://dx.doi.org/10.7717/peerj.148

Vines, T H, Albert, A Y K, Andrew, R L, Debarre, F, Bock, D G, Franklin, M T, Gilbert, K J, Moore, J S, Renaut, S, and **Rennison, D J** 2014 The availability of research data declines rapidly with article age. *Current Biology*, 24: 94–97. DOI: http://dx.doi.org/10.1016/j.cub.2013.11.014

Wicherts, J M 2011 Psychology must learn a lesson from fraud case. *Nature*, 480: 7. DOI: http://dx.doi.org/10.1038/480007a

Wicherts, J M, Bakker, M, and **Molenaar, D** 2011 Willingness to Share Research Data Is Related to the Strength of the Evidence and the Quality of Reporting of Statistical Results. *PLoS ONE*, 6(11). DOI: http://dx.doi.org/10.1371/journal.pone.0026828

Wicherts, J M, Borsboom, D, Kats, J, and **Molenaar, D** 2006 The poor availability of psychological research data for reanalysis. American Psychologist, 61(7): 726–728. DOI: http://dx.doi.org/10.1037/0003-066X.61.7.726

Young, N S, Ioannidis, J P A, and **Al-Ubaydli, O** 2008 Why current publication practices may distort science. *PloS Medicine*, 5: e201. DOI: http://dx.doi.org/10.1371/journal.pmed.0050201

How to cite this article: Hardwicke, T E, Jameel, L, Jones, M, Walczak, E J and Magis-Weinberg, L 2014 Only Human: Scientists, Systems, and Suspect Statistics. *Opticon1826,* (16): 25, pp. 1-12, DOI: http://dx.doi.org/10.5334/opt.ch

Published: 22 December 2014

]u[*Opticon1826* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS

Pernet and Poline *GigaScience* (2015) 4:15
DOI 10.1186/s13742-015-0055-8

(GIGA)n SCIENCE

COMMENTARY Open Access

Improving functional magnetic resonance imaging reproducibility

Cyril Pernet[1*] and Jean-Baptiste Poline[2*]

Abstract

Background: The ability to replicate an entire experiment is crucial to the scientific method. With the development of more and more complex paradigms, and the variety of analysis techniques available, fMRI studies are becoming harder to reproduce.

Results: In this article, we aim to provide practical advice to fMRI researchers not versed in computing, in order to make studies more reproducible. All of these steps require researchers to move towards a more open science, in which all aspects of the experimental method are documented and shared.

Conclusion: Only by sharing experiments, data, metadata, derived data and analysis workflows will neuroimaging establish itself as a true data science.

Keywords: Functional MRI, Reproducibility, Scripts, Workflows, Code, Open science

> "Experience has shown the advantage of occasionally rediscussing statistical conclusions, by starting from the same documents as their author. I have begun to think that no one ought to publish biometric results, without lodging a well arranged and well bound manuscript copy of all his data, in some place where it should be accessible, under reasonable restrictions, to those who desire to verify his work." Galton 1901 [1]

Introduction

Because current research is based on previous published studies, being able to reproduce an experiment and replicate a result is paramount to scientific progress. The extent to which results agree when performed by different researchers defines this tenet of the scientific method [2,3]. Recently, a number of authors have questioned the validity of many findings in epidemiology or in neuroscience [4,5]. Results can be found by chance (winner's curse effect), more often in poorly powered studies [6], or be declared significant after too many variations of the analysis procedure [7,8] without controlling appropriately for the overall risk of error (p-hacking effect [6,9]). Additionally, errors in code or in data manipulation are easy to make [10]: it is in general difficult to check for the correctness of neuroimaging analyses. Reproduction is one way to address these issues, given that the probability of a research finding being true increases with the number of reproductions (see Figure two in [4]).

If the reliability of a large proportion of functional magnetic resonance imaging (fMRI) results is questionable, this has serious consequences for our community. Mostly, this means that we are building future work on fragile ground. Therefore we need to ensure the validity of previous results. It is very possible, and some argue likely, that we - as a community - are wasting a large amount of our resources by producing poorly replicable results. We can, however, address the current situation on several fronts. First, at the statistical analysis level, one proposed solution is to be more disciplined and use pre-registration of hypotheses and methods [11]. Providing information about planned analyses and hypotheses being tested is crucial, as it determines the statistical validity of a result, and therefore the likelihood that it will be replicated. This would bring us closer to clinical trial procedures, leading to much more credible results. It

* Correspondence: cyril.pernet@ed.ac.uk; jbpoline@gmail.com
[1]Centre for Clinical Brain Sciences, Neuroimaging Sciences, University of Edinburgh Chancellor's Building, 49 Little France Crescent, Edinburgh EH16 4SB, UK
[2]Henry H Wheeler, Jr Brain Imaging Center, Helen Wills Neuroscience Institute, University of California at Berkeley, 3210 Tolman Hall, Berkeley, CA 94720-1650, USA

does not remove the possibility of analyzing data in an exploratory manner, but in that case p-values should not be attached to the results. Pre-registration is an effective solution to address the growing concern about poor reproducibility, as well as the 'file drawer' issue [9,12]. Second, we propose that better procedures and programming tools can improve the current situation greatly. We specifically address this question, because many of the researchers using fMRI have limited programming skills.

Although we aim for reproduction of results with other data and independent analysis methods, the first step is to ensure that results can be replicated within laboratories. This seems an easy task, but it is in fact common that results cannot be replicated after, say, a year or two, when the student or post-doc responsible for the analyses and the data management has left. Increasing our capacity to replicate the data analysis workflow has another crucial aspect: this will allow us to better document our work, and therefore communicate and share it much more easily. It is crucial that we remember that resources are limited, and part of our work is to make it easy for others to check and build upon our findings.

In computer science and related communities, a number of informatics tools and software are available (databases, control version system, virtual machines, etc.) to handle data and code, check results and ensure reproducibility. Neuroscientists working with functional MRI are, however, largely from other communities such as biology, medicine and psychology. Because of the differences in training and the field of research, such informatics tools are not necessarily sufficient, and are certainly not fully accessible to or mastered by all researchers. In this review, we address specifically the community of neuroscientists with little programming experience, and point to a number of tools and practices that can be used today by anyone willing to improve his or her research practices, with a view to better reproducibility. We also recommend observing how other communities are improving their reproducibility. For instance, B Marwick [13] gives an excellent summary of these issues and some solutions for the social sciences, and many of his recommendations may be shared between fields. Improving the capacity of other researchers to reproduce one's results involves some degree of sharing, through journals, repositories or dedicated websites (Annex 1). These practices, if followed, should be sufficient to allow any researcher to replicate a published fMRI experiment. Here we define replication as the capacity of a colleague to re-execute the analyses on the same dataset [14], but note that this definition varies in the literature [15]. In step 2 below ('Improving scripts and turning them into workflows'), we expand on good practice for writing and sharing code. Although this can seem daunting for people who do not often write code, our goal is to give some tips to improve everyone's analysis scripts.

Reproducible neuroimaging in 5 steps

We define reproducibility as the ability of an entire experiment to be reproduced [16], from data acquisition to results. In some fields, such as computational neuroscience, reproducibility can be readily dissociated from replicability, which is the capacity for exact analytical reproduction of the analysis pipeline, possibly using the same data [14,15]. For fMRI, as for other fields, reproduction is more of a continuum: analytic reproduction (the replication case), direct reproduction (reproducing a result using the same conditions, materials and procedures as in the original publication, but with other subjects), systematic reproduction (trying to obtain the same finding by using many different experimental conditions), and conceptual reproduction (reproducing the existence of a concept using different paradigms). The question we address here is to what extent we can share protocols, data, workflows and analysis code to make fMRI studies easier to replicate and directly reproduce.

Sharing experimental protocols

Every task-based fMRI study depends on an experimental procedure in which subjects are instructed to passively watch, listen, feel, taste, or smell, or to actively engage in a task. In all cases, stimuli are presented via a computer program that synchronizes with the MRI scanner. Although such procedures are always described in published articles, some details about the order of stimulus presentation, stimulus onset times or stimulus sizes, for example, can be missing. The issue is that such details can determine whether an effect is observed or not. It is therefore paramount to be able to replicate the experimental setup if one wants to reproduce a study. Sharing computer programs (and stimuli) is easily achievable: when publishing an article, the computer program can be made available either as supplementary material or, more usefully, through a repository. Repositories are large data storage servers with a website front-end that can be used to upload and share data publicly (e.g. Dryad [17], FigShare [18], OpenScience framework [19], or Zenodo [20]). A license allowing modification and resharing should be attached to these data to maximize the speed of research discoveries.

Document, manage and save data analysis batch scripts and workflows

Making analyses reproducible with limited programming skills

Functional MRI analyses are complex, involving many preprocessing steps as well as a multitude of possible statistical analyses. Even if the most important steps are reported using precise guidelines [21], there are too many parameters involved in the data analysis process to be able to provide a full description in any article. Carp [7] examined a simple event-related design using common neuroimaging tools, but varying the available settings (see also [8]). This

led to 6,912 unique analysis pipelines, and revealed that some analysis decisions contributed to variability in activation strength, location and extent, and ultimately to inflated false positive rates [4]. In the face of such variability, some have argued that 'anything less than release of actual source code is an indefensible approach for any scientific results that depend on computation, because not releasing such code raises needless, and needlessly confusing, roadblocks to reproducibility' [22].

In contrast with data analysts or software developers, many neuroimagers do not code their analysis from scratch - instead they rely on existing software and often reuse code gathered from others in the laboratory or on the web. Pressing buttons in a graphical user interface is not something that can be replicated, unless inputs and processing steps are saved in log files. To ensure reproducibility (even for oneself in a few months' time) one needs to set up an automatic workflow. Informatics and bioinformatics researchers have been discussing issues of code reproducibility for many years [23,24], and lessons can be learnt from their experience. Sandve et al. [24] have a few simple recommendations. First, keep track of every step, from data collection to results, and whenever possible keep track with electronic records. Most neuroimaging software has a so-called batch mode (SPM [25,26]) or pipeline engine (Nipype [27,28]), or is made up of scripts (AFNI [29,30], FSL [31,32]), and saving these is the best way to ensure that one can replicate the analysis. At each step, record electronically, and if possible automatically, what was done with what software (and its version). Second, minimize, and if possible eliminate, manual editing. For instance, if one needs to convert between file formats, this is better done automatically with a script, and this script should be saved. Third, for analyses that involve a random number generator, save the seed or state of the system, so that the exact same result can be obtained. As for the computer program used to run the experiment (step 1), the batch and scripts can be made available as supplementary material in a journal, and/or shared in repositories. If one ends up with a fully functional script that includes a new type of analysis, this can itself be registered as a tool on dedicated websites such as the NeuroImaging Tool and Resources Clearinghouse (NITRC [33]). Sharing the analysis batch and scripts is the only way to ensure reproducibility by allowing anyone to (i) check for potential errors that 'creep in' to any analyses [10]; (ii) reuse them on new data, possibly changing a few parameters to suit changes in scanning protocol - similar results should be observed if the effects were true [14] - and (iii) base new analysis techniques or further research on verifiable code.

Improving scripts and turning them into workflows

Although these recommendations are, we hope, useful, they are not generally sufficient. Analysis code depends on software, operating systems, and libraries that are regularly updated (see, e.g. [34] for an effect on imaging results). When the code is rerun, these changes should be tracked, and results attached to a specific version of the code and its environment. The only complete solution is to set up virtual machine or equivalent. For neuroimaging, the NeuroDebian project [35] integrates relevant software into the Debian operating system, where all software is unambiguously versioned and seamlessly available from a package repository. This makes it possible to define the whole environment and reconstruct it at any later time using snapshots of the Debian archive [36]. While such a solution is the most complete, investing in good revision control software is a first step that goes a long way in handling code (Wikipedia lists 36 types of such software [37]). We argue here that this investment is a necessity for reproducible science.

Although a simple text editor or word processing document could be used to precisely describe each analysis step, only an executable script and information on the associated software environment can give one a reasonable chance of reproducing an entire experiment. This implies that much more should be done to teach programming to students or researchers who need to work with neuroimaging data. Barriers to code sharing are not as great as for data, but they do exist. Researchers are often concerned that their code is too poor, and that there might be some errors. These, and the fear of being 'scooped', are some of the main reasons scientists give for not sharing code with others [38]. Yet, as Barnes [39] puts it, "software in all trades is written to be good enough for the job intended. So if your code is good enough to do the job, then it is good enough to release". A few simple rules can be applied to improve scripts [23]. First, make your code understandable to others (and yourself). Add comments to scripts, providing information not just about what is computed, but also reflecting what hypothesis is being tested, or question answered, by that specific piece of code [24]. Second, version control everything. Version control systems (VCSs) store and back up every previous version of the code, allowing one to 'roll back' to an older version of the code when things go wrong. Two of the most popular VCSs are Git [40] (which we recommend) and Subversion [41]. 'Social coding' platforms, such as GitHub [42] or Bitbucket [43], are also useful sharing and collaboration tools. Third, test your code effectively, to assure yourself and others that it does what it is supposed to. The software industry tells us that "untested code is broken code", but scientists lack incentives to invest time in this. For example, if you coded some statistical tests to be run on multiple voxels, compare the routine in one voxel against a prototype solution. Learning how to test and document one's code is a crucial skill to reduce bugs and ensure safe

reuse of code, an aspect that is not sufficiently emphasized and taught in curricula. In fact, the experience of the authors is that it is hardly ever mentioned.

Neuroimagers can also take advantage of a few easy-to-use tools to create complex scripts and make a workflow (a workflow consists of a repeatable pattern of activities that transform data and can be depicted as a sequence of operations, declared as work of a person or group (adapted from [44]). For Matlab-based analyses, we can recommend using Matlab-specific formatting[a] in the code, and a workflow engine such as the Pipeline System for Octave and Matlab (PSOM [45,46]) or the Automatic Analysis pipeline (AA [47,48]). For Python-based analyses, we recommend the IPython notebook ([49] now the Jupyter project) to sketch the analysis and explore results, along with the workflows provided in Nipype [27,28]. Packages such as SPM [25,26] have batch systems that create scripts of the whole analysis workflow, which should be learned for efficiency, reproducibility and provenance tracking. It is also possible to create entire workflows using general (e.g. Taverna [50], Kepler [51]) or dedicated libraries (LONI pipeline [52]) and thereby obtain analysis provenance information. Using these pipelines, one can create (via a graphical interface or a script) a workflow of the different steps involved in fMRI data processing, specifying parameters needed at each step, and save the workflow. Dedicated libraries or scripts can be called, and the impact of changing a parameter value in a specific implementation of a step can be studied. Most of these pipeline systems have ways to help distribute the processing using computers' multicore architectures, or job-scheduling systems installed on clusters, thereby reducing computation time. In general, these tools require some programming and software expertise (local installation and configuration issues seem to be largely underestimated issues) beyond what fMRI researchers can usually do (whereas PSOM, Nipype and using the SPM batch system are 'easy'). These more complex workflow or pipeline solutions can, however, ease replication of the analysis by others: see [53] for an example using the LONI pipeline.

Organize and share data and metadata

Besides replicating an analysis (running exactly the same code on the same data), sharing data provides guarantees of reproducibility by (i) allowing a comparison with newly collected data (are the patterns observed in the new dataset the same, independently of statistical significance?), (ii) allowing alternative analyses to be tested on the same data, and (iii) aggregating them with other data for meta-analyses [54]. Many funders now request that data are made available, and researchers must be prepared to do this and to identify where the data will be archived. When the data have obvious potential for reuse (e.g. [55]) or pose special challenges (e.g. [56]), their publication in journals such as *Data in Brief, Frontiers in Neuroscience, F1000 Research, GigaScience*, *Journal of Open Psychology Data*, or *Scientific Data* allow the creators to be acknowledged by citation. In any case, data can simply be put in a repository such as NITRC [33] or Open-fMRI [57] (task-based fMRI [58]). As of March 2015, OpenfMRI hosts 33 full datasets, and a more complete format describing the data is being developed. Previously, the major project that supported sharing of full fMRI datasets was the fMRI Data Center [59,60]. It currently has 107 datasets available on request, but has not accepted submission of additional datasets since 2007. The researcher must also be aware of the constraints involved in sharing MRI data. It is of course essential that consent forms indicate clearly that the data will be de-identified and shared anonymously, and it is the responsibility of the principal investigator to ensure proper de-identification [61], that is, not only removing any personal information from the image headers, but also removing facial (and possibly dental and ear) information from the T1-weighted image. Fortunately, personal information is removed automatically by most fMRI packages when converting from DICOM to NIfTI file format. Removing facial information can be trickier, but automated tools exist for this too (SPM [25,26], MBRIN defacer [62,63], Open fMRI face removal Python script[b]).

Another important issue to consider when sharing data is the metadata (information describing the data). Data reuse is only practical and efficient when data, metadata, and information about the process of generating the data are all provided [64]. Ideally, we would like all of the information about how the data came to existence (why and how) to be provided. The World Wide Web Consortium Provenance Group [65] defines information 'provenance' as the sum of all of the processes, people (institutions or agents), and documents (data included) that were involved in generating or otherwise influencing or delivering a piece of information. For fMRI data, this means that raw data would need to be available, along with (i) initial project information and hypotheses leading to the acquired data, including scientific background as well as people and funders involved; (ii) experimental protocol and acquisition details; and (iii) other subject information, such as demographics and behavioral or clinical assessments. There are currently no tools to do this metatagging, but we recommend checking with the database that will host the data and using their format from the start (that is, store data on your computer or server using the same structure). Functional MRI can have a complex data structure, and reorganizing the data post-hoc can be time-consuming (several hours for posting on OpenfMRI, if the reorganization is done manually [66]). In the future, efforts spearheaded by the International

Neuroinformatics Coordinating Facility (INCF [67]) data sharing task force (INCF-Nidash [68]) may provide a solution, with the development of the Neuro-Imaging Data Model (NIDM [69]), as well as some recommendations on the directory structure and metadata to be attached to the data. Some initial work already permits meta-information to be attached directly to SPM [25,26], FSL [31,32], and (soon) AFNI [29,30] fMRI data analysis results.

Make derived data available

Along with the raw data and the analysis batch and scripts, sharing derived data also increases reproducibility by allowing researchers to compare their results directly. Three types of derived data can be identified: intermediate derived data (from the data analysis workflow), primary derived data (results) and secondary derived data (summary measurements).

Providing intermediate derived data from the analysis workflow, such as the averaged echo-planar image (mean EPI) or statistical mask, makes it possible to judge whether an analysis provides reasonable-looking data, and what the residual brain coverage is after realignment, normalization and subject overlay. Intermediate derived data may not always be directly essential to reproducibility, but can improve the confidence in the data at hand and/or point to their limitations. More important for reproducibility is the sharing of primary derived data. Currently, fMRI studies only report significant results (regions that survive the statistical threshold), because one cannot list all regions or voxels tested. Yet results are more often reproduced when reported at a less conservative significance threshold (p-value) than is often used in our community [70]. The best way to validate that an experiment has been reproduced is by comparing effect sizes, independently of the significance level. Comparing peak coordinates of significant results can be useful, but is limited [66]. In contrast, providing statistical or parameter maps allows others to judge the significance and sparsity of activation clusters [71]. Statistical maps can be shared via NeuroVault [72,73]. NeuroVault allows the visualization and exploration of raw statistical maps and is thus a good way look not only at effect sizes, but also at the precise location of effects (rather than the crude cluster peak coordinate). Along with the statistical maps, some information about provenance currently has to be entered manually (taking 10 to 15 minutes). Again, this manual editing will soon be facilitated by the adoption of the NIDM [69]. Finally, as for statistical maps, secondary derived data should be shared - most likely as supplementary material data sheets. In a region of interest (ROI) analysis, for instance, the mean parameter values extracted across voxels are assembled into a matrix to compute statistics. This data matrix should be saved and distributed so that effect sizes can be compared across studies. Providing scatter plots along with the data of any zero-order, partial, or part correlations between brain activity or structure and behavioral measures also allows one to judge of the robustness of the results [74].

Publish

One aspect to consider when sharing data is to make them available online before publication, so that permanent links can be included in the article at the time of publication. We also recommend stating how you want data and code to be credited by using machine-readable licenses. Easy-to-implement licenses, many of which offer the advantage of being machine-readable, are offered by the Creative Commons organization [75] and Open Data Commons [76].

Discussion

Researchers are much more likely to be able to replicate experiments and reproduce results if material and procedures are shared, from the planning of an experiment to the fMRI result maps. This is also crucial if the global efficiency of our research field is to improve. To be able to do this, the single most important advice to consider would probably be to plan ahead, as lack of planning often prevents sharing[c]. Informed consent and ethics should be compliant with data sharing. When previous data are available, statistical power should be computed, sample size chosen accordingly and reported. Data, scripts and maps should be organized and written with the intention to share and allow reuse, and they should have licenses allowing redistribution.

To increase fMRI reproducibility, neuroscientists need to be trained, and to train others, to plan, document and code in a much more systematic manner than is currently done. Neuroimaging is a computational data science, and most biologists, medical doctors and psychologists lack appropriate programming, software and data science training. In that respect, sharing work has an additional educational value. By studying the code used by others, in order to replicate their results, one also learns what practices are useful when sharing. Piwowar et al. [77] showed that sharing data and code increases the trust and interest in papers, and citation of them. This also makes new collaborations possible more easily. Openness improves both the code used by scientists and the ability of the public to engage with their work [39]. Putting the code associated with a paper in a repository is likely to have as many benefits as sharing data or publications. For instance, the practice of self-archiving can increase citation impact by a dramatic 50 to 250% [78]. Data and code sharing can also be viewed as a more ethical and efficient use of public funding (as data acquired by public funds should be available to the scientific community at large), as well as a

much more efficient way of conducting science, by increasing the reuse of research products.

Conclusion

By adopting a new set of practices and by increasing the computational expertise of fMRI researchers, the reproducibility and validity of the field's results will improve. This calls for a much more open scientific attitude in fMRI, together with increased responsibility. This will advance our field more rapidly and yield a higher return on funding investment. Making neuroimaging reproducible will not make studies better; it will make scientific conclusions more verifiable, by accumulating evidence through replication, and ultimately make those conclusions more valid and research more efficient. Two of the main obstacles on this road are the lack of programming expertise in many neuroscience or clinical research laboratories, and the absence of widespread acknowledgement that neuroimaging is (also) a computational science.

Annex 1 - list of websites mentioned in the article that can be used for sharing

Bitbucket (https://bitbucket.org/) is "a web-based hosting service for projects that use either the Mercurial or Git revision control system" and allows managing and sharing code.

Dryad (http://datadryad.org/) "is a curated resource that makes the data underlying scientific publications discoverable, freely reusable, and citable" under a Creative Commons license. It is a nonprofit membership organization from an initiative among a group of leading journals and scientific societies in evolutionary biology and ecology. This repository now hosts any kind of biological data.

FigShare (http://figshare.com/) is a repository that "allows researchers to publish all of their data in a citable, searchable and sharable manner" under a Creative Commons license. It is supported by Digital Science, part of Macmillan Publishers Limited. This repository now hosts any kind of data.

GitHub (https://github.com/) is "a web-based Git repository hosting service" and allows managing and sharing code.

Kepler (https://kepler-project.org/) is a scientific workflow application "designed to help scientists, analysts, and computer programmers create, execute, and share models and analyses across a broad range of scientific and engineering disciplines".

LONI pipeline (http://pipeline.bmap.ucla.edu/) is an application to "create workflows that take advantage of all the tools available in neuroimaging, genomics [and] bioinformatics".

NeuroDebian (http://neuro.debian.net/) integrates neuroimaging and other related neuroscientific and computational software into Debian (Linux). It includes a repository of over 60 software and data packages. NeuroDebian also provides a virtual machine, simplifying deployment within any existing Linux, OS X or Windows environment.

NeuroImaging Tool and Resources Clearinghouse (http://www.nitrc.org/), is a web resource that "facilitates finding and comparing neuroimaging resources for functional and structural neuroimaging analyses". It is currently funded by the NIH Blueprint for Neuroscience Research, National Institute of Biomedical Imaging and Bioengineering, National Institute of Drug Addiction, National Institute of Mental Health, and National Institute of Neurological Disorders and Stroke.

NeuroVault (http://neurovault.org/) is a "public repository of unthresholded brain activation maps" under a data common license. It is managed by Krzysztof Gorgolewski, and supported by INCF and the Max Planck Society.

Open fMRI (https://openfmri.org/) is "a project dedicated to the free and open sharing of functional magnetic resonance imaging (fMRI) datasets, including raw data" under an open data common license. It is managed by Russ Poldrack and funded by a grant from the National Science Foundation.

OpenScience framework (https://osf.io/) is a project management system for an "entire research lifecycle: planning, execution, reporting, archiving, and discovery". It supports local archiving, but also links with other repositories. Multiple options for licensing are available. It is supported by the Center for Open Science.

Taverna (http://www.taverna.org.uk/) is a "domain-independent workflow management system - a suite of tools used to design and execute scientific workflows".

Zenodo (http://zenodo.org/) is a repository "that enables researchers, scientists, EU projects and institutions to share and showcase multidisciplinary research results", with a choice of open source licenses. It was launched within an EU funded project and is supported by the European Organization for Nuclear Research (CERN).

Endnotes

[a]Matlab Publishing Markup refers to specific keys such as %% or _ _ which allows not only inserting comments into your Matlab code, but also format it for then publish the code automatically into an executable and readable format, see http://uk.mathworks.com/help/matlab/matlab_prog/marking-up-matlab-comments-for-publishing.html.

[b]When uploading data to OpenfMRI you need to ensure the structural data are defaced appropriately – the website also offers to use their own defacing tool, see https://github.com/poldrack/openfmri/tree/master/pipeline/facemask.

[c]Thanks to Dorothy Bishop for pointing to this.

Abbreviations
AFNI: Analysis of functional neuroimages; fMRI: Functional magnetic resonance imaging; FSL: FMRIB software library; INCF: International neuroinformatics coordinating facility; NIDM: Neuro-imaging data model; Nipype: NeuroImaging in python pipelines and interfaces; PSOM: Pipeline system for octave and matlab; SPM: Statistical parametric mapping.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
CP and JBP drafted the manuscript. Both authors read and approved the final manuscript.

Received: 28 December 2014 Accepted: 15 March 2015
Published online: 31 March 2015

References

1. Galton F. Biometry. Biometrika. 1901;1(1):7–10.
2. Irreproducible LJ, Results E. Causes, (Mis)interpretations, and Consequences. Circulation. 2012;125:1211–4.
3. Stodden V, Leisch F, Peng RD. Implementing Reproducible Research. Victoria: Taylor and Francis group CRC Press; 2014.
4. Ioannidis JPA. Why Most Published Research Findings Are False. PLoS Med. 2005;2(8):e124.
5. Button KS, Ioannidis JPA, Mokrysz C, Nosek BA, Flint J, Robinson ESJ, et al. Power failure: why small sample size undermines the reliability of neuroscience. Nat Rev Neurosci. 2013;14(5):365–76.
6. Simonsohn U, Nelson LD, Simmons JP. P-curve: A key to the file-drawer. J Exp Psychol Gen. 2014;143:534–47.
7. Carp J. On the plurality of (methodological) worlds: Estimating the analytic flexibility of fMRI experiments. Front Neurosci. 2012;6(149).
8. Aurich NK, Alves Filho JO, MarquesdaSilva AM, Franco AR. Evaluating the Reliability of Different Preprocessing Steps to Estimate Graph Theoretical Measures in Resting State fMRI data. Front Neurosci. 2015;9:48.
9. Simmons JP, Nelson LD, Simonsohn U. False-Positive Psychology: Undisclosed Flexibility in Data Collection and Analysis Allows Presenting Anything as Significant. Psychol Sci. 2011;22(11):1359–66.
10. Donoho DL, Maleki A, Rahman I, Shahram M, Stodden V. Reproducible Research in Computational Harmonic Analysis. Comput Sci Eng. 2009;11(1):8–18.
11. Monogan J. The Controversy of Preregistration in Social Research [Internet]. [cited 2015 Mar 13]. Available from: http://bitss.org/2014/06/13/preregistration-controversy/.
12. Rosenthal R. The file drawer problem and tolerance for null results. Psychol Bull. 1979;86:638.
13. Marwick B. Reproducible Research: A Primer for the Social Sciences [Internet]. Rpres: Reproducibility; 2014. Available from: https://raw.githubusercontent.com/benmarwick/CSSS-Primer-Reproducible-Research/master/CSSS_WI14_Reproducibility.Rpres.
14. Drummond C. Replicability is not reproducibility: nor is it good science. Evaluation Methods for Machine Learning Workshop [Internet]. Montreal, Quebec, CA; 2009. Available from: http://cogprints.org/7691/7/ICMLws09.pdf.
15. Peng RD. Reproducible Research in Computational Science. Science. 2011;334:1226–7.
16. Reproducibility [Internet]. Wikipedia. [cited 2013 Mar 13]. Available from: http://en.wikipedia.org/wiki/Reproducibility.
17. Dryad [Internet]. [cited 2015 Mar 13]. Available from: http://datadryad.org/.
18. FigShare [Internet]. [cited 2015 Mar 13]. Available from: http://figshare.com/.
19. OpenScience framework [Internet]. [cited 2015 Mar 13]. Available from: https://osf.io/.
20. Zenodo [Internet]. [cited 2015 Mar 13]. Available from: http://zenodo.org/.
21. Poldrack RA, Fletcher PC, Henson RN, Worsley KJ, Brett M, Nichols TE. Guidelines for reporting an fMRI study. Neuroimage. 2008;40(2):409–14.
22. Ince DC, Hatton L, Graham-Cumming J. The case for open computer programs. Nature. 2012;482:485–8.
23. Osborne JM, Bernabeu MO, Bruna M, Calderhead B, Cooper J, Dalchau N, et al. Ten Simple Rules for Effective Computational Research. PLoS Comput Biol. 2014;10(3):e1003506.
24. Sandve GK, Nekrutenko A, Taylor J, Hovig E. Ten Simple Rules for Reproducible Computational Research. PLoS Comput Biol. 2013;9(10):e1003285.
25. Wellcome Trust Centre for Neuroimaging. Statistical Parametric Mapping [Internet]. [cited 2015 Mar 13]. Available from: http://www.fil.ion.ucl.ac.uk/spm/.
26. Flandin G, Friston KJ. Statistical parametric mapping (SPM). Scholarpedia. 2008;3(4):6332.
27. Ghosh S, Gorgolewski K. Neuroimaging in Pythom Pipelines and Interfaces [Internet]. [cited 2015 Mar 13]. Available from: http://nipy.sourceforge.net/nipype/.
28. Gorgolewski K, Burns CD, Madison C, Clark D, Halchenko YO, Waskom ML, et al. Nipype: A flexible, lightweight and extensible neuroimaging data processing framework. Front Neuroinformatics. 2011;5(13).
29. Cox RW. Analysis of Functional NeuroImages [Internet]. [cited 2015 Mar 13]. Available from: http://afni.nimh.nih.gov/afni/.
30. Cox RW. AFNI: software for analysis and visualization of functional magnetic resonance neuroimages. Comput Biomed Res. 1996;28:162–73.
31. FMRIB, Analysis Group. FMRIB Software Library [Internet]. [cited 2015 Mar 13]. Available from: http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/.
32. Jenkinson M, Beckmann CF, Behrens T, Woolrich MW, Smith SM. FSL. Neuroimage. 2012;62:782–90.
33. NeuroImaging Tool and Resources Clearinghouse [Internet]. [cited 2015 Mar 13]. Available from: http://www.nitrc.org/.
34. Gronenschild EHBM, Habets P, Jacobs HIL, Mengelers R, Rozendaal N, van Os J, et al. The Effects of FreeSurfer Version, Workstation Type, and Macintosh Operating System Version on Anatomical Volume and Cortical Thickness Measurements. PLoS One. 2012;7(6):e38234.
35. Halchenko Y, Hanke M. Open is not enough. Let's take the next step: An integrated, community-driven computing platform for neuroscience. Front. Neuroinformatics. 2012;6:22.
36. snapshot.debian.org [Internet]. [cited 2013 Mar 13]. Available from: http://snapshot.debian.org/.
37. Comparison of revision control software [Internet]. Wikipedia. [cited 2013 Mar 13]. Available from: http://en.wikipedia.org/wiki/Comparison_of_revision_control_software.
38. Stodden V. The scientific method in practice: Reproducibility in the computational sciences. MIT Sloan Res Pap. 2010;4773–10.
39. Barnes N. Publish your computer code: it is good enough. Nature. 2010;467:753.
40. git [Internet]. [cited 2015 Mar 13]. Available from: http://git-scm.com/.
41. Subversion [Internet]. [cited 2015 Mar 13]. Available from: http://subversion.apache.org/.
42. Github [Internet]. [cited 2015 Mar 13]. Available from: https://github.com/.
43. Bitbucket [Internet]. [cited 2015 Mar 13]. Available from: https://bitbucket.org/.
44. Workflow [Internet]. Wikipedia. [cited 2015 Mar 13]. Available from: http://en.wikipedia.org/wiki/Workflow.
45. Bellec P. PSOM [Internet]. [cited 2015 Mar 13]. Available from: https://github.com/SIMEXP/psom.
46. Bellec P, Lavoie-Courchesne S, Dickinson P, Lerch J, Zijdenbos A, Evans AC. The pipeline system for Octave and Matlab (PSOM): a lightweight scripting framework and execution engine for scientific workflows. Front Neuroinformatics. 2012;6(7).
47. Mitchell D, Auer T. Automatic Analysis pipeline [Internet]. [cited 2015 Mar 13]. Available from: http://imaging.mrc-cbu.cam.ac.uk/imaging/AA.
48. Cusack R, Vicente-Grabovetsky A, Mitchell DJ, Wild CJ, Auer T, Linke A, et al. Automatic analysis (aa): efficient neuroimaging workflows and parallel processing using Matlab and XML. Front Neuroinformatics. 2015;8(90).
49. IPython Notebook [Internet]. [cited 2015 Mar 13]. Available from: http://ipython.org/notebook.html.
50. Taverna [Internet]. [cited 2015 Mar 13]. Available from: http://www.taverna.org.uk/.
51. Kepler [Internet]. [cited 2015 Mar 13]. Available from: https://kepler-project.org/.
52. Laboratory of Neuro Imaging. LONI pipeline [Internet]. [cited 2015 Mar 13]. Available from: http://pipeline.bmap.ucla.edu/.
53. Torgerson CM, Quinn C, Dinov I, Liu Z, Petrosyan P, Pelphrey K, et al. Interacting with the National Database for Autism Research (NDAR) via the LONI Pipeline workflow environment. Brain Imaging Behav. 2015;9;89–103.
54. Poline J-B, Breeze JL, Ghosh S, Gorgolewski K, Halchenko YO, Hanke M, et al. Data sharing in neuroimaging research. Front Neuroinformatics. 2012.
55. Gorgolewski K, Storkey AJ, Bastin ME, Whittle I, Wardlaw J, Pernet CR. A test-retest fMRI dataset for motor, language and spatial attention functions. Gigascience. 2013;2:6.
56. Hanke M, Baumgartner FJ, Ibe P, Kaule FR, Pollmann S, Speck O, et al. A high-resolution 7-Tesla fMRI dataset from complex natural stimulation with an audio movie. Sci Data. 2014 May 27;1.

57. Poldrack RA. OpenfMRI [Internet]. [cited 2015 Mar 13]. Available from: https://openfmri.org/.
58. Poldrack RA, Barch DM, Mitchell JP, Wager TD, Wagner AD, Devlin JT, et al. Toward open sharing of task-based fMRI data: the OpenfMRI project. Front Neuroinformatics. 2013;7.
59. fMRI Data Center [Internet]. [cited 2015 Mar 13]. Available from: http://databib.org/repository/371.
60. Van Horn JD, Gazzaniga MS. Why share data? Lessons learned from the fMRIDC. Neuroimage. 2013;82:677–82.
61. Calhoun VD. A spectrum of sharing: maximization of information content for brain imaging data. GigaScience. 2015 Dec;4(1).
62. Laboratory for Computational Neuroimaging. MRI Deface. [cited 2015 Mar 13]. Available from: http://www.nitrc.org/projects/mri_deface/.
63. Bischoff-Grethe A, Ozyurt IB, Busa E, Quinn BT, Fennema-Notestine C, Clark CP, et al. A Technique for the Deidentification of Structural Brain MR Images. Hum Brain Mapp. 2007;28(9):892–903.
64. Goodman A, Pepe A, Blocker AW, Borgman C, Cranner K, Crosas M, et al. Ten Simple Rules for the Care and Feeding of Scientific Data. PLoS Comput Biol. 2014;10(4):e1003542.
65. World Wide Web Consortium Provenance Group [Internet]. [cited 2015 Mar 13]. Available from: http://www.w3.org/2011/prov/wiki/Main_Page.
66. Poldrack RA, Gorgolewski KJ. Making big data open: data sharing in neuroimaging. Nat Neurosci. 2014;17:11.
67. International Neuroinformatics Coordinating Facility [Internet]. Available from: http://www.incf.org/.
68. Neuroinformatics Coordinating Facility data sharing task force [Internet]. [cited 2015 Mar 13]. Available from: http://wiki.incf.org/mediawiki/index.php/Neuroimaging_Task_Force.
69. Neuro-Imaging Data Model [Internet]. [cited 2015 Mar 13]. Available from: http://nidm.nidash.org/.
70. Johnson VE. Revised Standards for Statistical Evidence. Proc Natl Acad Sci U S A. 2013;110(48):19313–7.
71. Jernigan TL, Gamst AC, Fennema-Notestine C, Ostergaard AL. More "mapping" in brain mapping: Statistical comparison of effects. Hum Brain Mapp. 2003;19(2):90–5.
72. Gorgolewski K. NeuroVault [Internet]. [cited 2015 Mar 13]. Available from: http://neurovault.org/.
73. Gorgolewski K, Varoquaux G, Rivers G, Schwartz Y, Ghosh SS, Maumet C, et al. NeuroVault.org: A web-based repository for collecting and sharing unthresholded statistical maps of the human brain. Bio Arch X. 2014;pre-print.
74. Rousselet GA, Pernet CR. Improving standards in brain-behavior correlation analyses. Front Hum Neurosci. 2012;6.
75. Creative Commons organization [Internet]. [cited 2015 Mar 13]. Available from: http://creativecommons.org/choose/.
76. Open Data Commons [Internet]. [cited 2015 Mar 13]. Available from: http://opendatacommons.org/licenses/pddl/.
77. Piwowar HA, Day RS, Fridsma DB. Sharing Detailed Research Data Is Associated with Increased Citation Rate. PLoS One. 2007;2(3):e308.
78. Harnad S. Publish or Perish — Self-Archive to Flourish: The Green Route to Open Access. Eur Res Consort Inform Math. 2006;64.

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 11 October 2012
doi: 10.3389/fnins.2012.00149

On the plurality of (methodological) worlds: estimating the analytic flexibility of fMRI experiments

Joshua Carp*

Department of Psychology, University of Michigan, Ann Arbor, MI, USA

Edited by:
Satrajit S. Ghosh, Massachusetts Institute of Technology, USA

Reviewed by:
Jonathan E. Peelle, Washington University, USA
Eugene Duff, University of Oxford, UK

****Correspondence:***
Joshua Carp, Department of Psychology, University of Michigan, 530 Church Street, Ann Arbor, MI 48109, USA.
e-mail: jmcarp@umich.edu

How likely are published findings in the functional neuroimaging literature to be false? According to a recent mathematical model, the potential for false positives increases with the flexibility of analysis methods. Functional MRI (fMRI) experiments can be analyzed using a large number of commonly used tools, with little consensus on how, when, or whether to apply each one. This situation may lead to substantial variability in analysis outcomes. Thus, the present study sought to estimate the flexibility of neuroimaging analysis by submitting a single event-related fMRI experiment to a large number of unique analysis procedures. Ten analysis steps for which multiple strategies appear in the literature were identified, and two to four strategies were enumerated for each step. Considering all possible combinations of these strategies yielded 6,912 unique analysis pipelines. Activation maps from each pipeline were corrected for multiple comparisons using five thresholding approaches, yielding 34,560 significance maps. While some outcomes were relatively consistent across pipelines, others showed substantial methods-related variability in activation strength, location, and extent. Some analysis decisions contributed to this variability more than others, and different decisions were associated with distinct patterns of variability across the brain. Qualitative outcomes also varied with analysis parameters: many contrasts yielded significant activation under some pipelines but not others. Altogether, these results reveal considerable flexibility in the analysis of fMRI experiments. This observation, when combined with mathematical simulations linking analytic flexibility with elevated false positive rates, suggests that false positive results may be more prevalent than expected in the literature. This risk of inflated false positive rates may be mitigated by constraining the flexibility of analytic choices or by abstaining from selective analysis reporting.

Keywords: fMRI, data analysis, analysis flexibility, selective reporting, false positive results

INTRODUCTION

How common are false positive results in the functional neuroimaging literature? Among functional MRI (fMRI) studies that apply statistical correction for multiple comparisons, most use a nominal false positive rate of 5%. However, Wager et al. (2009) estimate that between 10 and 40% of fMRI activation results are false positives. Furthermore, recent empirical (Ioannidis, 2005a) and mathematical modeling studies (Ioannidis, 2005b) argue that the true incidence of false positives may far exceed the nominal rate in the broader scientific literature. Indeed, under certain conditions, research findings are more likely to be false than true (Ioannidis, 2005b).

As described in a mathematical modeling study by Ioannidis (2005b), analytic flexibility is a key risk factor for inflated rates of false positive results when combined with selective reporting of favorable analysis methods. Analytic flexibility is defined here as the range of analysis outcomes across different acceptable analysis methods. Thus, if many analysis pipelines are considered valid, and if different methods yield different results, then analysis flexibility is high. When analytic flexibility is high, investigators may elect to report methods that yield favorable outcomes and omit methods that yield null results. This practice is known as selective analysis reporting. For example, a researcher may notice that an experiment yields positive results when analyzed using head motion regression, but not when analyzed without using head motion regression. The researcher may then choose to describe the former analysis but not the latter when reporting the results of the experiment. Indeed, investigators in some research fields appear to pursue this strategy. Reviews of randomized clinical trials show that many studies change outcome measures and other methodological parameters between study design and publication. Critically, these changes tend to make results appear more significant than they would have been under the original analysis plan (Chan et al., 2004a,b; Dwan et al., 2008; Mathieu et al., 2009).

A recent survey of fMRI methods shows that methodological decisions are highly variable from study to study (Carp, 2012). Across 241 published fMRI studies, authors reported using 32 unique software packages (e.g., SPM 2, FSL 3.3) and 207 unique combinations of design and analysis steps (e.g., spatial normalization, head motion regression). Parameter settings also showed considerable variability within each analysis step. For example, spatial smoothing kernels ranged from 3 to 12 mm full width at half maximum, and high-pass filter cutoffs ranged from 0.33 to 750 s. Because many studies did not describe critical analysis decisions, this survey likely understated the true diversity of experimental methods in the fMRI literature. In other words,

Table 1 | Pre-processing parameters.

Despiking		
Despiking using AFNI	No despiking	
Slice-timing correction		
Slice-timing correction	No slice-timing correction	
Spatial normalization		
Normalization of functional images to the SPM EPI template	Normalization of anatomical images to the SPM T1 template	Normalization with segmentation using unified normalization
Spatial smoothing		
Smoothing with kernel of 4 mm FWHM	Smoothing with kernel of 8 mm FWHM	Smoothing with kernel of 12 mm FWHM

Table 2 | Model estimation parameters.

Normalization-modeling order			
Normalize before modeling	Model before normalization		
High-pass filtering			
High-pass filtering using a cutoff of 128 s	No high-pass filtering		
Temporal autocorrelation correction			
AR(1) modeling	No correction for temporal autocorrelation		
Run concatenation			
Runs concatenated before model estimation	No run concatenation		
Model basis set			
Hemodynamic response function	Finite impulse response[1], time points 3–4 versus baseline	Finite impulse response[1], time by condition interaction	
Head motion regression			
Six regressors[2]	Twelve regressors[3]	Twenty-four regressors[4]	No motion regression

[1] *Eight basis functions.*

[2] *Raw motion parameters.*

[3] *Raw and time-shifted motion parameters.*

[4] *Raw, time-shifted, squared, and time-shifted squared motion parameters.*

fMRI researchers may choose from a wide array of acceptable methodological strategies.

Critically, methodological studies suggest that this variability in analytic strategies may translate into variability in research outcomes. Countless studies show that individual methodological decisions can have important effects on estimates of fMRI activation. For example, temporal filtering (Skudlarski et al., 1999), autocorrelation correction (Purdon and Weisskoff, 1998; Woolrich et al., 2001), global signal regression (Murphy et al., 2009; Weissenbacher et al., 2009), and head motion regression (Friston et al., 1996; Lund et al., 2005) can profoundly influence analysis outcomes. Activation estimates also vary with the order of analysis steps (Weissenbacher et al., 2009; Carp, 2011) and across analysis software packages (Smith et al., 2005; Poline et al., 2006). Further, combinations of analysis decisions may have interactive effects on research outcomes (Churchill et al., 2012a,b).

However, while many studies have examined the effects of individual analysis procedures or combinations of procedures on research outcomes, most of these studies have focused on optimizing the selection of analytic pipelines rather than quantifying variability across pipelines. For example, Skudlarski et al. (1999) investigated variations between analysis pipelines in receiver operating characteristic (ROC) measures; Della-Maggiore et al. (2002) assessed the effects of differing pipelines on statistical power; and Strother and colleagues (Strother et al., 2004; Churchill et al., 2012a,b) evaluated pipelines using reproducibility and prediction metrics. However, while these studies offer valuable insights into which procedures should be applied and which parameters should be used, they did not explicitly assess the variability of research outcomes across analysis pipelines. In contrast, Hopfinger et al. (2000) did measure variability in activation amplitude across 36 distinct pipelines. But this study examined just four analysis steps, rather than the complete pre-processing and modeling pipelines used in most current fMRI studies, and focused on regional rather than whole-brain activation results. Altogether, while a wealth of previous studies have investigated the question of pipeline optimization, relatively few have considered the question of pipeline variability.

Thus, expanding on previous studies of analytic flexibility, the present study estimated the variability of fMRI methods across 10 pre-processing and model estimation steps. Between two and four options were considered for each step (see **Tables 1** and **2**). Enumerating all combinations of each of the steps yielded a total of 6,912 unique analysis pipelines. Activation estimates from each pipeline were then statistically thresholded and corrected for multiple comparisons using five commonly used techniques, yielding 34,560 unique thresholded activation maps. By examining a range of analysis pipelines orders of magnitude greater than those considered in previous studies, the present investigation yields the most comprehensive picture of methodological flexibility in the fMRI literature available to date.

MATERIALS AND METHODS

DATA ACQUISITION

The present study re-analyzed a previously published fMRI study of response inhibition (Aron et al., 2007). Data were drawn from the Open fMRI database[1] (Accession Number: ds000008; Task: 001). Fifteen subjects completed three runs of a standard event-related stop-signal task and three runs of a conditional stop-signal task. Only data from the standard stop-signal task were considered here. The task included three trial types. On go trials, subjects were instructed to make a motor response; on successful stop trials, subjects were instructed to withhold a response and were able to do so; and on failed stop trials, subjects were instructed to withhold a response but failed to do so. Functional data were acquired using a 3 T Siemens Allegra MRI scanner (TR: 2 s; TE: 30 ms; flip angle: 90°; voxel dimensions: 3.125 mm × 3.125 mm × 4.0 mm). Each of the three functional scanning runs included 176 images. High-resolution T1 MPRAGE images were also acquired for use

[1] http://www.openfmri.org

in spatial normalization (TR: 2.3 s; TE: 2.1 ms; voxel dimensions: 1.0 mm × 1.33 mm × 1.33 mm). Complete imaging and behavioral data were only available for 13 of the subjects; the remaining two subjects were excluded from analysis. Further details on sample characteristics, task specifications, and imaging acquisition are given in the original report of these data (Aron et al., 2007).

PIPELINE GENERATION

To generate a large collection of analysis pipelines, five preprocessing decisions and five modeling decisions for which multiple strategies appear in the research literature were selected. Pre-processing decisions, detailed in **Table 1**, included despiking (despiking or no despiking), slice-timing correction (slice-timing correction or no correction), spatial normalization (normalization to a functional template, to an anatomical template, or using segmentation of anatomical images), and spatial smoothing (FWHM 4, 8, or 12 mm). Modeling decisions, detailed in **Table 2**, included the order of normalization and model estimation (images were normalized before or after model estimation), high-pass filtering (128 s cutoff or no filtering), autocorrelation correction [AR(1) correction or no correction], run concatenation (run concatenation or no run concatenation), basis set [canonical hemodynamic response function, finite impulse response (FIR) with the contrast of time points 3 and 4 versus fixation, and FIR with the interaction of time point by condition], and head motion regression (6, 12, or 24 motion parameters, or no motion regression). Taking all combinations of these options yielded 6,912 unique analysis pipelines.

Despiking was implemented using the 3dDespike tool in AFNI version 2011_05_26_1456. All other steps were implemented using SPM 8 release 4010 (Wellcome Trust Centre for Neuroimaging, UCL, UK) running under Matlab 2011b (The Mathworks, Inc., Natick, MA, USA).

Data from each subject were submitted to each analysis pipeline. Each single-subject model included separate regressors for go trials, successful stop trials, and failed stop trials. Single-subject models were combined using random-effects analysis. Test statistics (i.e., t and F values) were converted to Z-values after contrast estimation using a transformation adapted from the ttoz and ftoz utilities in FSL version 4.1.8. All further analysis was based on random-effects models of the contrast of successful stop trials versus go trials.

To assess the variability in activation strength across models, the range of Z-values (referred to hereafter as the analytic range) was computed for each voxel and for each contrast. In addition, the range of activation values associated with each analysis step (despiking, slice-timing correction, etc.) was estimated by computing the mean absolute difference of Z-values over all pairs of parameter options and over all settings of other analysis parameters. For example, to estimate the analytic range attributable to changes in spatial smoothing kernel, the absolute value of the differences between (a) 4 and 8 mm FWHM, (b) 4 and 12 mm FWHM, and (c) 8 and 12 mm FWHM were averaged over all combinations of all other analysis parameters for each voxel and each contrast. Because the analytic range metric used here is based on variability in Z-values, this metric is sensitive to differences in both parameter estimates and error variance across pipelines.

Table 3 | Statistical thresholding parameters.

	Uncorrected single-voxel threshold	Corrected single-voxel threshold	Cluster size threshold
Monte Carlo @ $p < 0.01$	$p < 0.01$	n/a	Determined by simulation
Monte Carlo @ $p < 0.001$	$p < 0.001$	n/a	Determined by simulation
Monte Carlo @ $p < 0.0001$	$p < 0.0001$	n/a	Determined by simulation
False discovery rate	n/a	$p < 0.05$	n/a
Gaussian random field theory	n/a	$p < 0.05$	n/a

Many neuroimaging studies report the locations of peak activation for contrasts of interest. Indeed, spatial precision is often advertised as one of the chief virtues of MRI as compared with other imaging techniques. Thus, the variability of peak activation coordinates across analysis pipelines was assessed as well. For each analysis pipeline and each contrast, the coordinates of the peak activation from each hemisphere were extracted. The distribution of peak coordinates was then plotted to assess the spatial dispersion of peak activation locations. To assess variability in localization within circumscribed regions of interest (ROIs), coordinates of peak activation were also extracted for each analysis pipeline within each of two ROIs: a right inferior frontal gyrus region (comprising the pars triangularis and pars opercularis regions of the right inferior frontal gyrus) and a right temporal cortex region (comprising the right superior and middle temporal gyri). All ROIs were defined using the Automatic Anatomical Labeling (AAL) atlas (Tzourio-Mazoyer et al., 2002).

The 6,912 random-effects statistical maps were also thresholded and corrected for multiple comparisons according to five strategies (**Table 3**), yielding 34,560 thresholded maps for each contrast. Activation maps were thresholded using three versions of a Monte Carlo simulation procedure, as implemented in the Resting-State fMRI Data Analysis Toolkit (REST; Song et al., 2011)[2]. These three thresholding approaches used uncorrected single-voxel thresholds of $p < 0.01$, $p < 0.001$, or $p < 0.0001$. Cluster size thresholds were then selected to set the cluster-wise false positive rate at 5% for each approach. Statistical maps were also thresholded using the false discovery rate (FDR; Genovese et al., 2002) and Gaussian random field theory (RFT; Nichols and Hayasaka, 2003) correction procedures, as implemented in SPM 8. Both the FDR and RFT procedures used a corrected single-voxel threshold of $p < 0.05$; neither of these methods employed cluster size thresholds.

It is important to note that these thresholding methods take different approaches to the problem of multiple comparisons. The Monte Carlo and RFT corrections used here attempt to control the family wise error at 5%. Using these corrections, 5% of activation maps should contain at least one false positive activation. In contrast, the FDR correction attempts to control the

[2]http://www.restfmri.net

proportion of false positive voxels, such that 5% of significantly activated voxels should be false positives in a given activation map. Further, while the RFT and FDR corrections control the false positive rate at the level of individual voxels, the Monte Carlo correction controls the false positive rate at the level of clusters. Because these thresholding strategies approach the problem of multiple comparisons in different ways, it was expected that different strategies would yield different results. However, all three strategies appear to be used interchangeably in published studies, with many reports describing their chosen approach simply as "correcting for multiple comparisons."

All code for generating analysis pipelines, calculating analytic variability, statistical thresholding, and plotting figures is freely available online[3].

RESULTS

ANALYTIC VARIABILITY OF ACTIVATION STRENGTH

Estimates of activation strength showed substantial variability across analysis pipelines. Analytic range values (i.e., the range of Z-values across pipelines) for the contrast of successful stop trials versus go trials are displayed in **Figure 1**. Range values varied from 1.14 in the right superior frontal gyrus to 8.83 in the right superior temporal gyrus, with a median analytic range value of 3.44. Analytic range also varied with mean activation across analysis pipelines. Mean activation and analytic range for the successful stop versus go contrast were highly correlated across voxels [$r(44{,}614) = 0.87$, $p < 0.001$], such that voxels with the strongest activation also showed the greatest variability across analysis pipelines.

While each analysis step contributed to variability in activation strength across pipelines, different steps were associated with distinct patterns of variability across brain regions. For the contrast of successful stop trials versus go trials, the analytic range values for choices of smoothing kernel (**Figure 2**) and model basis set (**Figure 3**) were greatest in regions of maximal mean activation, including superior temporal gyrus and precuneus. In contrast, the effects of despiking (**Figure 2**) and head motion regression (**Figure 3**) were generally greatest toward the edges of the brain, particularly in ventral frontal regions. Other steps, such as slice-timing correction and spatial normalization, exerted idiosyncratic patterns of focal effects in a variety of regions across the brain (**Figure 2**), whereas autocorrelation correction was associated with diffuse patterns of change across the brain and ventricles (**Figure 3**).

Finally, range maps were moderately correlated across analysis steps. The mean absolute correlation across voxels between range maps for all pairs of analysis steps was $r = 0.49$, with an average explained variance of $R^2 = 0.26$. In other words, while different analysis steps exerted spatially correlated effects on analysis outcomes across the brain, correlations among step-wise variability maps explained a minority of the variance associated with other analysis steps.

Thus, estimates of activation strength showed considerable variability across analytic pipelines; voxels that showed highly significant activations under some pipelines yielded null results under others. Pipeline-related variability was strongly correlated with average activation, such that activation estimates were most variable in regions showing the greatest overall activation. Finally, different analysis steps showed correlated but distinct patterns of influence across the brain.

ANALYTIC VARIABILITY OF ACTIVATION LOCATION

Activation localization also varied widely across analysis pipelines. To describe the spatial dispersion of peak activation locations, the coordinates of the most significant activation were extracted for each hemisphere and for each pipeline. As seen in **Figure 4**, the results showed a considerable degree of consistency across pipelines: many pipelines yielded maximal activation in the superior temporal gyrus, the supramarginal gyrus, and the right inferior frontal gyrus. Within these regions, however, peak locations were widely dispersed, with activations extending along the length of the sylvian fissure. And many pipelines yielded peak locations outside these regions. In the left hemisphere, 672 unique peak locations were observed, with standard deviations of 12.8, 38.5, and 21.8 mm along the x-, y-, and z-axes, respectively. Activation peaks extended along the anterior-posterior axis from the middle frontal gyrus ($y = 63.0$) to the middle occipital gyrus ($y = -108.875$); along the lateral-medial axis from the middle temporal gyrus ($x = -71.75$) to the middle occipital gyrus ($x = -18.625$); and along the inferior-superior axis from the posterior cerebellum ($z = -50$) to the postcentral gyrus ($z = 80.0$). In the right hemisphere, 534 unique peaks were observed, with standard deviations of 12.6, 30.4, and 16.4 mm. Peaks ranged along the anterior-posterior axis from the superior frontal gyrus ($y = 56.75$) to the middle occipital gyrus ($y = -108.875$); along the medial-lateral axis from the posterior cerebellum ($x = -15.5$) to the superior temporal gyrus ($x = 72.0$); and along the inferior-superior axis from the posterior cerebellum ($z = -50.0$) to the postcentral gyrus ($z = 75.0$). In all, peaks were identified in 69 of the 128 regions defined by the AAL atlas.

The foregoing analysis investigated pipeline variability in the localization of left- and right hemisphere activation peaks. However, investigators may be more interested in the localization of peak activation within specific brain regions rather than an entire cerebral hemisphere. To explore pipeline variability within circumscribed ROIs, peak activation coordinates were extracted for each pipeline within ROIs comprising the right inferior frontal gyrus and the right temporal cortex. This analysis identified 223 unique activation peaks in the right inferior frontal gyrus and 197 unique peaks in the right temporal cortex. As displayed in **Figure 5**, activation peaks were distributed widely across the right inferior frontal gyrus. Peaks in the right temporal cortex were relatively concentrated toward the center of the region, but nevertheless extended to span nearly the entire anterior-posterior and inferior-superior axes of the mask.

In sum, the localization of activation peaks also revealed both consistency and variability across analysis pipelines. While many pipelines yielded peak hemispheric activation locations in a network of regions thought to be related to response inhibition (Aron et al., 2004), peak locations were scattered widely throughout these regions, as well as additional regions throughout much of the brain. Analysis of peak activation distribution within inferior frontal and temporal regions also revealed considerable variability in localization across pipelines.

[3]https://github.com/jmcarp/fmri-pipe

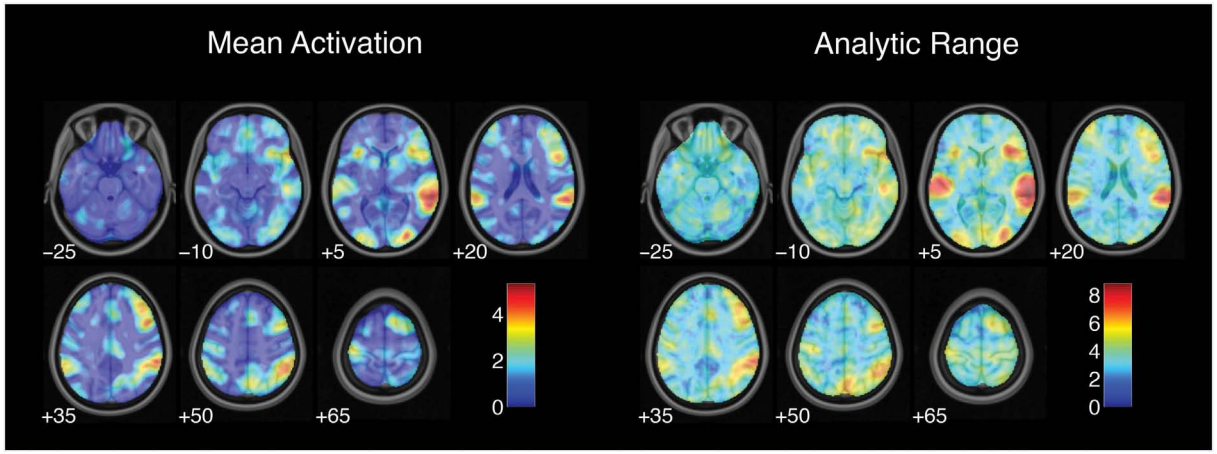


FIGURE 1 | Variation in activation strength across analysis pipelines. Mean activation denotes the average *Z*-value for each voxel across all analysis pipelines; analysis range denotes the range of *Z*-values across all pipelines. Images are presented in neurological orientation, with the left hemisphere displayed on the left. Note that color scales differ across panels.

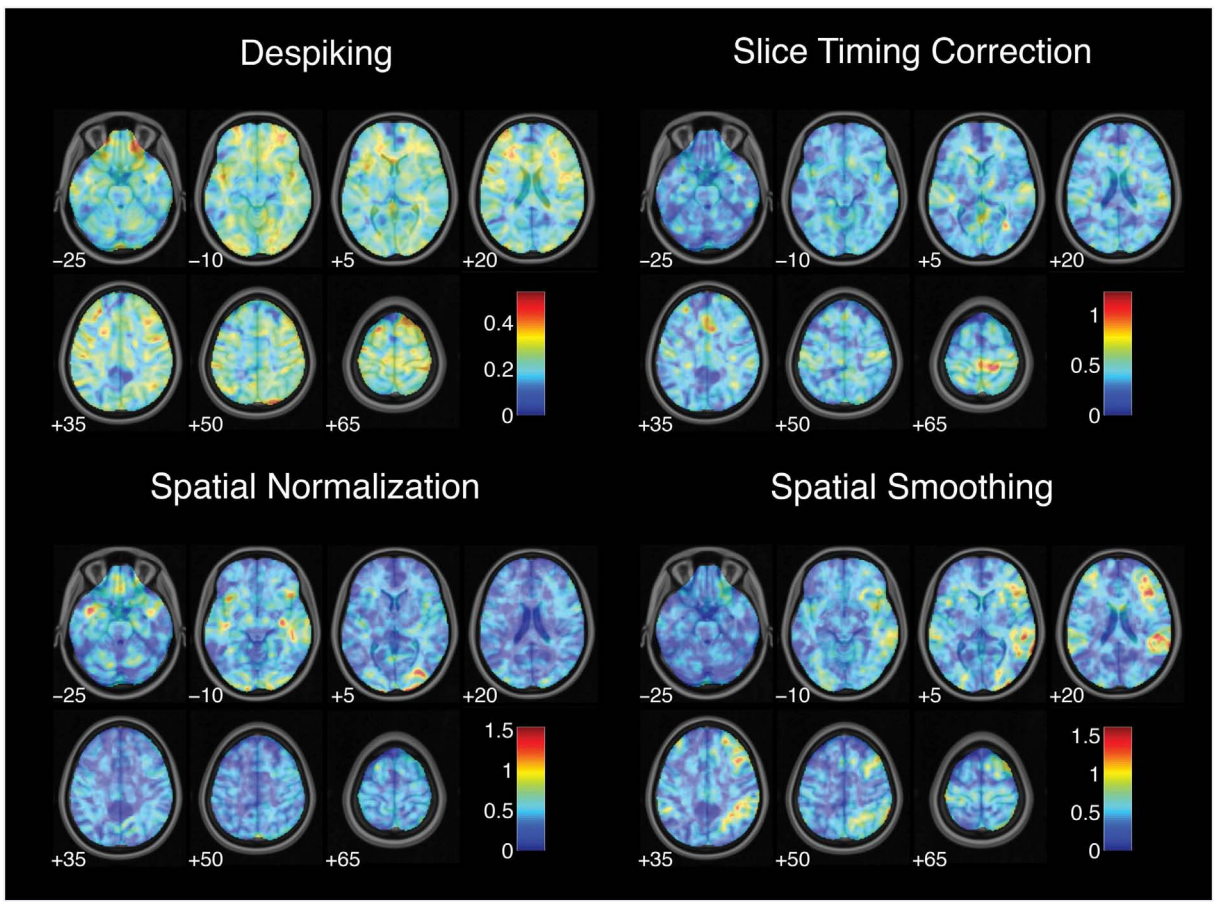


FIGURE 2 | Variation in activation strength attributable to pre-processing choices. Images are presented in neurological orientation, with the left hemisphere displayed on the left. Note that color scales differ across panels.

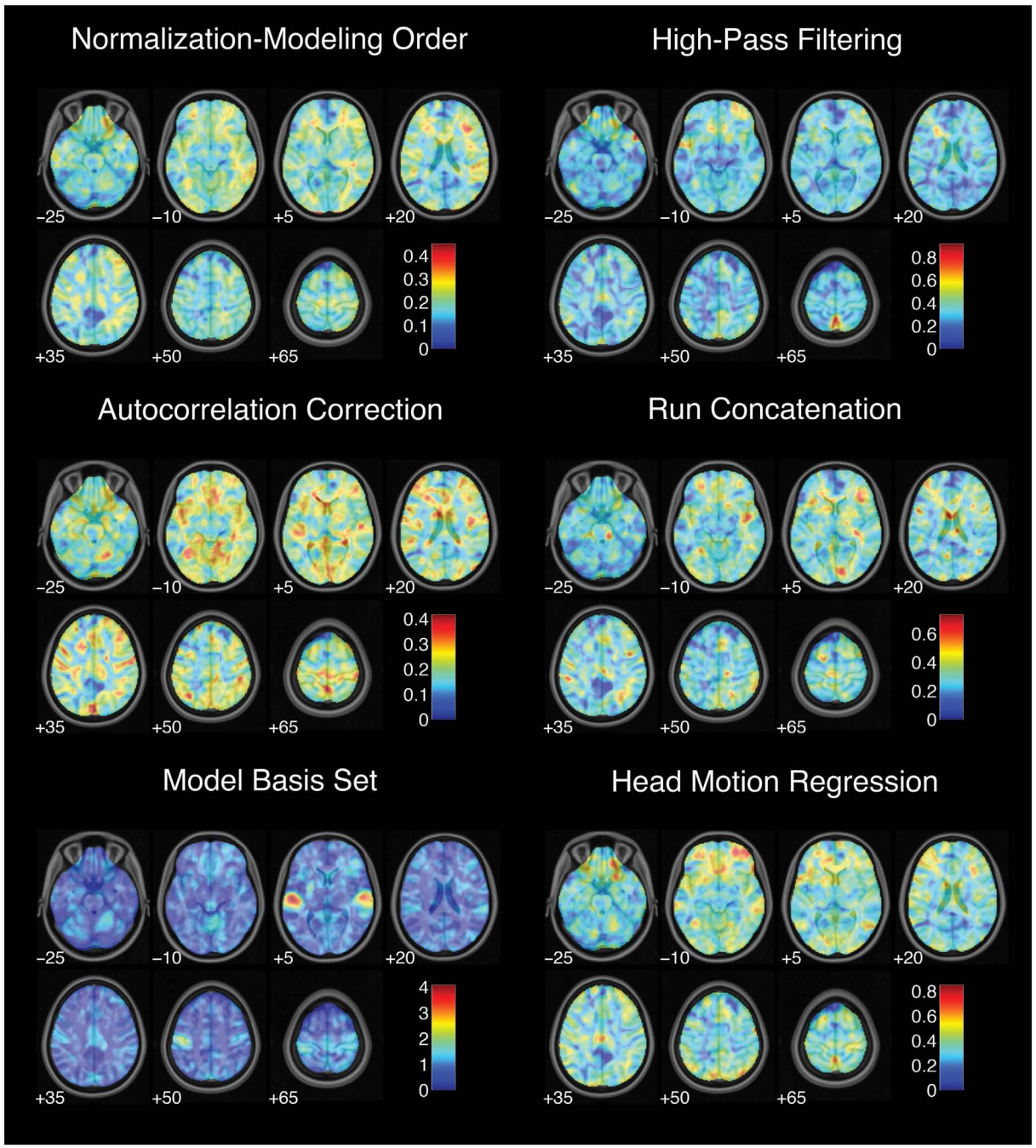


FIGURE 3 | Variation in activation strength attributable to model estimation choices. Images are presented in neurological orientation, with the left hemisphere displayed on the left. Note that color scales differ across panels.

ANALYTIC VARIABILITY OF ACTIVATION SIGNIFICANCE

The previous analyses revealed substantial quantitative variation in analysis outcomes (i.e., activation strength and location) across pipelines. Analysis of statistically thresholded images revealed that qualitative analysis outcomes (i.e., activation significance) varied with respect to methodological decisions as well. The 6,912 statistical maps were thresholded and corrected for multiple comparisons using five strategies: three variants of a Monte Carlo procedure, as well as FDR and Gaussian RFT corrections (**Table 3**). These parameters yielded 34,560 unique thresholded maps for each contrast.

For the successful stop versus go contrast, the proportion of significantly activated voxels (excluding non-brain voxels) varied from 0 to 26.3%, with a median of 4.6%. Monte Carlo

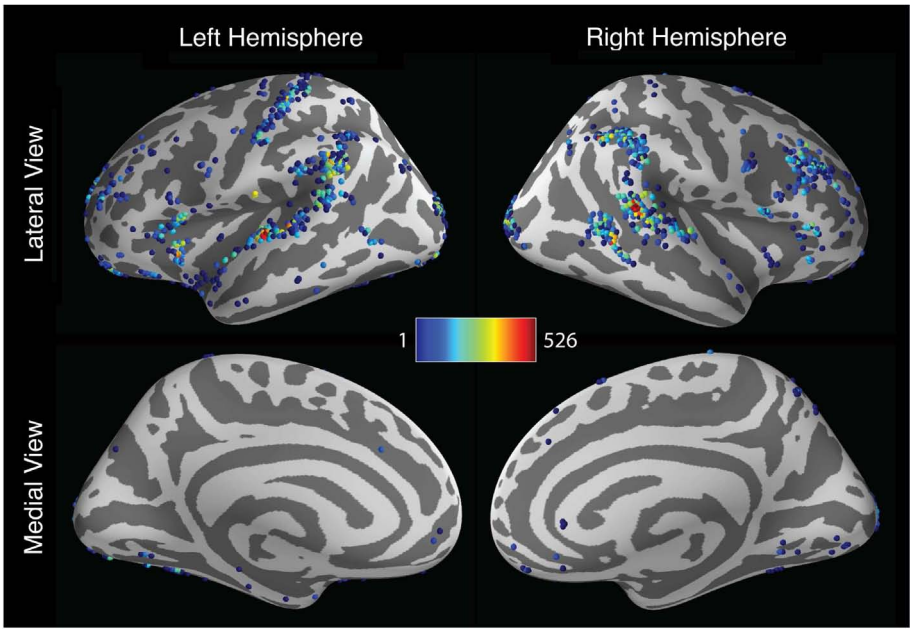


FIGURE 4 | Spatial distribution of peak activation locations across analysis pipelines across the cerebral hemispheres. Shaded spheres indicate the locations of activation peaks. Sphere colors denote the base-10 logarithm of the number of pipelines yielding maximal activation for that location; colors range from blue, indicating a single pipeline, to red, indicating 526 pipelines.

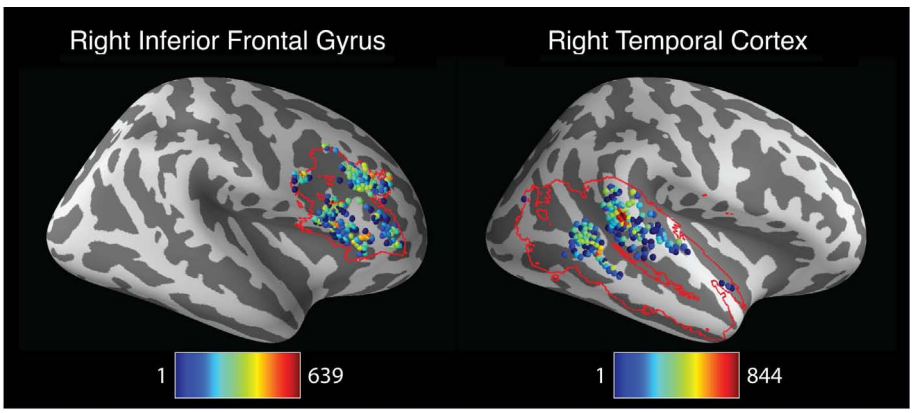


FIGURE 5 | Spatial distribution of peak activation locations across analysis within anatomically defined regions of interest (ROIs). Red contour lines indicate the boundaries of the ROIs. All images represent lateral views of the right hemisphere. Shaded spheres indicate the locations of activation peaks. Sphere colors denote the base-10 logarithm of the number of pipelines yielding maximal activation for that location. For the right inferior frontal gyrus ROI (left panel), colors range from blue, indicating a single pipeline, to red, indicating 639 pipelines. For the right temporal cortex ROI (right panel), colors range from blue, indicating a single pipeline, to red, indicating 844 pipelines.

simulation with a single-voxel threshold of $p < 0.01$ proved to be the most liberal procedure, with a median of 12.8% of brain voxels activated. Monte Carlo simulation with single-voxel thresholds of $p < 0.001$ and $p < 0.0001$ yielded median activation proportions of 5.4 and 1.9%, respectively. Using FDR correction yielded a median activation proportion of 10.8%. RFT correction was the most conservative approach, with a median of 0.16% of brain voxels activated. Critically, all five thresholding methods aimed to control the whole-brain false positive rate at 5%. Thus, these results suggest that some thresholding approaches are far more conservative than others, even when targeting the same corrected false positive rate – a point that has been raised in previous studies (e.g., Lieberman and Cunningham, 2009) but that merits being repeated here.

To characterize the likelihood of significant activation across all 34,560 thresholded maps, the proportion of pipelines yielding significant activation was computed for each voxel (**Figure 6**). This

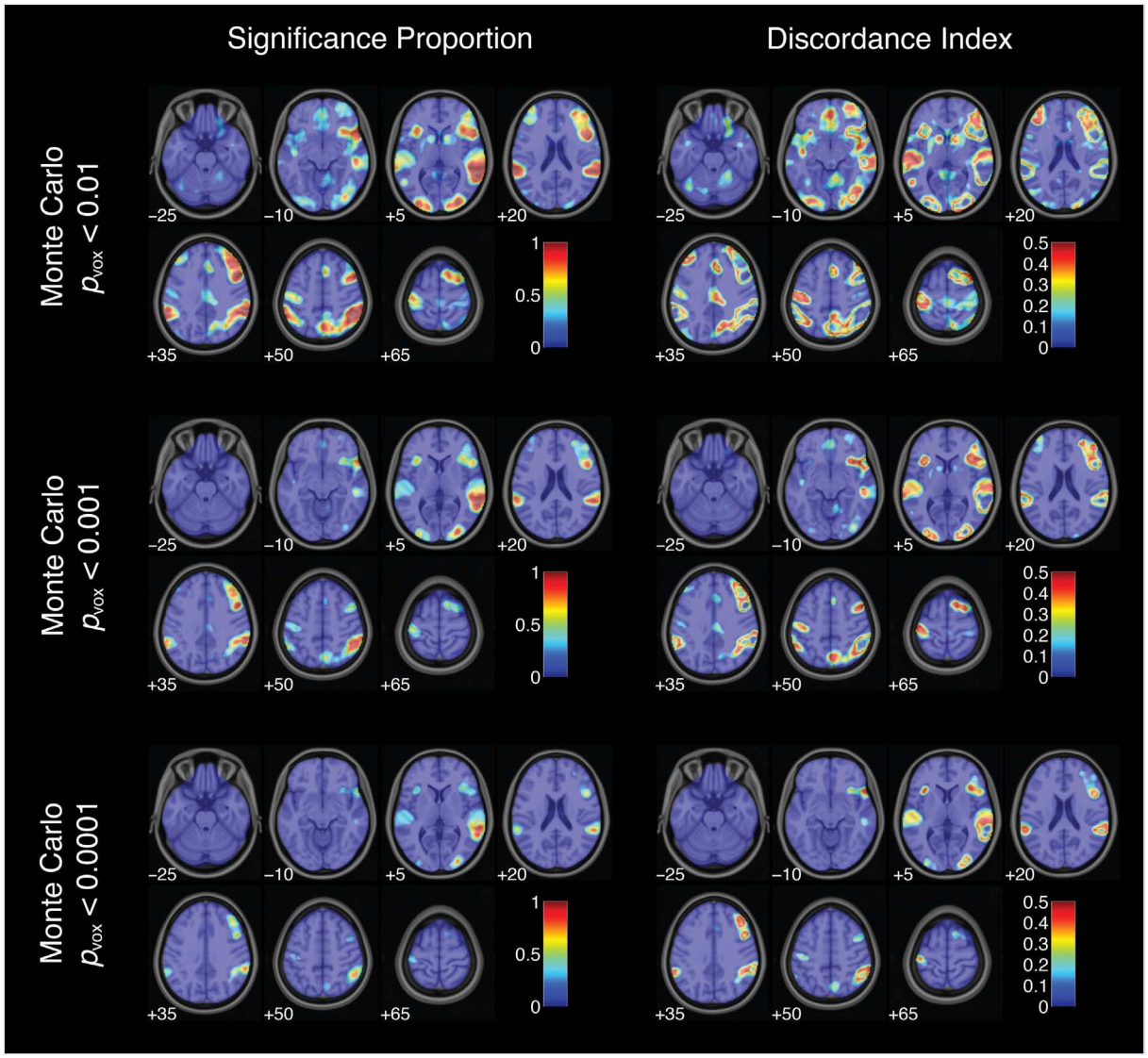


FIGURE 6 | Activation significance across analysis pipelines using three variants of a Monte Carlo thresholding procedure. Significance proportion denotes the fraction of thresholded maps yielding significant activation for each voxel. Discordance index denotes the level of disagreement across threshold maps. Images are presented in neurological orientation, with the left hemisphere displayed on the left. Note that color scales range from 0 to 1 for significance proportion and from 0 to 0.5 for discordance index.

index did not reach 1 (or 100%) for any voxel for the successful stop versus go contrast. In other words, no voxels showed significant activation under all analysis and thresholding pipelines. However, some voxels consistently showed significant activation over nearly every analytic approach. The peak significance proportion in the right superior temporal gyrus reached 0.93. A subset of voxels in the right inferior frontal gyrus and right middle occipital gyrus also showed significant activation across a majority of pipelines, with peak significance proportions of 0.77 and 0.83, respectively. In contrast, many voxels deep within the arcuate fasciculus yielded significance proportions of zero: these voxels did not show significant activation under any combination of analytic and thresholding strategies. Somewhat paradoxically, voxels showing relatively consistent activation (i.e., high significance proportion indices) also exhibited relatively strong quantitative variability across analysis pipelines (i.e., high analytic range values; $R^2 = 0.64$); analytic range values in the voxels of peak significance proportion in the right superior temporal gyrus, the right inferior frontal gyrus, and the right middle occipital gyrus were 8.13, 6.57, and 7.05 Z-units, respectively. Finally, nearly all voxels yielded non-zero significance proportions: 90.3% of brain voxels showed significant activation for at least some thresholded maps.

Thus, some voxels were significantly activated for nearly all analysis pipelines; others did not yield significant activation under any pipelines. However, some voxels yielded less consistent results across pipelines. This disagreement about qualitative

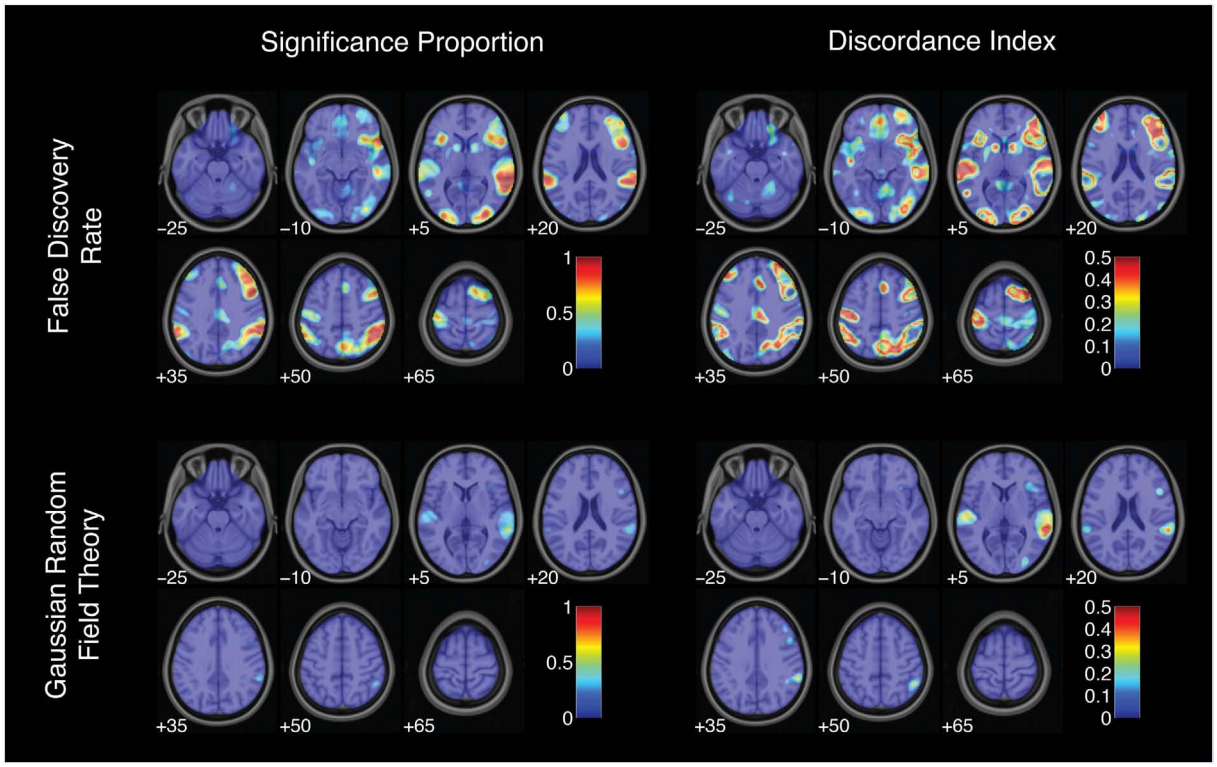


FIGURE 7 | Activation significance across analysis pipelines using false discovery rate and Gaussian random field theory error corrections. Significance proportion denotes the fraction of thresholded maps yielding significant activation for each voxel. Discordance index denotes the level of disagreement across threshold maps. Images are presented in neurological orientation, with the left hemisphere displayed on the left. Note that color scales range from 0 to 1 for significance proportion and from 0 to 0.5 for discordance index.

analysis outcomes was assessed at each voxel using the discordance index:

discordance = minimum(significance proportion, 1 − significance proportion).

This index ranged from 0 (when either 0 or 100% of analysis pipelines yielded significant activation) to 0.5 (when exactly 50% of pipelines yielded significant activation). Discordance indices were high, often reaching the theoretical maximum value of 0.5, in voxels surrounding regions of peak significance proportions (**Figures 6** and **7**). For example, voxels bordering the bilateral superior temporal gyrus and the right inferior frontal gyrus showed consistently high disagreement across analysis pipelines. These discordance rings around activation foci likely reflect the effects of differing spatial smoothing kernels on activation extent. Additional regions of disagreement included the precuneus (discordance index = 0.50), anterior cingulate cortex (discordance index = 0.44), and middle cingulate gyrus (discordance index = 0.30).

Altogether, estimates of the spatial extent of significant activation and the proportion of thresholded maps showing significant activation revealed substantial flexibility across methodological strategies. Furthermore, regions showing strong disagreement across pipelines were observed throughout the brain, both in the neighborhood of peak significance proportions and in additional isolated clusters.

DISCUSSION

According to a mathematical model of bias in scientific research (Ioannidis, 2005b), the prevalence of false positive results in published reports increases with the flexibility of research outcomes. Research outcomes are flexible to the extent that (a) researchers have access to a broad range of experimental design and data analytic strategies and (b) different research strategies yield different research outcomes. A recent survey of methods used in the fMRI literature shows that research strategies are highly flexible across published studies, with nearly as many unique methodological pipelines as studies in the sample (Carp, 2012). However, the extent to which flexible research strategies translate into flexible research outcomes remains unclear. Thus, the present study sought to estimate the flexibility of research outcomes across a wide range of complete analysis pipelines applied to a single fMRI experiment.

The present results revealed both consistency and variability across analysis pipelines. Some results were highly stable across pipelines. For example, voxels in the right superior temporal gyrus, the right inferior frontal gyrus, and the right middle occipital gyrus showed significant activation for the successful stop versus go contrast for at least 77% of the 34,560 thresholded maps considered here. Thus, although quantitative responses (i.e., activation

strength and location) in these regions proved variable across pipelines, their qualitative responses (i.e., activation significance) were quite stable. In other words, although we can be very confident that the level of stop-related activation in right inferior frontal gyrus is greater than zero, there is much greater uncertainty about the strength of this activation or its precise location within the inferior frontal gyrus. These observations are consistent with the view that the right inferior frontal gyrus is specialized for inhibitory control (e.g., Aron et al., 2004). These results also largely uphold the conclusions of the original stop-signal experiment by Aron and colleagues (2007).

However, results also varied considerably from one pipeline to another. Estimates of activation strength were highly variable across analytic pipelines: in regions of peak overall activation, significance estimates varied by over 8 Z-units. The localization of peak activation also proved to be strongly pipeline-dependent. Hundreds of unique peak coordinates were observed for each contrast, with peak locations scattered throughout much of the brain. For example, the contrast of failed stop trials versus baseline yielded activation peaks in 83 of the 128 regions defined by the AAL atlas. Finally, estimates of statistical significance showed substantial variability across pipelines as well. For example, for the successful stop versus go contrast, the proportion of activated brain voxels ranged across pipelines from 0 to 26.3%. While some voxels were consistently activated, others showed strong disagreement across analysis pipelines.

The flexibility of research outcomes illustrated here, along with mathematical models linking flexible research methods with elevated false positive rates (Ioannidis, 2005b), suggests that the risk of false positive results in fMRI research may be greater than expected. Nearly every voxel in the brain showed significant activation under at least one analysis pipeline. In other words, a sufficiently persistent researcher determined to find significant activation in virtually any brain region is quite likely to succeed. By the same token, no voxels were significantly activated across all pipelines. Thus, a researcher who hopes not to find any activation in a particular region (e.g., to rebut a competing hypothesis) can surely find a methodological strategy that will yield the desired null result. If investigators apply several analysis pipelines to an experiment and only report the analyses that support their hypotheses, then the prevalence of false positive results in the literature may far exceed the nominal rate.

It is important to note, however, that analytic flexibility only translates into elevated false positive rates when combined with selective analysis reporting. In other words, if fMRI researchers reported the results of all analysis pipelines used in their studies, then the flexibility documented here would not be problematic. To the author's knowledge, there is no evidence that fMRI researchers actually engage in selective analysis reporting. But researchers in other fields do appear to pursue this strategy. Surveys comparing research protocols to published articles show that a majority of randomized clinical trials add, omit, or replace study outcome variables – and, critically, that investigators are more likely to report significant outcomes than non-significant outcomes (Chan et al., 2004a,b; Dwan et al., 2008; Mathieu et al., 2009). Similarly, studies of putative brain volume abnormalities in patients with mental health disorders report far more positive results than would be expected given their power to detect such effects, likely reflecting the selective reporting of favorable analysis outcomes (Ioannidis, 2011). Thus, if fMRI researchers behave like researchers in other fields, then the methodological flexibility illustrated here would indeed imply an elevated rate of false positive results in the fMRI literature.

Critically, selective analysis reporting may occur without the intention or even the awareness of the investigator. For example, if the results of a new experiment do not concord with prior studies, researchers may adjust analysis parameters until the "correct" results are observed. Researchers may also elect not to describe the results of all analysis pipelines due to space limitations in journal articles or to preserve the narrative flow of a manuscript. Finally, researchers may simply not be aware of the risks posed by selective analysis reporting. Thus, although the practice of selective analysis reporting is deeply problematic, it need not reflect any malice on the part of the researchers who engage in it.

It is also important to note that bias related to analytic flexibility and selective analysis reporting is not unique to fMRI research. Indeed, previous studies have argued that selective analysis reporting can lead to false positive results in studies of randomized controlled trials (Chan et al., 2004a,b), brain volume abnormalities in psychiatric disorders (Ioannidis, 2011), and in the broader research literature (Ioannidis, 2005b). Selective analysis reporting can contaminate research results in any empirical field that allows for multiple analytic approaches – in other words, for nearly all empirical studies.

LIMITATIONS

Although the present study revealed a wide range of research outcomes for a single experiment, the approach used here likely underestimated the true flexibility of fMRI analysis methods. The present study considered two to four parameters for each analysis step, but many more parameters appear in the literature. For example, while this study considered three normalization targets, a methodological survey of recent fMRI studies (Carp, 2012) revealed a range of at least ten unique normalization targets. Similarly, while high-pass filtering cutoffs ranged from 0.33 to 750 s in this methodological survey, the present study only considered two filtering parameters: a cutoff of 128 s or no temporal filtering.

In addition, a number of key analysis steps were not considered in the present study. For example, the present approach did not investigate the effects of different strategies for coregistration between structural and functional images, for brain extraction and segmentation, for signal normalization, or for physiological noise reduction – e.g., as implemented in RETROICOR (Glover et al., 2000) or PHYCAA (Churchill et al., 2012c). Similarly, this study did not consider tools for the correction or deletion of noisy slices, brain volumes, or subjects, which may exert strong effects on analysis outcomes (Tohka et al., 2008; Power et al., 2012).

Furthermore, this study relied largely on analysis steps implemented in the SPM 8 software library. However, fMRI researchers use several versions of SPM and a wide variety of different software packages, with 32 unique libraries reported across a recent survey of fMRI studies (Carp, 2012). Studies may also combine analysis routines from multiple libraries, further increasing the flexibility of methodological approaches in the fMRI literature. This flexibility

across software options may also contribute to analytic flexibility. Different libraries may offer different strategies for the same analysis step. Further, even if multiple packages attempt to implement the same algorithms, ambiguities inherent in the translation from natural and mathematical language to computer programs may nonetheless result in differences between implementations (Ince et al., 2012). Indeed, informal comparisons suggest that choices of software package can have substantial effects on analysis outcomes (Poline et al., 2006).

The present study also relied on a relatively small sample size of 13 subjects. This sample size may have rendered many of the pipelines underpowered to detect true effects, leading to high rates of false negative results. However, the median sample size of single-group fMRI studies is approximately 15 subjects (Carp, 2012). Thus, while the present study is likely to be underpowered, it is also about as underpowered as the typical study of its kind. Thus, analytic flexibility in this sample is likely to be broadly representative of typical fMRI studies. Nevertheless, future studies should repeat this analysis using larger sample sizes to determine how or whether estimates of methods variability change with statistical power.

In addition, the extent to which the analysis pipelines investigated in this experiment resemble the true distribution of pipelines in the research literature is unclear. To the extent that the distribution of pipelines considered here differs from the distribution in the research literature, the present study may either underestimate or overestimate the true flexibility of analysis outcomes. For example, one third of the pipelines considered here estimated parameters for spatial normalization using the unified segmentation approach of SPM 8. But perhaps fewer or more than one third of published fMRI reports appear to use this approach. Analogously, all of the pipelines considered here included some form of correction for multiple comparisons. But a substantial fraction of published studies appear not to use such corrections (Carp, 2012). Thus, the pipelines examined in this study may not be fully representative of the pipelines used in published reports. However, because many published studies do not explicitly report which analysis steps and parameters were used (Carp, 2012), it is challenging to determine the true distribution of analysis pipelines in the literature. Future studies should continue to investigate the prevalence of different analysis pipelines and the effects of these pipelines on research outcomes.

Finally, it is important to note that the present study did not address the issue of which analysis pipelines should be used. Instead, this study merely sought to estimate the flexibility of research results across pipelines. As described in the Introduction, many previous studies have considered the problem of pipeline optimization (e.g., Strother et al., 2004; Churchill et al., 2012a,b).

RECOMMENDATIONS

What steps can investigators take to mitigate the risk of false positive results posed by flexible analysis methods in fMRI studies? As discussed above, the true range of fMRI methods cannot be estimated unless research reports describe analysis pipelines in detail. Thus, researchers should thoroughly describe the analysis methods chosen, as well as the reasoning behind those choices. Unfortunately, many published reports do not explicitly describe critical design and analysis decisions (Carp, 2012). Standardized reporting guidelines may help fMRI researchers to communicate methodological choices in greater detail. Such guidelines, which have been widely adopted by academic journals that publish studies of randomized controlled trials (Moher et al., 2001), diagnostic accuracy (Bossuyt et al., 2003), and observational epidemiology (von Elm et al., 2007), can significantly improve the quality of methods reporting (Plint et al., 2006). Although no consensus guidelines for the reporting of fMRI methods exist at present, the reporting recommendations by Poldrack et al. (2008) provide a useful starting point.

Flexibility in research methods may be particularly problematic when it is undisclosed (Simmons et al., 2011). For example, a hypothetical group of investigators might analyze an experiment using a range of methodological strategies and discover that only a few strategies yield positive results. If these investigators only report the pipelines that favor their hypotheses, then readers may not realize that the results of the experiment depend on (perhaps arbitrary) methodological decisions. Thus, it is critical that fMRI researchers report all analysis pipelines used in the course of data analysis, whether or not those pipelines yielded results favorable to the researchers' hypotheses. For example, if a research team initially used a canonical hemodynamic response function to model activation time series but later opted to use a finite impulse response basis set instead, the results of both strategies should be described in full. Similarly, if researchers discover that a contrast of interest yields significant activation using Monte Carlo correction but not using FDR correction, both sets of activation maps should be reported. If investigators only describe a single analysis pipeline, they should also certify that no additional pipelines were used. Finally, reviewers can work to mitigate selective analysis reporting as well. Indeed, Simmons and colleagues (2011) argue that "reviewers should require authors to demonstrate that their results do not hinge on arbitrary analytic decisions." If authors fail to indicate that they have fully described all analysis pipelines, reviewers should require them to do so; if reviewers suspect that critical results may depend on arbitrary methodological decisions, they may ask authors to defend their choices or to report the results of equally valid decisions.

Sharing data and analysis code may also help to unmask hidden flexibility in the analysis of fMRI experiments. If raw data for an experiment are freely available, then interested readers may reanalyze experiments on their own, searching out the analytic boundary conditions of reported results. Several promising data sharing initiatives focusing on resting-state imaging (the 1000 Functional Connectomes Project)[4], structural imaging (the Open Access Series of Imaging Studies database)[5], and task-based paradigms (the Open fMRI database)[6] are currently underway. Data from the present study were drawn from the Open fMRI database; analysis code is freely available online (see text footnote 3).

False positive results driven by analytic flexibility may also be mitigated by curtailing the range of available methodological

[4]http://fcon_1000.projects.nitrc.org
[5]http://www.oasis-brains.org
[6]http://openfmri.org

strategies. For example, investigators may develop standardized analysis pipelines that they apply to all of their experiments. Researchers may also simply adhere to the default options in their software packages of choice. However, while both of these approaches have the potential to reduce analytic flexibility and selective analysis reporting, they may not yield optimal analysis pipelines. Continued methodological research can also shrink the space of analytic approaches. For example, Sladky et al. (2011) argue that studies should perform slice-timing correction (but see also Poldrack et al., 2011, pp. 41–42); Purdon and Weisskoff (1998) suggest that studies should correct for temporal autocorrelation; and Lund et al. (2005) argue that studies should include head motion regression. Following these recommendations alone would reduce the number of pipelines in the present study from 6,912 to 1,296; additional research on optimal procedures and parameters may further reduce experimenter degrees of freedom. Pipeline optimization tools developed by Strother and colleagues can also be used to reduce analysis flexibility (e.g., Strother et al., 2004; Churchill et al., 2012a,b). These tools automatically identify the analysis pipelines that maximize reproducibility and prediction metrics estimated from the data on a subject-by-subject basis. Thus, using these methods reduces the risk that investigators might use a range of analysis pipelines and selectively report those that yield favorable results.

While these recommendations have the potential to reduce bias due to analytic flexibility and selective analysis reporting, they do not address other sources of error and bias. For example, while reporting the results of all analysis pipelines would (by definition) eliminate selective analysis reporting, it does not guarantee that any of the reported pipelines is optimal. As noted above, continued research on pipeline optimization may help to resolve this problem. In addition, none of these recommendations can address the problems of intentional misrepresentation or fraud. The voluntary guidelines described here cannot prevent researchers from covertly engaging in selective analysis reporting and claiming not to have done so – or from manipulating or fabricating results. Fortunately, though, relatively few scientists appear to engage in outright fraud (John et al., 2012).

CONCLUSION

The present study reveals both consistency and flexibility in the analysis of fMRI experiments. While some research outcomes were relatively stable across analysis pipelines, others varied widely from one pipeline to another. Given the extent of this variability, a motivated researcher determined to find significant activation in practically any brain region will very likely succeed – as will another researcher determined to find null results in the same region. To mitigate the effects of this flexibility on the prevalence of false positive results, investigators should either determine analysis pipelines *a priori* or identify optimal pipelines using data-driven metrics. If researchers use multiple pipelines to analyze a single experiment, the results of all pipelines should be reported – including those that yielded unfavorable results. If implemented, these steps could significantly improve the reproducibility of research in the fMRI literature.

ACKNOWLEDGMENTS

The author is supported by a National Defense Science and Engineering Graduate fellowship from the Department of Defense and a Graduate Research Fellowship from the National Science Foundation. He thanks Crystal Passmore, Kamin Kim, Emily Falk, and Russell Poldrack for helpful comments on earlier versions of this manuscript.

REFERENCES

Aron, A., Behrens, T., Smith, S., Frank, M., and Poldrack, R. (2007). Triangulating a cognitive control network using diffusion-weighted magnetic resonance imaging (MRI) and functional MRI. *J. Neurosci.* 27, 3743–3752.

Aron, A., Robbins, T., and Poldrack, R. (2004). Inhibition and the right inferior frontal cortex. *Trends Cogn. Sci. (Regul. Ed.)* 8, 170–177.

Bossuyt, P., Reitsma, J., Bruns, D., Gatsonis, C., Glasziou, P., Irwig, L., et al. (2003). Towards complete and accurate reporting of studies of diagnostic accuracy: the STARD initiative. *Br. Med. J.* 326, 41–44.

Carp, J. (2011). Optimizing the order of operations for movement scrubbing: comment on Power et al. *Neuroimage*. doi: 10.1016/j.neuroimage.2011.12.061

Carp, J. (2012). The secret lives of experiments: methods reporting in the fMRI literature. *Neuroimage* 63, 289–300.

Chan, A.-W., Hróbjartsson, A., Haahr, M., Gøtzsche, P., and Altman, D. (2004a). Empirical evidence for selective reporting of outcomes in randomized trials: comparison of protocols to published articles. *J. Am. Med. Assoc.* 291, 2457–2465.

Chan, A.-W., Krle~a-Jeric, K., Schmid, I., and Altman, D. (2004b). Outcome reporting bias in randomized trials funded by the Canadian Institutes of Health Research. *Can. Med. Assoc. J.* 171, 735–740.

Churchill, N., Oder, A., Abdi, H., Tam, F., Lee, W., Thomas, C., et al. (2012a). Optimizing preprocessing and analysis pipelines for single-subject fMRI. I. Standard temporal motion and physiological noise correction methods. *Hum. Brain Mapp.* 33, 609–627.

Churchill, N., Yourganov, G., Oder, A., Tam, F., Graham, S., and Strother, S. (2012b). Optimizing preprocessing and analysis pipelines for single-subject fMRI: 2. Interactions with ICA, PCA, task contrast and inter-subject heterogeneity. *PLoS ONE* 7, e31147. doi:10.1371/journal.pone.0031147

Churchill, N., Yourganov, G., Spring, R., Rasmussen, P., Lee, W., Ween, J., et al. (2012c). PHYCAA: data-driven measurement and removal of physiological noise in BOLD fMRI. *Neuroimage* 59, 1299–1314.

Della-Maggiore, V., Chau, W., Peres-Neto, P., and McIntosh, A. (2002). An empirical comparison of SPM preprocessing parameters to the analysis of fMRI data. *Neuroimage* 17, 19–28.

Dwan, K., Altman, D., Arnaiz, J., Bloom, J., Chan, A.-W., Cronin, E., et al. (2008). Systematic review of the empirical evidence of study publication bias and outcome reporting bias. *PLoS ONE* 3, e3081. doi:10.1371/journal.pone.0003081

Friston, K. J., Williams, S., Howard, R., Frackowiak, R. S., and Turner, R. (1996). Movement-related effects in fMRI time-series. *Magn. Reson. Med.* 35, 346–355.

Genovese, C., Lazar, N., and Nichols, T. (2002). Thresholding of statistical maps in functional neuroimaging using the false discovery rate. *Neuroimage* 15, 870–878.

Glover, G. H., Li, T. Q., and Ress, D. (2000). Image-based method for retrospective correction of physiological motion effects in fMRI: RETROICOR. *Magn. Reson. Med.* 44, 162–167.

Hopfinger, J. B., Büchel, C., Holmes, A. P., and Friston, K. J. (2000). A study of analysis parameters that influence the sensitivity of event-related fMRI analyses. *Neuroimage* 11, 326–333.

Ince, D., Hatton, L., and Graham-Cumming, J. (2012). The case for open computer programs. *Nature* 482, 485–488.

Ioannidis, J. (2005a). Contradicted and initially stronger effects in highly cited clinical research. *J. Am. Med. Assoc.* 294, 218–228.

Ioannidis, J. (2005b). Why most published research findings are false. *PLoS Med.* 2, e124. doi:10.1371/journal.pmed.0020124

Ioannidis, J. (2011). Excess significance bias in the literature on brain volume abnormalities. *Arch. Gen. Psychiatry* 68, 773–780.

John, L., Loewenstein, G., and Prelec, D. (2012). Measuring the prevalence of questionable research practices with incentives for truth telling. *Psychol. Sci.* 23, 524–532.

Lieberman, M., and Cunningham, W. (2009). Type I and Type II error concerns in fMRI research: re-balancing the scale. *Soc. Cogn. Affect. Neurosci.* 4, 423–428.

Lund, T., Nørgaard, M., Rostrup, E., Rowe, J., and Paulson, O. (2005). Motion or activity: their role in intra- and inter-subject variation in fMRI. *Neuroimage* 26, 960–964.

Mathieu, S., Boutron, I., Moher, D., Altman, D., and Ravaud, P. (2009). Comparison of registered and published primary outcomes in randomized controlled trials. *J. Am. Med. Assoc.* 302, 977–984.

Moher, D., Schulz, K. F., Altman, D., and Group, C. (2001). The CONSORT statement: revised recommendations for improving the quality of reports of parallel-group randomized trials. *J. Am. Med. Assoc.* 285, 1987–1991.

Murphy, K., Birn, R., Handwerker, D., Jones, T., and Bandettini, P. (2009). The impact of global signal regression on resting state correlations: are anti-correlated networks introduced? *Neuroimage* 44, 893–905.

Nichols, T., and Hayasaka, S. (2003). Controlling the familywise error rate in functional neuroimaging: a comparative review. *Stat. Methods Med. Res.* 12, 419–446.

Plint, A., Moher, D., Morrison, A., Schulz, K., Altman, D., Hill, C., et al. (2006). Does the CONSORT checklist improve the quality of reports of randomised controlled trials? A systematic review. *Med. J. Aust.* 185, 263–267.

Poldrack, R., Fletcher, P., Henson, R., Worsley, K., Brett, M., and Nichols, T. (2008). Guidelines for reporting an fMRI study. *Neuroimage* 40, 409–414.

Poldrack, R., Mumford, J., and Nichols, T. (2011). *Handbook of Functional MRI Data Analysis.* Cambridge: Cambridge University Press.

Poline, J.-B., Strother, S., Dehaene-Lambertz, G., Egan, G., and Lancaster, J. (2006). Motivation and synthesis of the FIAC experiment: reproducibility of fMRI results across expert analyses. *Hum. Brain Mapp.* 27, 351–359.

Power, J., Barnes, K., Snyder, A., Schlaggar, B., and Petersen, S. (2012). Spurious but systematic correlations in functional connectivity MRI networks arise from subject motion. *Neuroimage* 59, 2142–2154.

Purdon, P. L., and Weisskoff, R. M. (1998). Effect of temporal autocorrelation due to physiological noise and stimulus paradigm on voxel-level false-positive rates in fMRI. *Hum. Brain Mapp.* 6, 239–249.

Simmons, J., Nelson, L., and Simonsohn, U. (2011). False-positive psychology: undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychol. Sci.* 22, 1359–1366.

Skudlarski, P., Constable, R. T., and Gore, J. C. (1999). ROC analysis of statistical methods used in functional MRI: individual subjects. *Neuroimage* 9, 311–329.

Sladky, R., Friston, K., Tröstl, J., Cunnington, R., Moser, E., and Windischberger, C. (2011). Slice-timing effects and their correction in functional MRI. *Neuroimage* 58, 588–594.

Smith, S., Beckmann, C., Ramnani, N., Woolrich, M., Bannister, P., Jenkinson, M., et al. (2005). Variability in fMRI: a re-examination of inter-session differences. *Hum. Brain Mapp.* 24, 248–257.

Song, X. W., Dong, Z. Y., Long, X. Y., Li., S. F., Zuo, X. N., Zhu, C. Z., et al. (2011). REST: a toolkit for resting-state functional magnetic resonance imaging data processing. *PLoS ONE* 6, e25031. doi:10.1371/journal.pone.0025031

Strother, S., La Conte, S., Hansen, L., Anderson, J., Zhang, J., Pulapura, S., et al. (2004). Optimizing the fMRI data-processing pipeline using prediction and reproducibility performance metrics: I. A preliminary group analysis. *Neuroimage* 23(Suppl. 1), S196–S207.

Tohka, J., Foerde, K., Aron, A., Tom, S., Toga, A., and Poldrack, R. (2008). Automatic independent component labeling for artifact removal in fMRI. *Neuroimage* 39, 1227–1245.

Tzourio-Mazoyer, N., Landeau, B., Papathanassiou, D., Crivello, F., Etard, O., Delcroix, N., et al. (2002). Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain. *Neuroimage* 15, 273–289.

von Elm, E., Altman, D., Egger, M., Pocock, S., Gøtzsche, P., Vandenbroucke, J., et al. (2007). The Strengthening the Reporting of Observational Studies in Epidemiology (STROBE) statement: guidelines for reporting observational studies. *PLoS Med.* 4, e296. doi:10.1371/journal.pmed.0040296

Wager, T., Lindquist, M., Nichols, T., Kober, H., and Van Snellenberg, J. (2009). Evaluating the consistency and specificity of neuroimaging data using meta-analysis. *Neuroimage* 45, S210–S221.

Weissenbacher, A., Kasess, C., Gerstl, F., Lanzenberger, R., Moser, E., and Windischberger, C. (2009). Correlations and anticorrelations in resting-state functional connectivity MRI: a quantitative comparison of preprocessing strategies. *Neuroimage* 47, 1408–1416.

Woolrich, M. W., Ripley, B. D., Brady, M., and Smith, S. M. (2001). Temporal autocorrelation in univariate linear modeling of FMRI data. *Neuroimage* 14, 1370–1386.

Conflict of Interest Statement: The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 06 August 2012; accepted: 18 September 2012; published online: 11 October 2012.

*Citation: Carp J (2012) On the plurality of (methodological) worlds: estimating the analytic flexibility of fMRI experiments. Front. Neurosci. **6**:149. doi: 10.3389/fnins.2012.00149*

This article was submitted to Frontiers in Brain Imaging Methods, a specialty of Frontiers in Neuroscience.

Towards standard practices for sharing computer code and programs in neuroscience

Stephen J. Eglen[1,†], Ben Marwick[2], Yaroslav O. Halchenko[3], Michael Hanke[4,5], Shoaib Sufi[6], Padraig Gleeson[7], R. Angus Silver[7], Andrew P. Davison[8], Linda Lanyon[9], Mathew Abrams[9], Thomas Wachtler[10], David J. Willshaw[11], Christophe Pouzat[12], Jean-Baptiste Poline[13,†]

February 8, 2016

[1] Cambridge Computational Biology Institute, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, UK
[2] Department of Anthropology, University of Washington, Seattle, WA 98195-3100 USA
[3] Department of Psychological and Brain Sciences, Dartmouth College, Hanover, NH 03755 USA
[4] Institute of Psychology II, Otto-von-Guericke-University Magdeburg, 39106 Magdeburg, Germany
[5] Center for Behavioral Brain Sciences, 39106 Magdeburg, Germany
[6] Software Sustainability Insitute, University of Manchester, UK
[7] Department of Neuroscience, Physiology and Pharmacology, University College London, UK
[8] Unité de Neurosciences, Information et Complexité, CNRS, Gif sur Yvette, France
[9] International Neuroinformatics Coordinating Facility, Karolinska Institutet, Stockholm, Sweden
[10] Department of Biology II, Ludwig-Maximilians-Universität München, Germany
[11] Institute for Adaptive and Neural Computation, School of Informatics, University of Edinburgh, UK
[12] MAP5 Paris-Descartes University and CNRS UMR 8145, 45 rue des Saints-Pères, 75006 Paris, France
[13] Henry H. Wheeler, Jr. Brain Imaging Center, Helen Wills Neuroscience Institute, University of California, Berkeley, USA

† Corresponding authors: S.J.Eglen@damtp.cam.ac.uk, jbpoline@gmail.com

Background

Many areas of neuroscience are now critically dependent on computational tools to help understand the large volumes of data being created. Furthermore, computer models are increasingly being used to help predict and understand the function of the nervous system. Many of these computations are complex and often cannot be concisely reported in the methods section of a scientific article. In a few areas there are widely used software packages for analysis (e.g., SPM, FSL, AFNI, BrainVoyager, FreeSurfer in neuroimaging) or simulation (e.g. NEURON, NEST, Brian). However, we often write new computer programs to solve specific problems in the course of our research. Some of these programs may be relatively small scripts that help analyze all of our data, and these rarely get described in papers. As authors, how best can we maximize the chances that other scientists can reproduce our computations or reuse our methods on their data? Is our research reproducible[1]?

To date, the sharing of computer programs underlying neuroscience research has been the exception (see below for some examples), rather than the rule. However, there are many potential benefits to sharing these programs, including increased understanding and reuse of your work. Furthermore, open source programs can be scrutinized and improved, whereas the functioning of closed source programs remains forever unclear[2]. Funding agencies, research institutes and publishers are all gradually developing policies to reduce the withholding of computer programs relating to research[3]. The Nature family of journals has recently published opinion pieces in favor of sharing whatever code is available, in whatever form[4,5]. More recently, since October 2014, all Nature journals require papers to include a statement declaring *whether* the programs underlying central results in a paper are available. In April 2015 *Nature Biotechnology* offered recommendations for providing code with papers and began asking referees to give feedback on their ability to test code that accompanies submitted manuscripts[6]. In July 2015 F1000Research stated that "Software papers describing non-open software, code and/or web tools will be rejected" (`http://f1000research.com/channels/f1000-faculty-reviews/for-authors/article-guidelines/software-tool-articles`). Also in July 2015, BioMed Central introduced a minimum standards of reporting checklist for BMC Neuroscience and several other journals, requiring submissions to include a code availability statement and for code to be cited using a DOI or similar unique identifier[7]. **We believe that all journals should adopt policies that highly encourage, or even mandate, the sharing of software relating to journal publications.**

What should be shared?

It may not be obvious what to share, especially for complex projects with many collaborators. As advocated by Claerbout and Donoho, for computational sciences the scholarship is not the article; the "scholarship is the complete software [...]"[8,9]. So, ideally, you should share as much code and data as is needed to allow others to reproduce your work, but

this may not be possible or practical. However, it is expected that you will share key parts of the work, e.g. implementations of novel algorithms or analyses. At a minimum, we suggest following the recommendation of submission of work to ModelDB[10], i.e. to share enough code, data and documentation to allow at least one key figure from your manuscript to be reproduced. However, by adopting appropriate software tools, as mentioned in the next section, it is now relatively straightforward to share the materials required to regenerate *all* figures and tables. On the other hand, code that is not novel because it is already available, or that you feel that is unlikely to be of use to others need not be shared. This includes code that performs simple preprocessing or statistical tests, or code that deals with local computing issues such as hardware and software configurations. Finally, if your work is computationally intensive and requires a long time to run (e.g. many weeks), you may prefer to provide a small "toy" example to demonstrate your code.

By getting into the habit of sharing as much as possible, not only do you help others who wish to reproduce your work (which is a basic tenet of the scientific method), you will be helping other members of your laboratory, or even yourself in the future. By sharing your code publicly, you are more likely to write higher-quality code[11], and you will know where to find it after you've moved on from the project[12], rather than the code disappearing on a colleague's laptop when they leave your group. You will be part of a community and benefit from the code shared by others, thus contributing to a reduction in software development time for yourself and others.

Simple steps to help you share your code

Once you have decided *what* you plan to share, here are some simple guidelines for *how* to share your work. Ideally, these principles should be followed throughout the lifetime of your project, not just at the end when you wish to publish your results. Guidelines similar to these have been proposed recently in many areas of science[13–15], suggesting that they are part of norms that are emerging across disciplines. In the 'further reading' section below, we list some specific proposals from other fields that expand on the guidelines we suggest here.

Version control Use a version control system (such as Git) to develop the code[16]. The version control database can then be easily and freely shared with others using sites such as `http://github.com`[17] or `https://bitbucket.org`. These sites allow you fine control over private versus public access to your code. This means that you can keep your code repository private during its development, and then publicly share the repository at a later stage e.g. at the time of publication. It also makes it easy for others to contribute to your code, and to adapt it for their own uses.

Persistent URLs Generate stable URLs (such as a DOI) for key versions of your software. Unique identifiers are a key element in demonstrating the integrity and re-

producibility of research[18], and allow referencing of the exact version of your code used to produce figures. DOIs can be obtained freely and routinely with sites such as `http://zenodo.org` and `http://figshare.com`. If your work includes computer models of neural systems, you may wish to consider depositing these models in established repositories such as ModelDB[10], Open Source Brain[19], INCF Software Center[20] or NITRC[21]. Some of these sites allow for private sharing of repositories with anonymous peer reviewers. Journal articles that include a persistent URL to code deposited in a trusted repository meet the requirements of level two of the 'analytic methods (code) transparency' standard of the TOP guidelines[13].

License Choose a suitable license for your code to assert how you wish others to reuse your code. For example, to maximize reuse, you may wish to use a permissive license such as MIT or BSD[22]. Licenses are also important to protect you from others misusing your code. Visit `http://choosealicense.com/` to get a simple overview of which license to choose, or `http://www.software.ac.uk/resources/guides/adopting-open-source-licence` for a detailed guide.

Etiquette When working with code written by others, observe Daniel Kahneman's 'reproducibility etiquette'[23] and have a discussion with the authors of the code to give them a chance to fix bugs or respond to issues you have identified before you make any public statements. Cite their code in an appropriate fashion.

Documentation Contrary to popular expectations, you do not need to write extensive documentation or a user's guide for the code still be to useful to others[4]. However, it is worth providing a minimal README file to give an introduction to what the code does, and how to run it. For example, you should provide instructions on how to regenerate a key result, or a particular figure from a paper. Literate programming methods, where code and narrative text are interwoven in the same document, make documentation semi-automatic and can save a lot of time when preparing code to accompany a publication[24,25]. However, these methods admittedly take more time to write in the first instance, and you should be prepared to rewrite documentation when rewriting code. In any cases, well-documented code allows for easier re-use and checking.

Tools Consider using modern, widely used software tools that can help with making your computational research reproducible. Many of these tools have already been used in neuroscience and serve as good examples to follow, for example Org mode[26], IPython/Jupyter[27] and Knitr[28]. Virtualization environments, such as VirtualBox appliances and Docker containers, can also be used to encapsulate or preserve all of the computational environment so that other users can run your code without having to install numerous dependencies[29].

Case studies As well as the examples listed above in Tools[26–28], there are many prior

examples to follow when sharing your code. For example, some prominent examples of reproducible research in computational neuroscience include Vogels et al.[30] and Waskom et al.[31]; see `https://github.com/WagnerLabPapers` for details. The ModelDB repository contains over 1000 computational models deposited with instructions for reproducing key figures to papers e.g. `https://senselab.med.yale.edu/ModelDB/showModel.cshtml?model=93321` for a model of activity-dependent conductances[32].

Data Any experimental data collected alongside the software should also be released. For small datasets, this could be stored alongside the software, although it may be preferable to store experimental data separately in an appropriate repository. Both PLOS and Scientific Data maintain useful lists of subject-specific and general repositories for data storage, see `http://journals.plos.org/plosbiology/s/data-availability#loc-recommended-repositories` and `http://www.nature.com/sdata/data-policies/repositories`.

Standards Use of community standards where appropriate should be encouraged. In computational neuroscience for example, PyNN[33] and NeuroML[34] are widely used formats for making models more accessible and portable across multiple simulators.

Tests Testing the code has long been recognized as a critical step in software industry but the practice is not widely adopted yet by researchers. We recommend including test suites that demonstrate the code is producing the correct results[35]. These tests can be at a low level (testing each individual function, called unit testing) or at a higher level (e.g. testing that the program yields correct answers on simulated data)[36]. Linking tests to continuous integration services (such as Travis CI, https://travis-ci.org) allows these tests to be automatically run each time a change is made to the code, ensuring failing tests are immediately flagged and can be dealt with quickly.

Further reading (note to editor: please make this a box feature)

Varsha Khodiyar 2015 Code Sharing — read our tips and share your own. Scientific Data Blog, February 19, 2015. http://blogs.nature.com/scientificdata/2015/02/19/code-sharing-tips/

Leveque Randall 2013. Top ten reasons to not share your code (and why you should anyway). SIAM News, April 2013, http://sinews.siam.org/DetailsPage/tabid/607/ArticleID/386/Top-Ten-Reasons-To-Not-Share-Your-Code-and-why-you-should-anyway.aspx

Stodden V., & Miguez, S., 2014. Best practices for computational science: software infrastructure and environments for reproducible and extensible research. Journal of Open Research Software. 2(1), p.e21. DOI: http://doi.org/10.5334/jors.ay

Stodden, V., Leisch, F., & Peng, R. (Eds.). (2014). Implementing reproducible research. CRC press, Chapman and Hall.

Halchenko, Y. O. and Hanke, M. (2015). Four aspects to make science open "by design" and not as an after-thought. GigaScience, 4. DOI: http://doi.org/10.1186/s13742-015-0072-7

Sandve, G. K., Nekrutenko, A., Taylor, J., & Hovig E (2013) Ten simple rules for reproducible computational research. PLoS Comput Biol 9:e1003285.

Online communities discussing code sharing (note to editor: please make this a box feature)

StackExchange and related projects StackExchange is a network of free and highly active question-and-answer websites. Two members of the network are relevant to questions of code sharing: `http://stackoverflow.com/` which is dedicated to questions about programming in any language in any context, and `http://academia.stackexchange.com/questions/tagged/reproducible-research` which is focused questions relating to reproducible research in academic context. A related project is `https://neurostars.org/` which is a similar free public Q&A website focused on neuroinformatics questions, and with many questions on software packages, etc.

Scientists for Reproducible Research This is an international multi-disciplinary email list that discusses a wide range of issues relating to code sharing: `https://groups.google.com/forum/#!forum/reproducible-research`

GitHub GitHub is an online repository for computer code and programs that has a large community of researchers that develop and share their code openly on the site. GitHub is the largest and most active code sharing site (others include BitBucket and GitLab) and has convenient tools for facilitating efficient collaborative coding[37]. If you are using an open source program you may find a community of users and developers active on GitHub, where you can ask questions and report problems.

Closing remarks

Changing the behaviors of neuroscientists so that they make their code more available will likely be resisted by those who do not see the community benefits as outweighing the personal costs of the time and effort required to share code[38]. The community benefits, in our view, are obvious and substantial: we can demonstrate more robustly and transparently the reliability of our results, we can more easily adapt methods developed by others to our data, and the impact of our work increases as others can similarly reuse our methods on their data. Thus, we will endeavor to lead by example, and follow all these practices as part of our future work in all scientific publications. Even if the code we produce today will not run ten years from now, it will still be a more precise and complete expression of our analysis than the text of the methods section in our paper.

However, exhortations such as this editorial are only a small part of making code sharing a normal part of doing neuroscience; many other activities are important. All researchers should be trained in sound coding principles; such training is provided by organizations such as Software Carpentry[36] and through national neuroinformatics nodes, e.g. `http://python.g-node.org`. Furthermore, we should request code and data when reviewing, and submit to and review for journals that support code sharing. Grant pro-

posals should be checked for mentions of code availability, and we should encourage efforts toward openness in hiring, promotion, and reference letters[39]. Funding agencies and publishers should also consider mandating code sharing by default. This combination of efforts on a variety of fronts will increase the visibility of research accompanied by open source code, and demonstrate to others in the discipline that code sharing is a desirable activity that helps move the field forward.

We believe that the sociological barriers to code sharing are harder to overcome than the technical ones. Currently, academic success is strongly linked to publications and there is little recognition for producing and sharing code. Code may also be seen as providing a private competitive advantage to researchers. We challenge this view and propose that code be regarded as part of the research products which should be shared by default, and that there should be an obligation to share code for those conducting publicly funded research. We hope our code availability review will help establish such sharing as the norm. Moreover, we are advocating for code sharing as part of a broader culture change embracing transparency, reproducibility, and re-usability of research products.

Acknowledgments

This article is based upon discussions from a workshop to encourage sharing in neuroscience, held in Cambridge, December 2014. It was financially supported and organized by the International Neuroinformatics Coordinating Facility (`http://www.incf.org`), with additional support from the Software Sustainability institute (`http://www.software.ac.uk`).

References

1. *Challenges in irreproducible research* <`http://www.nature.com/nature/focus/reproducibility`>.
2. Vihinen, M. No more hidden solutions in bioinformatics. *Nature* **521,** 261 (2015).
3. Morin, A *et al.* Research priorities. Shining light into black boxes. *Science* **336,** 159–160 (2012).
4. Barnes, N. Publish your computer code: it is good enough. *Nature* **467,** 753 (2010).
5. Ince, D. C., Hatton, L. & Graham-Cumming, J. The case for open computer programs. *Nature* **482,** 485–488 (2012).
6. Rebooting review. *Nature Biotech* **33,** 319–319 (2015).
7. Kenall, A. *et al.* Better reporting for better research: a checklist for reproducibility. *BMC Neuroscience* **16,** 1–3 (2015).
8. Claerbout, J. & Karrenbach, M. *Electronic documents give reproducible research a new meaning* in *Proc. 62nd Ann. Int. Meeting of the Soc. of Exploration Geophysics* (1992), 601–604. <`http://library.seg.org/doi/pdf/10.1190/1.1822162`>.

9. Donoho, D. L. An invitation to reproducible computational research. *Biostatistics (Oxford, England)* **11,** 385–8 (2010).

10. Hines, M. L., Morse, T., Migliore, M., Carnevale, N. T. & Shepherd, G. M. ModelDB: A Database to Support Computational Neuroscience. *Journal of Computational Neuroscience* **17,** 7–11 (2004).

11. Easterbrook, S. M. Open code for open science? *Nature Geosci* **7,** 779–781 (2014).

12. Halchenko, Y. O. & Hanke, M. Four aspects to make science open "by design" and not as an after-thought. *GigaScience* **4,** 31 (2015).

13. Nosek, B. A. *et al.* Promoting an open research culture. *Science* **348,** 1422–1425 (2015).

14. Miguel, E. *et al.* Promoting transparency in social science research. *Science (New York, NY)* **343,** 30 (2014).

15. Stodden, V., Guo, P. & Ma, Z. *How journals are adopting open data and code policies* in *The First Global Thematic IASC Conference on the Knowledge Commons: Governing Pooled Knowledge Resources* (2012).

16. Blischak, J. D., Davenport, E. R. & Wilson, G. A Quick Introduction to Version Control with Git and GitHub. *PLoS Comput. Biol.* **12,** e1004668 (2016).

17. Ram, K. Git can facilitate greater reproducibility and increased transparency in science. *Source Code Biol. Med.* **8,** 7 (2013).

18. Vasilevsky, N. A. *et al.* On the reproducibility of science: unique identification of research resources in the biomedical literature. *PeerJ* **1,** e148 (2013).

19. Gleeson, P., Silver, A. & Cantarelli, M. in *Encyclopedia of Computational Neuroscience* (eds Jaeger, D. & Jung, R.) 1–3 (Springer New York, 2014).

20. *INCF Software Center* <`http://software.incf.org/`>.

21. Poline, J.-B. & Kennedy, D. in *Encyclopedia of Computational Neuroscience* (eds Jaeger, D. & Jung, R.) (Springer, 2014).

22. Stodden, V. Enabling reproducible research: Open licensing for scientific innovation. *International Journal of Communications Law and Policy* **13,** 1–25 (2009).

23. Kahneman, D. A new etiquette for replication. *Soc. Psychol.* **45,** 310 (2014).

24. Schulte, E., Davison, D., Dye, T. & Dominik, C. A multi-language computing environment for literate programming and reproducible research. *Journal of Statistical Software* **46,** 1–24 (2012).

25. Gentleman, R. & Lang, D. T. Statistical analyses and reproducible research. *Journal of Computational and Graphical Statistics* (2012).

26. Delescluse, M., Franconville, R., Joucla, S., Lieury, T. & Pouzat, C. Making neurophysiological data analysis reproducible: why and how? *J. Physiol. Paris* **106,** 159–170 (2011).

27. Stevens, J.-L. R., Elver, M. & Bednar, J. A. An automated and reproducible workflow for running and analyzing neural simulations using Lancet and IPython Notebook. *Front. Neuroinform.* **7,** 44 (2013).

28. Eglen, S. J. *et al.* A data repository and analysis framework for spontaneous neural activity recordings in developing retina. *Gigascience* **3,** 3 (2014).

29. Boettiger, C. An introduction to Docker for reproducible research. *ACM SIGOPS Operating Systems Review* **49,** 71–79 (2015).

30. Vogels, T. P., Sprekeler, H, Zenke, F, Clopath, C & Gerstner, W. Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks. *Science* **334,** 1569–1573 (2011).

31. Waskom, M. L., Kumaran, D., Gordon, A. M., Rissman, J. & Wagner, A. D. Frontoparietal representations of task context support the flexible control of goal-directed cognition. *J. Neurosci.* **34,** 10743–10755 (2014).

32. Liu, Z, Golowasch, J, Marder, E & Abbott, L. F. A model neuron with activity-dependent conductances regulated by multiple calcium sensors. *J. Neurosci.* **18,** 2309–2320 (1998).

33. Davison, A. P. *et al.* PyNN: A Common Interface for Neuronal Network Simulators. *Frontiers in Neuroinformatics* **2,** 11 (2009).

34. Cannon, R. C. *et al.* LEMS: A language for expressing complex biological models in concise and hierarchical form and its use in underpinning NeuroML 2. *Frontiers in Neuroinformatics* **8** (2014).

35. Axelrod, V. Minimizing bugs in cognitive neuroscience programming. *Front. Psychol.* **5,** 1435 (2014).

36. Wilson, G. *et al.* Best practices for scientific computing. *PLoS Biol.* **12,** e1001745 (2014).

37. Tippmann, S. My digital toolbox: Nuclear engineer Katy Huff on version-control systems. *Nature.* <`http://www.nature.com/news/my-digital-toolbox-nuclear-engineer-katy-huff-on-version-control-systems-1.16014`> (2014).

38. Stodden, V. The scientific method in practice: Reproducibility in the computational sciences. *MIT Sloan School Working Paper 4773-10* (2010).

39. LeVeque, R. J., Mitchell, I. M. & Stodden, V. Reproducible research for scientific computing: Tools and strategies for changing the culture. *Computing in Science and Engineering* **14,** 13 (2012).

Halchenko and Hanke *GigaScience* (2015) 4:31
DOI s13742-015-0072-7

(GIGA)n SCIENCE

COMMENTARY Open Access

Four aspects to make science open "by design" and not as an after-thought

Yaroslav O. Halchenko[1,2,3*] and Michael Hanke[3,4,5]

Abstract

Unrestricted dissemination of methodological developments in neuroimaging became the propelling force in advancing our understanding of brain function. However, despite such a rich legacy, it remains not uncommon to encounter software and datasets that are distributed under unnecessarily restricted terms, or that violate terms of third-party products (software or data). With this brief correspondence we would like to recapitulate four important aspects of scientific research practice, which should be taken into consideration as early as possible in the course of any project. Keeping these in check will help neuroimaging to stay at the forefront of the open science movement.

Keywords: Neuroimaging, Open science, Intellectual property

Background

A long-standing relationship already exists between open science and neuroimaging research, primarily due to the fact that most research software in the field is free and open source software (FOSS). Many software toolkits for stimulus delivery and neuroimaging data processing were either developed as such from the beginning, or were relicensed under open-source licenses at some point. This rich collection prompted centralized software and data "clearing houses" such as the Neuroimaging Informatics Tools and Resources Clearinghouse (http://nitrc.org (NITRC)) [1, 2], and integrated turnkey software platforms such as the authors' NeuroDebian (http://neuro.debian.net) [3, 4]. Increasingly, the software aspect of open science in neuroimaging is accompanied by open data, with public datasets being made available from archives such as OpenFMRI (http://openfmri.org) [5], the NITRC image repository (http://nitrc.org/ir (NITRC-IR)) [2, 6], and the Collaborative Research in Computational Neuroscience (http://crcns.org (CRCNS)) [7, 8] web portal. Despite these successes, incidents of neglected intellectual property (IP) norms, especially in scientific software, are not rare, even though neglecting or postponing IP issues poses a threat to a product's (software or data) longevity and availability, and in turn the reproducibility of associated scientific results. For instance, the discovery of just a small, possibly even unused, snippet of code covered by a restrictive incompatible license can render *all* affected releases of a piece of software illegal, requiring their removal from public servers. A frequent example of this issue is the inclusion of example code shipped with the "Numerical Recipes" books (*e.g.*, [9]), in order to facilitate development by adoption of readily available implementations.

Planning ahead

To enable future reproducibility, we first need to ensure the continued availability of today's open science products. Therefore, we must be diligent in our compliance with established norms regulating IP, which are conversely the legal tool we can use to enforce persistent "openness". We must make sure to obtain all necessary permissions to re-use or re-distribute third-party products and, in addition, determine under what conditions we can release our own work under open terms. It is important to understand that making your research products open to everyone *now* could be the only way to make them available to yourself in the *future*; for example, in case of a change of employment, or of a company policy. As it is impossible to provide an exhaustive advisory regarding IP laws, we will only outline

*Correspondence: yaroslav.o.halchenko@onerussian.com
[1]Center for Cognitive Neuroscience, Dartmouth College, 3 Maynard Street, 03755 Hanover, NH, USA
[2]Department of Psychology and Brain Sciences, Dartmouth College, 3 Maynard Street, 03755 Hanover, NH, USA
Full list of author information is available at the end of the article

the most important aspects, the first three of which concern both data and software projects, while the last one is mostly data-specific.

Respect trademarks

Trademarks (commonly names and logos) exist to protect the identity of products or services and claim their exclusive properties. Trademark owners might pursue legal action if they find their trademark infringed upon, *e.g.,* if your related product has a similar name, or contains a trademarked name. Despite usually being resolved in private, we are aware of at least a few cases where authors of FOSS projects were contacted with *cease and desist* letters from corporations and were forced to pay fines for trademark infringement.

Whenever deciding on a new project name or logo, verify that you are not infringing on an existing registered trademark, or in conflict with another open project. Both the US Patent and Trademark Office (http://www.uspto.gov (USPTO)) website and generic web search engines could be used to make a quick check. In the case of reusing names/logos of FOSS projects, check their trademark policies and consult the project owners.

Clarify ownership

The term *copyright* refers to the exclusive rights that may be enforced by some property owners. In the research context, there are typically three copyright-related issues to consider: 1) is a product *copyrightable*; and if so 2) who is the *owner*; and finally 3) do rights needs to be *transferred* to a third-party (*e.g.*, to a publisher)? Copyright applies to "any expressible form of an idea or information that is substantive and discrete" [10]. This also means that some materials may not be subject to copyright law. It is widely accepted that software (code and binaries), writing (articles, *etc.*), and artwork are copyrightable. The situation is less clear (and varies widely across different jurisdictions) in the case of application program interfaces (APIs) [see e.g., [11]] and data. For example, Creative Commons (CC) originally considered its license inappropriate for data [12], but this position was later rectified, recommending the data-oriented CC0 "no rights reserved" license [13], or the Public Domain Dedication and License (PDDL) [14], but also advising the use of CC licenses "where applicable/desired" [15, 16].

Generally authors hold the copyright of authored products, but if the product is a result of "work for hire", the copyright is commonly either owned by the employer in some jurisdictions (*e.g.*, USA), or exclusively licensed to the employer where personal authors' rights could not be transferred, as is the case in Germany [17]. It is common practice, then, that through the available legal norms, principal investigators sign off their rights to the work they were hired to do (often including off-work hours). Furthermore, rights to written works (*e.g.*, articles, books) are often transferred or exclusively licensed to a publisher, even for open access articles.

Limitations and exceptions to copyright [18], such as "fair use" in the USA [19] and "fair dealing" in the Commonwealth of Nations [20], exist to allow copyrighted works to be used without a license. However, their applicability is limited, varies widely across jurisdictions, and is open to interpretation, thus making reuse of those copyrighted works vulnerable to litigation.

To guarantee perpetual open availability of your work it is first necessary to establish whether you could make it open. If unsure, make use of a "technology transfer" department or similar (e.g., a Copyright Specialist at the library and their online resources [e.g., [21]]). Clarify whether your product could be copyrighted, and who would own said copyright, given the details of the project funding and your status/contract. Be considerate when reusing any copyrighted materials. State the copyright (years, owner) for your copyrightable product and any third-party products you incorporate. When publishing, consider venues that do not require you to surrender your copyright or to provide exclusive rights.

Choose appropriate licenses

Licenses are tightly linked to the notion of copyright, defining rights granted by an IP's owner that dictate how a product can be used and (re)distributed by a licensee. Moreover, many of the standard free and open source licenses include a disclaimer of any implicit warranty that could be associated with the product. Importantly, this is different from plain deposition of a product into the public domain (where applicable), as it may not provide this safety net.

The most common problem with licenses in the research context is related to the "borrowing" of source code from another product that was not released under a license permitting redistribution (as in the previously mention "Numerical Recipes" example) or imposing restrictions (*e.g.*, non-commercial use). The longer such incidents go unnoticed, the greater the negative impact for studies employing such products, and the greater the threat to the longevity of the product itself. A striking example of such a case is Astrolabe, Inc. vs. Olson et al. (tzdata database), in which Astrolabe claimed infringement by distributing factual data snippets copied from published atlases [22]. The authors of the tzdata database needed legal support from the Electronic Frontiers Foundation (EFF) to have the case dismissed. For sustainable open science we believe it is critical to release your work under a free and open license; it is just as critical to be pedantic in order to ensure the same freedom for all borrowed code and used products.

If your institution/employer owns a product and the copyright, negotiate the choice of license with them. If work was performed as part of a grant submitted through your institution, chances are that an open license provision is already in place. Under all circumstances, avoid creating a custom license—use a standard one from Creative Commons (http://creativecommons.org) or Open Data Commons (http://opendatacommons.org/licenses), and ideally one that is known to conform to Debian Free Software Guidelines (http://www.debian.org/social_contract#guidelines) [23] and/or is Open Source Initiative (OSI) (http://opensource.org/licenses)-approved. License wording is non-trivial legalese; products with custom licenses are often neglected by third-party users because their legal implications are not fully understood. Do not impose additional (e.g., "no clinical use") restrictions, unless unavoidable, to guarantee the widest possible adoption (see e.g., [24] for an analysis of common misconceptions about the conflict between open-source licenses and commercial interests). Choose a license appropriate to the product's domain: software, web framework, documentation, artwork, data—they might require different licenses. Respect the licenses of the third-party products you use and make sure your license is compatible with their terms.

Obtain permission to share

Whenever products are shared, permission to do so must be given for all components with third-party rights. In general, this is implemented as a license. In neuroimaging research, there is one important special case: human subject data. For projects with human participants, protection of the participants' privacy is of paramount importance when making imaging data publicly available. The respective norms are generally implemented as laws, such as [[25], 45 Code of Federal Regulations Part 46] in the US; adherence to these is scrutinized by institutional ethics committees, also known as institutional review boards (IRB). The decentralization of IRBs and the heterogeneity in their interpretation of the legal situation is one reason for the present lack of a *commonly accepted* language for participant consent forms to enable the sharing of research data. Consequently, many researchers simply exclude any data sharing statement in their consent forms to avoid frustration and delays in IRB evaluations. It is often neglected that the signed consent form is a document to protect researchers in the case that data has to be shared, for example, in order to comply with rules and regulations imposed by funding agencies, or publishers.

Although IRBs could warrant sharing of data previously collected without participants' explicit agreement that their anonymized data may be publicly shared, it is in the experimenter's interest to obtain explicit permission from participants to preclude any possible future legal trouble.

Provision public data sharing via data archives in your consent forms before you begin collecting the data. The Open Brain Consent project (http://open-brain-consent.readthedocs.org) [26] can be used to obtain samples of consent forms used at other institutions, and software for anonymization of data for sharing.

Conclusion

Established norms behind intellectual property and participant privacy cannot simply be ignored if we would like to ensure the longevity of our open scientific projects. Due attention to the four aforementioned aspects from the beginning will reduce risks and foster sharing of methodologies, data, and results of your work later on—all activities inherent to "open science".

Abbreviations

API: Application Program Interface; CC: Creative Commons; CRCNS: Collaborative Research in Computational Neuroscience; EFF: Electronic Frontiers Foundation; FOSS: Free and Open Source Software; IP: Intellectual Property; IRB: Institutional Review Board; NITRC: Neuroimaging Informatics Tools and Resources Clearinghouse; NITRC-IR: Neuroimaging Informatics Tools and Resources Clearinghouse Image Repository; OSI: Open Source Initiative; PDDL: Public Domain Dedication and License; USTPO: United States Trademarks and Patents Office.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

YOH conceived the idea for this correspondence. YOH and MH worked equally to conceptualize and write this article. Both authors read and approved the final manuscript.

Acknowledgements

We thank Samuel A. Nastase and James E. Dobson for their feedback on the manuscript. We would also like to express our gratitude to the debian-legal (http://lists.debian.org/debian-legal) community, which helped to identify and mitigate legal concerns in some problematic cases.

Author details

[1]Center for Cognitive Neuroscience, Dartmouth College, 3 Maynard Street, 03755 Hanover, NH, USA. [2]Department of Psychology and Brain Sciences, Dartmouth College, 3 Maynard Street, 03755 Hanover, NH, USA. [3]Debian Project, http://www.debian.org. [4]Psychoinformatics lab, Institute of Psychology II, Otto-von-Guericke-University, Magdeburg, Germany. [5]Center for Behavioral Brain Sciences, Magdeburg, Germany.

Received: 17 April 2015 Accepted: 7 July 2015
Published online: 18 July 2015

References

1. Neuroimaging Informatics Tools and Resources Clearinghouse. http://www.nitrc.org Accessed 13- Mar- 2013.
2. Kennedy DN, Haselgrove C, Riehl J, Preuss N, Buccigrossi R. The three NITRCs: A guide to neuroimaging neuroinformatics resources. Neuroinformatics. 2015. doi:10.1007/s12021-015-9263-8. Accessed 2015-04-15.
3. NeuroDebian – turnkey platform for neuroscience. 2009. http://neuro.debian.net Accessed 25-Apr-2011.
4. Halchenko YO, Hanke M. Open is not enough. Let's take the next step: An integrated, community-driven computing platform for neuroscience. Front. Neuroinformatics. 2012;6(00022). doi:10.3389/fninf.2012.00022. PMC3458431.
5. Portal for open sharing of functional magnetic resonance imaging (fMRI) data 2010. http://opefmri.org Accessed 6-July-2015.

6. NITRC imaging repository. http://www.nitrc.org/ir Accessed 13-Mar-2015.
7. Marketplace and discussion forum for sharing tools and data in neuroscience 2007. http://crcns.org Accessed 17-June-2015.
8. Teeters JL, Harris KD, Millman KJ, Olshausen BA, Sommer FT. Data sharing for computational neuroscience. Neuroinformatics. 2008;6(1): 47–55. doi:10.1007/s12021-008-9009-y.
9. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical Recipes 3rd Edition: The Art of Scientific Computing, 3rd edn. New York, NY, USA: Cambridge University Press; 2007.
10. Wikipedia: Copyright (2001). http://en.wikipedia.org/wiki/Copyright Accessed 27-May-2014.
11. Wikipedia: Oracle v. Google (2012). http://en.wikipedia.org/wiki/Oracle_v._Google Accessed 27-May-2014.
12. Nguyen T. Freedom to research (2008). http://sciencecommons.org/wp-content/uploads/freedom-to-research.pdf Accessed 28-May-2014.
13. Creative Common: CC0 1.0 Universal (CC0 1.0) Public Domain Dedication (2013). http://creativecommons.org/publicdomain/zero/1.0/ Accessed 09-June-2014.
14. Open Data Commons: Open Data Commons Public Domain Dedication and License (PDDL) (2007). http://opendatacommons.org/licenses/pddl/ Accessed 28-May-2014.
15. Linksvayer M. CC and data[bases]: huge in 2011, what you can do (2011). http://creativecommons.org/weblog/entry/26283 Accessed 28-May-2014.
16. Creative Commons: Data and CC licenses (2013). http://wiki.creativecommons.org/Data_and_CC_licenses Accessed 28-May-2014.
17. Wikipedia: Copyright law of Germany (2006). http://en.wikipedia.org/wiki/Copyright_law_of_Germany Accessed 18-June-2014.
18. Wikipedia: Limitations and exceptions to copyright (2003). http://en.wikipedia.org/wiki/Limitations_and_exceptions_to_copyright Accessed 27-May-2014.
19. U.S. Copyright Office. §107. Limitations on exclusive rights: Fair use, (2011). http://www.copyright.gov/title17/92chap1.html#107 Accessed 6-Apr-2015.
20. Wikipedia: Fair dealing (2002). http://en.wikipedia.org/wiki/Fair_dealing Accessed 6-Apr-2015.
21. Stim R. Copyright Overview (NOLO) (2010). http://fairuse.stanford.edu/overview/ Accessed 17-June-2014.
22. Wikipedia: Tzdata (2012). http://en.wikipedia.org/wiki/Tzdata Accessed 27-May-2014.
23. Perens B, Schuessler E. Debian Project: Debian Free Software Guidelines (DFSG). Debian, (1997). Debian. v.1.2. http://www.debian.org/social_contract#guidelines Accessed 13-March-2013.
24. Sonnenburg S, Braun M, Ong CS, Bengio S, Bottou L, Holmes G, et al. The need for open source software in machine learning. J Mach Learn Res. 2007;8:2443–466.
25. U.S. Department of Health and Human Services: Code of Federal Regulations. Title 45: Public Welfare. Part 46. Protection of Human Subjects (2009). http://www.hhs.gov/ohrp/humansubjects/guidance/45cfr46.html Accessed 17-June-2014.
26. Halchenko YO. Open Brain Consent project (2015). http://open-brain-consent.readthedocs.org Accessed 6-July-2015.

frontiers in Neuroinformatics

METHODS
published: 10 April 2015
doi: 10.3389/fninf.2015.00008

NeuroVault.org: a web-based repository for collecting and sharing unthresholded statistical maps of the human brain

Krzysztof J. Gorgolewski[1,2], Gael Varoquaux[3], Gabriel Rivera[4], Yannick Schwarz[3], Satrajit S. Ghosh[5], Camille Maumet[6], Vanessa V. Sochat[2], Thomas E. Nichols[7], Russell A. Poldrack[2], Jean-Baptiste Poline[8], Tal Yarkoni[9] and Daniel S. Margulies[1]*

[1] Max Planck Research Group for Neuroanatomy and Connectivity, Max Planck Institute for Human Cognitive and Brain Sciences, Leipzig, Germany, [2] Department of Psychology, Stanford University, Stanford, CA, USA, [3] INRIA Parietal, Neurospin Bat 145, CEA, Saclay, Gif sur Yvette, France, [4] InfoCortex UG, Frankfurt am Main, Germany, [5] McGovern Institute for Brain Research, Massachusetts Institute of Technology, Cambridge, MA, USA, [6] Warwick Manufacturing Group, University of Warwick, Coventry, UK, [7] Department of Statistics, University of Warwick, Coventry, UK, [8] Helen Wills Neuroscience Institute, University of California, Berkeley, CA, USA, [9] Department of Psychology, University of Texas at Austin, Austin, TX, USA

OPEN ACCESS

Edited by:
Sean L. Hill, International Neuroinformatics Coordinating Facility, Sweden

Reviewed by:
Angela R. Laird, Florida International University, USA
Daniel Marcus, Washington University in St. Louis, USA

***Correspondence:**
Krzysztof J. Gorgolewski, Max Planck Research Group for Neuroanatomy and Connectivity, Max Planck Institute for Human Cognitive and Brain Sciences, Stephanstrasse 1a, 04103 Leipzig, Germany
krzysztof.gorgolewski@gmail.com

Received: *13 October 2014*
Accepted: *21 March 2015*
Published: *10 April 2015*

Citation:
Gorgolewski KJ, Varoquaux G, Rivera G, Schwarz Y, Ghosh SS, Maumet C, Sochat VV, Nichols TE, Poldrack RA, Poline J-B, Yarkoni T and Margulies DS (2015) NeuroVault.org: a web-based repository for collecting and sharing unthresholded statistical maps of the human brain. Front. Neuroinform. 9:8. doi: 10.3389/fninf.2015.00008

Here we present NeuroVault—a web based repository that allows researchers to store, share, visualize, and decode statistical maps of the human brain. NeuroVault is easy to use and employs modern web technologies to provide informative visualization of data without the need to install additional software. In addition, it leverages the power of the Neurosynth database to provide cognitive decoding of deposited maps. The data are exposed through a public REST API enabling other services and tools to take advantage of it. NeuroVault is a new resource for researchers interested in conducting meta- and coactivation analyses.

Keywords: data sharing, statistical parameter mapping (SPM), meta-analysis, repository, database

Introduction

Non-invasive neuroimaging techniques such as MRI and PET have enabled unprecedented insight into the localization of various functions in the human brain. As the number of studies using such techniques continues to grow exponentially, the challenge of assessing, summarizing, and condensing their findings poses ever-greater difficulty. Even though a single study can take years to conduct, cost hundreds of thousands of dollars, and require the effort of dozens of highly trained scientists and volunteers, the output is usually reduced to an academic article, and the original data are rarely shared (Poline et al., 2012). Unfortunately, due to the historical legacy of reporting knowledge in written form (of an academic paper), the final documented results consist mostly of subjective interpretation of data with very little machine-readable information. While the introduction of common stereotaxic spaces (e.g., Talairach and MNI305) has provided an initial framework for a standard of reporting activation locations to subsequently enable meta-analyses, there are several issues with this coordinate-based strategies. First, peak coordinates are not able to fully describe the 3D shape and extent of a suprathreshold volume on a statistical map. Many papers use figures (2D or 3D) to present these statistical maps, but authors must decide which aspects of the 3D data cube to show. To fully explore all layers of the data one would need to be able to interrogate it in an interactive fashion. Furthermore, published figures are not machine-readable, and researchers that are interested in comparing their own results with published literature are forced to manually

reconstruct regions of interest (ROIs) using spheres placed at the limited reported activation locations.

A second issue is the difficulty of putting one's results in the context of other studies. The overwhelming number of brain imaging results published each year makes manual comparison both unfeasible and prone to bias. There are attempts to automatically aggregate knowledge across large sets of neuroimaging studies. For example, Neurosynth (Yarkoni et al., 2011) is a meta-analysis database that collects coordinates of activation foci from published papers and generates topic maps based on the spatial distribution of those coordinates. Such maps can aid in interpretation of new results. However, comparing a new result to a set of topic maps has so far not been implemented in a user-friendly fashion.

Finally, and most importantly, making meta-analytic inferences using only peak coordinates (or statistically thresholded maps) is problematic. It is easy to imagine a subthreshold effect that is consistent across many studies. Such an effect would not be picked up by existing meta-analysis methods (Laird et al., 2005; Yarkoni et al., 2011) because it would never be reported in the tables of peak coordinates. Considering how underpowered most human neuroscience studies are, this situation is not that unlikely. Discarding information that is below threshold in this fashion is akin to not publishing null results (Rosenthal, 1979), a dangerous practice that creates a publication bias skewing our perception of accumulated knowledge.

Using fully unthresholded statistical maps instead of solely peak coordinates would provide a significant advance in meta-analytic power. Coordinate-based meta-analysis (CBMA) methods show only modest overlap with image-based meta-analysis (IBMA; meta-analysis based on unthresholded statistical maps) methods and are less powerful (Salimi-Khorshidi et al., 2009). However, IBMA methods struggle with access to the data. Peak coordinates are easier to obtain and share because coordinate tables are an integral component of traditional neuroimaging papers, whereas very few papers provide links to unthresholded statistical maps (usually by an *ad hoc* means such as the author's web site).

NeuroVault.org is an attempt to solve these problems. It is a web-based repository that makes it easy to deposit and share statistical maps. It provides attractive visualization and cognitive decoding of the maps that can improve collaboration efforts and readability of the results. At the same time, it also provides an API for methods researchers to download the data, perform powerful analyses, or build new tools.

Results

In the following section we describe the architecture and features of NeuroVault and present two example analyses.

Platform

One of the key features of NeuroVault is the ease of uploading and sharing statistical brain maps. **Figure 1** presents a schematic overview of the platform. After logging in, users can upload a broad range of neuroimaging images and associated metadata. These data are then immediately accessible (subject to user-controlled privacy settings) via both an interactive HTML-based interface, and a comprehensive RESTful web API that facilitates programmatic interoperability with other resources. In the following sections, we discuss different aspects of the platform.

Image Upload

The NeuroVault upload process emphasizes speed and ease of use. Users can rely on existing social media accounts (Google or Facebook) to log in, and can upload individual images, or entire folders (see **Figure 1**). Users can arrange their maps into collections or to group them with tags. Each collection and statistical image in NeuroVault gets a permanent link (URL) that can be shared with other researchers or included in papers or other forms of publication (blogs, tweets, etc…). Users can specify whether each collection is public or private. The latter have a unique obfuscated URL that is not discoverable on the NeuroVault website, and thus are accessible only by whomever the owner decides to share the URL with. The option of creating private collections gives users freedom to decide who can access their data, and can facilitate a scenario in which a collection is shared privately during the pre-publication peer review process and then made public upon acceptance of a manuscript. Using a third-party (such as NeuroVault) to share data that are part of the peer review process eliminates concerns about the reviewers' anonymity. Even though we opted to minimize the required amount of metadata[1] for collections and statistical maps (to streamline the process) we give users an option to provide more information to maximize the usability of maps (see Supplementary Tables 1, 2). Most importantly, we provide ability to link a collection to a paper via a DOI to promote the associated paper and facilitate meta-analysis.

Data Types

NeuroVault is able to handle a plethora of different types of brain maps as long as they are represented as 3D NIFTI files in MNI space. This includes Z or T maps derived from task-based, resting state fMRI, and PET experiments as well as statistics derived from analyses of structural data (e.g., Voxel Based Morphometry, VBM). In addition, results from electroencephalography (EEG) and magnetoencephalography (MEG) experiments can be used with NeuroVault as long as they are converted to NIFTI volumes through source localization (Phillips et al., 2002). NeuroVault can also handle mask files (for describing ROIs), label maps (a result of parcellation studies), posterior probability maps (coming from Bayesian methods; Woolrich et al., 2004), weight maps (coming from multivariate pattern analysis methods; Haxby, 2012), and group-level lesion maps (from clinical studies). In addition, NeuroVault is able to automatically extract some metadata from SPM.mat files and FEAT folders if they are uploaded along with the statistical maps. NeuroVault also supports FSL brain atlas file format (NIFTI file with a side car XML file). When users upload such data the parcel labels are exposed through the user interface

[1]Collections require only name or DOI fields to be filled. Statistic maps require name, map type (T, Z, F, etc…), modality (BOLD-fMRI, diffusion, EEG, etc…), and cognitive paradigm [chosen from the list of Tasks in the Cognitive Atlas (Poldrack et al., 2011)] fields to be entered. For more details see Supplementary Tables 1, 2.

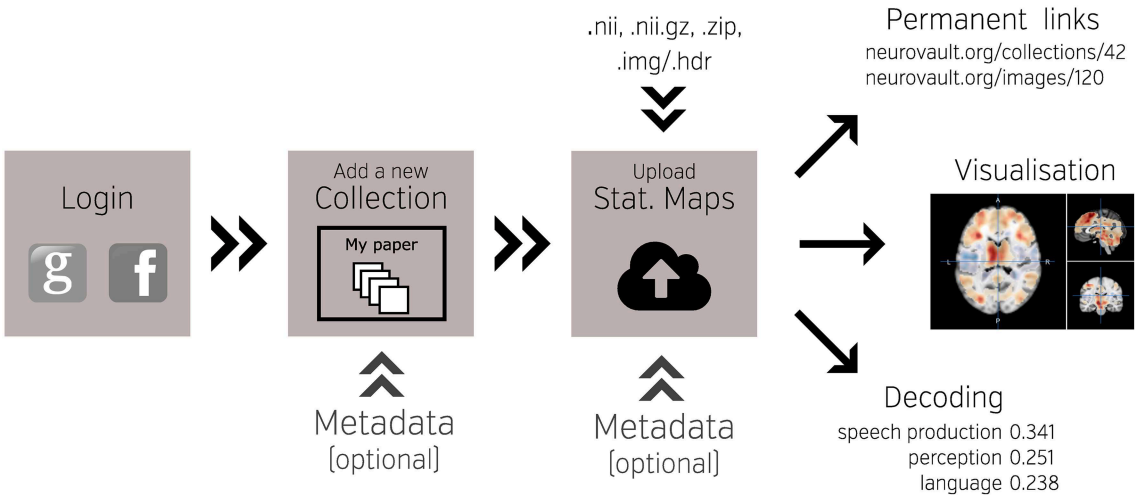


FIGURE 1 | Schematic overview of the NeuroVault platform. To begin working with NeuroVault, users are asked to create an account or log in using their Facebook or Google account. After login, the user creates a collection (representing a paper or a study). At this stage, users can provide a DOI pointing to a paper associated with the collection and/or fill in a number of fields describing the study (see Supplementary Table 1 for details). This additional information is, however, optional. After the collection is created, users can add images. This can be done one-by-one or in bulk by uploading whole folders. Again, there is an option to add more metadata describing the images. The process of creating a collection and uploading statistical maps to NeuroVault takes only 5–10 min. When the maps are uploaded, users can start benefiting from permanent link to their results, interactive web-based visualization, and real-time image decoding.

and the API (the API provides the ability to query atlases by a set of coordinates or a region name).

User Interface

NeuroVault is designed to provide intuitive, interactive visualization of uploaded images. Each image is assigned its own unique URL with an embedded JavaScript 2D/3D viewer. In contrast to traditional, static figures in published articles, users can dynamically interact with images—adjusting statistical thresholds, selecting different color maps, and loading additional brain volumes into the viewer for comparison. Using two embedded open-source JavaScript viewers (Papaya–https://github.com/rii-mango/Papaya and pycortex–https://github.com/gallantlab/pycortex), users can interrogate the data both in the volumetric space as well as on the surface (see **Figure 2**). Both viewers work inside modern web browsers and do not require any additional software to be installed. In addition to the visual representation of the volume, each page also displays any metadata associated with that image (e.g., experimental contrast, statistic type, etc. . .).

Interoperability

A major goal of NeuroVault is to directly interoperate with other existing web-based neuroimaging resources, ensuring that users can take advantage of a broad range of computational tools and resources without additional effort. There are two components to this. First, in cases where other relevant resources implemented a public API, NeuroVault can provide a direct interface to those resources. For example, at the push of a single button, each map deposited in NeuroVault can be near-instantly "decoded" using Neurosynth (see **Figure 3**). In the time of 1–2 s, the uploaded image is analyzed for its spatial correlation with a subset of the concept-based meta-analysis maps in the Neurosynth database. The user is then presented with a ranked, interactive list of maximally similar concepts, providing a quantitative, interactive way of interpreting individual statistical images that is informed by a broader literature of nearly 10,000 studies. Second, NeuroVault exposes its own public RESTful web API that provides fully open programmatic access to all public image collections and enables direct retrieval and filtering of images and associated metadata (see http://neurovault.org/api-docs for detailed description). This feature allows other researchers to leverage NeuroVault data in a broad range of desktop and web applications. To maximize the impact of data stored in NeuroVault the access to the API is unrestricted, does not require any terms of use agreements, and the data itself is distributed under the CC0 license (http://creativecommons.org/publicdomain/zero/1.0).

Accessibility

Another advantage of depositing statistical maps in NeuroVault is the increase in longevity and impact of one's research outputs. By providing a free, publicly accessible, centralized repository of whole-brain images, NeuroVault has the potential to increase the flow of data between different researchers and lab groups. Maps deposited in NeuroVault can be used by other researchers to create detailed regions-of-interest for hypothesis-driven studies or to compare results of replications. However, one of the most interesting cases of reusing statistical maps from previous studies is IBMA. Researchers wanting to perform meta-analyses can obtain the statistical maps from NeuroVault and perform annotation using various external tools/platforms

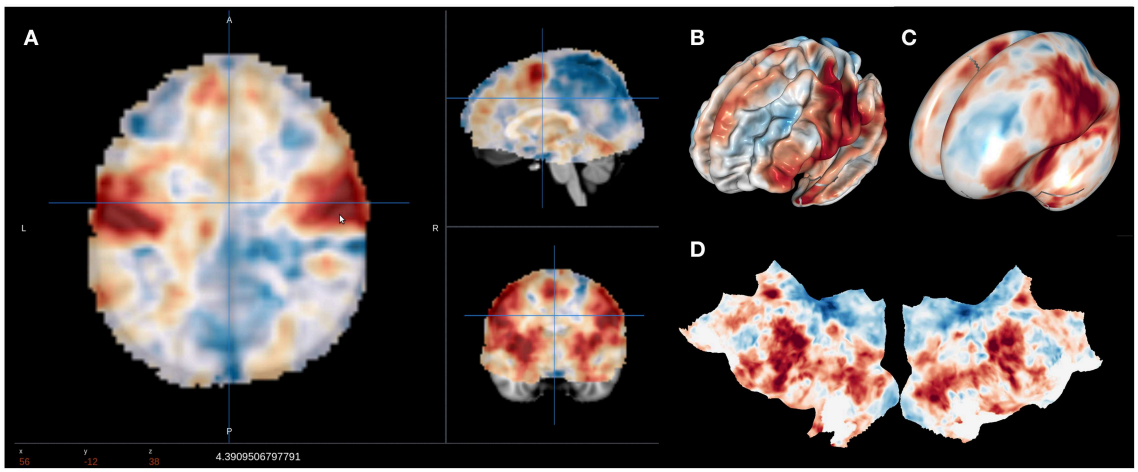


FIGURE 2 | Visualization options available in NeuroVault. The user can choose to interactively interrogate the images using 2D volumetric view **(A)**, 3D fiducial view **(B)**, 3D inflated view **(C)**, or a flattened cortical surface map **(D)**.

such as BrainMap (manual annotation; Laird et al., 2005), BrainSpell[2] (crowd-sourced annotation; http://brainspell.org), or Neurosynth (automatic annotation; Yarkoni et al., 2011). It is worth noting that so far meta-analysis in neuroimaging have rarely been performed based on labels and annotation provided by the study authors, and thus we feel outsourcing data annotation is the best current approach. Here we present a proof of concept meta-analysis based on NeuroVault data collected to date. It gives a taste of the potential this platform provides for aggregating knowledge about the human brain.

Meta-Analysis Using the Neurovault Data

At the time of submitting this publication, there were 135 non-empty public collections (53 of them associated to a publication; for up to date stats see http://neurovault.org/collections/stats) comprised of 692 images labeled as Z, T, or F statistics. Out of these, we removed 14 outliers, and selected 678 maps to perform proof of concept analyses. The outliers were detected by using a PCA on all the statistical maps (Fritsch et al., 2012). We found wrongly labeled images such as brain atlases, cropped images, and images thresholded at a very high threshold. We performed meta-analyses using the remaining set of curated images with the goal of determining whether results could be obtained using a limited set of unthresholded maps that are similar to results from large coordinate-based databases. The analyses focused on two aspects: (i) spatial distribution of activations across all maps (ii) example meta analysis of response inhibition. Code for the analyses is available at https://github.com/NeuroVault/neurovault_analysis.

Spatial Distribution of Activations

The goal of this analysis is to explore the spatial distribution of activations across all maps in Neurovault in relation to results previously reported in the literature. The analysis aims to quantify the base rate of activation at each voxel across the entire brain–i.e., to identify regions that are activated more or less often across different tasks.

Using coordinate data from the the Neurosynth database, we generated a prior activation probability map based on over 300,000 coordinates drawn from nearly 10,000 published studies. To facilitate fair comparison with the Neurosynth map, we thresholded each map from NeuroVault at a Z or T value of 3 (F maps were excluded). This discretization step approximates the standard Neurosynth procedure of taking discrete peaks reported in studies and convolving them with 3D spheres. We then generated an activation frequency map by counting the proportion of all NeuroVault maps that surpassed the threshold at each voxel.

Figure 4 (middle) shows the NeuroVault frequency map. The distribution is strikingly non-uniform throughout gray matter. In particular, the most frequently activated regions include the frontal part of the insula and dorsal anterior cingulate cortex, which form a well-known cingulate-insulate control network associated with salience processing (Seeley et al., 2007) or maintenance of task sets (ADD: Dosenbach et al., 2006, Neuron). The other structures highlighted in **Figure 4** are the inferior parietal sulcus—regions sometimes called the "task-positive network" (Fox et al., 2005)—as well as the occipital lobe, encompassing the visual cortex. The presence of the latter likely reflects the fact that the majority of experiments rely on visual stimuli. Interestingly, the networks that are most prominent on this map are largely related to attention and executive control.

The Neurosynth prior activation map is shown for comparison in **Figure 4**, top. It displays a similar density of activation, with visible attentional networks. It is worth noting that other studies have also reported similar activation density maps (e.g., Nelson et al., 2010). However, the visual cortex is much less present in the Neurosynth map compared to the NeuroVault frequency map. This could potentially be explained by the fact that results NeuroVault includes many statistical maps from fMRI

[2]BrainSpell is a web platform that allows many users to annotate neuroimaging papers and the results described in them using existing ontologies. It is based on the crowdsourcing principle–anyone is able to contribute their annotations with the assumption that the effort can be spread across multiple people and the consensus will maintain high quality.

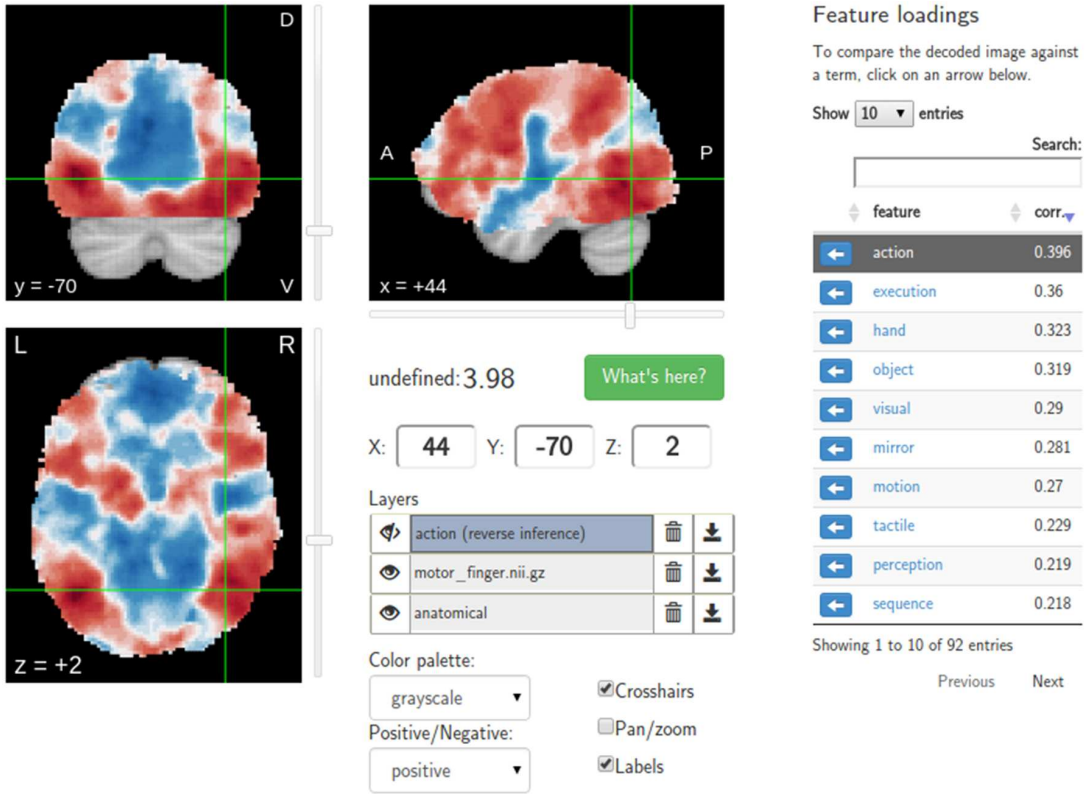


FIGURE 3 | Results of the Neurosynth decoding of a statistical map obtained through NeuroVault API. Users are able to interactively compare their maps with Neurosynth topic maps.

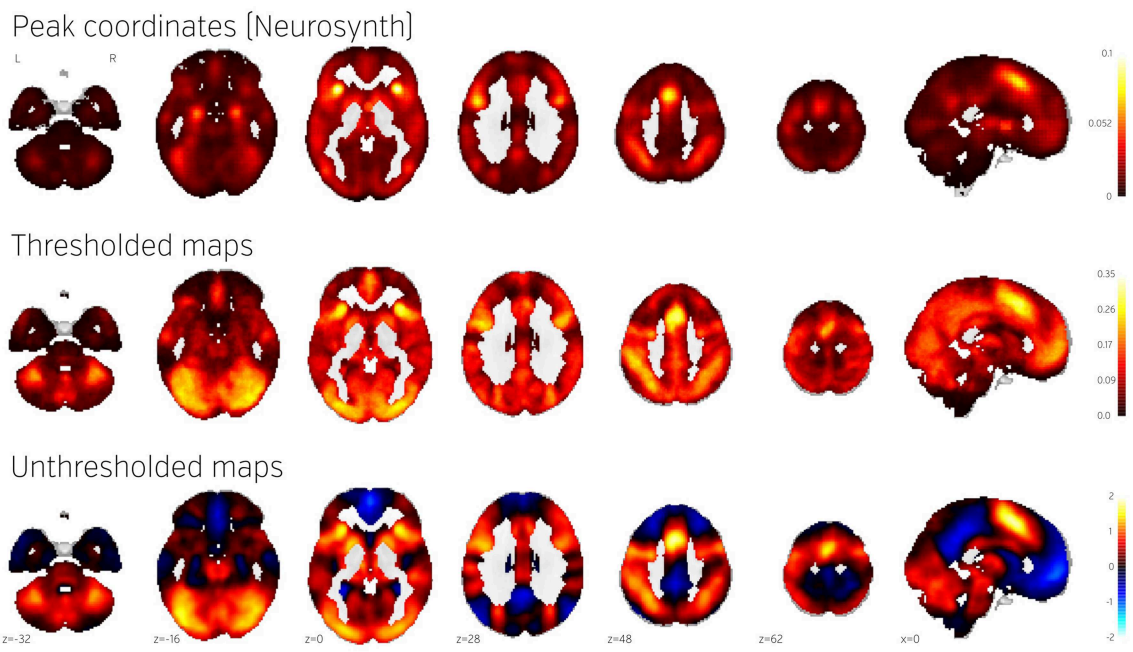


FIGURE 4 | Comparison of frequency of activation across human brain studies obtained using different methods. **Top**: Prior activation probability map obtained from coordinate-based meta-analysis using NeuroSynth. **Middle**: Proportion of maps in NeuroVault exhibiting values of T or Z higher than 3. **Bottom**: Mean of all T and Z maps (also deposited in NeuroVault). Maps from this figure are available at http://neurovault.org/collections/439/.

experiments contrasting a single condition with fixation cross baseline. However, most papers report contrasts between conditions removing the effect of the visual stimuli and thus the coordinate database will contain fewer activations in the visual cortex.

We initially thresholded the NeuroVault frequency map in order to facilitate comparison with conventional coordinate based approaches (e.g., the Neurosynth map). However, one important benefit of using unthresholded maps is the retention of additional information in the form of continuous values at all voxels. To investigate what one can gain by using unthresholded maps, we calculated a simple average of all T and Z maps across the entire NeuroVault database (**Figure 4**, bottom). Unlike the frequency map, as well as the CBMA, this analysis also captures the dominant sign of the activation, accumulating power in regions that may not cross threshold in analyses from individual studies (note that doing a principled statistical inference, e.g., computing a *p*-value or a posterior from this heterogeneous collection of maps would require methodological developments outside of the scope of this article). For example, the average unthresholded map clearly shows regions that respond, on average, by deactivating in the experimental condition relative to the baseline condition (depicted in shades of blue). This pattern spans the default-mode network (DMN), which was historically discovered in a similar analysis through observation of consistent decreases in activity across a variety of tasks (Shulman et al., 1997).

TABLE 1 | Details of the four studies included in the example meta-analysis.

NeuroVault collection ID	Number of "stop – go" maps	Number of subjects	References
42	2	15	"Triangulating a Cognitive Control Network Using Diffusion-Weighted Magnetic Resonance Imaging (MRI) and Functional MRI" (Aron et al., 2007)
98	1	24	"The generality of self-control"https://openfmri.org/dataset/ds000009 (not published)
413	2	8	"Classification learning and stop-signal (1 year test–retest)"g6 ghttps://openfmri.org/dataset/ds000017 (not published)
423	3	20	"Common Neural Substrates for Inhibition of Spoken and Manual Responses" (Xue et al., 2008)

Example Image-Based Meta Analysis Using Neurovault: Response Inhibition

To demonstrate how NeuroVault can be used for meta-analysis, we turn to the subject of response inhibition. This cognitive concept involves interrupting a prepared or ongoing response to a stimuli as a result of being presented with new information (for review see Verbruggen and Logan, 2008). We began by querying the NeuroVault API for statistical maps containing "stop signal" in the task description. This returned 66 maps. We then filtered our set to maps contrasting "stop" and "go" conditions, which resulted in eight maps across four studies (see **Table 1**). Using the NeuroVault API, we downloaded and visually inspected the maps. Since all of them contained T statistics, we converted them to standardized Z maps prior to the analysis. We estimated the degrees of freedom from the number of participants participating in each study, and this information was also obtained through the NeuroVault API. Since some of the studies contained multiple maps (one study used a test–retest protocol, and one used three different variants of the stop signal task) we created one average Z map for each study. We then used Stouffer's Z-score method (Stouffer et al., 1949; Lazar et al., 2002) to combine the results across studies in a fixed-effects meta-analysis (see **Figure 5** top)[3].

The results show consistent activation across the four studies in both left and right inferior frontal gyri and anterior insula as well as left and right parietal cortex. Similar locations have been reported in previous coordinate based meta analyses (Levy and Wagner, 2011; Swick et al., 2011). In contrast to coordinate based meta analyses our analysis also found a deactivation in medial prefrontal cortex. This brain region is one of the hubs of the default mode network, and has been found to be anticorrelated with response inhibition performance (Congdon et al., 2010). This discrepancy is likely caused by the fact that most studies do not report coordinates of deactivation and thus such patterns cannot be picked up by coordinate based meta analyses.

To validate our findings we also compared our results to the "response inhibition" topic map generated by Neurosynth, which is based on 151 studies (see **Figure 5**). The two maps exhibit remarkable similarity, with the exception of the presence of deactivations and larger cluster extents in the NeuroVault map–further validating the notion that an image-based meta analysis approach compares favorably to the widely accepted coordinate-based approach (cf. Salimi-Khorshidi et al., 2009). It is worth noting that our analysis yields plausible results despite being limited to only four studies and a limited number of subjects per study.

Discussion

We present NeuroVault, a web based platform that allows researchers to store, share, visualize, and decode maps of the human brain. This new resource can improve how human brain mapping experiments are presented, disseminated, and reused. Due to its web-based implementation NeuroVault does not require any additional software to be installed and thus is very easy to use.

[3]An alternate approach would be to submit all eight Z maps to Stouffer's method, but this neglects the intra-study correlation; our practical approach averaging each study's Zs is conservative but valid. We have presented the results of the analysis of eight maps here: https://github.com/NeuroVault/neurovault_analysis.

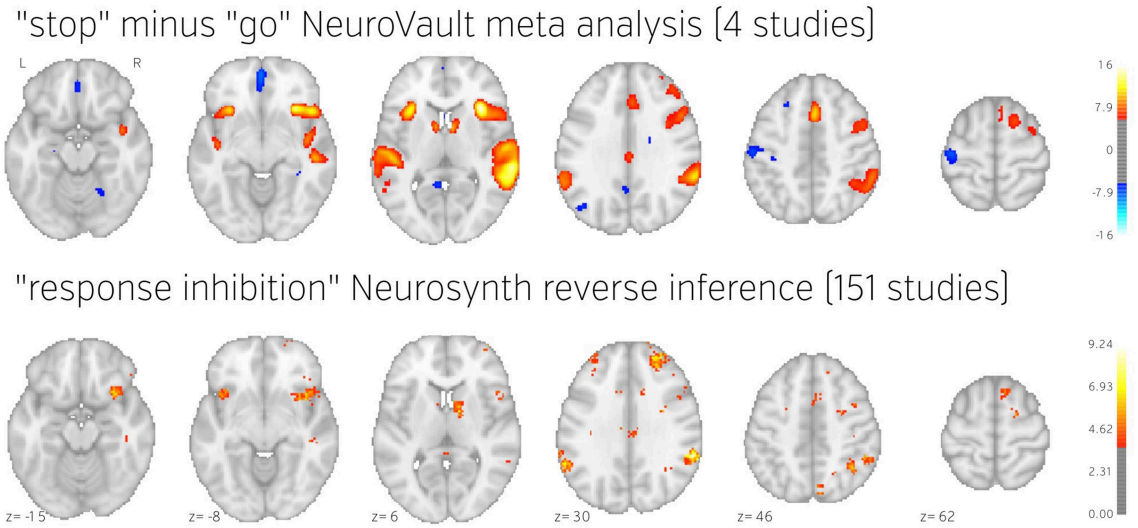


FIGURE 5 | Comparison of image based and coordinate based meta analysis of response inhibition. Meta analysis based on unthresholded statistical maps obtained from NeuroVault (top row) managed to recover the pattern of activation obtained using traditional methods despite including much fewer studies. NeuroVault map has been thresholded at $z = 6$, response inhibition map has been thresholded at $z = 1.77$ (the threshold values were chosen for visualization purposes only, but both are statistically significant at $p < 0.05$). Unthresholded versions of these maps are available at http://neurovault.org/collections/439/.

One of the biggest challenges of data sharing platforms is sustainability. Users contributing their data trust that they will be available over an extended period of time. While we cannot make any certain claims about the future, we designed the service in a way to maximize its robustness. NeuroVault is an open source project (the code is available at https://github.com/NeuroVault/NeuroVault) that is dependent only on free and open source components (web servers, content management systems, databases, etc…). This means that if the need arises, an individual with minimum web administration experience can set up NeuroVault to run on a new server. Software is not, however, the most important part of the project. To preserve the data we are performing daily offsite backups that are later copied to other locations. The procedure of restoring the service from scratch using the freely available code combined with these backups has been heavily tested. The last component of the service reliability is hardware. It is worth noting that statistical maps take considerably less space than other types of data such as raw fMRI datasets. A 500 GB hard drive (available for $50) can store almost 500,000 statistical maps. Furthermore, the cost of server maintenance and the connection to the Internet can easily be leveraged by existing academic institutions' infrastructures. In short, we argue that even though no one is able to guarantee long term availability of NeuroVault, due to the nature of its design and the type of data it is dealing with, it is easy and cheap to maintain or host at a new location given there is enough interest and the service will prove to be useful to the scientific community.

NeuroVault is not only a helpful tool for researchers who want to share, visualize, and decode their maps, it is also a resource for researchers wanting to perform meta- and coactivation analyses. Thanks to the public RESTful API and the CC0 licensing of the data there are no restrictions in terms of how and by whom the data can be used. We hope that this will accelerate progress in the field of human brain imaging and better integrate the growing compendium of resources, as there are many services that could benefit from interaction with NeuroVault. We suggest that Neurosynth and BrainMap can boost the power of their meta-analyses by working with unthresholded maps stored in NeuroVault instead of peak coordinates extracted from papers. In our analyses we have showed promising results [replication of Neurosynth frequency map, DMN deactivation and ICA topic maps similar to Smith et al. (2009)] even with an initial heterogenous set of few maps. The power of an image based meta-analysis approach is exemplified by the by the fact that using only a few 100 maps replicated results from much bigger (coordinate-based) databases (BrainMap and NeuroSynth cover, respectively 2500 and 9000 papers). We are convinced that an increased amount of data will lead to discovering new organizational principles of brain function.

The sharing of neuroimaging data can potentially raise ethical issues related to subject confidentiality (Brakewood and Poldrack, 2013). As NeuroVault is mainly focused on group data analyses, there is little chance that personal information will be included and lead to ethical issues, but the platform allows single subject analysis results to be uploaded. Uploading such data would require researchers to take extra care not to expose the the identity of their subjects.

To minimize the amount of effort needed to create a new collection, the addition of annotated metadata is optional in NeuroVault. Nevertheless, at the users' discretion, a rich set of metadata can be manually included and stored with the statistical maps. We envision that, in the future, more and more

machine-readable information will be shared and these metadata will be populated automatically to increase the potential re-use of the datasets hosted at NeuroVault. Current efforts (e.g., the previously mentioned BrainSpell), can aid the process of annotating papers (and their corresponding maps) through crowdsourcing. Ideally, machine-readable metadata would be made available directly by the software packages used to generate the statistical maps. For example, the NeuroImaging Data Model (NIDM; Keator et al., 2013) is a metadata standard that could be used to withstand metadata loss between an analysis and the upload of the statistical maps into NeuroVault. The NIDM-Results standard captures not only the statistic map, but also the design matrix, residuals, group mask, and many other pieces of information useful for future analysis. Currently only SPM natively exports to this file format, but we have adopted third party scripts to convert outputs of the FSL analyses (FEAT folders) to NIDM-Results on the server side and thus capture richer metadata in a fully automated way, and a solution for AFNI is currently being implemented. To exemplify the importance of such metadata, we present a hypothetical study that aims to train a classifier to predict some outcome from activation maps. It could be the case that effects are due to metadata variables such as the source, software, or scanner, and this finding would only be apparent given that this information is available.

It is also worth pointing out that NeuroVault is not only supporting task-based fMRI results. Results from resting state fMRI, PET, VBM, DWI, and most interestingly source reconstructed EEG/MEG experiments can be used with the platform as long as they are NIFTI files in MNI space. We plan to expand this to FreeSurfer surfaces, CIFTI files, and connectomes in the near future. Historically, aggregating results across modalities has been difficult, and we hope that this platform can start to improve upon this situation, by providing one common place for storing and sharing statistical maps.

NeuroVault is also integrated with the Resource Identification Initiative through The Neuroscience Information Framework (NIF, see Gardner et al., 2008 and http://neuinfo.org/). This interdisciplinary project assigns identifiers to resources and tools used in research that are then included in publications and later indexed by Google Scholar and PubMed. These identifiers work with the PubMed LinkOut service (http://www.ncbi.nlm.nih.gov/projects/linkout/) so that links can automatically be made between the tools and publications on web pages describing either. Assigning these resource identifiers to statistical maps, then, would both allow for the creators to track how the maps are used and grant academically acknowledge credit (even in the case when the maps come from unpublished studies).

Limitations and Future Directions

One of the biggest limitations of the NeuroVault database is its size and the voluntary nature of data contributions. For any meta-analysis to be meaningful the sample of included studies needs to be representative. Including only papers that have corresponding statistical maps in NeuroVault instead of all papers might create unpredictable biases (although this bias is most likely to be toward inclusion of more trustworthy results; see Wicherts et al., 2011). One-way of dealing with this is to enforce deposition of statistical maps across all published research. This would be a drastic move, and some data sharing initiatives in neuroimaging in the past were met with considerable opposition from the community (Van Horn and Gazzaniga, 2013). Instead we have reached out to leading journals in the field to encourage (but not require) authors of accepted papers to deposit statistical maps in NeuroVault. So far, NeuroImage, F1000Research and Frontiers in Brain Imaging Methods have joined us in the quest of providing better and more open representation of experimental results. We hope that with time publishing statistical maps will become standard practice.

NeuroVault fills a specific niche in the neuroinformatics ecosystem. The main purpose is to collect, store, and share statistical maps. We leave the task of extracting knowledge (tags, labels terms) out of papers and associating them with the statistical maps to other platforms BrainSpell, Neurosynth, and BrainMap. We also do not aspire to provide a platform for performing meta analyses (neurosynth and BrainMap facilitate this). This decision is intentional and was made to focus on one specific task and do it well. Thus, in the future we want to focus on (i) making the platform more attractive for researchers (so the motivation for data deposition will increase), (ii) making the data deposition process easier and automatic extraction of metadata more effective, and (iii) reaching out to the community to make sharing of statistical maps a common practice. In terms of the first goal we are working hard on adding new features that will help researchers to understand and visualize their maps. One of such features (currently in beta) is map comparison: users will be able to compare their map with all the other maps deposited in the database and thus easily find experiments with similar imaging results. The second goal will involve tighter integration with the most popular software packages (capitalizing on the NIDM-Results standard). We plan to provide a single click solution for uploading maps to NeuroVault that will be available within analysis software such as SPM, FSL, and AFNI. Finally the third goal, probably the most important, and also the hardest, involves continuous conversations with academic journals and conference organizations such as the OHBM. We hope that by including all of the interested parties in these conversation we will be able to convince the community about the pressing need for sharing statistical maps.

Conclusion

In this work we have described NeuroVault—a web-based repository that allows researchers to store, share, visualize, and decode unthresholded statistical maps of the human brain. This project not only helps individual researchers to disseminate their results and put them in the context of existing literature, but it also enables aggregation of data across studies. Through our analyses we have shown that with only a few hundred statistical maps we can achieve results comparable to those obtained with thousands of sets of coordinates. NeuroVault is free and unencumbered by data use agreements. The data is available and the database queryable via the web interface and RESTful API. This simple and

modern platform opens the door to developing novel methods to draw inferences from a meta-analytic database.

Acknowledgments

This work was partially funded by the National Institutes of Health (NIH), R01MH096906 [TY], NSF OCI-1131441 [RP], International Neuroinformatics Coordinating Facility (INCF) and the Max Planck Society [KJG, DSM]. We thank the International Neuroinformatics Coordinating Facility (INCF) Neuroimaging Data Sharing (Nidash) task force members for their input during several discussions.

Supplementary Material

The Supplementary Material for this article can be found online at: http://www.frontiersin.org/journal/10.3389/fninf.2015.00008/abstract

References

Aron, A. R., Behrens, T. E., Smith, S., Frank, M. J., and Poldrack, R. A. (2007). Triangulating a cognitive control network using diffusion-weighted magnetic resonance imaging (MRI) and functional MRI. *J. Neurosci.* 27, 3743–3752. doi: 10.1523/JNEUROSCI.0519-07.2007

Brakewood, B., and Poldrack, R. A. (2013). The ethics of secondary data analysis: considering the application of Belmont principles to the sharing of neuroimaging data. *Neuroimage* 82, 671–676. doi: 10.1016/j.neuroimage.2013.02.040

Congdon, E., Mumford, J. A., Cohen, J. R., Galvan, A., Aron, A. R., Xue, G., et al. (2010). Engagement of large-scale networks is related to individual differences in inhibitory control. *Neuroimage* 53, 653–663. doi: 10.1016/j.neuroimage.2010.06.062

Dosenbach, N. U. F., Visscher, K. M., Palmer, E. D., Miezin, F. M., Wenger, K. K., Kang, H. C., et al. (2006). A core system for the implementation of task sets. *Neuron* 50, 799–812. doi: 10.1016/j.neuron.2006.04.031

Fox, M. D., Snyder, A. Z., Vincent, J. L., Corbetta, M., Van Essen, D. C., and Raichle, M. E. (2005). The human brain is intrinsically organized into dynamic, anticorrelated functional networks. *Proc. Natl. Acad. Sci. U.S.A.* 102, 9673–9678. doi: 10.1073/pnas.0504136102

Fritsch, V., Varoquaux, G., Thyreau, B., Poline, J.-B., and Thirion, B. (2012). Detecting outliers in high-dimensional neuroimaging datasets with robust covariance estimators. *Med. Image Anal.* 16, 1359–1370. doi: 10.1016/j.media.2012.05.002

Gardner, D., Akil, H., Ascoli, G. A., Bowden, D. M., Bug, W., Donohue, D. E., et al. (2008). The neuroscience information framework: a data and knowledge environment for neuroscience. *Neuroinformatics* 6, 149–160. doi: 10.1007/s12021-008-9024-z

Haxby, J. V. (2012). Multivariate pattern analysis of fMRI: the early beginnings. *Neuroimage* 62, 852–855. doi: 10.1016/j.neuroimage.2012.03.016

Keator, D. B., Helmer, K., Steffener, J., Turner, J. A., Van Erp, T. G. M., Gadde, S., et al. (2013). Towards structured sharing of raw and derived neuroimaging data across existing resources. *Neuroimage* 82, 647–661. doi: 10.1016/j.neuroimage.2013.05.094

Laird, A. R., Lancaster, J. J., and Fox, P. T. (2005). BrainMap. *Neuroinformatics* 3, 65–77. doi: 10.1385/NI:3:1:065

Lazar, N. A., Luna, B., Sweeney, J. A., and Eddy, W. F. (2002). Combining brains: a survey of methods for statistical pooling of information. *Neuroimage* 16, 538–550. doi: 10.1006/nimg.2002.1107

Levy, B. J., and Wagner, A. D. (2011). Cognitive control and right ventrolateral prefrontal cortex: reflexive reorienting, motor inhibition, and action updating. *Ann. N.Y. Acad. Sci.* 1224, 40–62. doi: 10.1111/j.1749-6632.2011.05958.x

Nelson, S. M., Dosenbach, N. U. F., Cohen, A. L., Wheeler, M. E., Schlaggar, B. L., and Petersen, S. E. (2010). Role of the anterior insula in task-level control and focal attention. *Brain Struct. Funct.* 214, 669–680. doi: 10.1007/s00429-010-0260-2

Phillips, C., Rugg, M. D., and Friston, K. J. (2002). Anatomically informed basis functions for EEG source localization: combining functional and anatomical constraints. *Neuroimage* 16, 678–695. doi: 10.1006/nimg.2002.1143

Poldrack, R. A., Kittur, A., Kalar, D., Miller, E., Seppa, C., Gil, Y., et al. (2011). The cognitive atlas: toward a knowledge foundation for cognitive neuroscience. *Front. Neuroinform.* 5:17. doi: 10.3389/fninf.2011.00017

Poline, J.-B., Breeze, J. L., Ghosh, S., Gorgolewski, K., Halchenko, Y. O., Hanke, M., et al. (2012). Data sharing in neuroimaging research. *Front. Neuroinform.* 6:9. doi: 10.3389/fninf.2012.00009

Rosenthal, R. (1979). The file drawer problem and tolerance for null results. *Psychol. Bull.* 86, 638. doi: 10.1037/0033-2909.86.3.638

Salimi-Khorshidi, G., Smith, S. M., Keltner, J. R., Wager, T. D., and Nichols, T. E. (2009). Meta-analysis of neuroimaging data: a comparison of image-based and coordinate-based pooling of studies. *Neuroimage* 45, 810–823. doi: 10.1016/j.neuroimage.2008.12.039

Seeley, W. W., Menon, V., Schatzberg, A. F., Keller, J., Glover, G. H., Kenna, H., et al. (2007). Dissociable intrinsic connectivity networks for salience processing and executive control. *J. Neurosci.* 27, 2349–2356. doi: 10.1523/JNEUROSCI.5587-06.2007

Shulman, G. L., Fiez, J. A., Corbetta, M., Buckner, R. L., Miezin, F. M., Raichle, M. E., et al. (1997). Common blood flow changes across visual tasks: II. Decreases in cerebral cortex. *J. Cogn. Neurosci.* 9, 648–663. doi: 10.1162/jocn.1997.9.5.648

Smith, S. M., Fox, P. T., Miller, K. L., Glahn, D. C., Fox, P. M., Mackay, C. E., et al. (2009). Correspondence of the brain's functional architecture during activation and rest. *Proc. Natl. Acad. Sci. U.S.A.* 106, 13040–13045. doi: 10.1073/pnas.0905267106

Stouffer, S. A., Suchman, E. A., Devinney, L. C., Star, S. A., and Williams, R. M. Jr. (1949). *The American Soldier: Adjustment During Army Life. (Studies in Social Psychology in World War II, Vol. 1.)*. Available online at: http://psycnet.apa.org/psycinfo/1950-00790-000

Swick, D., Ashley, V., and Turken, U. (2011). Are the neural correlates of stopping and not going identical? Quantitative meta-analysis of two response inhibition tasks. *Neuroimage* 56, 1655–1665. doi: 10.1016/j.neuroimage.2011.02.070

Van Horn, J. D., and Gazzaniga, M. S. (2013). Why share data? Lessons learned from the fMRIDC. *Neuroimage* 82, 677–682. doi: 10.1016/j.neuroimage.2012.11.010

Verbruggen, F., and Logan, G. D. (2008). Response inhibition in the stop-signal paradigm. *Trends Cogn. Sci.* 12, 418–424. doi: 10.1016/j.tics.2008.07.005

Wicherts, J. M., Bakker, M., and Molenaar, D. (2011). Willingness to share research data is related to the strength of the evidence and the quality of reporting of statistical results. *PLoS ONE* 6:e26828. doi: 10.1371/journal.pone.0026828

Woolrich, M. W., Behrens, T. E. J., Beckmann, C. F., Jenkinson, M., and Smith, S. M. (2004). Multilevel linear modelling for FMRI group analysis using Bayesian inference. *Neuroimage* 21, 1732–1747. doi: 10.1016/j.neuroimage.2003.12.023

Xue, G., Aron, A. R., and Poldrack, R. A. (2008). Common neural substrates for inhibition of spoken and manual responses. *Cereb. Cortex* 18, 1923–1932. doi: 10.1093/cercor/bhm220

Yarkoni, T., Poldrack, R. A., Nichols, T. E., Van Essen, D. C., and Wager, T. D. (2011). Large-scale automated synthesis of human functional neuroimaging data. *Nat. Methods* 8, 665–670. doi: 10.1038/nmeth.1635

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in
NEUROINFORMATICS

METHODS ARTICLE
published: 08 July 2013
doi: 10.3389/fninf.2013.00012

Toward open sharing of task-based fMRI data: the OpenfMRI project

Russell A. Poldrack[1]*, Deanna M. Barch[2], Jason P. Mitchell[3], Tor D. Wager[4], Anthony D. Wagner[5], Joseph T. Devlin[6], Chad Cumba[1], Oluwasanmi Koyejo[7] and Michael P. Milham[8]

[1] *Imaging Research Center, University of Texas, Austin, TX, USA*
[2] *Department of Psychology, Washington University, St. Louis, MO, USA*
[3] *Department of Psychology, Harvard University, Cambridge, MA, USA*
[4] *Department of Psychology, University of Colorado, Boulder, CO, USA*
[5] *Department of Psychology and Neurosciences Program, Stanford University, Stanford, CA, USA*
[6] *Cognitive, Perceptual and Brain Sciences, University College London, London, UK*
[7] *Electrical and Computer Engineering, University of Texas, Austin, TX, USA*
[8] *Center for the Developing Brain, Child Mind Institute, New York, NY, USA*

Edited by:
Daniel Gardner, Weill Cornell Medical College, USA

Reviewed by:
John Van Horn, University of California at Los Angeles, USA
David N. Kennedy, University of Massachusetts Medical School, USA
Robert Dougherty, S-Dougherty Law Offices, USA

***Correspondence:**
Russell A. Poldrack, Imaging Research Center, University of Texas, 100 E. 24th St., Austin, 78712 TX, USA
e-mail: poldrack@gmail.com

The large-scale sharing of task-based functional neuroimaging data has the potential to allow novel insights into the organization of mental function in the brain, but the field of neuroimaging has lagged behind other areas of bioscience in the development of data sharing resources. This paper describes the OpenFMRI project (accessible online at http://www.openfmri.org), which aims to provide the neuroimaging community with a resource to support open sharing of task-based fMRI studies. We describe the motivation behind the project, focusing particularly on how this project addresses some of the well-known challenges to sharing of task-based fMRI data. Results from a preliminary analysis of the current database are presented, which demonstrate the ability to classify between task contrasts with high generalization accuracy across subjects, and the ability to identify individual subjects from their activation maps with moderately high accuracy. Clustering analyses show that the similarity relations between statistical maps have a somewhat orderly relation to the mental functions engaged by the relevant tasks. These results highlight the potential of the project to support large-scale multivariate analyses of the relation between mental processes and brain function.

Keywords: informatics, data sharing, metadata, multivariate, classification

1. INTRODUCTION

The sharing of data has become commonplace in many parts of science, and the availability of large databases of shared data has led to impressive advances that could not have been made without such sharing. For example, the GenBank database (http://www.ncbi.nlm.nih.gov/genbank/) contains all publicly available DNA sequences, which currently number more than 100 million annotated sequences. Using these data, a large number of data mining tools have been developed that allow computational gene discovery (i.e., mapping from sequences to specific genes) as well as prediction of the proteins that are encoded by a sequence. Such tools have greatly increased the power of molecular biology and genomics research. An excellent example of the power of these tools comes from the outbreak of *E. coli* O104:H4 infection in Germany in 2011. The genetic sequences obtained from these organisms were made public on the Internet, and within days researchers around the world had determined the genes responsible for the especially high virulence of the bacterium as well as its relation to other known *E. coli* strains. Such applications highlight one of the most important benefits of data sharing: By combining shared data into large databases, it is possible to identify relationships between effects at different levels of analysis (e.g., genetic sequence and bacterial virulence) that otherwise would be much more difficult to identify.

The open sharing of fMRI data has the potential to revolutionize cognitive neuroscience in much the same way (Van Horn and Gazzaniga, 2002; Poline et al., 2012). First, doing so would allow investigators to search for similar patterns of activity in multiple datasets, and thus to identify relations between cognitive tasks that result in these similarities. This could help address the common problem of reverse inference (Poldrack, 2006), wherein patterns of activation are informally used to infer putative mental function. The sharing of fMRI data would allow researchers to more formally assess the specificity of observed brain activity with various cognitive tasks, thereby permitting probabilistic inferences about the role of various brain regions or networks in mental function. Second, by allowing researchers to decompose the mental processes involved in each study and then test for associations between these processes and brain activity, large databases would support more direct identification of relations between mental processes and brain networks, rather than relying on associations with activation on single tasks (Poldrack et al., 2009; Yarkoni et al., 2011). Third, by making published datasets available to a wide range of researchers, open sharing would

encourage the re-analysis of existing data with new analysis methods (e.g., Greicius et al., 2004). Doing so would not only obviate the need for additional data collection in some cases, but would also allow more direct comparison between previous and new analysis methods. In addition to these judicious effects on scientific knowledge, the availability of a large database of published datasets would also have a powerful impact on education and training, as new trainees and individuals at institutions without imaging resources would have access to extensive datasets. Finally, there is an ethical argument to be made that sharing of data is essential in order to fully respect the contributions of the human subjects who participate in research studies (Brakewood and Poldrack, 2013).

1.1. CHALLENGES OF fMRI DATA SHARING

Although the benefits of sharing fMRI data are clear, the challenges of doing so are even clearer. The sharing of fMRI data is made difficult by a number of factors including large datasets, need for common data formats, complex metadata, and social factors.

1.1.1. Large datasets

The usual fMRI dataset comprises a set of functional images (usually 4–8 scanning runs lasting 6–10 min each) along with structural brain images and other associated measurements (such as physiological and behavioral data). The functional data typically consist of 4-dimensional data sets (3 spatial dimensions x time); depending upon the number and length of scanning runs, spatial resolution, and the number of slices acquired, the raw functional data for a single subject in an fMRI study can range in size from 50 MB to more than 1 GB, and most studies have at least 15 subjects. Datasets of this size require substantial resources for storage and processing, although improvements in computing technology have made it feasible to store and process such datasets on commodity hardware. In addition, cloud-based resources make the sharing and analysis of very large datasets possible without purchasing any physical hardware.

1.1.2. The need for common data formats

Ten years ago, the field of neuroimaging was a virtual Tower of Babel, with a number of incompatible image data formats used across different software packages and scanner platforms. This made early efforts at data sharing very difficult. In recent years, the field has gravitated toward two standard formats for data storage which have addressed this problem to some degree. Most MRI scanners now save the raw MRI data to the DICOM format. However, DICOM is not convenient for everyday analysis due to the fact that it requires a large number of small files. In addition, the DICOM standard varies between implementations across different scanners. Within the neuroimaging community, a standard known as NIfTI (http://nifti.nimh.nih.gov/nifti-1/) has been widely adopted in the field and is now supported by every major fMRI analysis package. The NIfTI format provides support for 3D and 4D images and supports rich metadata including orientation information, which can help alleviate problems with left-right orientation that were common in early days of fMRI. Although the NIfTI format is a step forward, differences in its implementation remain between software packages, such that problems can still arise when using data processed across multiple packages.

1.1.3. Complex metadata

To describe a fMRI study fully, researchers must specify a large number of details. These include:

- MRI acquisition parameters
- Design and timing of the experimental task
- Description of the participants
- Data preprocessing and analysis procedures
- Description of the mental processes being examined
- Description of behavioral data during task performance

Recent work has begun to develop frameworks for minimal information regarding fMRI studies (Poldrack et al., 2008) and for more detailed descriptions of cognitive tasks (Turner and Laird, 2012). However, a systematic framework for describing these metadata does not yet exist, and fully describing an fMRI study thus remains a significant challenge.

1.1.4. Researcher participation

Successful data sharing requires researchers who are not just willing but motivated to share their data. However, there are a number of reasons why investigators might not wish to share their data. First, data sharing requires significant effort on behalf of an investigator, and the perceived benefits have often not been sufficient to motivate this extra work [though the move toward "data papers" could help by providing published credit for data sharing; cf. Gorgolewski et al. (2013a)]. Second is the desire for exclusive rights to re-analyze the data in the future, either to test different hypotheses or to apply different analysis techniques. This is particularly the case with high-value datasets (e.g., data from special populations), where keeping the dataset private can provide a significant competitive advantage. Others might be reluctant to share data due to fear of subsequent analyses that could uncover problems with the data or invalidate the results from their publications. A recent study provided direct evidence that concerns about followup analyses may underlie the unwillingness to share; an analysis of psychology papers for which data were shared upon request vs. those that were not shared found that papers for which data were not shared had a higher rate of apparent errors in statistical reporting as well as having smaller effect sizes on average (Wicherts et al., 2011).

1.2. PREVIOUS fMRI DATA SHARING PROJECTS

The first effort to openly share fMRI data was the fMRI Data Center (fMRIDC) (Van Horn et al., 2001), which was originated at Dartmouth and subsequently moved to Santa Barbara in 2007. Van Horn and Gazzaniga (2012) recently outlined the history of the project and discussed its impact and the lessons learned in the project. The fMRIDC amassed 107 fMRI datasets which remain available for shipment via physical media. Data obtained from the fMRIDC were used in at least ten papers that presented novel analyses, utilizing both single datasets as well as mega-analyses combining multiple datasets (see Van Horn and

Ishai, 2007). These ranged from analyses of task-related connectivity (Mechelli et al., 2003) to one of the earliest studies of the "default mode" in Alzheimer's disease (Greicius et al., 2004) to an exploration of consciousness that combined data across multiple studies (Lloyd, 2002). The fMRIDC also aroused controversy within the neuroimaging community early in its existence when it was announced that some journals (including the *Journal of Cognitive Neuroscience* and *Proceedings of the National Academy of Sciences*) would require authors to submit their data to the center (Editorial, 2000). This reluctance of the community to participate in data sharing via fMRIDC was likely due to a number of factors including a lack of social consensus at the time regarding the value of data sharing as well as concern about the significant amount of effort required to submit datasets to the fMRIDC. Ultimately the project discontinued addition of new datasets due to a lack of continued funding, but the data remain available and it clearly played a role in establishing the utility of sharing complete fMRI data sets. The fMRIDC stands as a very important guiding example for data sharing in neuroimaging.

A more recent project has focused on open sharing of resting state fMRI data. Originally known as the *1000 Functional Connectomes Project* (FCP), and now as the International Neuroimaging Data-sharing Initiative (INDI) (Mennes et al., 2012), this project has already shared nearly 5000 subjects' worth of resting state fMRI data collected from centers around the world, making the data openly available via the web. Initial mega-analysis (i.e., reanalysis of the full combined dataset, as opposed to meta-analysis of summary statistics) of this dataset provided novel insights into the stability and variability of resting state networks (Biswal et al., 2010), and other groups have already used the data to make new discoveries about the organization of resting brain networks (Tomasi and Volkow, 2010). A limitation of the initial FCP dataset was that there was very little phenotype data included other than sex and age; however, more recently this group has begun prospectively sharing data with a greater amount of phenotype information, including the deeply-phenotyped NKI-Rockland sample (Nooner et al., 2012).

The FCP/INDI project shows how a community effort can result in the availability of large, freely-available datasets that can be used to enable novel scientific discoveries. The success of FCP/INDI also suggests that the neuroimaging community has a greater appreciation for the benefits of data sharing than it did when the fMRIDC first began 10 years ago. At the same time, it is important to highlight that by focusing on resting state fMRI, the FCP/INDI project sidesteps many of the difficult metadata problems that are present for task fMRI (in particular, the need to represent task paradigms and behavioral data in a systematic way).

In addition to these efforts at sharing raw data, another set of efforts has focused on sharing of highly processed data, namely the activation coordinates reported in papers. These include Brainmap (http://www.brainmap.org) (Laird et al., 2005), SumsDB (http://sumsdb.wustl.edu/sums/), and Neurosynth (http://www.neurosynth.org) (Yarkoni et al., 2011), each of which provides tools to perform coordinate-based meta-analyses. This approach has been very powerful, but at the same time is clearly limited by the coarseness of the data at every level; the shortcomings of coordinate-based meta-analysis in comparison to meta-analysis based on full image data have been shown by Salimi-Khorshidi et al. (2009). These results suggest that in addition to the sharing of raw data, there is likely utility in the sharing of processed data (e.g., statistical images).

2. THE OpenfMRI PROJECT

Here, we describe a new resource for the open dissemination of functional neuroimaging data, called the OpenfMRI Project (accessible online at http://www.openfmri.org). The goal of the project is to support the free and open distribution of both raw and processed neuroimaging datasets, focused primarily on whole-brain datasets from task-based fMRI studies. The project aims to use what was learned in previous data sharing efforts and take advantage of subsequent improvements in computing and information technology as well as changes in the social landscape that have made open data sharing more viable.

Some lessons learned from previous data sharing projects (such as fMRIDC and FCP/INDI) include:

- Data sharing can and should emerge from a community agreement regarding its benefits.
- The metadata required for sharing should be tailored to the specific research goals, rather than aiming for a complete representation of all possible variables of potential interest.
- Data should strictly adhere to a common organizational scheme, so that researchers can reanalyze very large datasets in a straightforward manner using automated means.
- Data should be instantly accessible over the internet, with minimal restrictions on access (except where necessary, e.g., for reasons of subject confidentiality).

2.1. REQUIREMENTS FOR INCLUSION

One of the major goals of the OpenfMRI project is to enable whole-brain meta-analyses. For this reason, a dataset must include task-based fMRI data with coverage of the whole brain in order to be included in the OpenFMRI database; missing data at the edges of the volume can be accommodated, but datasets including coverage of only a portion of the brain will not be included (similar to the inclusion requirements for the BrainMap database). In addition, a high-resolution structural scan is necessary for each individual; additional structural scans, such as an in-plane structural image or diffusion-weighted images, are welcome if available but are not required. Finally, the metadata necessary to perform a standard statistical analysis (i.e., event onset times and durations for each experimental condition) are required. In cases where trial-by-trial behavioral data is necessary to perform the primary analysis of interest then those data are required; in other cases the submission of behavioral data is encouraged but not required.

2.2. DATA ORGANIZATION

Precise organization and naming is necessary to allow automated processing of large datasets. We have developed an initial scheme for data organization, based on the framework in use in a number of laboratories. The scheme is described in some detail by Poldrack et al. (2011b) and an overview of the current version

is shown in **Figure 1**. The datasets currently available on the site have been organized according to this scheme. As shown in **Figure 1**, each study is also associated with a set of key files that describe the conditions, tasks, contrasts, and MRI data acquisition details (including order of scans) that are necessary for proper analysis. This scheme will likely need to be modified to accommodate unexpected features of future data sets, such as different types of task designs. In addition, while currently organized using flat text files, we envision that in the future this scheme will be migrated toward a more formal metadata representation scheme such as XCEDE (Gadde et al., 2012).

2.3. REPRESENTING fMRI DESIGNS

Probably the single most difficult challenge of sharing task-based fMRI data is the representation of metadata describing the study. A common complaint about the process of sharing data via the fMRIDC was the requirement to formally specify a very extensive body of metadata. Whereas the fMRIDC process embodied a completist philosophy about metadata, we have chosen a more minimalist approach. In particular, the metadata that we absolutely require for submission are only those metadata that are necessary for specifying the analysis of the fMRI data using standard software packages. This includes minimal details regarding

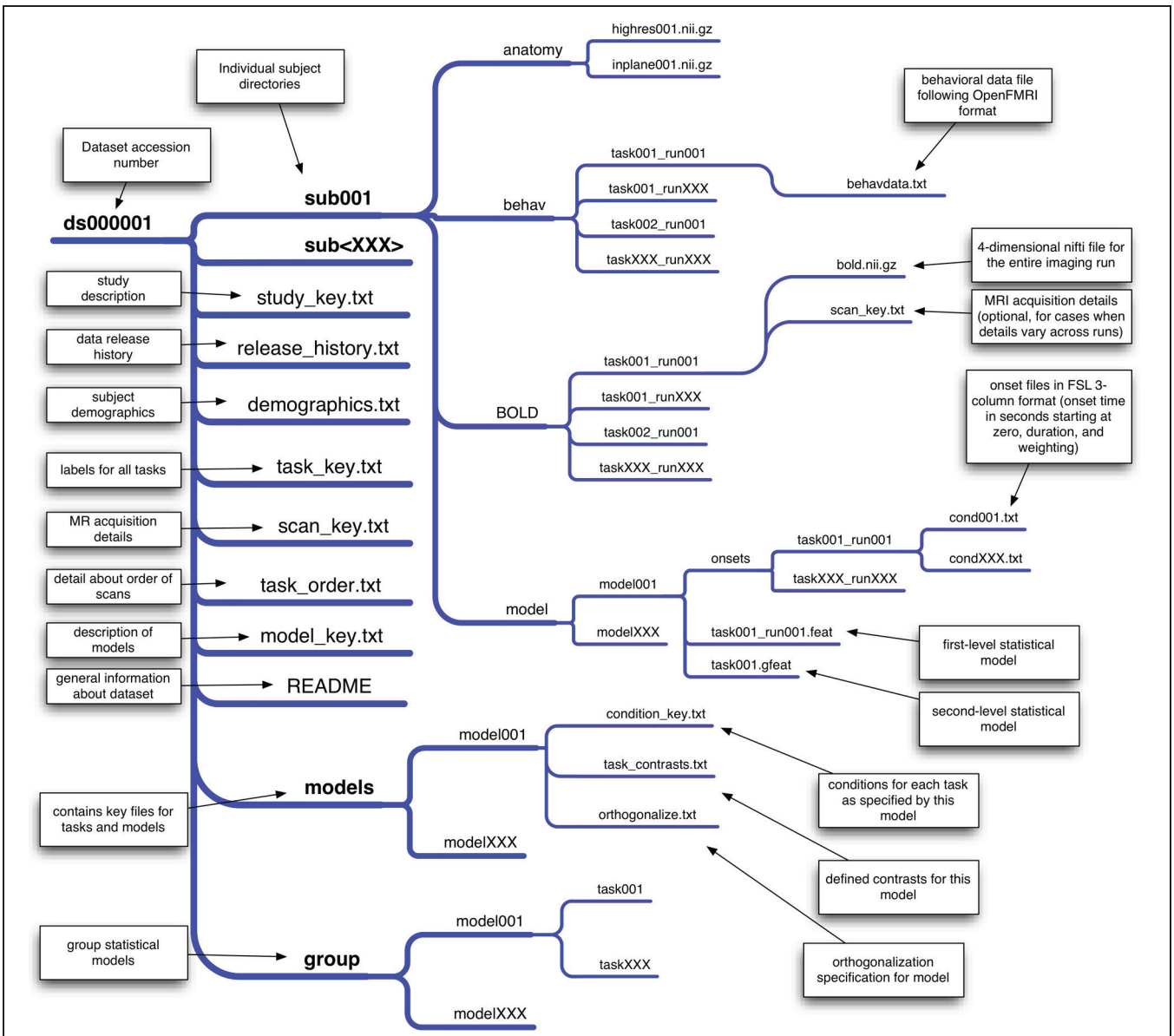


FIGURE 1 | An overview of the draft data organization scheme for the OpenFMRI project. A schematic of the directory tree for a dataset is presented, with each subdirectory shown on a separate branch. This structure allows specification of an arbitrary set of tasks, runs, and statistical models. The key files included in the base directory for the dataset specify features that are consistent across all of the data (such as demographics, task naming, and scan ordering), while key files in subdirectories specify details that may change across models or runs.

the MR acquisition (e.g., the repetition time), along with a specification of the onset times and event durations for each experimental condition (which may include behaviorally-defined conditions as well as experimenter-defined conditions). In particular, we will use the flexible 3-column onset file format developed within the FSL software package; because of the flexibility of this format (which includes onset times, lengths, and weightings for each event), it is also possible to specify many different complex designs, from simple blocked designs to complex parametric event-related designs. It is also relatively easy to transform design specifications from other software packages (e.g., SPM, AFNI) into this format. When stimuli are available, they will be included either in the behavioral data file described below (e.g., for single word stimuli) or within a separate directory (e.g., for image or sound files). Each dataset will also be accompanied by a textual description of the methods (usually the methods section from an associated paper or an equivalent description for unpublished data), so that additional details can be obtained from that description even if they are not represented in the dataset. In addition, whenever DICOM header information or other detailed MRI acquisition information (e.g., a dump of the scanner protocol) is available for a study, it will be included for each scan.

2.4. BEHAVIORAL DATA

Another challenging issue surrounds the representation of behavioral data, which are essential to the modeling of fMRI data for many studies (e.g., for modeling of accuracy or reaction times). Because there is no general framework for the representation of behavioral data, we have developed a simple protocol for behavioral data storage for the OpenFMRI project. This is a trial-based scheme in which any number of variables can be specified, including independent variables (such as condition names or stimulus identities) and dependent variables (such as response time or accuracy). An additional key file describes the meaning and possible values for each variable. If additional variables need to be represented in a way that is not trial-based (e.g., eye position measured at every timepoint), these data can be specified in additional files. In this way, we allow maximal flexibility with minimal need for reformatting (since the data for most studies will already be stored in a trial-based manner within a spreadsheet). As the project progresses, we plan to develop a more formal representation of the behavioral data (e.g., using XML).

2.5. PSYCHOLOGICAL CONSTRUCTS

A final challenge arises from the need to specify the psychological constructs that are meant to be indexed by each experimental comparison in a dataset. This is a much more difficult undertaking than describing the task and imaging metadata because of the lack of common agreement about what psychological constructs are measured by any particular comparison. In a separate project known as the Cognitive Atlas [http://www.cognitiveatlas.org; Poldrack et al. (2011a)], we have begun to develop an online knowledge base (or *ontology*) that aims to capture the structure of mental processes and their relation to specific tasks. The Cognitive Atlas currently provides the basis for annotation of datasets within the OpenfMRI database; tasks included in the OpenFMRI dataset are automatically linked to the Cognitive Atlas task database, and relations between these tasks and mental processes can then be specified by researchers in the community.

3. CONFIDENTIALITY

Confidentiality of research participants is of critical importance in data sharing (cf. Van Horn et al., 2001; Nooner et al., 2012). The upload policy for the project specifies that data should be anonymized before uploading by removing all of the 18 possible unique identifiers specified by HIPAA. The investigators submitting data are responsible for anonymization, but once data are uploaded a curator will doublecheck the data to ensure that no identifying information remains. Because high-resolution structural images may contain information about facial structures, all structural images will have facial features removed prior to sharing.

4. INTELLECTUAL PROPERTY AND CREDIT

Another common concern about data sharing for researchers relates to intellectual property. We believe that sharing of data with highly restrictive terms and conditions would defeat the purpose of an open data sharing repository, and we trust that the community will largely be responsible in their use of the data and attribution of its provenance. For this reason, data shared by the project will be released by default under the Public Domain Dedication and License developed by the Open Data Commons (ODC). This license states that users can download the data and use them for any purpose they wish, with no requirement for permission, citation, or coauthorship. We will encourage users to follow the ODC Attribution/Share-Alike Community Norms, which request that users give credit to the originator of the data and share any resulting products in a similar manner. We realize that some investigators may wish to share high-value datasets but may not be comfortable with public domain dedication; in this case we will consider more restrictive licensing on a case-by-case basis. In cases where investigators wish to stage a dataset for release on a specific date (e.g., to coincide with the publication of a paper), we will allow investigators to specify an embargo period for submitted datasets (generally not to exceed 6 months), which will provide sufficient curation time for the dataset to be ready for release on the intended date.

Individuals sharing data should reasonably expect to receive credit for having gone to the effort of data sharing. On the OpenFMRI web site, credit is given via a link to the publication on the associated data page as well as a list of investigators involved in collecting the data. In addition, the inclusion of the OpenFMRI database within the Neuroscience Information Framework (NIF: Gardner et al., 2008) allows links to the dataset to be added automatically to the PubMed listing for each associated paper, using the NIF Link-Out Broker (Marenco et al., 2008). With the advent of venues for data publication including *Nature Scientific Data* and *GigaScience*, it is also possible for contributors to publish a separate paper that describes the dataset (for a recent example, see Gorgolewski et al., 2013b).

5. PROCESSING STREAM

We have implemented an automated processing stream for the data in the OpenFMRI database; the processing steps are listed

in **Table 1**, and the code for these analyses is freely available via the OpenFMRI web site. Because of the use of a precise organizational scheme and metadata format, it is possible to completely automate every step of data processing, including the generation of FSL design files for each level of analysis. Eventually, the results from each intermediate processing step will be made available in the future through the OpenFMRI web site along with the raw data. Because of the computationally intensive nature of such processing on a large dataset, analysis is performed using the high-performance Lonestar cluster at the Texas Advanced Computing Center. All code used to implement this processing stream is available at http://github.com/poldrack/openfmri.

The specific processing stream was selected based on its current use in the first author's laboratory, but represents a fairly standard processing stream in the field. After conversion to NIfTI format and organization via the standard data scheme, the BOLD data are motion-corrected using MCFLIRT (FSL) and the brain is extracted using BET (FSL). The event onsets for each experimental condition are represented using the 3-column (onset time, length, and weighting) format from FSL. Using custom code, we automatically generate the FSL design files from these onset files, with extracted motion parameters and their temporal derivatives included as nuisance regressors. First-level statistical modeling is performed using FEAT (FSL) and contrasts are automatically generated for each experimental condition compared to baseline, in addition to any other potential contrasts of interest. For studies with multiple runs per task, second-level modeling is performed using a fixed-effects model. Third-level modeling is performed using FLAME (FSL), implementing a mixed-model that treats subjects as a random effect.

High-resolution anatomical images are first brain-extracted using FreeSurfer. The anatomical image is aligned to the MNI152 template using a combination of boundary-based registration and linear registration with FNIRT (FSL). The functional images are aligned to the high resolution image and the warps are combined to provide a transformation of the functional data into the MNI152 space, which is applied to the results of the statistical analysis at the higher levels. Cortical surface generation and automated anatomical parcellation are performed using FreeSurfer.

Quality control is performed using the fmriqa package (https://github.com/poldrack/fmriqa) for raw data, and the fsl-qa package (https://github.com/poldrack/fsl-qa) for analyzed data. QA results for the raw data are included in the base download. Reports are also generated that allow manual inspection of defacing, spatial registration of structural and functional images to standard space, and statistical analyses; these reports are examined and validated by the OpenFMRI staff before the data are made publicly available.

When data are uploaded to the OpenFMRI database, they are processed by the curators through the level of group analysis, in an attempt to replicate the results of the original analysis (e.g., in a published paper associated with the dataset). Given the multiplicity of different analysis streams and likelihood of different results between streams (Carp, 2012), it is expected that the results will sometimes fail to exactly match those of the original analysis. In such a case, we first contact the investigators to ensure that the task has been properly modeled in our analysis (e.g., that there are no mistakes in the event timing files). If the modeling is confirmed to be correct, then the authors will be given a chance to withdraw their submission, or to have the data shared despite this mismatch in results.

Table 1 | List of processing steps applied to data, and tools used for each operation.

Operation	Tool used
Motion correction	mcflirt (FSL)
Brain extraction (highres)	Freesurfer
Brain extraction (BOLD)	bet (FSL)
Quality assurance and generation of confound files	fmriqa (custom)
Creation of design files	custom code
First-level (within-run) statistical modeling	feat (FSL)
Second-level statistical modeling (for multi-run datasets)	feat (FSL)
Group statistical modeling	feat (FSL)
Cortical surface generation and parcellation (highres)	Freesurfer

5.1. DATA VERSIONING AND SOFTWARE UPDATES

The web page for each dataset currently contains versioning information that describes any changes in the dataset. In addition, a revision history file is included with each dataset download. While the raw data are largely independent of any processing software, the distribution of processed data is made challenging by the constant stream of software updates for packages such as FSL. Fortunately, the implementation of our processing stream within a high performance computing environment makes it relatively straightforward to reprocess the entire database within a relatively short time (generally within 1 day). For existing processed data, we will reprocess the data and release updated versions of the data for all major revisions of the FSL package, once they have been vetted and ensured to work properly with our processing stream. We do not expect substantial changes in results across major versions of the software, but if any such differences are noticed, we will first discuss with the software developers to ensure that they do not reflect software problems. If they are determined to be true methodological differences, then these differences will be described on the web site.

5.2. INFORMATICS PLATFORM

The OpenFMRI website storage and processing mechanisms have been chosen to provide an extensible software platform. Datasets are stored in an XNAT server (Marcus et al., 2007), and processing streams access the datasets through XNAT's built in REST API. In our initial model, the Lonestar cluster at TACC accesses data from XNAT, performs its processing operation, and then writes the processed data back into XNAT. Using XNAT's web services, we can expose that read/write API to other applications on a case by case basis. This will allow qualified users to apply their own analysis methods to the OpenFMRI database and then expose the results via the database. This platform model will give end users a variety of choices in how their data are processed, while providing automated documentation and quality control.

In addition to being hosted directly by the openfmri.org web site, the shared dataset is also available via the INCF Dataspace (http://www.incf.org/resources/data-space), which is a federated data sharing environment based on the iRODS data management system.

6. CURRENT STATUS AND FUTURE PLANS

The OpenFMRI database currently contains 18 full datasets from seven different laboratories, with a total of 347 unique scanning sessions, all of which are available for download directly from the web site. The database remains heavily skewed toward datasets from the Poldrack lab, but is increasingly diverse with the addition of new datasets. The site is currently accepting uploads, and has a number of datasets in the process of curation in addition to those currently available for download. As of March 2013, there have been 914 downloads of full datasets from the database, and four publications using data from the OpenfMRI database (Carp, 2012; Pedregosa et al., 2012; Varoquaux et al., 2012; Park et al., 2013).

Development, curation, and further population of the site are currently funded by a set of linked grants from the National Science Foundation. However, as Van Horn and Gazzaniga (2012) point out, it is essential to have a plan for longevity once the initial funding period ends. We are hopeful that national funding agencies will continue to view this project as worth supporting, but cannot rely on this. The Texas Advanced Computing Center has committed to long-term storage and accessibility of the data, but continued operation of the project beyond the funding window will require volunteer curators. Given the increased attention to data management and data sharing by federal funding agencies, it is possible that curation could also be supported by "data managers" funded by grants in participating labs. We will also explore other options for long-term funding such as development of a non-profit organization (similar to Wikipedia).

7. PRELIMINARY ANALYSES

To demonstrate the potential utility of mega-analysis on a large task-based fMRI dataset, below we present results from initial analyses of a subset of the current database (as of March 2013). The datasets, tasks, and contrasts included in this analysis are listed in **Table 2**. Three datasets from the current database were excluded from this analysis, due to small sample size (ds105) or exact replication of tasks and subjects from other datasets (ds006B and ds017B). Most of the datasets include multiple runs, for a total of 479 images from 337 unique subjects for run 1, and 429 images from 317 unique subjects for run 2.

Table 2 | List of datasets used in the preliminary analyses below.

Dataset #	Accession #	Task #	Task/contrast description	References
1	ds001	1	Balloon analog risk task: Parametric pump effect vs. control	Schonberg et al., 2012
2	ds002	1	Classification learning task: Task vs. baseline	Aron et al., 2006
3	ds002	2	Classification learning task: Feedback vs. baseline	Aron et al., 2006
4	ds002	3	Classification decision: Task vs. baseline	Aron et al., 2006
5	ds003	1	Rhyme judgment: Task vs. baseline	Xue and Poldrack, 2007
6	ds005	1	Mixed-gambles task: Parametric gain response	Tom et al., 2007
7	ds006A	1	Mirror reading task: Mirror-reversed vs. plain items	Jimura et al., in preparation
8	ds007	1	Stop signal task: Letter classification vs. baseline	Xue et al., 2008
9	ds007	2	Stop signal task: Letter naming vs. baseline	Xue et al., 2008
10	ds007	3	Stop signal task: Pseudoword naming vs. baseline	Xue et al., 2008
11	ds008	1	Stop signal task: Successful stop vs. baseline	Aron et al., 2007
12	ds008	2	Conditional stop signal task: Successful stop vs. baseline	Aron et al., 2007
13	ds011	1	Tone-counting task: Task vs. baseline	Foerde et al., 2006
14	ds011	2	Single-task classification learning: Task vs. baseline	Foerde et al., 2006
15	ds011	3	Dual-task classification learning: Task vs. baseline	Foerde et al., 2006
16	ds011	4	Classification decision: Task vs. baseline	Foerde et al., 2006
17	ds017A	2	Conditional stop signal task: Go-critical vs baseline	Rizk-Jackson et al., unpublished
18	ds051	1	Abstract-concrete task: novel vs. repeated words	Alvarez and Poldrack, unpublished
19	ds052	1	Classification learning task: Positive feedback vs. baseline	Poldrack et al., 2001
20	ds052	2	Classification reversal learning task: Negative feedback vs. baseline	Poldrack et al., 2001
21	ds101	1	Simon task: incorrect vs. correct	Kelley and Milham, unpublished
22	ds102	1	Flanker task: incongruent vs. congruent	Kelly et al., 2008
23	ds107	1	One-back task: words vs. consonants	Duncan et al., 2009
24	ds108	1	Emotion regulation task: Regulate-negative vs. Look-negative	Wager et al., 2008
25	ds109	1	False belief task: False belief story vs. false picture story	Moran et al., 2012
26	ds110	1	Incidental memory encoding task with cueing: Valid cue high confidence hits vs. misses	Uncapher et al., 2011

Accession and task numbers refer to the specific descriptors used in the database.

All analyses reported below were performed on spatially normalized Z statistic maps obtained for the contrast of interest using the processing stream described in **Table 1**. The data used to generate all for the foregoing analyses and figures are available from the OpenFMRI web site at http://openfmri.org/dataset/ds900001, and the code used to perform all analyses is available at http://github.com/poldrack/openfmri. Thus, anyone should be able to run these same analyses on their own system in order to reproduce the results reported here.

7.1. ICA ANALYSIS

Although the statistical maps obtained in the analyses described above include more than 200,000 voxels, a significant amount of information is carried in the coordinated activity of a much smaller number of large-scale neural systems, which can be identified using dimensionality reduction methods such as independent components analysis (ICA). To characterize the large-scale networks that emerged across the different tasks in the dataset, statistical (Z) images for each contrast/task/dataset were submitted to ICA using the FSL melodic tool (Beckmann and Smith, 2004) after spatial smoothing (6mm FWHM). Based on similar recent analyses (Smith et al., 2009; Congdon et al., 2010), we first specified 20 components in order to identify a small set of large-scale networks. The components identified in this analysis are shown in **Figure 2**. A number of these components reflected the basic sensorimotor aspects of the tasks, including visual regions (components 1, 3, and 19), auditory regions (component 10), and motor regions (components 15 and 20). Others reflected higher-order networks, including fronto-parietal (component 2) and cingulo-opercular (component 5) control networks identified by Dosenbach et al. (2010) and the left-hemisphere language network (component 6). In addition, there were components reflecting the "default-mode" network generally identified during the resting state (component 4) as well as one component reflecting coherent white matter signal (component 14). These results are highly consistent with the results of Smith et al. (2009), which were based on meta-analytic maps from the BrainMap database.

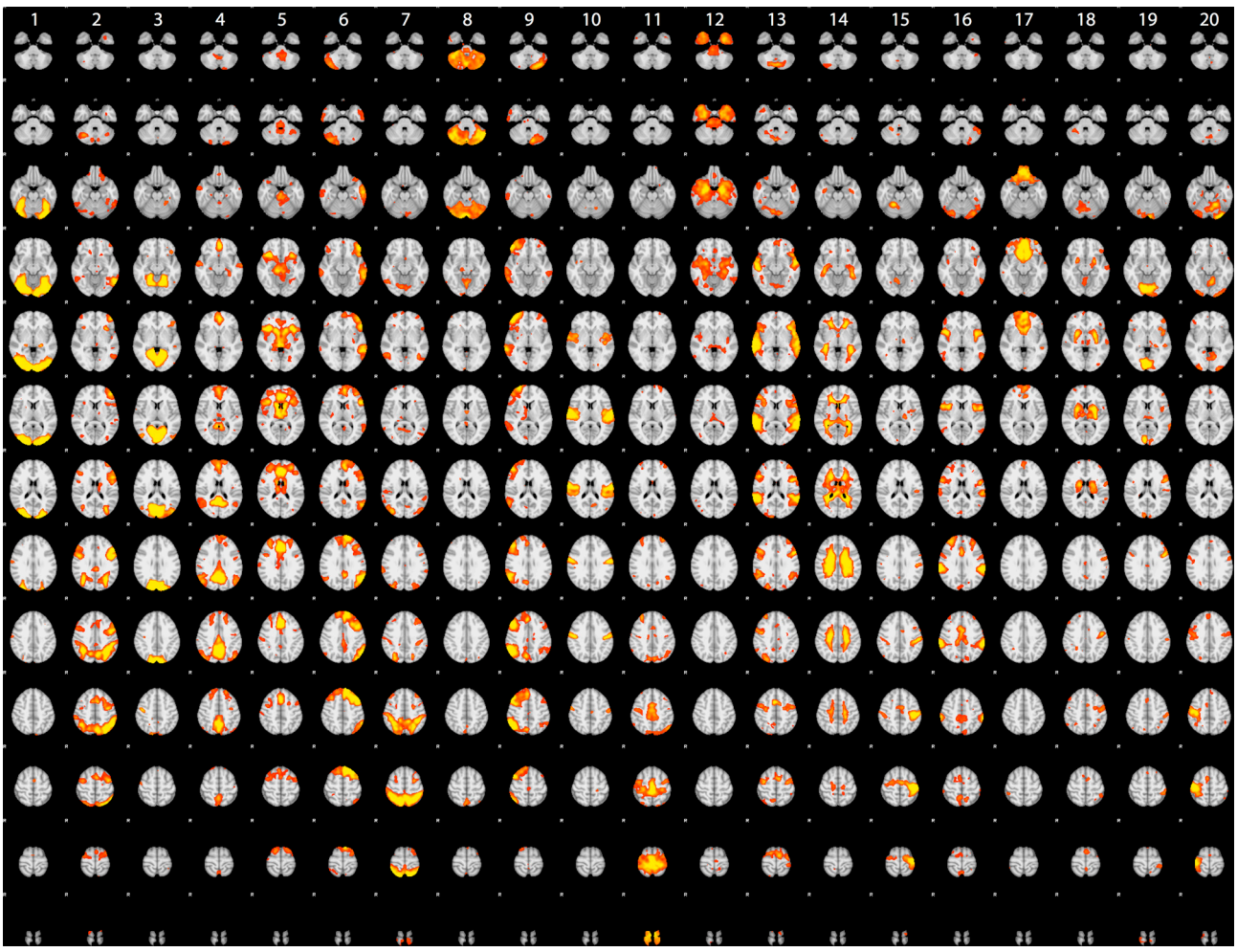


FIGURE 2 | Rendered maps of the voxels with significant loadings on the 20 ICA components identified statistical images for the datasets listed in Table 1. Each column displays the loading for a single component; voxels shown in red had positive loading for that component.

7.2. CLASSIFICATION ANALYSIS

Previous work (using some of the same data analyzed here) has shown that it is possible to classify which task a subject is performing using a classifier that was trained on other individuals performing a broad range of tasks (Poldrack et al., 2009). We performed a set of similar analyses in order to examine the replicability of those analyses in a dataset that overlapped partially with those used by Poldrack et al. (2009) but using different dimensionality reduction and classification methods. We trained a classifier to identify the task being performed by each subject out of 26 possible tasks. To reduce the dimensionality of the dataset, the whole-brain data from run 1 were regressed against the spatial ICA components obtained from the run 2 data (in order to maintain strict separation of training and test data). ICA was estimated using several different dimensionality levels, in order to examine the relation between classification accuracy and degree of dimensionality reduction; in each case, the loadings on each component (ranging from 2 to 200 components) were used as features in the classification. Twenty-six-way classification was performed using three methods: Linear support vector machine (SVM) (implemented in Liblinear: Fan et al., 2008), non-linear support vector machine (with radial basis kernel, implemented in LibSVM: Chang and Lin, 2011) and regularized logistic regression (LR) with an L2 penalty (implemented in the scikit-learn package: http://scikit-learn.sourceforge.net/). Classifier parameters (cost parameter for SVMs, gamma for non-linear SVM, and penalty parameter for LR) were optimized using the run 2 data, ensuring no crosstalk between parameter identification and classification testing. Classification accuracy was assessed on the run 1 data using leave-one-out cross validation, and an empirical null distribution was obtained by performing the classification 1000 times using randomly permuted labels.

Accuracy for the 26-way classification for each method across all dimensionality levels is shown in **Figure 3**. Accuracy rose incrementally as the number of components was increased from 2 to 100 and then remained relatively stable (around 50%) after 100. Accuracy was quite similar for the two linear classifiers, and only slightly higher for the non-linear SVM. All classification values were substantially greater than chance. Thus, highly significant classification was possible across subjects on a range of tasks, even after very substantial dimensionality reduction, consistent with the findings of Poldrack et al. (2009). It should be noted that because some subjects contributed data to multiple datasets in the classification analysis, not all data points were independent; this likely led to reduced classification accuracy due to confusions caused by subject-specific rather than task-specific patterns. Analysis of classifier confusion matrices showed that discriminability between tasks was compromised in some cases where the subjects overlapped, but in many cases also reflected overlap in task content across different datasets. A further analysis of the effects of non-independence is beyond the scope of the present paper but will be explored in future publications.

We also examined whole-brain classification of Z-statistic maps with the same dataset using a linear support vector machine (SVM) classifier with default cost value ($C = 1$) and eight-fold balanced cross-validation. Voxels with missing data for more than 3 subjects were excluded, leaving a total of 174,264 voxels.

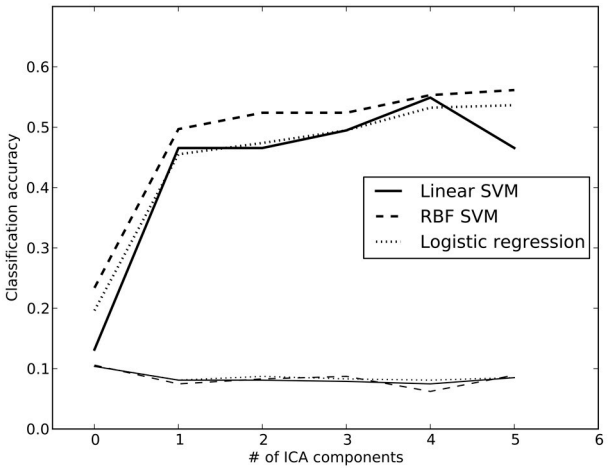


FIGURE 3 | Classification accuracy using reduced-dimension data, as a function of the number of ICA components in the dimensionality reduction step. Dimensionality reduction was first performed on an independent set of data (from run 2 for each subject), and the data from run 1 were then projected onto those components. Reported accuracy (thick lines) reflects average accuracy of task classification across all data points, from a total of 25 possible labels. The thin lines at the bottom reflect the empirically derived 95% cutoff for the null hypothesis of chance accuracy, obtained by performing the same classification 100 times with randomized labels, and taking the 95th largest value. RBF, radial basis function; SVM, support vector machine.

Classification accuracy for this analysis was moderate (mean = 48.8%; 95th percentile of empirical null distribution = 7.7%). The decreased accuracy compared to the previously reported task-classification results (Poldrack et al., 2009) likely reflects the fact that the present analysis included more fine-grained contrasts, as well as including a larger number of heavily overlapping tasks. It is interesting that classification performance was not appreciably greater for whole-brain analysis than for the model using ICA components, suggesting that most of the differentiation between tasks is being carried by differences across large-scale networks.

Finally, we examined whether it was possible to identify individual subjects based on their statistical maps. A one-vs.-all multi-class linear SVM was trained to classify each subject into a separate class using the default cost parameter ($C = 1.0$); for some subjects data were available for multiple tasks, whereas for others there was only a single training instance. To ensure that the classification was not driven by missing data, only voxels with non-zero values for all subjects were included in the analysis, leaving a total of 152,704 voxels. Generalization was tested on the data from run 2, which was available for 317 of the 337 subjects. Classification accuracy of 66.9% was achieved; with random subject relabeling, the mean classifier accuracy was 0.3% and the 95th percentile of the null distribution was 0.9%, showing that it was possible to identify individual subjects using their statistical maps from different scanning runs with highly significant accuracy. This finding is consistent with previous arguments regarding the importance of stable individual differences in neuroimaging data (e.g., Miller et al., 2009).

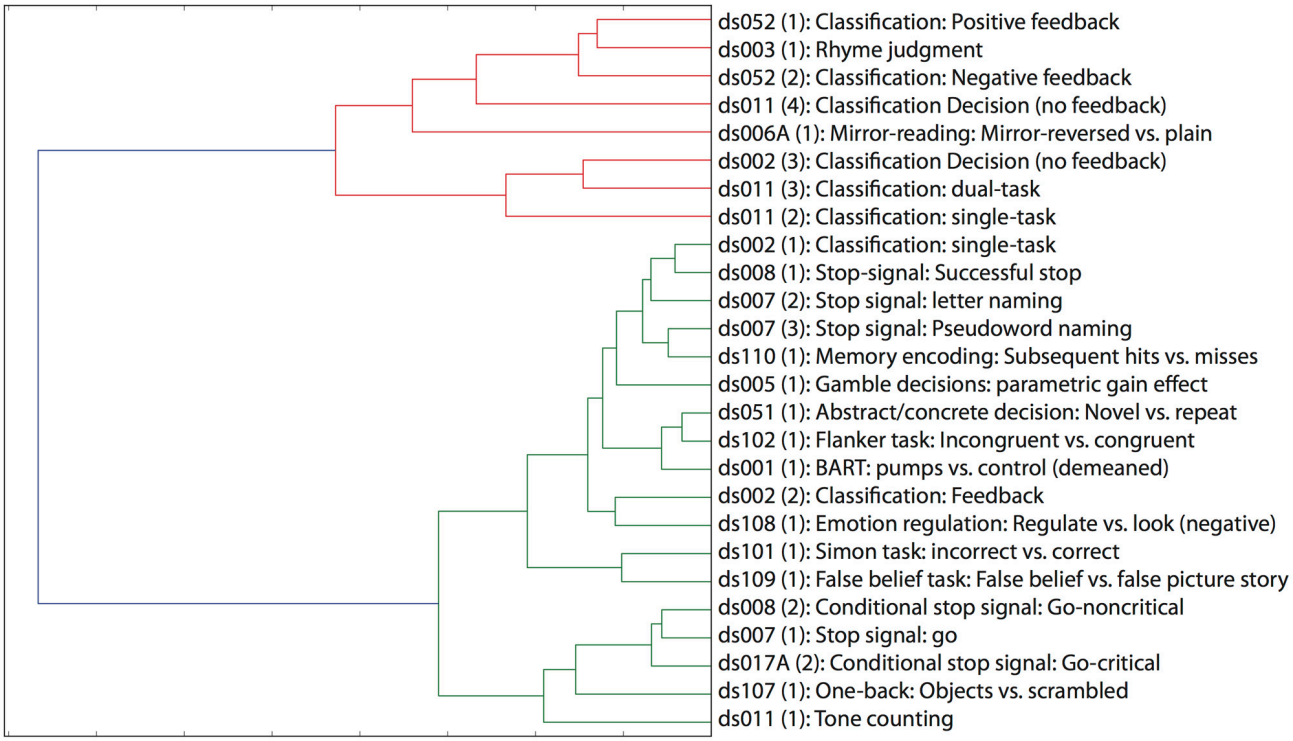


FIGURE 4 | Hierarchical clustering of statistical maps across tasks, after projection into the 20-dimensional ICA space depicted in Figure 2.

7.3. CLUSTER ANALYSIS

One of the great advantages of large datasets like those in the OpenFMRI database is the ability to examine the large-scale multidimensional neural space that characterizes different cognitive tasks. In order to examine this, we performed a hierarchical clustering analysis on whole-brain statistical maps after projection into the 20-dimension ICA space depicted in **Figure 2** and averaging across subjects within each task; clustering was performed using Ward's method with a Euclidean distance metric as implemented in scikit-learn. The resulting dendrogram is shown in **Figure 4**, and shows that there is a noisy but surprisingly consistent similarity in the neural activity patterns between tasks that engage common processes [as found previously by Poldrack et al. (2009)]. In several cases, maps from similar tasks within the same dataset were clustered together (e.g., the pseudoword naming and letter naming conditions from ds007, both of which come from the same subjects), whereas in other cases, the same task from different datasets were clustered together; particularly striking is the fact that the classification decision and classification learning datasets from multiple studies are clustered together, even though they were collected on very different versions of the tasks and different scanner platforms. These results highlight the degree to which different tasks exist within a larger similarity space of neural activity, which could potentially provide insights into the latent neurocognitive bases of mental functions.

8. CONCLUSION

Data sharing has revolutionized other areas of biomedical science, and we believe that it has the potential to do the same for cognitive neuroscience. The OpenfMRI data sharing project has developed the infrastructure for the sharing of task-based fMRI datasets, and has begun making datasets openly available online. We are optimistic that this project will help encourage widespread voluntary data sharing by providing a powerful resource that makes sharing as straightforward as possible. Preliminary analyses of the database have confirmed the ability to classify mental states across individuals, as well as demonstrating the novel ability to classify the identity of individual subjects from their fMRI patterns. In addition, multivariate analyses provide new glimpses into the multidimensional relations between mental function and brain function. We foresee many additional insights arising from these data as the database grows and other novel analysis methods are applied to the data.

ACKNOWLEDGMENTS

This work was supported by NSF Grants OCI-1131441 (Russell A. Poldrack), OCI-1130086 (Anthony D. Wagner), OCI-1131291 (Deanna M. Barch), OCI-1131338 (Jason P. Mitchell), and OCI-1131801 (Tor D. Wager). Thanks to Tom Schonberg, Chris Gorgolewski, and Jean-Baptise Poline, Laurens van der Maaten, and Jack Van Horn for helpful comments and Natalie Picchetti for assistance with data analysis.

REFERENCES

Aron, A. R., Behrens, T. E., Smith, S., Frank, M. J., and Poldrack, R. A. (2007). Triangulating a cognitive control network using diffusion-weighted magnetic resonance imaging (MRI) and functional MRI. *J. Neurosci.* 27, 3743–3752. doi: 10.1523/JNEUROSCI.0519-07.2007

Aron, A. R., Gluck, M. A., and Poldrack, R. A. (2006). Long-term test-retest reliability of functional MRI in a classification learning task. *Neuroimage* 29, 1000–1006. doi: 10.1016/j.neuroimage.2005.08.010

Beckmann, C. F., and Smith, S. M. (2004). Probabilistic independent component analysis for functional magnetic resonance imaging. *IEEE Trans. Med. Imag.* 23, 137–152. doi: 10.1109/TMI.2003.822821

Biswal, B. B., Mennes, M., Zuo, X.-N., Gohel, S., Kelly, C., Smith, S. M., et al. (2010). Toward discovery science of human brain function. *Proc. Natl. Acad. Sci. U.S.A.* 107, 4734–4739. doi: 10.1073/pnas.0911855107

Brakewood, B., and Poldrack, R. A. (2013). The ethics of secondary data analysis: considering the application of belmont principles to the sharing of neuroimaging data. *Neuroimage*. doi: 10.1016/j.neuroimage.2013.02.040. [Epub ahead of print].

Carp, J. (2012). On the plurality of (methodological) worlds: estimating the analytic flexibility of fMRI experiments. *Front Neurosci* 6:149. doi: 10.3389/fnins.2012.00149

Chang, C., and Lin, C. (2011). Libsvm : a library for support vector machines. *ACM Trans. Intell. Syst. Technol.* 2, 1–27. doi: 10.1145/1961189.1961199

Congdon, E., Mumford, J. A., Cohen, J. R., Galvan, A., Aron, A. R., Xue, G., et al. (2010). Engagement of large-scale networks is related to individual differences in inhibitory control. *Neuroimage* 53, 653–663. doi: 10.1016/j.neuroimage.2010.06.062

Dosenbach, N. U. F., Nardos, B., Cohen, A. L., Fair, D. A., Power, J. D., Church, J. A., et al. (2010). Prediction of individual brain maturity using fMRI. *Science* 329, 1358–1361. doi: 10.1126/science.1194144

Duncan, K. J., Pattamadilok, C., Knierim, I., and Devlin, J. T. (2009). Consistency and variability in functional localisers. *Neuroimage* 46, 1018–1026. doi: 10.1016/j.neuroimage.2009.03.014

Editorial, N. N. (2000). A debate over fMRI data sharing. *Nat. Neurosci.* 3, 845–846. doi: 10.1038/78728

Fan, R. E., Chang, K., Hsieh, C., Wang, X., and Lin., C. (2008). Liblinear: a library for large linear classification. *J. Mac. Learn. Res.* 9, 1871–1874. doi: 10.1145/1390681.1442794

Foerde, K., Knowlton, B. J., and Poldrack, R. A. (2006). Modulation of competing memory systems by distraction. *Proc. Natl. Acad. Sci. U.S.A.* 103, 11778–11783. doi: 10.1073/pnas.0602659103

Gadde, S., Aucoin, N., Grethe, J. S., Keator, D. B., Marcus, D. S., Pieper, S., et al. (2012). Xcede: an extensible schema for biomedical data. *Neuroinformatics* 10, 19–32. doi: 10.1007/s12021-011-9119-9

Gardner, D., Akil, H., Ascoli, G. A., Bowden, D. M., Bug, W., Donohue, D. E., et al. (2008). The neuroscience information framework: a data and knowledge environment for neuroscience. *Neuroinformatics* 6, 149–160. doi: 10.1007/s12021-008-9024-z

Gorgolewski, K. J., Margulies, D. S., and Milham, M. P. (2013a). Making data sharing count: a publication-based solution. *Front. Neurosci.* 7:9. doi: 10.3389/fnins.2013.00009

Gorgolewski, K. J., Storkey, A., Bastin, M. E., Whittle, I. R., Wardlaw, J. M., and Pernet, C. R. (2013b). A test-retest fmri dataset for motor, language and spatial attention functions. *Gigascience* 2:6. doi: 10.1186/2047-217X-2-6

Greicius, M. D., Srivastava, G., Reiss, A. L., and Menon, V. (2004). Default-mode network activity distinguishes alzheimer's disease from healthy aging: evidence from functional MRI. *Proc. Natl. Acad. Sci. U.S.A.* 101, 4637–4642. doi: 10.1073/pnas.0308627101

Kelly, A. M. C., Uddin, L. Q., Biswal, B. B., Castellanos, F. X., and Milham, M. P. (2008). Competition between functional brain networks mediates behavioral variability. *Neuroimage* 39, 527–537. doi: 10.1016/j.neuroimage.2007.08.008

Laird, A. R., Lancaster, J. L., and Fox, P. T. (2005). Brainmap: the social evolution of a human brain mapping database. *Neuroinformatics* 3, 65–78. doi: 10.1385/NI:3:1:065

Lloyd, D. (2002). Functional MRI and the study of human consciousness. *J. Cogn. Neurosci.* 14, 818–831. doi: 10.1162/089892902760191027

Marcus, D. S., Olsen, T. R., Ramaratnam, M., and Buckner, R. L. (2007). The extensible neuroimaging archive toolkit: an informatics platform for managing, exploring, and sharing neuroimaging data. *Neuroinformatics* 5, 11–34. doi: 10.1385/NI:5:1:11

Marenco, L., Ascoli, G. A., Martone, M. E., Shepherd, G. M., and Miller, P. L. (2008). The nif linkout broker: a web resource to facilitate federated data integration using ncbi identifiers. *Neuroinformatics* 6, 219–227. doi: 10.1007/s12021-008-9025-y

Mechelli, A., Price, C. J., Noppeney, U., and Friston, K. J. (2003). A dynamic causal modeling study on category effects: bottom-up or top-down mediation? *J. Cogn. Neurosci.* 15, 925–934. doi: 10.1162/089892903770007317

Mennes, M., Biswal, B. B., Castellanos, F. X., and Milham, M. P. (2012). Making data sharing work: the fcp/indi experience. *Neuroimage*. doi: 10.1016/j.neuroimage.2012.10.064. [Epub ahead of print].

Miller, M. B., Donovan, C.-L., Van Horn, J. D., German, E., Sokol-Hessner, P., and Wolford, G. L. (2009). Unique and persistent individual patterns of brain activity across different memory retrieval tasks. *Neuroimage* 48, 625–635. doi: 10.1016/j.neuroimage.2009.06.033

Moran, J. M., Jolly, E., and Mitchell, J. P. (2012). Social-cognitive deficits in normal aging. *J. Neurosci.* 32, 5553–5561. doi: 10.1523/JNEUROSCI.5511-11.2012

Nooner, K. B., Colcombe, S. J., Tobe, R. H., Mennes, M., Benedict, M. M., Moreno, A. L., et al. (2012). The nki-rockland sample: a model for accelerating the pace of discovery science in psychiatry. *Front. Neurosci.* 6:152. doi: 10.3389/fnins.2012.00152

Park, M., Koyejo, O., Ghosh, J., Poldrack, R., and Pillow, J. W., (2013). "Bayesian structure learning for functional neuroimaging," in *Sixteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, (Scottsdale, AZ).

Pedregosa, F., Gramfort, A., Varoquaux, G., Cauvet, E., Pallier, C., and Thirion, B. (2012). "Learning to rank from medical imaging data," in *3rd International Workshop on Machine Learning in Medical Imaging*, Nice, France, INRIA.

Poldrack, R. A. (2006). Can cognitive processes be inferred from neuroimaging data? *Trends Cogn. Sci.* 10, 59–63. doi: 10.1016/j.tics.2005.12.004

Poldrack, R. A., Clark, J., Paré-Blagoev, E. J., Shohamy, D., Creso Moyano, J., Myers, C., and Gluck, M. A. (2001). Interactive memory systems in the human brain. *Nature* 414, 546–550. doi: 10.1038/35107080

Poldrack, R. A., Fletcher, P. C., Henson, R. N., Worsley, K. J., Brett, M., and Nichols, T. E. (2008). Guidelines for reporting an fMRI study. *Neuroimage* 40, 409–414. doi: 10.1016/j.neuroimage.2007.11.048

Poldrack, R. A., Halchenko, Y. O., and Hanson, S. J. (2009). Decoding the large-scale structure of brain function by classifying mental states across individuals. *Psychol. Sci.* 20, 1364–1372. doi: 10.1111/j.1467-9280.2009.02460.x

Poldrack, R. A., Kittur, A., Kalar, D., Miller, E., Seppa, C., Gil, Y., et al. (2011a). The cognitive atlas: towards a knowledge foundation for cognitive neuroscience. *Front. Neuroinform.* 5:17. doi: 10.3389/fninf.2011.00017

Poldrack, R. A., Mumford, J. A., and Nichols, T. E. (2011b). *Handbook of Functional MRI Data Analysis*. New York, NY: Cambridge University Press. doi: 10.1017/CBO9780511895029

Poline, J.-B., Breeze, J. L., Ghosh, S., Gorgolewski, K., Halchenko, Y. O., Hanke, M., et al. (2012). Data sharing in neuroimaging research. *Front. Neuroinform.* 6:9. doi: 10.3389/fninf.2012.00009

Salimi-Khorshidi, G., Smith, S. M., Keltner, J. R., Wager, T. D., and Nichols, T. E. (2009). Meta-analysis of neuroimaging data: a comparison of image-based and coordinate-based pooling of studies. *Neuroimage* 45, 810–823. doi: 10.1016/j.neuroimage.2008.12.039

Schonberg, T., Fox, C. R., Mumford, J. A., Congdon, E., Trepel, C., and Poldrack, R. A. (2012). Decreasing ventromedial prefrontal cortex activity during sequential risk-taking: an fMRI investigation of the balloon analog risk task. *Front. Neurosci.* 6:80. doi: 10.3389/fnins.2012.00080

Smith, S. M., Fox, P. T., Miller, K. L., Glahn, D. C., Fox, P. M., Mackay, C. E., et al. (2009). Correspondence of the brain's functional architecture during activation and rest. *Proc. Natl. Acad. Sci. U.S.A.* 106, 13040–13045. doi: 10.1073/pnas.0905267106

Tom, S. M., Fox, C. R., Trepel, C., and Poldrack, R. A. (2007). The neural basis of loss aversion in decision-making under risk. *Science* 315, 515–518. doi: 10.1126/science.1134239

Tomasi, D., and Volkow, N. D. (2010). Functional connectivity density mapping. *Proc. Natl. Acad. Sci. U.S.A.* 107, 9885–9890. doi: 10.1073/pnas.1001414107

Turner, J. A., and Laird, A. R. (2012). The cognitive paradigm ontology: design and application. *Neuroinformatics* 10, 57–66. doi: 10.1007/s12021-011-9126-x

Uncapher, M. R., Hutchinson, J. B., and Wagner, A. D. (2011). Dissociable effects of top-down and bottom-up attention during episodic encoding. *J. Neurosci.* 31, 12613–12628. doi: 10.1523/JNEUROSCI.0152-11.2011

Van Horn, J. D., and Gazzaniga, M. S. (2002). Opinion: databasing fMRI studies towards a 'discovery science' of brain function. *Nat. Rev. Neurosci.* 3, 314–318. doi: 10.1038/nrn788

Van Horn, J. D., and Gazzaniga, M. S. (2012). Why share data? lessons learned from the fmridc. *Neuroimage*. doi: 10.1016/j.neuroimage.2012.11.010. [Epub ahead of print].

Van Horn, J. D., Grethe, J. S., Kostelec, P., Woodward, J. B., Aslam, J. A., Rus, D., et al. (2001). The functional magnetic resonance imaging data center (fmridc): the challenges and rewards of large-scale databasing of neuroimaging studies. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 356, 1323–1339. doi: 10.1098/rstb.2001.0916

Van Horn, J. D., and Ishai, A. (2007). Mapping the human brain: new insights from fmri data sharing. *Neuroinformatics* 5, 146–153. doi: 10.1007/s12021-007-0011-6

Varoquaux, G., Gramfort, A., and Thirion, B. (2012). "Small-sample brain mapping: sparse recovery on spatially correlated designs with randomization and clustering," in *International Conference on Machine Learning*, eds L. John and P. Joelle (Edimbourg: Andrew McCallum).

Wager, T. D., Davidson, M. L., Hughes, B. L., Lindquist, M. A., and Ochsner, K. N. (2008). Prefrontal-subcortical pathways mediating successful emotion regulation. *Neuron* 59, 1037–1050. doi: 10.1016/j.neuron.2008.09.006

Wicherts, J. M., Bakker, M., and Molenaar, D. (2011). Willingness to share research data is related to the strength of the evidence and the quality of reporting of statistical results. *PLoS ONE* 6:e26828. doi: 10.1371/journal.pone.0026828

Xue, G., Aron, A. R., and Poldrack, R. A. (2008). Common neural substrates for inhibition of spoken and manual responses. *Cereb. Cortex* 18, 1923–1932. doi: 10.1093/cercor/bhm220

Xue, G., and Poldrack, R. A. (2007). The neural substrates of visual perceptual learning of words: implications for the visual word form area hypothesis. *J. Cogn. Neurosci.* 19, 1643–1655. doi: 10.1162/jocn.2007.19.10.1643

Yarkoni, T., Poldrack, R. A., Nichols, T. E., Van Essen, D. C., and Wager, T. D. (2011). Large-scale automated synthesis of human functional neuroimaging data. *Nat. Methods* 8, 665–670. doi: 10.1038/nmeth.1635

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 14 January 2013; accepted: 03 June 2013; published online: 08 July 2013.

Citation: Poldrack RA, Barch DM, Mitchell JP, Wager TD, Wagner AD, Devlin JT, Cumba C, Koyejo O and Milham MP (2013) Toward open sharing of task-based fMRI data: the OpenfMRI project. Front. Neuroinform. 7:12. doi: 10.3389/fninf.2013.00012

frontiers in
NEUROINFORMATICS

METHODS ARTICLE
published: 22 August 2011
doi: 10.3389/fninf.2011.00013

Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in Python

Krzysztof Gorgolewski[1]*, Christopher D. Burns[2], Cindee Madison[2], Dav Clark[3], Yaroslav O. Halchenko[4], Michael L. Waskom[5,6], Satrajit S. Ghosh[7]

[1] *Neuroinformatics and Computational Neuroscience Doctoral Training Centre, School of Informatics, University of Edinburgh, Edinburgh, UK*
[2] *Helen Wills Neuroscience Institute, University of California, Berkeley, CA, USA*
[3] *Department of Psychology, University of California, Berkeley, CA, USA*
[4] *Department of Psychological and Brain Sciences, Dartmouth College, Hanover, NH, USA*
[5] *Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA, USA*
[6] *McGovern Institute for Brain Research, Massachusetts Institute of Technology, Cambridge, MA, USA*
[7] *Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA, USA*

Edited by:
Andrew P. Davison, CNRS, France

Reviewed by:
Gael Varoquaux, INSERM, France
Ivo Dinov, University of California, USA

****Correspondence:***
Krzysztof Gorgolewski, School of Informatics, University of Edinburgh, Informatics Forum, 10 Crichton Street, Edinburgh EH8 9AB, UK.
e-mail: krzysztof.gorgolewski@gmail.com

Current neuroimaging software offer users an incredible opportunity to analyze their data in different ways, with different underlying assumptions. Several sophisticated software packages (e.g., AFNI, BrainVoyager, FSL, FreeSurfer, Nipy, R, SPM) are used to process and analyze large and often diverse (highly multi-dimensional) data. However, this heterogeneous collection of specialized applications creates several issues that hinder replicable, efficient, and optimal use of neuroimaging analysis approaches: (1) No uniform access to neuroimaging analysis software and usage information; (2) No framework for comparative algorithm development and dissemination; (3) Personnel turnover in laboratories often limits methodological continuity and training new personnel takes time; (4) Neuroimaging software packages do not address computational efficiency; and (5) Methods sections in journal articles are inadequate for reproducing results. To address these issues, we present Nipype (Neuroimaging in Python: Pipelines and Interfaces; http://nipy.org/nipype), an open-source, community-developed, software package, and scriptable library. Nipype solves the issues by providing Interfaces to existing neuroimaging software with uniform usage semantics and by facilitating interaction between these packages using Workflows. Nipype provides an environment that encourages interactive exploration of algorithms, eases the design of Workflows within and between packages, allows rapid comparative development of algorithms and reduces the learning curve necessary to use different packages. Nipype supports both local and remote execution on multi-core machines and clusters, without additional scripting. Nipype is Berkeley Software Distribution licensed, allowing anyone unrestricted usage. An open, community-driven development philosophy allows the software to quickly adapt and address the varied needs of the evolving neuroimaging community, especially in the context of increasing demand for reproducible research.

Keywords: neuroimaging, data processing, workflow, pipeline, reproducible research, Python

INTRODUCTION

Over the past 20 years, advances in non-invasive *in vivo* neuroimaging have resulted in an explosion of studies investigating human cognition in health and disease. Current imaging studies acquire multi-modal image data (e.g., structural, diffusion, functional) and combine these with non-imaging behavioral data, patient and/or treatment history, and demographic and genetic information. Several sophisticated software packages (e.g., AFNI, BrainVoyager, FSL, FreeSurfer, Nipy, R, SPM) are used to process and analyze such extensive data. In a typical analysis, algorithms from these packages, each with its own set of parameters, process the raw data. However, data collected for a single study can be diverse (highly multi-dimensional) and large, and algorithms suited for one dataset may not be optimal for another. This complicates analysis methods and makes data exploration and inference challenging, and comparative analysis of new algorithms difficult.

CURRENT PROBLEMS

Here we outline issues that hinder replicable, efficient, and optimal use of neuroimaging analysis approaches.

No uniform access to neuroimaging analysis software and usage information

For current multi-modal datasets, researchers typically resort to using different software packages for different components of the analysis. However, these different software packages are accessed, and interfaced with, in different ways, such as: shell scripting (FSL, AFNI, Camino), MATLAB (SPM), and Python (Nipy). This has resulted in a heterogeneous set of software with no uniform way to use these tools or execute them. With the primary focus on algorithmic improvement, academic software development often lacks a rigorous software engineering framework that involves extensive testing and documentation and ensures compatibility with other

tools. This often necessitates extensive interactions with the authors of the software to understand their parameters, their quirks, and their usage.

No framework for comparative algorithm development and dissemination

Except for some large software development efforts (e.g., SPM, FSL, FreeSurfer), most algorithm development happens in-house and stays within the walls of a lab, without extensive exposure or testing. Furthermore, testing comparative efficacy of algorithms often requires significant effort (Klein et al., 2010). In general, developers create software for a single package (e.g., VBM8 for SPM), create a standalone cross-platform tool (e.g., Mricron), or simply do not distribute the software or code (e.g., normalization software used for registering architectonic atlases to MNI single subject template – Hömke, 2006).

Personnel turnover in laboratories often limits methodological continuity and training new personnel takes time

Most cognitive neuroscience laboratories aim to understand some aspect of cognition. Although a majority of such laboratories gather and analyze neuroimaging data, very few of them have the personnel with the technical expertise to understand methodological development and modify laboratory procedures to adopt new tools. Lab personnel with no prior imaging experience often learn by following online tutorials, taking organized courses or, as is most often the case, by learning from existing members of the lab. While this provides some amount of continuity, understanding different aspects of neuroimaging has a steep learning curve, and steeper when one takes into account the time and resources needed to learn the different package interfaces and algorithms.

Neuroimaging software packages do not address computational efficiency

The primary focus of neuroimaging analysis algorithms is to solve problems (e.g., registration, statistical estimation, tractography). While some developers focus on algorithmic or numerical efficiency, most developers do not focus on efficiency in the context of running multiple algorithms on multiple subjects, a common scenario in neuroimaging analysis. Creating an analysis workflow for a particular study is an iterative process dependent on the quality of the data and participant population (e.g., neurotypical, presurgical, etc.). Researchers usually experiment with different methods and their parameters to create a workflow suitable for their application, but no suitable framework currently exists to make this process efficient. Furthermore, very few of the available neuroimaging tools take advantage of the growing number of parallel hardware configurations (multi-core, clusters, clouds, and supercomputers).

Method sections of journal articles are often inadequate for reproducing results

Several journals (e.g., PNAS, Science, PLoS) require mandatory submission of data and scripts necessary to reproduce results of a study. However, most current method sections do not have sufficient details to enable a researcher knowledgeable in the domain to reproduce the analysis process. Furthermore, as discussed above, typical neuroimaging analyses integrate several tools and current analysis software do not make it easy to reproduce all the analysis steps in the proper order. This leaves a significant burden on the user to satisfy these journal requirements as well as ensure that analysis details are preserved with the intent to reproduce.

CURRENT SOLUTIONS

There have been several attempts to address these issues by creating pipeline systems (for comparison see **Table 1**). Taverna (Oinn et al., 2006), VisTrails (Callahan et al., 2006) are general pipelining systems with excellent support for web-services, but they do not address problems specific to neuroimaging. BrainVisa (Cointepas et al., 2001), MIPAV (McAuliffe et al., 2001), SPM include their own batch processing tools, but do not allow mixing components from other packages. Fiswidgets (Fissell et al., 2003), a promising initial approach, appears to have not been developed and does not support state of the art methods. A much more extensive and feature rich solution is the LONI Pipeline (Rex et al., 2003; Dinov et al., 2009, 2010). It provides an easy to use graphical interface for choosing processing steps or nodes from a predefined library and defining their dependencies and parameters. Thanks to an advanced client–server architecture, it also has extensive support for parallel execution on an appropriately configured cluster (including data transfer, pausing execution, and combining local and remote software). Additionally, the LONI Pipeline saves information about executed steps (such as software origin, version, and architecture) providing provenance information (Mackenzie-Graham et al., 2008).

However, the LONI Pipeline does not come without limitations. Processing nodes are defined using eXtensible Markup Language (XML). This "one size fits all" method makes it easy to add new nodes as long as they are well-behaved command lines. However, many software packages do not meet this criterion. For example, SPM, written in MATLAB, does not provide a command line interface. Furthermore, for several command line programs, arguments are not easy to describe in the LONI XML schema (e.g., ANTS – Avants and Gee, 2004). Although it provides a helpful graphical interface, the LONI Pipeline environment does not provide an easy option to script a workflow or for rapidly exploring parametric variations within a workflow (e.g., VisTrails). Finally, due to restrictive licensing, it is not straightforward to modify and redistribute the modifications.

To address issues with existing workflow systems and the ones described earlier, we present Nipype (Neuroimaging in Python: Pipelines and Interfaces), an open-source, community-developed, Python-based software package that easily interfaces with existing software for efficient analysis of neuroimaging data and rapid comparative development of algorithms. Nipype uses a flexible, efficient, and general purpose programming language – Python – as its foundation. Processing modules and their inputs and outputs are described in an object-oriented manner providing the flexibility to interface with any type of software (not just well-behaved command lines). The workflow execution engine has a plug-in architecture and supports both local execution on multi-core machines and remote execution on clusters. Nipype is distributed with a Berkeley Software Distribution (BSD) license allowing anyone to make changes and redistribute it. Development is done openly with collaborators from many different labs, allowing adaptation to the varied needs of the neuroimaging community.

Table 1 | Feature comparison of selected pipeline frameworks.

	Local multi-processing[1]	Grid engine	Scripting support	XNAT	Web-services[2]	Platforms	Graphical user interface	Designed for neuroimaging
Taverna	Yes	PBS	Java, R	Yes	Yes	Mac, Unix, Windows	Yes	No
VisTrails	Yes	n/a	Python	Yes	Yes	Mac, Unix, Windows	Yes	No
Fiswidgets	No	n/a	Java	No	No	Mac, Unix, Windows	Yes	Yes
LONI	No	DRMAA	No	Yes	No	Mac, Unix, Windows	Yes	Yes
Nipype	Yes	SGE, PBS, IPython	Python	Yes	No	Mac, Unix	No	Yes

BrainVisa, MIPAV, and SPM were not included due to their inability to combine software from different packages.
[1]Without additional dependencies.
[2]Support for executing processing steps defined as web-services.

IMPLEMENTATION DETAILS

Nipype consists of three components (see **Figure 1**): (1) *interfaces* to external tools that provide a unified way for setting inputs, executing, and retrieving outputs; (2) *a workflow engine* that allows creating analysis pipelines by connecting inputs and outputs of interfaces as a directed acyclic graph (DAG); and (3) *plug-ins* that execute workflows either locally or in a distributed processing environment (e.g., Torque[1], SGE/OGE). In the following sections, we describe key architectural components and features of this software.

INTERFACES

Interfaces form the core of Nipype. The goal of Interfaces[2] is to provide a uniform mechanism for accessing analysis tools from neuroimaging software packages (e.g., FreeSurfer, FSL, SPM). Interfaces can be used directly as a Python object, incorporated into custom Python scripts or used interactively in a Python console. For example, there is a Realign Interface that exposes the SPM realignment routine, while the MCFLIRT Interface exposes the FSL realignment routine. In addition, one can also implement an algorithm in Python within Nipype and expose it as an Interface. Interfaces are flexible and can accommodate the heterogeneous software that needs to be supported, while providing unified and uniform access to these tools for the user. Since, there is no need for the underlying software to be changed (recompiled or adjusted to conform to a certain standard), developers can continue to create software using the computer language of their choice.

An Interface definition consists of: (a) input parameters, their types (e.g., file, floating point value, list of integers, etc…) and dependencies (e.g., does input "a" require input "b"); (b) outputs and their types, (c) how to execute the underlying software (e.g., run a MATLAB script, or call a command line program); and (d) a mapping which defines the outputs that are produced given a particular set of inputs. Using an object-oriented approach, we minimize redundancy in interface definition by creating a hierarchy of base Interface classes (see **Figure 2**) to encapsulate common functionality (e.g., Interfaces that call command line programs are derived from the CommandLine class, which provides methods to translate Interface inputs into command line parameters and for calling the command). Source code of an example Interface is shown in **Listing 1**.

We use Enthought Traits[3] to create a formal definition for Interface inputs and outputs, to define input constraints (e.g., type, dependency, whether mandatory) and to provide validation (e.g., file existence). This allows malformed or underspecified inputs to be detected prior to executing the underlying program. The input definition also allows specifying relations between inputs. Often, some input options should not be set together (mutual exclusion) while other inputs need to be set as a group (mutual inclusion). Part of the input specification for the "bet" (Brain Extraction Tool) program from FSL is shown in **Listing 2**.

Currently, Nipype (version 0.4) is distributed with a wide range of interfaces (see **Table 2**). Adding new Interfaces is simply a matter of writing a Python class definition as was shown in **Listing 1**. When a formal specification of inputs and outputs are provided by the underlying software, Nipype can support these programs automatically. For example, the Slicer command line execution modules come with an XML specification that allows Nipype to wrap them without creating individual interfaces.

NODES, MAPNODES, AND WORKFLOWS

Nipype provides a framework for connecting Interfaces to create a data analysis Workflow. In order for Interfaces to be used in a Workflow they need to be encapsulated in either Node or MapNode objects. Node and MapNode objects provide additional functionality to Interfaces. For example, creating a hash of the input state, caching of results, and the ability to iterate over inputs. Additionally, they execute the underlying interfaces in their own uniquely named directories (almost like a sandbox), thus providing a mechanism to isolate and track the outputs resulting from execution of the Interfaces. These mechanisms allow not only for provenance tracking, but aid in efficient pipeline execution.

The MapNode class is a sub-class of Node that implements a MapReduce-like architecture (Dean and Ghemawat, 2008). Encapsulating an Interface within a MapNode allows Interfaces that normally operate on a single input to execute the Interface on multiple inputs. When a MapNode executes, it creates a separate instance of the underlying Interface for every value of an input list and executes these instances independently. When all instances finish running, their results are collected into a list and exposed through the MapNode's outputs (see **Figure 4D**). This approach improves

[1]http://www.clusterresources.com/products/torque-resource-manager.php

[2]Throughout the rest of the paper we are going to use upper case for referring to classes (such as Interfaces, Workflows, etc…) and lower case to refer to general concepts.

[3]http://code.enthought.com/projects/traits/

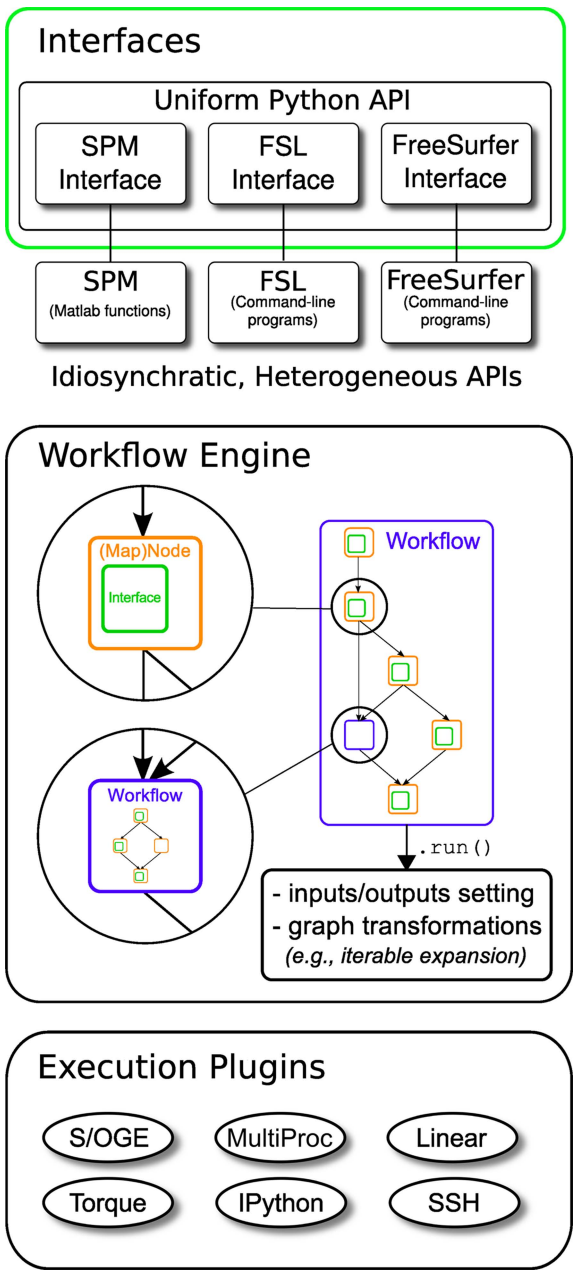


FIGURE 1 | Architecture overview of the Nipype framework. Interfaces are wrapped with Nodes or MapNodes and connected together as a graph within a Workflow. Workflows themselves can act as a Node inside another Workflow, supporting a composite design pattern. The dependency graph is transformed before being executed by the engine component. Execution is performed by one of the plug-ins. Currently Nipype supports serial and parallel (both local multithreading and cluster) execution.

granularity of the Workflow and provides easy support for Interfaces that can only process one input at a time. For example, the FSL "bet" program can only run on a single input, but wrapping the BET Interface in a MapNode allows running "bet" on multiple inputs.

A Workflow object captures the processing stages of a pipeline and the dependencies between these processes. Interfaces encapsulated into Node or MapNode objects can be connected

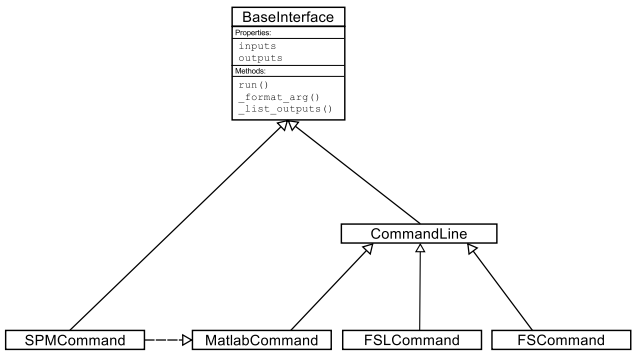


FIGURE 2 | Simplified hierarchy of Interface classes. An object-oriented design is used to reduce code redundancy by defining common functionality in base classes, and makes adding new interfaces easier and quicker. MatlabCommand, FSLCommand, and FSCommand extend the CommandLine class to provide functionality specific to executing MATLAB, FSL, and FreeSurfer programs. The SPMCommand class defines functions that simplify wrapping SPM functionality. The dashed line indicates that the SPMCommand class uses the MatlabCommand class to execute the SPM matlab scripts generated by the SPM interfaces.

together within a Workflow. By connecting outputs of some Nodes to inputs of others, the user implicitly specifies dependencies. These are represented internally as a DAG. The current semantics of Workflow do not allow conditionals and hence the graph needs to be acyclic. Workflows themselves can be a node of the Workflow graph (see **Figure 1**). This enables a hierarchical architecture and encourages Workflow reuse. The Workflow engine validates that all nodes have unique names, ensures that there are no cycles, and prevents connecting multiple outputs to a given input. For example in an fMRI processing Workflow, preprocessing, model fitting, and visualization of results can be implemented as individual Workflows connected together in a main Workflow. This not only improves clarity of designed Workflows but also enables easy exchange of whole subsets. Common Workflows can be shared across different studies within and across laboratories thus reducing redundancy and increasing consistency.

While a neuroimaging processing pipeline could be implemented as a Bash, MATLAB, or a Python script, Nipype explicitly implements a pipeline as a graph. This makes it easy to follow what steps are being executed and in what order. It also makes it easier to go back and change things by simply reconnecting different outputs and inputs or by inserting new Nodes/MapNodes. This alleviates the tedious component of scripting where one has to manually ensure that the inputs and outputs of different processing calls match and that operations do not overwrite each other's outputs.

A Workflow provides a detailed description of the processing steps and how data flows between Interfaces. Thus it is also a source of provenance information. We encourage users to provide Workflow definitions (as scripts or graphs) as supplementary material when submitting articles. This ensures that at least the data processing part of the published experiment is fully reproducible. Additionally, exchange of Workflows between researchers stimulates efficient use of methods and experimentation.

```
from nipype.interfaces.base import (TraitedSpec, CommandLineInputSpec,
                                    CommandLine, File)

import os

class GZipInputSpec(CommandLineInputSpec):
    input_file = File(desc = "File", exists = True, mandatory = True,
                      argstr = "%s")

class GZipOutputSpec(TraitedSpec):
    output_file = File(desc = "Zip file", exists = True)

class GZipTask(CommandLine):
    input_spec = GZipInputSpec
    output_spec = GZipOutputSpec
    cmd = 'gzip'

    def _list_outputs(self):
        outputs = self.output_spec().get()
        outputs['output_file'] = os.path.abspath(self.inputs.input_file + ".gz")
        return outputs

if __name__ == '__main__':
    zipper = GZipTask(input_file='an_existing_file')
    print zipper.cmdline
    zipper.run()
```

LISTING 1 | An example interface wrapping the *gzip* command line tool and a usage example. This Interface takes a file name as an input, calls gzip to compress it and returns a name of the compressed output file.

```
class BETInputSpec(FSLCommandInputSpec):
    in_file = File(exists=True,
                   desc = 'input file to skull strip',
                   argstr='%s', position=0, mandatory=True)
    out_file = File(desc = 'name of output skull stripped image',
                    argstr='%s', position=1, genfile=True)
    mask = traits.Bool(desc = 'create binary mask image',argstr='-m')
    functional = traits.Bool(argstr='-F', xor=('functional','reduce_bias'),
                             desc="apply to 4D fMRI data")
    ...
```

LISTING 2 | Part of the input specification for the Brain Extraction Tool (BET) Interface. Full specification covers 18 different arguments. Each attribute of this class is a Traits object which defines an input and its data type (i.e., list of integers), constraints (i.e., length of the list), dependencies (when for example setting one option is mutually exclusive with another, see the xor parameter), and additional parameters (such as argstr and position which describe how to convert an input into a command line argument).

EXAMPLE – BUILDING A WORKFLOW FROM SCRATCH

In this section, we describe how to create and extend a typical fMRI processing Workflow. A typical fMRI Workflow can be divided into two sections: (1) preprocessing and (2) modeling. The first one deals with cleaning data from confounds and noise and the second one fits a model to the cleaned data based on the experimental design. The preprocessing stage in this Workflow will consist of only two steps: (1) motion correction (aligns all volumes in a functional run to the mean realigned volume) and (2) smoothing (convolution with a 3D Gaussian kernel). We use SPM Interfaces to define the processing Nodes.

```
from nipype.pipeline.engine import Node, Workflow

realign = Node(interface=spm.Realign(),
               name="realign")

realign.inputs.register_to_mean = True

smooth = Node(interface=spm.Smooth(),
              name="smooth")

smooth.inputs.fwhm = 4
```

Table 2 | Supported software.

Name	URL
AFNI	afni.nimh.nih.gov/afni
BRAINS	www.psychiatry.uiowa.edu/mhcrc/IPLpages/BRAINS.htm
Camino	www.cs.ucl.ac.uk/research/medic/camino
Camino-TrackVis	www.nitrc.org/projects/camino-trackvis
ConnecomeViewerToolkit	www.connectomeviewer.org
dcm2nii	www.cabiatl.com/mricro/mricron/dcm2nii.html
Diffusion Toolkit	www.trackvis.org/dtk
FreeSurfer	freesurfer.net
FSL	www.fmrib.ox.ac.uk/fsl
Nipy	nipy.org/nipy
NiTime	nipy.org/nitime
Slicer	www.slicer.org/
SPM	www.fil.ion.ucl.ac.uk/spm
SQLite	www.sqlite.org
PyXNAT	github.com/pyxnat, xnat.org

List of software packages fully or partially supported by Nipype. For more details see http://nipy.org/nipype/interfaces.

We create a Workflow to include these two Nodes and define the data flow from the output of the *realign* Node (*realigned_files*) to the input of the *smooth* Node (*in_files*). This creates a simple preprocessing workflow (see **Figure 3**).

```
preprocessing = Workflow
  (name="preprocessing")
preprocessing.connect(realign,
  "realigned_files", smooth, "in_files")
```

A modeling Workflow is constructed in an analogous manner, by first defining Nodes for model design, model estimation, and contrast estimation. We again use SPM Interfaces for this purpose. However, Nipype adds an extra abstraction Interface for model specification whose output can be used to create models in different packages (e.g., SPM, FSL, and Nipy). The nodes of this Workflow are: SpecifyModel (Nipype model abstraction Interface), Level1Design (SPM design definition), ModelEstimate, and ContrastEstimate. The connected modeling Workflow can be seen on **Figure 3**.

We create a master Workflow that connects the preprocessing and modeling Workflows, adds the ability to select data for processing (using DataGrabber Interface) and a DataSink Node to save the outputs of the entire Workflow. Nipype allows connecting Nodes between Workflows. We will use this feature to connect *realignment_parameters* and *smoothed_files* to modeling workflow.

The DataGrabber Interface allows the user to define flexible search patterns which can be parameterized by user defined inputs (such as subject ID, session, etc.). This Interface can adapt to a wide range of directory organization and file naming conventions. In our case we will parameterize it with subject ID. In this way we can run the same Workflow for different subjects. We automate this by iterating over a list of subject IDs, by setting the *iterables* property of the DataGrabber Node for the input *subject_id*. The DataGrabber Node output is connected to the realign Node from preprocessing Workflow.

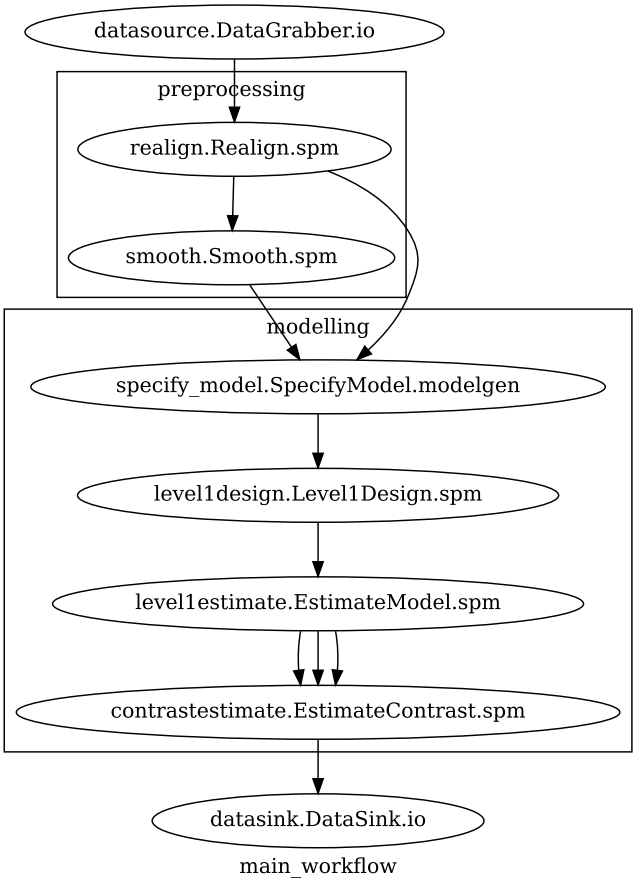


FIGURE 3 | Graph depicting the processing steps and dependencies for a first level functional analysis workflow. Every output–input connection is represented with a separate arrow. Nodes from every subworkflow are grouped in boxes with labels corresponding to the name of the subworkflow. Such graphs can be automatically generated from a Workflow definition and provide a quick overview of the pipeline.

DataSink on the other side provides means for storing selected results in a specified location. It supports automatic creation of folders, simple substitutions, and regular expressions to alter target filenames. In this example we store the statistical (T maps) resulting from contrast estimation.

A Workflow defined this way (see **Figure 3**, for full code see Supplementary Material) is ready to run. This can be done by calling *run()* method of the master Workflow.

If the *run()* method is called twice, the Workflow input hashing mechanism ensures that none of the Nodes are executed during the second run if the inputs remain the same. If, however, a highpass filter parameter of *specify_model* is changed, some of the Nodes (but not all) would have to rerun. Nipype automatically determines which Nodes require rerunning.

ITERABLES – PARAMETER SPACE EXPLORATION

Nipype provides a flexible approach to prototype and experiment with different processing strategies, through the unified and uniform access to a variety of software packages (Interfaces) and creating data flows (Workflows). However, for various neuroimaging

tasks, there is often a need to explore the impact of variations in parameter settings (e.g., how do different amounts of smoothing affect group statistics, what is the impact of spline interpolation over trilinear interpolation). To enable such parametric exploration, Nodes have an attribute called *iterables*.

When an *iterable* is set on a Node input, the Node, *and* its subgraph are executed for each value of the iterable input (see **Figure 4** iterables_vs_mapnode). Iterables can also be set on multiple inputs of a Node (e.g., somenode.iterables = [("input1," [1,2,3]), ("input2," ["a," "b"])]). In such cases, every combination of those values is used as a parameter set (the prior example would result in the following parameter sets: (1, "a"), (1, "b"), (2, "a"), etc…). This feature is especially useful to investigate interactions between parameters of intermediate stages with respect to the final results of a workflow. A common use-case of *iterables* is to execute the same Workflow for many subjects in an fMRI experiment and to simultaneously look at the impact of parameter variations on the results of the Workflow.

It is important to note that unlike MapNode, which creates copies of the underlying interface for every element of an input of type list, *iterables* operate on the subgraph of a node and create copies not only of the node but also of all the nodes dependent on it (see **Figure 4**).

PARALLEL DISTRIBUTION AND EXECUTION PLUG-INS

Nipype supports executing Workflows locally (in series or parallel) or on load-balanced grid-computing clusters (e.g., SGE, Torque, or even via SSH) through an extensible plug-in interface. No change is needed to the Workflow to switch between these execution modes. One simply calls the Workflow's run function with a different plug-in and its arguments. Very often different components of a Workflow can be executed in parallel and even more so when the same Workflow is being repeated on multiple parameters (e.g., subjects). Adding support for additional cluster management systems does not require changes in Nipype, but simply writing a plug-in extension conforming to the plug-in API.

The Workflow engine sends an execution graph to the plug-in. Executing the Workflow in series is then simply a matter of performing a topological sort on the graph and running each node in the sorted order. However, Nipype also provides additional plug-ins that use Python's multi-processing module, use IPython (includes SSH-based, SGE, LSF, PBS, among others) and provide native interfaces to SGE or PBS/Torque clusters. For all of these, the graph structure defines the dependencies as well as which nodes can be executed in parallel at any given stage of execution.

One of the biggest advantages of Nipype's execution system is that parallel execution using local multi-processing plug-in does not require any additional software (such as cluster managers like SGE) and therefore makes prototyping on a local multi-core workstations easy. However for bigger studies and complex Workflows, a high-performance computing cluster can provide substantial improvements in execution time. Since there is a clear separation between definition of the Workflow and its execution, Workflows can be executed in parallel (locally or on a cluster) without any modification. Transitioning from developing a processing pipeline on a single subject on a local workstation to executing it on a bigger cohort on a cluster is therefore seamless.

Rerunning workflows has also been optimized. When a Node or MapNode is run, the framework will actually execute the underlying interface only if inputs have changed relative to prior execution. If not, it will simply return cached results.

THE FUNCTION INTERFACE

One of the Interfaces implemented in Nipype requires special attention: The Function Interface. Its constructor takes as arguments Python function pointer or code, list of inputs, and list of outputs. This allows running any Python code as part of a Workflow. When combined with libraries such as Nibabel (neuroimaging data input and output), Numpy/Scipy (array representation and processing) and scikits-learn or PyMVPA (machine learning and data mining) the Function Interface provides means for rapid prototyping of complex data processing methods. In addition, by using the

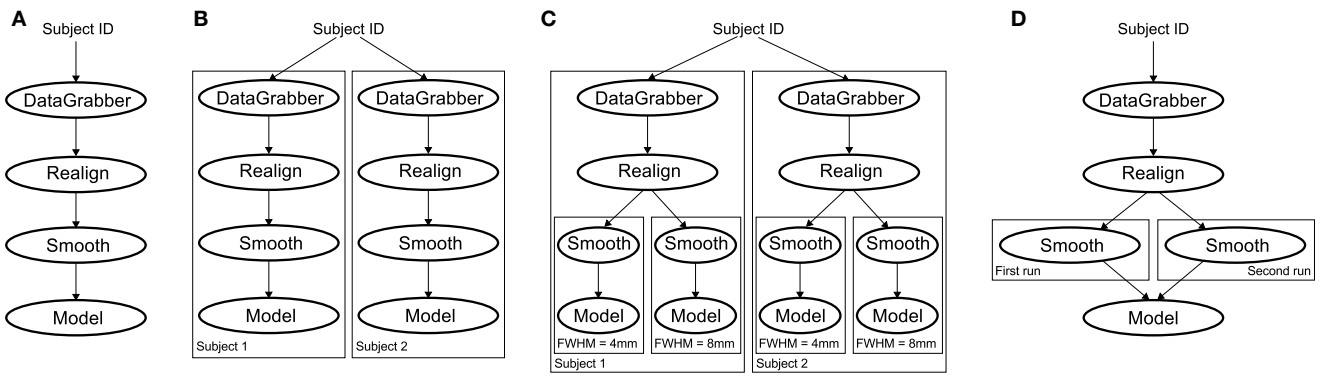


FIGURE 4 | Workflow modification using *iterables* and MapNodes. If we take the processing pipeline **(A)** and set *iterables* parameter of DataGrabber to a list of two subjects, Nipype will effectively execute graph **(B)**. Identical processing will be applied to every subject from the list. Iterables can be used in a graph on many levels. For example, setting iterables on Smooth FWHM to a list of 4 and 8 mm will result in graph **(C)**. In contrast to *iterables*, MapNode branches within a node of the graph and also merges the results of the branches, effectively performing a MapReduce operation **(D)**.

Function Interface users can avoid writing their own Interfaces which is especially useful for *ad hoc* solutions (e.g., calling an external program that has not yet been wrapped as an Interface).

WORKFLOW VISUALIZATION

To be able to efficiently manage and debug Workflows, one has to have access to a graphical representation. Using graphviz (Ellson et al., 2002), Nipype generates static graphs representing Nodes and connections between them. In the current version four types of graphs are supported: *orig* – does not expand inner Workflows, *flat* – expands inner workflows, *exec* – expands workflows and iterables, and *hierarchical* – expands workflows but maintains their hierarchy. Graphs can be saved in a variety of file formats including Scalable Vector Graphics (SVG) and Portable Network Graphics (PNG; see **Figures 3 and 6**).

CONFIGURATION OPTIONS

Certain options concerning verbosity of output and execution efficiency can be controlled through configuration files or variables. These include, among others, *hash_method* and *remove_unecessary_outputs*. As explained before, rerunning a Workflow only recomputes those Nodes whose inputs have changed since the last run. This is achieved by recording a hash of the inputs. For files there are two ways of calculating the hash (controlled by the *hash_method* config option): *timestamp* – based only on the size and modification time and *content* – based on the content of the file. The first one is faster, but does not deal with the situation when an identical copy overwrites the file. The second one can be slower especially for big files, but can tell that two files are identical even if they have different modification times. To allow efficient recomputation Nipype has to store outputs of all Nodes. This can generate a significant amount of data for typical neuroimaging studies. However, not all outputs of every Node are used as inputs to other Nodes or relevant to the final results. Users can decide to remove those outputs (and save some disk space) by setting the *remove_unecessary_outputs* to *True*. These and other configuration options provide a mechanism to streamline the use of Nipype for different applications.

DEPLOYMENT

Nipype supports GNU/Linux and Mac OS X operating systems (OS). A recent Internet survey based study (Hanke and Halchenko, 2011) showed that GNU/Linux is the most popular platform in the neuroimaging community and together with Mac OS X is used by over 70% of neuroimagers. There are not theoretical reasons why Nipype should not work on Windows (Python is a cross-platform language), but since most of the supported software (for example FSL) requires a Unix based OS, Nipype has not been tested on this platform.

We currently provide three ways of deploying Nipype on a new machine: manual installation from sources[4], PyPi repository[5], and from package repositories on Debian-based systems. Manual installation involves downloading a source code archive and running a standard Python installation script (distutils). This way the user has to take care of installing all of the dependencies. Installing from PyPI repository lifts this constraint by providing dependency information and automatically installing required packages. Nipype is available from standard repositories on recent Debian and Ubuntu releases. Moreover, NeuroDebian[6] (Hanke et al., 2010) repository provides the most recent releases of Nipype for Debian-based systems and a NeuroDebian Virtual Appliance making it easy to deploy Nipype and other imaging tools in a virtual environment on other OS, e.g., Windows. In addition to providing all core dependencies and automatic updates NeuroDebian also provides many of the software packages supported by Nipype (AFNI, FSL, Mricron, etc.), making deployment of heterogeneous Nipype pipelines more straightforward.

DEVELOPMENT

Nipype is trying to address the problem of interacting with the ever changing universe of neuroimaging software in a sustainable manner. Therefore the way its development is managed is a part of the solution. Nipype is distributed under BSD license which allows free copying, modification, and distribution and additionally meets all the requirements of open-source definition (see Open-Source Initiative[7]) and Debian Free Software Guidelines[8]. Development is carried out openly through distributed version control system (*git* via GitHub[9]) in an online community. The current version of the source code together with complete history is accessible to everyone. Discussions between developers and design decisions are done on an open access mailing list. This setup encourages a broader community of developers to join the project and allows sharing of the development resources (effort, money, information, and time).

In these previous paragraphs, we presented key features of Nipype that facilitate rapid development and deployment of analysis procedures in laboratories, and address all of the issues described earlier. In particular, Nipype provides: (1) uniform access to neuroimaging analysis software and usage information; (2) a framework for comparative algorithm development and dissemination; (3) an environment for methodological continuity and paced training of new personnel in laboratories; (4) computationally efficient execution of neuroimaging analysis; and (5) a mechanism to capture the data processing details in compact scripts and graphs. In the following section, we provide examples to demonstrate these solutions.

USAGE EXAMPLES

UNIFORM ACCESS TO TOOLS, THEIR USAGE, AND EXECUTION

Users access Interfaces by importing them from Nipype modules. Each neuroimaging software distribution such as FSL, SPM, Camino, etc., has a corresponding module in the *nipype.interfaces* namespace.

```
from nipype.interfaces.camino import DTIFit
```

The *help()* function for each interface prints the inputs and the outputs associated with the interface.

[4]http://nipy.org/nipype/

[5]http://pypi.python.org/pypi/nipype/

[6]http://neuro.debian.net

[7]http://www.opensource.org/docs/osd

[8]http://www.debian.org/social_contract#guidelines

[9]http://github.com/nipy/nipype

```
>>> DTIFit.help()
Inputs
------

Mandatory:
 in_file: voxel-order data filename
 scheme_file: Camino scheme file
 (b values / vectors, see camino.fsl2scheme)

Optional:
 args: Additional parameters to the command
 environ: Environment variables (default={})
 ignore_exception: Print an error message
instead of throwing an exception in case
the interface fails to run (default=False)
 non_linear: Use non-linear fitting instead
of the default linear regression to the log
measurements.
 out_file: None

Outputs
-------
tensor_fitted: path/name of 4D volume in voxel
 order
```

The output of the *help()* function is standardized across all Interfaces. It is automatically generated based on the traited input and output definitions and includes information about required inputs, types, and default value. Alternatively, extended information is available in the form of auto-generated HTML documentation on the Nipype website (see **Figure 5**). This extended information includes examples that demonstrate how the interface can be used.

For every Interface, input values are set through the *inputs* field:

```
fit.inputs.scheme_file = 'A.scheme'
fit.inputs.in_file = \
                'tensor_fitted_data.Bfloat'
```

When trying to set an invalid input type (for example a non-existing input file, or a number instead of a string) the Nipype framework will display an error message. Input validity checking before actual Workflow execution saves time. To run an Interface user needs to call run() method:

```
fit.run()
```

At this stage the framework checks if all mandatory inputs are set and all input dependencies are satisfied, generating an error if either of these conditions are not met.

DTIFit

Wraps command **dtfit**

Reads diffusion MRI data, acquired using the acquisition scheme detailed in the scheme file, from the data file.

Use non-linear fitting instead of the default linear regression to the log measurements. The data file stores the diffusion MRI data in voxel order with the measurements stored in big-endian format and ordered as in the scheme file. The default input data type is four-byte float. The default output data type is eight-byte double. See modelfit and camino for the format of the data file and scheme file. The program fits the diffusion tensor to each voxel and outputs the results, in voxel order and as big-endian eight-byte doubles, to the standard output. The program outputs eight values in each voxel: [exit code, ln(S(0)), D_xx, D_xy, D_xz, D_yy, D_yz, D_zz]. An exit code of zero indicates no problems. For a list of other exit codes, see modelfit(1). The entry S(0) is an estimate of the signal at q=0.

Example

```
>>> import nipype.interfaces.camino as cmon
>>> fit = cmon.DTIFit()
>>> fit.inputs.scheme_file = 'A.scheme'
>>> fit.inputs.in_file = 'tensor_fitted_data.Bfloat'
>>> fit.run()
```

Inputs:

```
[Mandatory]
in_file : (an existing file name)
        voxel-order data filename
scheme_file : (an existing file name)
        Camino scheme file (b values / vectors, see camino.fsl2scheme)

[Optional]
args : (a string)
        Additional parameters to the command
environ : (a dictionary with keys which are a value of type 'str' and with values which are a value of type 'str')
        Environment variables
ignore_exception : (a boolean)
        Print an error message instead of throwing an exception in case the interface fails to run
non_linear : (a boolean)
        Use non-linear fitting instead of the default linear regression to the log measurements.
out_file : (a file name)
        Unknown
```

Outputs:

```
tensor_fitted : (an existing file name)
        path/name of 4D volume in voxel order
```

FIGURE 5 | HTML help page for dtfit command from Camino. This was generated based on the Interface code: description and example was taken from the class docstring and inputs/outputs were list was created using traited input/output specification.

Nipype standardizes running and accessing help information irrespective of whether the underlying software is a MATLAB program, a command line tool, or Python module. The framework deals with translating inputs into appropriate form (e.g., command line arguments or MATLAB scripts) for executing the underlying tools in the right way, while presenting the user with a uniform interface.

A FRAMEWORK FOR COMPARATIVE ALGORITHM DEVELOPMENT AND DISSEMINATION

Uniform semantics for interfacing with a wide range of processing methods not only opens the possibility for richer Workflows, but also allows comparing algorithms that are designed to solve the same problem across and within such diverse Workflows. Typically, such an exhaustive comparison can be time-consuming, because of the need to deal with interfacing different software packages. Nipype simplifies this process by standardizing the access to the software. Additionally, the *iterables* mechanism allows users to easily extend such comparisons by providing a simple mechanism to test different parameter sets.

Accuracy or efficiency of algorithms can be determined in an isolated manner by comparing their outputs or execution time or memory consumption on a given set of data. However, researchers typically want to know how different algorithms used at earlier stages of processing might influence the final output or statistics they are interested in. As an example of such use, we have compared voxelwise isotropic, voxelwise anisotropic, and surface based smoothing all for two levels of FWHM – 4 and 8 mm. First one is the standard convolution with Gaussian kernel as implemented in SPM. Second one involves smoothing only voxels of similar intensity in attempt to retain structure. This was implemented in SUSAN from FSL (Smith, 1992). Third method involves reconstructing surface of the cortex and smoothing along it (Hagler et al., 2006). This avoids bleeding of signal over sulci.

Establishing parameters from data and smoothing using SUSAN is already built into Nipype as a Workflow. It can be created using *create_susan_smooth()* function. It has similar inputs and outputs as SPM Smooth Interface. Smoothing on a surface involves doing a full cortical reconstruction from T1 volume using FreeSurfer (Fischl et al., 1999) followed by coregistering functional images to the reconstructed surface using BBRegister (Greve and Fischl, 2009). Finally, the surface smoothing algorithm from FreeSurfer is called.

Smoothed EPI volumes (direct/local influence) and statistical maps (indirect/global influence), along with the pipeline used to generate them can be found in **Figures 6 and 7**. Full code used to generate this data can be found in the Supplementary Material. This comparison serves only to demonstrate Nipype capabilities; a comparison between smoothing methods is outside of the scope of this paper.

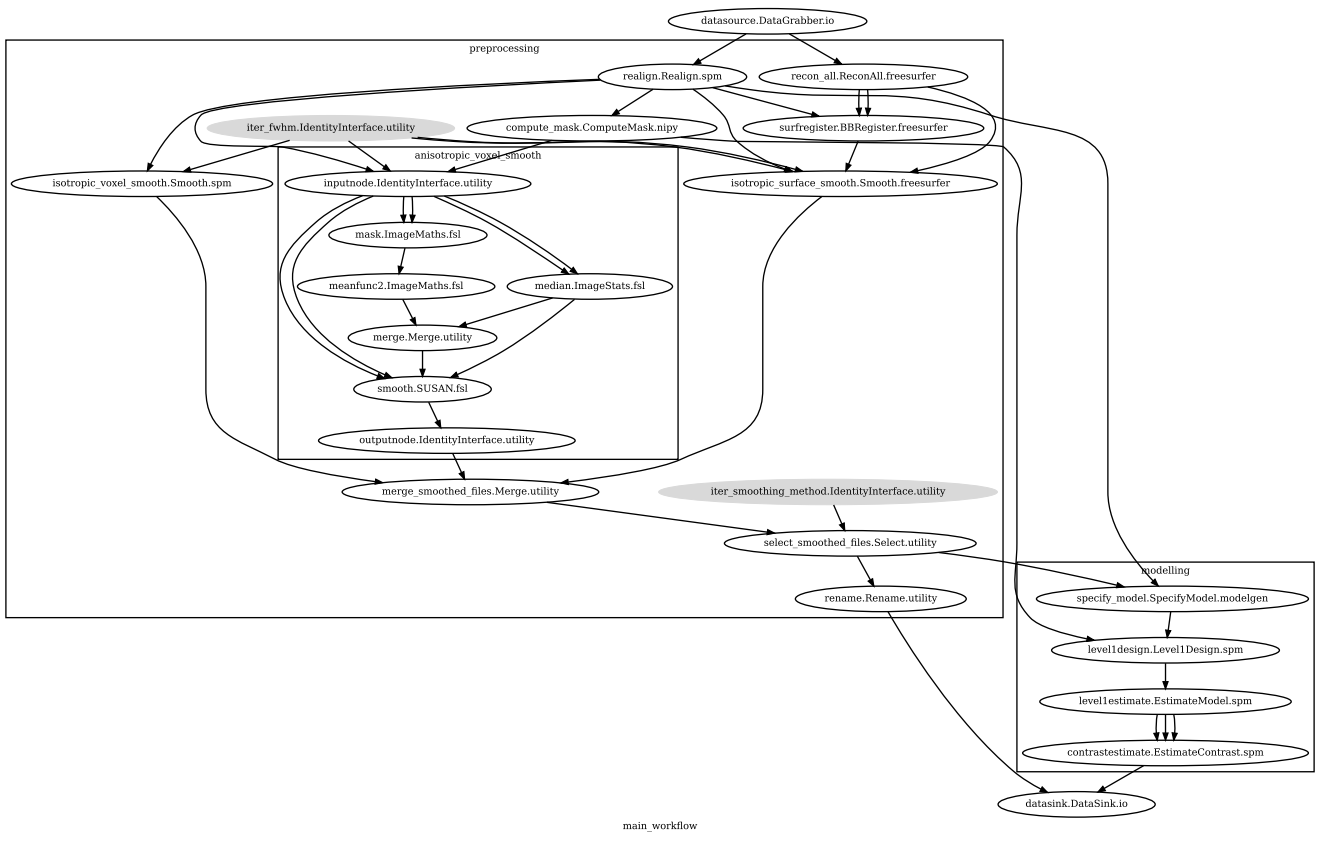


FIGURE 6 | Graph showing the workflow used for the smoothing methods and parameters comparison. The gray shaded nodes have *iterables* parameter set. This allows easy iteration over all combinations of FWHM and smoothing algorithms used in the comparison.

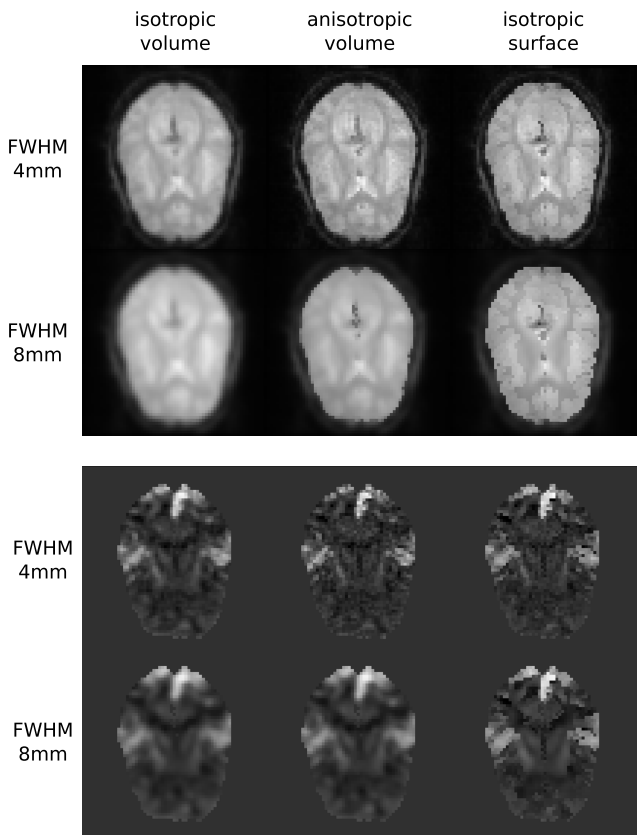


FIGURE 7 | Influence of different smoothing methods and their parameters. Upper half shows direct influence of smoothing on the EPI sequence (slice 16, volume 0, run 2). Lower half shows indirect influence of smoothing on the T maps (same slice) of the main contrast.

Algorithm comparison is not the only way Nipype can be useful for a neuroimaging methods researcher. It is in the interest of every methods developer to make his or hers work most accessible. This usually means providing ready to use implementations. However, because the field is so diverse, software developers have to provide several packages (SPM toolbox, command line tool, C++ library, etc.) to cover the whole user base. With Nipype, a developer can create one Interface and expose a new tool, written in any language, to a greater range of users, knowing it will work with the wide range of software currently supported by Nipype.

A good example of such scenario is ArtifactDetection toolbox[10]. This piece of software uses EPI timeseries and realignment parameters to find timepoints (volumes) that are most likely artifacts and should be removed (by including them as confound regressors in the design matrix). The tool was initially implemented as a MATLAB script, compatible only with SPM and used locally within the lab. The current Nipype interface can work with SPM or FSL Workflows, thereby not limiting its users to SPM.

[10]http://www.nitrc.org/projects/artifact_detect/

AN ENVIRONMENT FOR METHODOLOGICAL CONTINUITY AND PACED TRAINING OF NEW PERSONNEL IN LABORATORIES

Neuroimaging studies in any laboratory typically use similar data processing methods with possibly different parameters. Nipype Workflows can be very useful in dividing the data processing into reusable building blocks. This not only improves the speed of building new Workflows but also reduces the number of potential errors, because a well tested piece of code is being reused (instead of being reimplemented every time). Since a Workflow definition is an abstract and simplified representation of the data processing stream, it is much easier to describe and hand over to new project personnel. Furthermore, a data independent Workflow definition (see **Figure 8**) enables sharing Workflows within and across research laboratories. Nipype provides a high-level abstraction mechanism for exchanging knowledge and expertise between researchers focused on methods in neuroimaging and those interested in applications.

The uniform access to Interfaces and the ease of use of Workflows in Nipype helps with training new staff. Composition provided by Workflows allows users to gradually increase the level of details when learning how to perform neuroimaging analysis. For example user can start with a "black box" Workflow that does analysis from A to Z, and gradually learn what the sub-components (and their sub-components) do. Playing with Interfaces in an interactive console is also a great way to learn how different algorithms work with different parameters without having to understand how to set them up and execute them.

COMPUTATIONALLY EFFICIENT EXECUTION OF NEUROIMAGING ANALYSIS

A computationally efficient execution allows for multiple rapid iterations to optimize a Workflow for a given application. Support for optimized local execution (running independent processes in parallel, rerunning only those steps that have been influenced by the changes in parameters or dependencies since the last run) and exploration of parameter space eases Workflow development. The Nipype package provides a seamless and flexible environment for executing Workflows in parallel on a variety of environments from local multi-core workstations to high-performance clusters. In the SPM workflow for single subject functional data analysis (see **Figure 9**), only a few components can be parallelized. However, running this Workflow across several subjects provides room for embarrassingly parallel execution. Running this Workflow in distributed mode for 69 subjects on a compute cluster (40 cores distributed across 6 machines) took 1 h and 40 min relative to the 32-min required to execute the analysis steps in series for a single subject on the same cluster. The difference from the expected runtime of 64 min (32 min for the first 40 subjects and another 32 min for the remaining 29 subjects) stems from disk I/O and other network and processing resource bottlenecks.

CAPTURES DETAILS OF ANALYSIS REQUIRED TO REPRODUCE RESULTS

The graphs and code presented in the examples above capture all the necessary details to rerun the analysis. Any user, who has the same versions of the tools installed on their machine and access to the data and scripts, will be able to reproduce the results of the study. For example, running Nipype within the NeuroDebian framework can

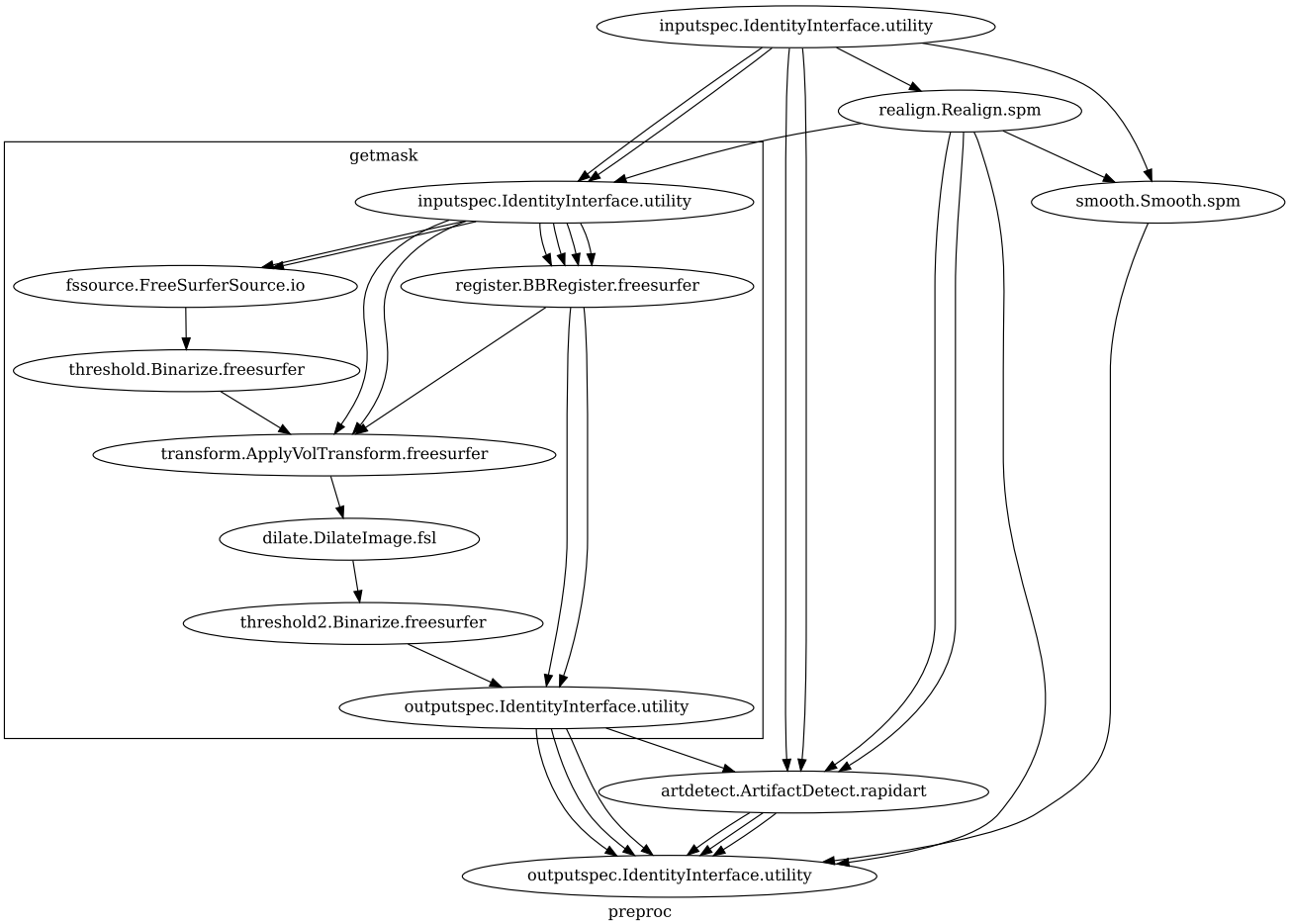


FIGURE 8 | create_spm_preproc() functions returns this reusable, data independent Workflow. It implements typical fMRI preprocessing with smoothing (SPM), motion correction (SPM), artifact detection (Nipype), and coregistration (FreeSurfer). Inputs and outputs are grouped using IdentityInterfaces. Thanks to this, changes in the configuration of the nodes will not break backward compatibility. For full source code see Supplementary Material.

provide access to specific versions of the underlying tools. This provides an easy mechanism to be compliant with the submitting data and scripts/code mandates of journals such as PNAS and Science.

DISCUSSION

Current neuroimaging software offer users an incredible opportunity to analyze their data in different ways, with different underlying assumptions. However, this heterogeneous collection of specialized applications creates several problems: (1) No uniform access to neuroimaging analysis software and usage information; (2) No framework for comparative algorithm development and dissemination; (3) Personnel turnover in laboratories often limit methodological continuity and training new personnel takes time; (4) Neuroimaging software packages do not address computational efficiency; and (5) Method sections of journal articles are often inadequate for reproducing results.

We addressed these issues by creating Nipype, an open-source, community-developed initiative under the umbrella of Nipy. Nipype, solves these issues by providing uniform Interfaces to existing neuroimaging software and by facilitating interaction between these packages within Workflows. Nipype provides an environment that encourages interactive exploration of algorithms from different packages (e.g., SPM, FSL), eases the design of Workflows within and between packages, and reduces the learning curve necessary to use different packages. Nipype is addressing limitations of existing pipeline systems and creating a collaborative platform for neuroimaging software development in Python, a high-level scientific computing language.

We use Python for several reasons. It has extensive scientific computing and visualization support through packages such as SciPy, NumPy, Matplotlib, and Mayavi (Pérez et al., 2010; Millman and Aivazis, 2011). The Nibabel package provides support for reading and writing common neuroimaging file formats (e.g., NIFTI, ANALYZE, and DICOM). Being a high-level language, Python supports rapid prototyping, is easy to learn and adopt and is available across all major OS. At the same time Python allows to seamlessly bind with C code (using Weave package) for improved efficiency of critical subroutines.

Python is also known to be a good choice for the first programming language to learn (Zelle, 1999) and is chosen as the language for introductory programming at many schools and universities[11].

[11]http://wiki.python.org/moin/SchoolsUsingPython

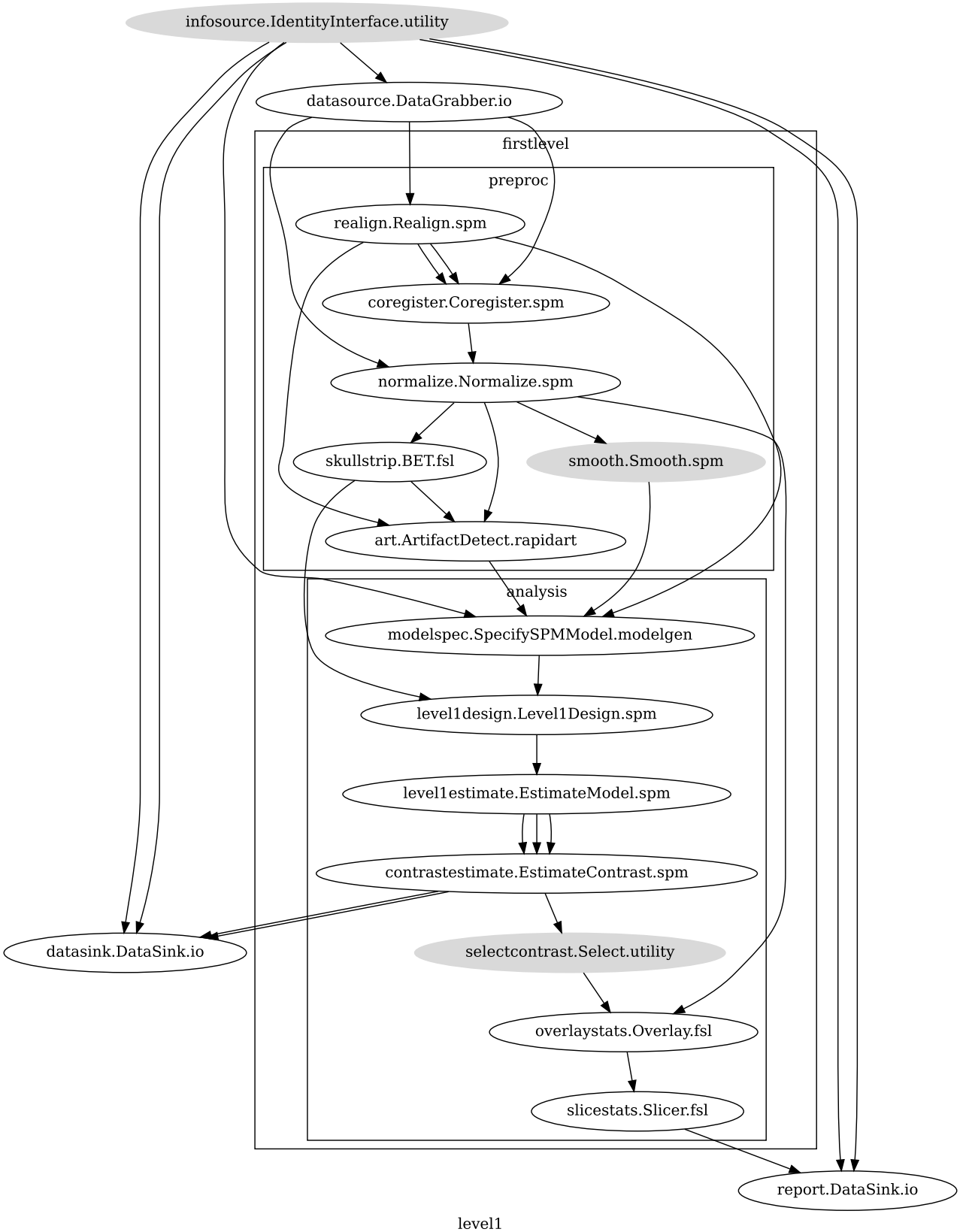


FIGURE 9 | Single subject fMRI Workflow used for benchmarking parallel execution.

Being a generic and free language, with various extensions available "out of the box," it has allowed many researchers to start implementing and sharing their ideas with minimal knowledge of Python, while learning more of the language and programming principles along the way. Many such endeavors later on became popular community-driven FOSS projects, attracting users and contributors, and even outlasting the involvement of the original authors. Python has already been embraced by the neuroscientific community and is rapidly gaining popularity (Bednar, 2009; Goodman and Brette, 2009). The Connectome Viewer Toolkit (Gerhard et al., 2011), Dipy (Garyfallidis et al., 2011), NiBabel[12], Nipy[13], NiTime (Rokem et al., 2009), PyMVPA (Hanke et al., 2009), PyXNAT (Schwartz et al., 2011), and Scikits-Learn[14] are just a few examples of neuroimaging related software written in Python. Nipype, based on Python, thus has immediate access to this extensive community and its software, technological resources and support structure.

Nipype provides a formal and flexible framework to accommodate the diversity of imaging software. Within the neuroimaging community, not all software is limited to well-behaved command line tools. Furthermore, a number of these tools do not have well defined inputs, outputs, or usage help. Although, currently we use Enthought Traits to define inputs and outputs of interfaces, such definitions could be easily translated into instances of XML schemas compatible with other pipeline frameworks. On the other hand, when a tool provides a formal XML description of their inputs and outputs (e.g., Slicer 3D, BRAINS), it is possible to take these definitions and automatically generate Nipype wrappers for those classes.

Nipype development welcomes input and contributions from the community. The source code is freely distributed under a BSD license allowing anyone any use of the software and Nipype conforms to the Open Software Definition of the Open-Source Initiative. Development process is fully transparent and encourages contributions from users from all around the world. The diverse and geographically distributed user and developer base makes Nipype a flexible project that takes into account needs of many scientists.

Improving openness, transparency, and reproducibility of research has been a goal of Nipype since its inception. A Workflow definition is, in principle, sufficient to reproduce the analysis. Since it was used to analyze the data, it is more detailed and accurate than a typical methods description in a paper, but also has the advantage of being reused and shared within and across laboratories. Accompanying a publication with a formal definition of the processing pipeline (such as a Nipype script) increases reproducibility and transparency of research. The Interfaces and Workflows of Nipype capture neuroimaging analysis knowledge and the evolution of methods. Although, at the execution level, Nipype already captures a variety of provenance information, this aspect can be improved by generating provenance reports defined by a standardized XML schema (Mackenzie-Graham et al., 2008).

Increased diversity of neuroimaging data processing software has made systematic comparison of performance and accuracy of underlying algorithms essential (for examples, see Klein et al., 2009, 2010). However, a platform for comparing algorithms, either by themselves or in the context of an analysis workflow, or determining optimal workflows in a given application context (e.g., Churchill et al., 2011), does not exist. Furthermore, in this context of changing hardware and software, traditional analysis approaches may not be suitable in all contexts (e.g., data from 32-channel coils which show a very different sensitivity profile, or data from children). Nipype can make such evaluations, design of optimal workflows, and investigations easier (as demonstrated via the smoothing example above), resulting in more efficient data analysis for the community.

SUMMARY

We presented Nipype, an extensible Python library and framework that provides interactive manipulation of neuroimaging data through uniform Interfaces and enables reproducible, distributed analysis using the Workflow system. Nipype has encouraged the scientific exploration of different algorithms and associated parameters, eased the development of Workflows within and between packages and reduced the learning curve associated with understanding the algorithms, APIs, and user interfaces of disparate packages. An open, community-driven development philosophy provides the flexibility required to address the diverse needs in neuroimaging analysis. Overall, Nipype represents an effort toward collaborative, open-source, reproducible, and efficient neuroimaging software development and analysis.

ACKNOWLEDGMENTS

A list of people who have contributed code to the project is available at http://github.com/nipy/nipype/contributors. We thank Fernando Perez, Matthew Brett, Gael Varoquaux, Jean-Baptiste Poline, Bertrand Thirion, Alexis Roche, and Jarrod Millman for technical and social support and for design discussions. We would like to thank Prof. John Gabrieli's laboratory at MIT for testing Nipype through its evolutionary stages, in particular, Tyler Perrachione and Gretchen Reynolds. We would also like to thank the developers of FreeSurfer, FSL, and SPM for being supportive of the project and providing valuable feedback on technical issues. We would like to thank James Bednar, Stephan Gerhard, and Helen Ramsden for providing feedback during the preparation of the manuscript. Satrajit Ghosh would like to acknowledge support from NIBIB R03 EB008673 (PI: Ghosh and Whitfield-Gabrieli), the Ellison Medical Foundation, Katrien Vander Straeten, and Amie Ghosh. Krzysztof Gorgolewski would like to thank his supervisors Mark Bastin, Cyril Pernet, and Amos Storkey for their support during this project.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at http://www.frontiersin.org/neuroinformatics/10.3389/fninf.2011.00013/abstract

[12]http://nipy.sourceforge.net/nibabel/

[13]http://nipy.sourceforge.net/nipy/

[14]http://scikit-learn.sourceforge.net

REFERENCES

Avants, B., and Gee, J. C. (2004). Geodesic estimation for large deformation anatomical shape averaging and interpolation. *Neuroimage* 23(Suppl. 1), S139–S150.

Bednar, J. A. (2009). Topographica: building and analyzing map-level simulations from Python, C/C++, MATLAB, NEST, or NEURON components. *Front. Neuroinform.* 3:8. doi: 10.3389/neuro.11.008.2009

Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., Silva, C. T., and Vo, H. T. (2006). "VisTrails: visualization meets data management," in *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, Chicago, IL, 745–747.

Churchill, N. W., Oder, A., Abdi, H., Tam, F., Lee, W., Thomas, C., Ween, J. E., Graham, S. J., and Strother, S. C. (2011). Optimizing preprocessing and analysis pipelines for single-subject fMRI. I. Standard temporal motion and physiological noise correction methods. *Hum. Brain Mapp.* doi: 10.1002/hbm.21238. [Epub ahead of print].

Cointepas, Y., Mangin, J., Garnero, L., and Poline, J. (2001). BrainVISA: software platform for visualization and analysis of multi-modality brain data. *Neuroimage* 13, 98.

Dean, J., and Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1–13.

Dinov, I., Lozev, K., Petrosyan, P., Liu, Z., Eggert, P., Pierce, J., Zamanyan, A., Chakrapani, S., Van Horn, J., Parker, D. S., Magsipoc, R., Leung, K., Gutman, B., Woods, R., and Toga, A. (2010). Neuroimaging study designs, computational analyses and data provenance using the LONI pipeline. *PLoS ONE* 5, e13070. doi: 10.1371/journal.pone.0013070

Dinov, I. D., Van Horn, J. D., Lozev, K. M., Magsipoc, R., Petrosyan, P., Liu, Z., Mackenzie-Graham, A., Eggert, P., Parker, D. S., and Toga, A. W. (2009). Efficient, distributed and interactive neuroimaging data analysis using the LONI pipeline. *Front. Neuroinform.* 3:22. doi: 10.3389/neuro.11.022.2009

Ellson, J., Gansner, E., Koutsofios, L., North, S., and Woodhull, G. (2002). "Graphviz – open source graph drawing tools," in *Proceeding of 9th Internation Symposium on Graph Drawing*, Vienna, 594–597.

Fischl, B., Sereno, M. I., and Dale, A. M. (1999). Cortical surface-based analysis. II: inflation, flattening, and a surface-based coordinate system. *Neuroimage* 9, 195–207.

Fissell, K., Tseytlin, E., Cunningham, D., Iyer, K., Carter, C. S., Schneider, W., and Cohen, J. D. (2003). A graphical computing environment for neuroimaging analysis. *Neuroinformatics* 1, 111–125.

Garyfallidis, E., Brett, M., Amirbekian, B., Nguyen, C., Yeh, F.-C., Halchenko, Y., and Nimmo-Smith, I. (2011). "Dipy – a novel software library for diffusion MR and tractography," in *17th Annual Meeting of the Organization for Human Brain Mapping*, Quebec City, QC.

Gerhard, S., Daducci, A., Lemkaddem, A., Meuli, R., Thiran, J. P., and Hagmann, P. (2011). The connectome viewer toolkit: an open source framework to manage, analyze, and visualize connectomes. *Front. Neuroinform.* 5:3. doi: 10.3389/fninf.2011.00003

Goodman, D. F., and Brette, R. (2009). The brian simulator. *Front. Neurosci.* 3:2. doi:10.3389/neuro.01.026.2009

Greve, D. N., and Fischl, B. (2009). Accurate and robust brain image alignment using boundary-based registration. *Neuroimage* 48, 63–72.

Hagler, D. J., Saygin, A. P., and and Sereno, M. I. (2006). Smoothing and cluster thresholding for cortical surface-based group analysis of fMRI data. *Neuroimage* 33, 1093–1103.

Hanke, M., and Halchenko, Y. O. (2011). Neuroscience runs on GNU/Linux. *Front. Neuroinform.* 5: 8. doi: 10.3389/fninf.2011.00008

Hanke, M., Halchenko, Y. O., Haxby, J. V., and and Pollmann, S. (2010). "Improving efficiency in cognitive neuroscience research with NeuroDebian," in *Cognitive Neuroscience Society*, Montréal.

Hanke, M., Halchenko, Y. O., Sederberg, P. B., Hanson, S. J., Haxby, J. V., and Pollmann, S. (2009). PyMVPA: a Python toolbox for multivariate pattern analysis of fMRI data. *Neuroinformatics* 7, 37–53.

Hömke, L. (2006). A multigrid method for anisotropic PDEs in elastic image registration. *Numer. Linear Algebra Appl.* 13, 215–229.

Klein, A., Andersson, J., Ardekani, B. A., Ashburner, J., Avants, B., Chiang, M. C., Christensen, G. E., Collins, D. L., Gee, J., Hellier, P., Song, J. H., Jenkinson, M., Lepage, C., Rueckert, D., Thompson, P., Vercauteren, T., Woods, R. P., Mann, J. J., and Parsey, R. V. (2009). Evaluation of 14 nonlinear deformation algorithms applied to human brain MRI registration. *Neuroimage* 46, 786–802.

Klein, A., Ghosh, S. S., Avants, B., Yeo, B. T., Fischl, B., Ardekani, B., Gee, J. C., Mann, J. J., and Parsey, R. V. (2010). Evaluation of volume-based and surface-based brain image registration methods. *Neuroimage* 51, 214–220.

Mackenzie-Graham, A. J., Van Horn, J. D., Woods, R. P., Crawford, K. L., and Toga, A. W. (2008). Provenance in neuroimaging. *Neuroimage* 42, 178–195.

McAuliffe, M. J., Lalonde, F. M., McGarry, D., Gandler, W., Csaky, K., and Trus, B. L. (2001). "Medical image processing, analysis and visualization in clinical research," in *Proceedings 14th IEEE Symposium on Computer-Based Medical Systems*, Bethesda, MD, 381–386.

Millman, K. J., and and Aivazis, M. (2011). Python for scientists and engineers. *Comput. Sci. Eng.* 13, 9–12.

Oinn, T., Greenwood, M., Addis, M. J., Alpdemir, M. N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D. J., Li, P., Lord, P., Pocock, M. R., Senger, M., Stevens, R., Wipat, A., and Wroe, C. (2006). Taverna: lessons in creating a workflow environment for the life sciences. *Concurr. Comput.* 18, 1067–1100.

Pérez, F., Granger, B. E., and Hunter, J. D. (2010). Python: an ecosystem for scientific computing. *Comput. Sci. Eng.* 13, 13–21.

Rex, D. E., Ma, J. Q., and Toga, A. W. (2003). The LONI pipeline processing environment. *Neuroimage* 19, 1033–1048.

Rokem, A., Trumpis, M., and Perez, F. (2009). "Nitime: time-series analysis for neuroimaging data," in *Proceedings of the 8th Python in Science Conference*, Pasadena.

Schwartz, Y., Barbot, A., Vincent, F., Thyreau, B., Varoquaux, G., Thirion, B., and Poline, J. (2011). "PyXNAT: a Python interface for XNAT," in *17th Annual Meeting of the Organization for Human Brain Mapping*, Quebec City, QC.

Smith, S. M. (1992). "A new class of corner finder," in *Proceedings 3rd British Machine Vision Conference* (Leeds: University of Leeds), 139–148.

Zelle, J. M. (1999). "Python as a first language," in *Proceedings of 13th Annual Midwest Computer Conference*, Lisle, IL, 2.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 23 June 2011; accepted: 23 July 2011; published online: 22 August 2011.
Citation: Gorgolewski K, Burns CD, Madison C, Clark D, Halchenko YO, Waskom ML, Ghosh SS (2011) Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in Python. Front. Neuroinform. 5:13. doi: 10.3389/fninf.2011.00013

frontiers in
NEUROINFORMATICS

ORIGINAL RESEARCH ARTICLE
published: 03 April 2012
doi: 10.3389/fninf.2012.00007

The pipeline system for Octave and Matlab (PSOM): a lightweight scripting framework and execution engine for scientific workflows

***Pierre Bellec*[1,2]*, *Sébastien Lavoie-Courchesne*[1,2,3], *Phil Dickinson*[1,3], *Jason P. Lerch*[4,5], *Alex P. Zijdenbos*[6] *and Alan C. Evans*[3]**

[1] *Centre de Recherche de l'Institut Universitaire de Gériatrie de Montréal, Montréal, QC, Canada*
[2] *Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Montréal, QC, Canada*
[3] *McConnell Brain Imaging Centre, Montreal Neurological Institute, McGill University, Montréal, QC, Canada*
[4] *Mouse Imaging Centre, The Hospital for Sick Children, Toronto, ON, Canada*
[5] *Department of Medical Biophysics, University of Toronto, Toronto, ON, Canada*
[6] *Biospective Incorporated, Montréal, QC, Canada*

Edited by:
Andrew P. Davison, Centre National de la Recherche Scientifique, France

Reviewed by:
Ivo Dinov, University of California, USA
Yann Cointepas, CEA - NeuroSpin, France

***Correspondence:**
Pierre Bellec, Centre de Recherche de l'Institut Universitaire de Gériatrie de Montréal, 4545 chemin Queen-Mary, Montréal, QC H3W 1W5, Canada.
e-mail: pierre.bellec@criugm.qc.ca

The analysis of neuroimaging databases typically involves a large number of inter-connected steps called a pipeline. The pipeline system for Octave and Matlab (PSOM) is a flexible framework for the implementation of pipelines in the form of Octave or Matlab scripts. PSOM does not introduce new language constructs to specify the steps and structure of the workflow. All steps of analysis are instead described by a regular Matlab data structure, documenting their associated command and options, as well as their input, output, and cleaned-up files. The PSOM execution engine provides a number of automated services: (1) it executes jobs in parallel on a local computing facility as long as the dependencies between jobs allow for it and sufficient resources are available; (2) it generates a comprehensive record of the pipeline stages and the history of execution, which is detailed enough to fully reproduce the analysis; (3) if an analysis is started multiple times, it executes only the parts of the pipeline that need to be reprocessed. PSOM is distributed under an open-source MIT license and can be used without restriction for academic or commercial projects. The package has no external dependencies besides Matlab or Octave, is straightforward to install and supports of variety of operating systems (Linux, Windows, Mac). We ran several benchmark experiments on a public database including 200 subjects, using a pipeline for the preprocessing of functional magnetic resonance images (fMRI). The benchmark results showed that PSOM is a powerful solution for the analysis of large databases using local or distributed computing resources.

Keywords: pipeline, workflow, Octave, Matlab, open-source, parallel computing, high-performance computing, neuroimaging

1. INTRODUCTION

The rapid development of public databases in neuroimaging (e.g., Evans, 2006; Biswal et al., 2010; Burton, 2011) is opening exciting avenues for data mining. The analysis of a neuroimaging database typically involves a large number of inter-connected processing steps, collectively referred to as a pipeline (or workflow) (Deelman et al., 2009). Neuroimaging pipelines can be implemented as a Matlab script, e.g., DPARSF (Chao-Gan and Yu-Feng, 2010), fMRIstat[1] (Worsley et al., 2002), SPM[2] (Ashburner, 2011), or brainstorm[3] (Tadel et al., 2011). Matlab is a programming language for general scientific computing, well-adapted to the rapid prototyping of new algorithms. It can also wrap heterogeneous tools implemented in a variety of languages. To facilitate the inclusion of these computational tools in complex scientific workflows, we developed a general-purpose pipeline system in Octave and Matlab (PSOM)[4]. To contrast PSOM against alternative projects, we reviewed key features of popular packages within four areas of a pipeline life cycle (Deelman et al., 2009): (1) composition of the pipeline; (2) mapping of the pipeline to the underlying resources; (3) execution of the pipeline; (4) recording of the metadata and provenance.

1.1. PIPELINE COMPOSITION

The composition of a pipeline is the generation of a (possibly abstract) representation of all steps of analysis and associated dependencies, including access to datasets. Many extensions of existing languages have been developed for that purpose, such as matlabbatch[5] for Matlab, or Nipype[6] (Gorgolewski et al., 2011) and the Soma-workflow[7] (Laguitton et al., 2011) for Python.

[1] http://www.math.mcgill.ca/keith/fmristat/
[2] www.fil.ion.ucl.ac.uk/spm/
[3] http://neuroimage.usc.edu/brainstorm/
[4] http://code.google.com/p/psom/
[5] http://sourceforge.net/apps/trac/matlabbatch/wiki
[6] nipy.org/nipype
[7] http://brainvisa.info/soma-workflow

Some scripting languages were also developed specifically to compose pipelines, e.g., DAGMan[8], Swift[9] (Wilde et al., 2011) and Pegasus (Deelman et al., 2005). All these systems differ by the way the dependencies between jobs are encoded. DAGMan and Soma-workflow are both based on an explicit declaration of dependencies between jobs by users. The pipeline thus takes the form of a directed acyclic graph (DAG) with jobs as nodes and dependencies as (directed) edges. The Pegasus package also uses a DAG as input, yet this DAG is represented in an XML format called DAX. DAX graphs can be generated by any scripting language. By contrast, Nipype, Swift, and PSOM build on the notion of *futures* (Baker and Hewitt, 1977), i.e., a list of datasets (or variables) that will be generated by a job at run-time. The data-flow then implicitly defines the dependencies: all the inputs of a job have to exist before it can be started. An alternative to scripting approaches for pipeline composition is to rely on graphical abstractions. A number of projects offer sophisticated interfaces based on "box and arrow" graph representations, e.g., Kepler[10] (Ludäscher et al., 2006), Triana[11] (Harrison et al., 2008), Taverna[12] (Oinn et al., 2006), VisTrails[13] (Callahan et al., 2006), Galaxy (Goecks et al., 2010) and LONI pipeline[14] (Dinov et al., 2009). Because the graph representations can get really large, various mechanisms have been developed to keep the representation compact, such as encapsulation (the ability to represent a sub-pipeline as one box) and the use of control operations, e.g., iteration of a module over a grid of parameters, instead of a pure data-flow dependency system. Note that complex control mechanism are also necessary in systems geared toward data-flow dependencies to give the ability to, e.g., branch between pipelines or iterate a subpart of the pipeline until a data-dependent condition is satisfied. Finally, systems that put a strong emphasis on pipeline composition and re-use, such as Taverna, Nipype, and LONI pipeline, critically depend on the availability of a library of modules to build pipelines. Taverna claims to have over 3500 such modules, developed in a variety of domains such as bioinformatics or astronomy. Nipype and LONI both offer extensive application catalogue for neuroimaging analysis.

1.2. PIPELINE MAPPING

When a pipeline representation has been generated, it needs to be mapped onto available resources. For example, in grid computing, multiple production sites may be available, and a subset of sites where the pipeline will run has to be selected. This selection process can simply be a choice left to the user, e.g., Kepler, Taverna, VisTrails, Soma-workflow. It can also be automatically performed based on the availability and current workload at each registered production site, e.g., CBRAIN (Frisoni et al., 2011) and Pegasus, as well as quality of service issues. Another typical mapping task is the synchronization of the datasets across multiple data servers to the production site(s), an operation that can itself involve some interactions through web services with a database system, such as XNAT (Marcus et al., 2007) or LORIS (Das et al., 2012). The Pegasus project recompose pipelines at the mapping stage. This feature proceeds by grouping tasks in order to limit the over-head related to job submission and more generally optimize the pipeline for the infrastructure where it will be executed. Such mapping operation is central to achieve high performance in grid or cloud computing settings. Note that some pipeline systems have no or limited mapping capabilities. The PSOM project as well as matlabbatch, Nipype, and DAGMan for example were designed to work locally on the production server. The Soma-workflow can map pipelines in remote execution sites, but does not recompose the pipeline to optimize the performance of execution as Pegasus does. On the other end of the spectrum, CBRAIN is essentially a mapping/execution/provenance tool where pipelines have to be first composed in another system (such as PSOM).

1.3. PIPELINE EXECUTION

A dedicated execution engine is used to run the pipeline after mapping on computational resources. It will detect the degree of parallelism present in the pipeline at any given time, and process jobs in parallel depending on available computational resources. All pipeline systems reviewed here, including PSOM, can execute jobs in parallel on a multi-core machine or a supercomputer through submissions to a queuing mechanism such as SGE qsub, after a proper configuration has been set. Some of them (e.g., Taverna, Triana, Pegasus, CBRAIN) can also run jobs concurrently on one or multiple supercomputers in a computing grid, and are able to accommodate the variety of queuing mechanism found across production sites. Some execution engines, e.g., Nipype, will support a pipeline that builds dynamically, for example with a data-dependent branching in the pipeline. Fault tolerance is also an important feature. A first level of fault-tolerance is the notification of errors to the user, coupled with the ability to restart the pipeline where it stopped (e.g., PSOM, Nipype, Soma-workflow). The execution engine can also check that the expected output files have properly been generated (e.g., Pegasus, PSOM). In addition, after an error occurred, an execution engine may resubmit a job a number of times before considering that it has definitely failed (e.g., Swift, PSOM) because some random failures can occur due to, e.g., improper configuration, memory, or disk space exhaust on one execution node. An execution engine can also feature the ability to perform a "smart update," i.e., restart a pipeline while re-using the results from prior executions as much as possible (e.g., Kepler, Nipype, PSOM).

1.4. PIPELINE PROVENANCE

The final stage of a pipeline life cycle is provenance tracking, which represents the comprehensive recording of the processing steps applied to the datasets. This can also be extended to the archiving of the computing environment used for production (e.g., the version of the software that was used for processing), and the origin of the datasets that were used as inputs (MacKenzie-Graham et al., 2008). Provenance is a critical step to achieve reproducible research, which is itself considered as a

[8]http://research.cs.wisc.edu/condor/dagman/
[9]http://www.ci.uchicago.edu/swift/
[10]kepler-project.org
[11]http://www.trianacode.org/
[12]taverna.org.uk
[13]http://www.vistrails.org/
[14]http://pipeline.loni.ucla.edu/

cornerstone of the scientific method (Mesirov, 2010). A competition on provenance generation demonstrated that several pipeline systems captured similar informations (Bose et al., 2006). How these informations can be accessed easily and shared remains an area of development[15]. The quality of provenance tracking also depends on the quality of the interface between the pipeline system and the tools applied by each job: a comprehensive list of underlying parameters has to be generated before it is recorded. The PSOM development framework was notably designed to facilitate the systematic recording of the default job parameters as part of the provenance, in a way that scales well with the number of parameters. An innovative feature introduced by the VisTrails package is the capacity to graphically represent the changes made to a pipeline, not only providing a provenance mechanism for the pipeline execution but also for the steps of pipeline generation and/or variations in employed parameters.

1.5. PSOM FEATURES

The PSOM is a lightweight scripting solution for pipeline composition, execution, and provenance tracking. The package is intended for scientists who prototype new algorithms and pipelines using Octave or Matlab (O/M). PSOM is actively developed since 2008, and it has been inspired by several PERL pipeline systems (called RPPL, PCS, and PMP) used at the McConnell Brain Imaging Centre, Canada, over the past fifteen years (Zijdenbos et al., 1998). PSOM is based on a new standard to represent all steps of a pipeline analysis as a single O/M variable. This representation defines dependencies between processing steps implicitly by the data-flow. We established a limited number of scripting guidelines with the goal of maintaining a concise and modular code. These guidelines are suggestions rather than mandates, and the pipeline representation can be generated using any coding strategy. PSOM comes with a generic pipeline execution engine offering the following services:

1. Parallel computing: Automatic detection and execution of parallel components in the pipeline. The same code can run in a single matlab session, on a multi-core machine or on a distributed architecture with hundreds of execution nodes just by changing the PSOM configuration.
2. Provenance tracking: Generation of a comprehensive record of the pipeline stages and the history of execution. These records are detailed enough to fully reproduce an analysis, and profile the components of the pipeline.
3. Fault tolerance: Multiple attempts will be made to run each job before it is considered as failed. Failed jobs can be automatically re-started by the user after termination of the pipeline.
4. Smart updates: When an analysis is started multiple times, the parts of the pipeline that need to be reprocessed are automatically detected and those parts only are executed.

1.6. COMPARISON BETWEEN PSOM AND OTHER PACKAGES

As reviewed above, there are several alternatives with broader functionality than PSOM, such as LONI pipeline, VisTrails, Pegasus, Kepler, Triana, Galaxy, and Taverna. These systems notably support a graphical composition of the pipeline, database interfaces, and mapping capabilities. They, however, require users to write dedicated interfaces for importing computational modules. The DAGMan and Soma-workflow systems even leave to the user the task to generate the dependency graph of the pipeline using a third-party software, and concentrate mainly on the pipeline mapping, execution, and provenance. The aim of the PSOM project was to propose a single environment where computational modules and pipelines could be developed jointly. This is achieved by building a pipeline representation using native data structures of O/M. As our intended audience is developers, a graphical tool for pipeline composition was not a priority and is not currently available. PSOM also does not offer pipeline mapping capabilities because PSOM pipelines can be easily interfaced after the development phase with projects specifically focused on pipeline mapping, such as CBRAIN. By contrast, PSOM features powerful pipeline execution capabilities, in terms of fault tolerance and smart updates. Thanks to these features, users can modify, debug, or optimize the computational modules of a PSOM pipeline at the same time they are implementing (and testing) it.

The closest alternatives to PSOM are matlabbatch and Nipype. Both offer a simple scripting strategy to implement complex pipelines using data structures that are native to Matlab and Python, respectively. The pipeline composition is based on a set of dedicated scripting constructs, which may result in a highly concise code. Two projects have recently pursued this idea even further by adding coding constructs inspired by the Swift scripting language to Python, the PYDflow (Armstrong, 2011) package, and R, the SwiftR[16] package. PSOM pipelines are not as concise as the ones implemented with these systems, but they can be constructed with common O/M operations only. This choice was made to limit the learning curve for new users, who will hopefully find PSOM syntax very intuitive if they are already familiar with O/M. The distinctive features of PSOM are:

1. Minimally invasive: No new programming construct is introduced to script a pipeline.
2. Portable: PSOM is distributed under an MIT open-source license, granting the rights to modify, use and redistribute the code, free of charge, as part of any academic or commercial project. Moreover, the installation of PSOM is straightforward: it has no dependency and does not require compilation. Any system that supports Matlab or Octave (i.e., Linux, Windows, and Mac OS X) will run PSOM.
3. Dual O/M compatibility: PSOM users can benefit of the comfort of the Matlab environment for development purposes (graphical debugging tools, advanced profiling capabilities) and of the free open-source Octave interpreter to execute a code on hundreds of cores.

1.7. PAPER OUTLINE

The standard representation of a pipeline using a O/M variable is first presented in Section 2. Section 3 provides an overview of the key features of the execution engine on simple examples, while

[15] www.w3.org/2011/prov/

[16] http://people.cs.uchicago.edu/~tga/swiftR/

Section 4 details how these features were implemented. Section 5 provides further coding guidelines designed to keep the generation of pipelines concise, re-usable, and readable. Finally, Section 6 reviews some neuroinformatics projects that were implemented with PSOM. A preprocessing pipeline for functional magnetic resonance imaging (fMRI) was used for a benchmark evaluation of PSOM execution perfomance with several computing environments and execution configurations. The paper ends with a discussion of current PSOM capabilities and directions for future developments.

2. PIPELINE REPRESENTATION

A pipeline is a collection of jobs, which is implemented using the so-called O/M structure data type. The fields used in the `pipeline` are arbitrary, unique names for the jobs. Each job can have up to five fields, in which all but the first one are optional:

- `command`: (mandatory) a string describing the command that will be executed by the job.
- `files_in`: (optional) a list of input files.
- `files_out`: (optional) a list of output files.
- `files_clean`: (optional) a list of files that will be deleted by the job.
- `opt`: (optional) some arbitrary parameters.

The jobs are executed by PSOM in a protected environment where the only available variables are `files_in`, `files_out`, `files_clean`, and `opt`. The following code is a toy example of a simple pipeline:

```
% Job "sample" : No input, generate a
                  random vector a
command = 'a = randn([opt.nb_samps 1]);
               save(files_out,''a'')';
pipeline.sample.command      = command;
pipeline.sample.files_out    = 'sample.mat';
pipeline.sample.opt.nb_samps = 10;
% Job "quadratic" : Compute a.^2 and
                    save the results
command = 'load(files_in); b = a.^2;
           save(files_out,''b'')';
pipeline.quadratic.command   = command;
pipeline.quadratic.files_in  =
           pipeline.sample.files_out;
pipeline.quadratic.files_out =
           'quadratic.mat';
```

The first job, named `sample`, does not take any input file, and will generate one output file called `'sample.mat'`. It takes one parameter `nb_samps`, equals to 10. The field `opt` can be of any of the O/M data types. The second job, named `quadratic`, uses the output of `sample` as its input (`quadratic.files_in` is filled using `sample.files_out`). This convention avoids the generation of file names at multiple places in the script. It also makes explicit the dependence between `sample` and `quadratic` when reading the code: as the input of `quadratic` is the output of `sample`, `sample` has to be completed before `quadratic` can be started. This type of dependency between jobs, called "file-passing," is translated into a directed dependency graph, see **Figure 1A**. The dependency graph dictates the order of job execution. It can be represented using the following command:

```
psom_visu_dependencies(pipeline)
```

Let's now assume that the output of `sample` is regarded as an intermediate file that does not need to be retained. A new job `cleanup` is added to delete the output of `sample`, which is declared using the field `files_clean`:

```
% Adding a job "cleanup" : delete the
                    output of "sample"
pipeline.cleanup.command      =
          'delete(files_clean)';
pipeline.cleanup.files_clean  =
          pipeline.sample.files_out;
```

Because `cleanup` will delete the input file of `quadratic`, it is mandatory to wait until `quadratic` is successfully executed before `cleanup` is started. This type of dependency, called "cleanup", is again included as a directed link in the dependency graph, see **Figure 1B**.

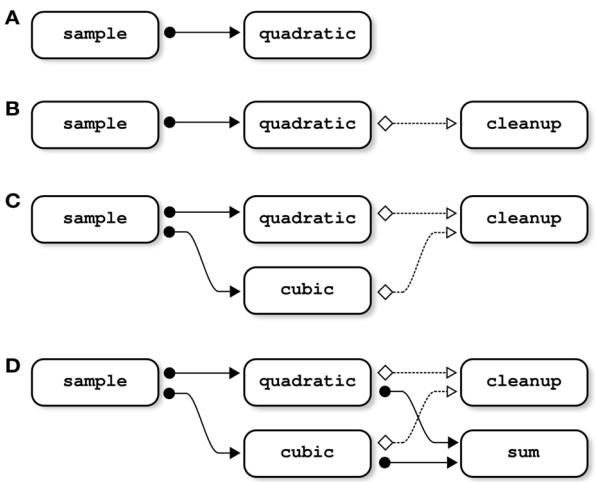


FIGURE 1 | Examples of dependency graphs. In panel **(A)**, the input file of the job `quadratic` is an output of the job `sample`; `sample` thus needs to be completed before starting `quadratic`. This type of dependency ("file-passing") can be represented as a directed dependency graph. In panel **(B)**, the job `cleanup` deletes an input file of `quadratic`; `quadratic` thus needs to be completed before starting `cleanup`. Note that such "cleanup" dependencies may involve more than two jobs: if `cleanup` deletes some input files used by both `quadratic` and `cubic`, `cleanup` depends on both of them (panel **C**). The same property holds for "file-passing" dependencies: if sum is using the outputs of both `quadratic` and `cubic`, sum depends on both jobs (panel **D**).

The order in which the jobs are added to the pipeline does not have any implications on the dependency graph, and is thus independent of the order of their execution. For example, if a new job `cubic` is added:

```
% Adding a job "cubic" : Compute a.^3 and
                         save the results
command = 'load(files_in);
          c = a.^3; save(files_out,''c'')';
pipeline.cubic.command     =  command;
pipeline.cubic.files_in    =
              pipeline.sample.files_out;
pipeline.cubic.files_out   =
              'cubic.mat';
```

the job `cleanup` will be dependent upon `quadratic` and `cubic`, because the latter jobs are using the output of `sample` as an input, a file that is deleted by `cleanup` (**Figure 1C**).

The type of `files_in`, `files_out`, and `files_clean` is highly flexible. It can be a string, a cell of strings, or a nested structure whose terminal fields are strings or cells of strings. The following job for example uses two inputs, generated by two different jobs (see **Figure 1D**):

```
% Adding a job "sum" : Compute a.^2+a.^3
                      and save the results
command = 'load(files_in{1});
        load(files_in{2}); d = b+c, ...
        save(files_out,''d'')';
pipeline.sum.command      = command;
pipeline.sum.files_in{1} =
          pipeline.quadratic.files_out;
pipeline.sum.files_in{2} =
          pipeline.cubic.files_out;
pipeline.sum.files_out    = 'sum.mat';
```

3. PIPELINE EXECUTION

3.1. A FIRST PASS THROUGH THE TOY PIPELINE

When a `pipeline` structure has been generated by the user, PSOM offers a generic command to execute the pipeline:

```
psom_run_pipeline(pipeline,opt_pipe)
```

where `opt_pipe` is a structure of options that can be used to set the configuration of PSOM, see Section 4.6. The main configuration option is the name of a folder used to store the logs of the pipeline, which is the "memory" of the pipeline system. When invoked, PSOM first determines which jobs need to be restarted using the logs folder. The jobs are then executed in independent sessions, as soon as all their dependencies are satisfied. The next section (Section 4) describes the implementation of all stages of pipeline execution in details. This section outlines the key mechanisms using simple examples, starting with the toy pipeline presented in the last section without the cleanup job (see **Figure 2**). Initially, only one job (`sample`) can be started because it does not have any parent in the dependency graph (**Figure 2A**). As soon as this job has been successfully completed, its two children (`quadratic` and `cubic`) are started. This is assuming of course that the configuration allows PSOM to execute at least two jobs in parallel (e.g., background execution on a dual-core machine), see **Figure 2B**. The job `sum` is started only when both of its dependencies have been satisfied, see **Figures 2C,D**. When all jobs are completed, the pipeline manager finally exits (**Figure 2E**).

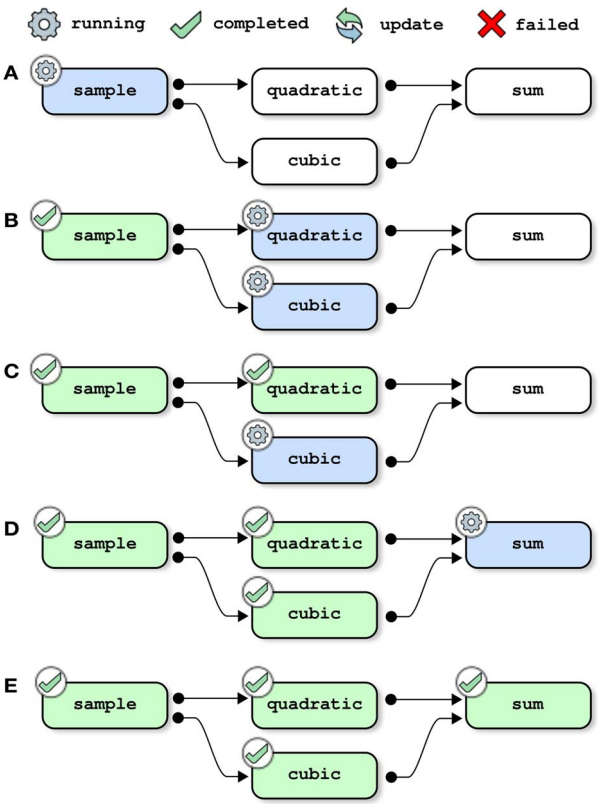


FIGURE 2 | Pipeline execution: a first pass through the toy pipeline. Each panel represents one step in the execution of the toy pipeline presented in Section 2, without the `cleanup` job. This example assumes that at least two jobs can run in parallel, and that the pipeline was not executed before. All jobs are executed as soon as all of their dependencies are satisfied, possibly with some jobs running in parallel.

3.2. UPDATING A PIPELINE (WITH A BUG)

This next example shows how the pipeline manager deals with the update of a pipeline. That is to say that a pipeline is submitted for execution after it was previously executed using the same logs folder. If one of the jobs has changed since the last submission, this job along with all of its children in the dependency graph are scheduled to be reprocessed. Here, the job `quadratic` is modified to introduce a bug, before restarting the pipeline:

```
% Changing the job quadratic to
  introduce a bug
pipeline.quadratic.command = 'BUG!';
% Restart the pipeline
psom_run_pipeline(pipeline,opt_pipe)
```

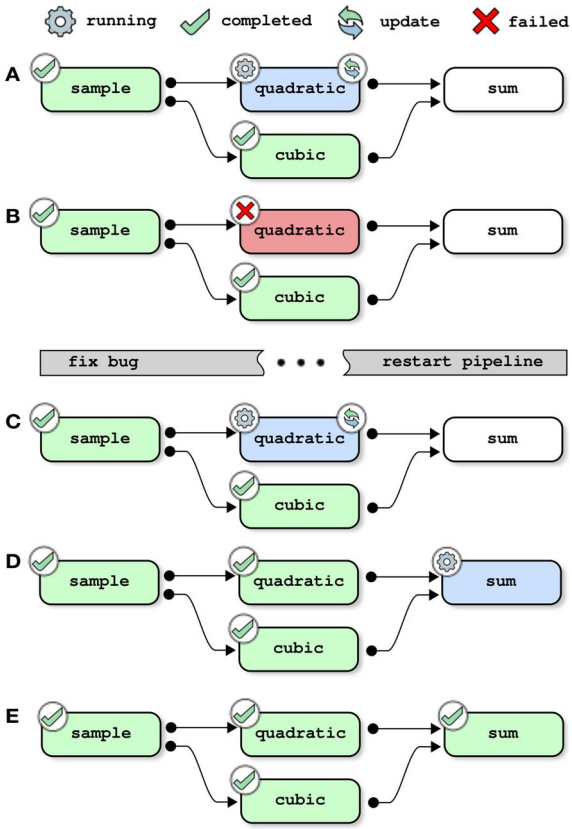


FIGURE 3 | Pipeline management, example 2: updating a pipeline (with one bug). Each panel represents one step in the execution of the toy pipeline presented in Section 2, without the cleanup job. This example assumes that at least two jobs can run in parallel, and that the pipeline has already been executed once as outlined in **Figure 2**. The pipeline is first started after changing the job `quadratic` to introduce a bug (panels **A–B**). When the execution of the pipeline fails, the job `quadratic` is modified to fix the bug. The pipeline is then restarted and completes successfully (panels **C–E**).

The pipeline manager first restarts the job `quadratic` because a change is detected in its description (**Figure 3A**). After the execution of the job is completed, the job is tagged with a "failed" status (panel **B**). The job `sum` is not started because it has a dependency that cannot be solved, and the pipeline manager simply exits. It is then possible to access the logs of the failed job, i.e., a text description of the job, start time, user name, system used as well as end time and all text outputs:

```
>> psom_pipeline_visu
    (opt.path_logs,'log','quadratic');
**********************************
Log of the (octave) job : quadratic
Started on 19-Jul-2011 16:01:36
User: pbellec
host : sorbier
system : unix
**********************************
command      = BUG!
files_in     = /home/pbellec/database/
               demo_psom/sample.mat
files_out    = /home/pbellec/database/
               demo_psom/quadratic.mat
files_clean  = {}(0x0)
opt          = {}(0x0)
********************
The job starts now !
********************
Something went bad ... the job has FAILED !
The last error message occured was :
parse error:
  syntax error
>>> BUG!
File /home/pbellec/svn/psom/trunk/
      psom_run_job.m at line 110
****************
Checking outputs
****************
The output file or directory ...
   /home/pbellec/database/demo_psom/
  quadratic.mat has not been generated!

****************************************
19-Jul-2011 16:01:36 : The job has FAILED
Total time used to process the
          job : 0.00 sec.
****************************************
```

The pipeline is then modified to fix the bug in `quadratic`. After restarting the pipeline, the jobs `quadratic` and `sum` run sequentially and are successfully completed (**Figures 3C–E**).

3.3. ADDING A JOB

Updating the pipeline is not solely restricted to changing the description of a job that was previously a part of the pipeline. It is also possible to add new jobs and resubmit the pipeline. **Figure 4** shows the steps of resolution of the full toy pipeline (including the cleanup job) when the subpipeline (not including the cleanup pipeline) had already been successfully completed prior to submission. In that case, there is no job that depends on the outputs of `cleanup`, so the only job that needs to be processed is `cleanup` itself and the pipeline is successfully completed immediately after this job is finished.

3.4. RESTARTING A JOB AFTER CLEAN UP

It is sometimes useful to force a job to restart, for example a job that executes a modified script while the job description remains identical. PSOM is not able to detect this type of change in the pipeline (it assumes that all libraries are identical across multiple runs of the pipeline). The following option will force a job to restart:

```
opt_pipe.restart = {'quadratic'};
psom_run_pipeline(pipeline,opt_pipe);
```

In this example, all jobs whose name includes `quadratic` will be restarted by the pipeline manager. Further we will assume that the full toy pipeline (including the `cleanup` job) has already

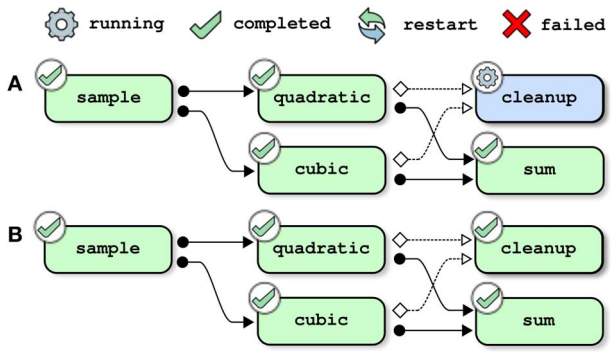


FIGURE 4 | Pipeline management, example 3: adding a (cleanup) job. This example assumes that the toy pipeline (without the `cleanup` job) had already been successfully completed. The full toy pipeline (with the `cleanup` job) is then submitted for execution. The only job that is not yet processed is `cleanup`, and the pipeline execution ends after `cleanup` successfully completes.

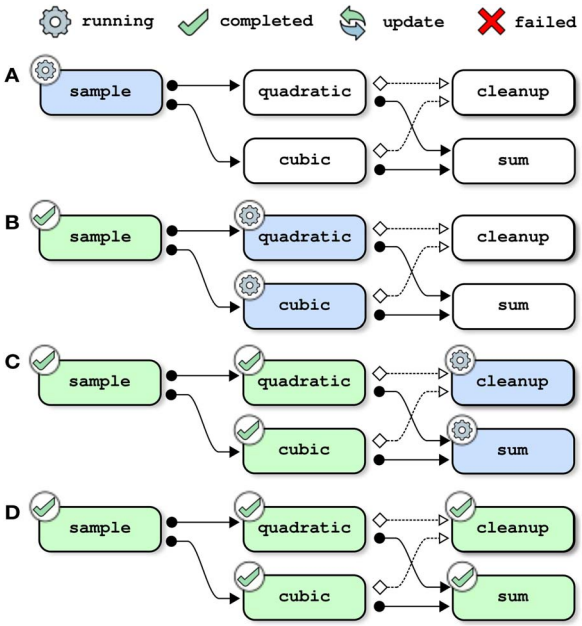


FIGURE 5 | Pipeline management, example 4: restarting a job after its inputs have been cleaned up. This example assumes that the full toy pipeline (including the `cleanup` job) has already been successfully completed. The same pipeline is then submitted for a new run and the job `quadratic` is forced to be restarted. Because the inputs of `quadratic` (generated by `sample`) have been deleted by `cleanup`, the pipeline manager also restarts the job `sample` (panel **A**). Because all jobs depend indirectly on sample, all jobs in the pipeline have to be reprocessed (panels **B–D**).

been completed. In the absence of the `cleanup` job, the job `quadratic` would be restarted as well as all of its children. The inputs of `quadratic`, however, have been deleted by `cleanup`. It is therefore, not possible to restart the pipeline at this stage. The pipeline manager will automatically detect that the missing inputs can be re-generated by restarting the job `sample`. It will thus restart this job as well as all of its children, including `cubic` (see **Figure 5** for a step-by-step resolution of the pipeline). Note that this behavior is iterative, such that if some inputs from `sample` had been missing, the pipeline manager would look for jobs that could be restarted to generate those files.

3.5. PIPELINE HISTORY

When PSOM is solving a pipeline, it is not generating a color-coded graph such as those presented in **Figures 2–5**. Rather, it outputs a text summary of all operations, such as job submission, job completion, and job failure. Each event is reported along with the time of its occurrence. This is presented in the following example for the first execution of the toy pipeline (**Figure 2**):

```
****************************************
The pipeline PIPE is now being processed.
Started on 21-Jul-2011 09:37:45
user: pbellec, host: berry, system: unix
****************************************
21-Jul-2011 09:37:45 -
...The job sample has been submitted to the
   queue (1 jobs in queue).
21-Jul-2011 09:37:48 -
...The job sample has been successfully
   completed (0 jobs in queue).
21-Jul-2011 09:37:48 -
...The job quadratic has been submitted to
   the queue (1 jobs in queue).
21-Jul-2011 09:37:48 -
...The job cubic has been submitted to the
   queue (2 jobs in queue).
21-Jul-2011 09:37:52 -
...The job quadratic has been successfully
   completed (1 jobs in queue).
21-Jul-2011 09:37:52 -
...The job cubic has been successfully
   completed (0 jobs in queue).
21-Jul-2011 09:37:52 -
...The job sum has been submitted to the
   queue (1 jobs in queue).
21-Jul-2011 09:37:55 -
...The job sum has been successfully
   completed (0 jobs in queue).
****************************************
The processing of the pipeline is
terminated.
See report below for job completion status.
21-Jul-2011 09:37:55
****************************************
All jobs have been successfully completed.
```

These logs are concatenated across all instances of pipeline executions, and they are saved in the logs folder. They can be accessed using a dedicated M-command:

```
psom_pipeline_visu
   (opt_pipe.path_logs,'monitor')
```

The logs of individual jobs can also be accessed with the same command, using a different option:

```
psom_pipeline_visu
    (opt_pipe.path_logs,'log',JOB_NAME)
```

as shown in Section 3.2. Finally, it is possible to get access to the execution time for all jobs from the pipeline, which can be useful for benchmarking purposes:

```
>> psom_pipeline_visu
    (opt_pipe.path_logs,'time','')
**********
cleanup    : 0.07 s, 0.00 mn,
             0.00 hours, 0.00 days.
cubic      : 0.07 s, 0.00 mn,
             0.00 hours, 0.00 days.
quadratic  : 0.08 s, 0.00 mn,
             0.00 hours, 0.00 days.
sample     : 0.13 s, 0.00 mn,
             0.00 hours, 0.00 days.
sum        : 0.11 s, 0.00 mn,
             0.00 hours, 0.00 days.
**********
Total computation time :  0.46 s, 0.01 mn,
                    0.00 hours, 0.00 days.
```

4. IMPLEMENTATION OF THE PIPELINE EXECUTION ENGINE

4.1. OVERVIEW

At the user level, PSOM requires two objects to be specified: (1) a `pipeline` structure which describes the jobs, see Section 2; (2) an `opt_pipe` structure which configures how the jobs will be executed, see Section 4.6. The configuration notably includes the name of a so-called logs folder, where a comprehensive record of the pipeline execution is kept. The pipeline execution itself is initiated by a call to the function `psom_run_pipeline`, which comprises three distinct modules:

1. The **initialization stage** starts off with basic viability checks. If the same logs folder is used multiple times, the current pipeline is compared against older records. This determines which jobs need to be (re)started.
2. When the initialization stage is finished, a process called the **pipeline manager** is started. The pipeline manager remains active as long as the pipeline is running. Its role is to create small scripts to run individual jobs, and then submit those scripts for execution as soon as their dependencies are satisfied and sufficient resources, as determined by the configuration, become available.
3. Each job is executed in an independent session by a **job manager**. Upon termination of the job, the completion status ("failed" or "finished") is checked and reported to the pipeline manager using a "tag file" mechanism.

This section describes the implementation of each module, as well as the configuration of PSOM and the content of the logs folder. An overview is presented in **Figure 6**.

4.2. PIPELINE INITIALIZATION

The initialization of pipeline execution includes the following steps:

1. Check that the (directed) dependency graph of the pipeline is acyclic. A dependency graph that includes a cycle is impossible to solve.
2. Check that all of the output files are generated only once (otherwise the results of the pipeline may depend on an arbitrary order of job executions).
3. If available, retrieve the history of previous pipeline executions. Determine which jobs need to be processed based on the history. Update the pipeline history accordingly. This step will be further detailed below.
4. Check that all of the input files that are not generated as part of the pipeline are present on the disk. If not, issue a warning because some jobs may fail when input files are missing. This, however, depends on the behavior of the commands specified by the user and cannot be tested by PSOM. The decision to continue is thus left to the user who may decide to interrupt the execution at this stage.
5. Create all the necessary folders for output files. This feature circumvents the repetitive task of coding the creation of the output folder(s) inside each individual job.
6. If some of the output files already exist, delete them. This step is intended to avoid possible errors in the pipeline execution due to some jobs not overwriting the output files.

To determine what jobs from the pipeline actually need to be processed, the jobs submitted for execution are compared with those previously executed in the same logs folder (if any), along with their completion status. There are three possible status results:

- `'none'` means that the job has never been started (this is the default if no previous status exists).
- `'finished'` means that the job was previously executed and successfully completed.
- `'failed'` means that the job was previously executed and had failed.

A job will be added to the "to-do list" (i.e., will be executed by the pipeline manager) if it meets one of the following conditions:

- the job has a `'failed'` status.
- the job has a `'none'` status.
- the description of the job has changed.
- the user forced a restart of the job using `opt_pipe.restart`. See Section 3.4.

Every time a job `A` is added to the to-do list, the following actions are taken:

- Change the status of the job `A` to `'none'`.
- Add all jobs with a dependency on `A` to the "to-do list".
- If an input file of `A` is missing and a job of the pipeline can generate this file, add this last job to the "to-do list".

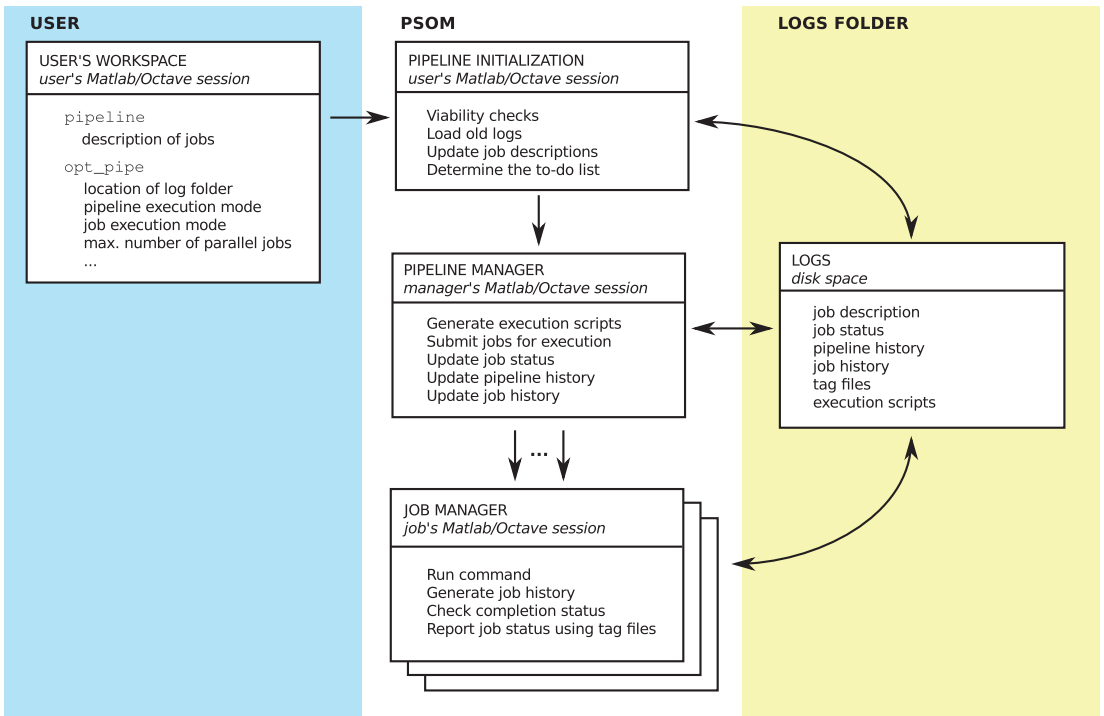


FIGURE 6 | Overview of the PSOM implementation. On the user's side (left panel), a structure `pipeline` is built to describe the list of jobs, and a structure `opt_pipe` is used to configure PSOM. The memory of the pipeline is a logs folder located on the disk space (right panel), in which a series of files are stored to provide a comprehensive record of multiple runs of pipeline execution. The PSOM proceeds in three stages (center panel). At the initialization stage, the current pipeline is compared with previous executions to set up a "to-do" list of jobs that needs to be (re)started. Then, the pipeline manager is started, which constantly submits jobs for execution and monitors the status of on-going jobs. Finally, each job is executed independently by a job manager which reports the completion status upon termination (either "failed" or "finished").

Note that the process of adding a job to the to-do list is recursive and it can lead to restarting a job with a `'finished'` status, e.g., if that job has changed or if it is dependent on a job that has changed.

4.3. PIPELINE MANAGER

After the pipeline has been initialized, a small process called the "pipeline manager" is started. The pipeline manager is essentially a long loop that constantly monitors the state of the pipeline, and submits jobs for execution. The pipeline manager as well as the individual jobs can run within the current O/M session, or in an independent session running either locally (on the same machine) or remotely (on another computer/node). At any given point in time, the pipeline manager submits all of the jobs that do not have an unsatisfied dependency, as long as there are enough resources available to process the jobs. The following rules apply to determine if the dependencies of a job are satisfied:

1. If a job has been successfully completed, the dependencies to all the children in the dependency graph are considered satisfied.
2. Conversely, the dependencies of a job are all satisfied if there are no dependencies in the first place or if the parents in the dependency graph all have a `'finished'` status.

Depending on the selected configuration, there may also be a limit to the maximal number of jobs that can be submitted for execution simultaneously. This was implemented because some high-performance computing facilities impose such a limit. Upon completion or failure, the jobs report their status using tag files located in the **logs** folder. A tag file is an empty file with a name of the form `JOB_NAME.failed` or `JOB_NAME.finished`, which indicates the completion status. If the pipeline system was fully based on tag files to store status, a pipeline with thousands of jobs would create thousands of tag files. This would cause very important delays when accessing the file system. The pipeline manager thus monitors these tag files and removes them as soon as their presence is detected. The tag files are used to update a single O/M file coding for the status of all jobs in the pipeline. As the tag files are empty files, there is no possible race condition between their creation and their subsequent removal by the pipeline manager. The pipeline manager also adds status updates in a plain text "history" file which can be monitored while being updated in a terminal or from O/M through the dedicated command `psom_pipeline_visu`.

4.4. JOB MANAGER

When a job is submitted for execution by the pipeline manager, the command specified by the user is always executed by a generic job manager. The job manager is a matlab function (`psom_run_job`) which automates the generation of a job profile, logs, as well as the tag files that are used to report the

completion status to the pipeline manager. This function notably executes the command in a `try ... catch` block, which means that errors in the command will not crash the job manager. When the command has finished to run, the job manager will check that all of the output files have been properly generated. If an error occurs, or if one of the output files is missing, then the job is marked as `'failed'`. Otherwise it is considered `'finished'`. The job manager reports back the completion status of the job to the pipeline manager using a tag file mechanism already described in Section 4.3. The job manager also automatically generates logs, i.e., a text record of the execution of the command, as well as other automatically generated informations such as the user name, the date, the time, and the type of operating system, see Section 3.5 for an example. Finally, the job manager measures and saves the execution time of the command for profiling purposes.

4.5. LOGS FOLDER

The logs folder contain the following files:

- `PIPE_history.txt`: A plain text file with the history of the execution of the pipeline manager (see Section 3.5 for an example).
- `PIPE_jobs.mat`: An O/M file were each job is saved as a variable. This structure includes the latest version of all jobs executed from the logs folder.
- `PIPE_status.mat`: An O/M file where the status of each job is saved as one (string) variable.
- `PIPE_logs.mat`: An O/M file where the logs of each job is saved as one (string) variable.
- `PIPE_profile.mat`: An O/M file where each job appears as a variable. This variable is an O/M structure, notably including the execution time of the command.
- `PIPE.mat`: An O/M file where PSOM configuration variables are saved.

Importantly, using `PIPE_jobs.mat`, it is possible to re-execute the pipeline from scratch at any point in time, or to access any of the parameters that were used for the analysis. The logs folder thus contains enough information to fully reproduce the results of the pipeline. Moreover, with this information being stored in the form of an M-structure, it is easy to access and fully scalable. This can support jobs with potentially hundreds or even thousands of parameters. Octave and Matlab both use the HDF5 file format (Poinot, 2010). This format offers internal compression, yet still allows PSOM to read or write individual variables without accessing the rest of the file. This is a key technical feature that enables PSOM to quickly update the logs/status/profile files for each job, regardless of the size of the pipeline. Note that the logs folder also contain other files generated temporarily as part of the pipeline submission/execution process, as well as backup files in the event the main files are corrupted.

4.6. PSOM CONFIGURATION

The only necessary option to start a pipeline is setting where to store the logs folder:

```
>> opt_pipe.path_logs =
 '/home/pbellec/database/demo_psom/logs/';
```

It is highly recommended that the logs folder be used solely for the purposes of storing the history of the pipeline. Another important, yet optional parameter is setting how the individual jobs of the pipeline are executed:

```
>> opt_pipe.mode = 'batch';
```

Five execution modes are available:

- `'session'`: The jobs are executed in the current O/M session, one after the other.
- `'background'`: This is the default. Each job is executed in the background as an independent O/M session, using an "asynchronous" system call. If the user's session is interrupted, the pipeline manager and the jobs are interrupted as well.
- `'batch'`: Each job is executed in the background as an independent O/M session, using the `at` command on Linux and the `start` command on windows. If the user's session is interrupted, the pipeline manager and the jobs are not interrupted. This mode is less robust than `background` and may not be available on some platforms.
- `'qsub'`: The jobs are executed on a remote execution server through independent submissions to a queuing scheduler using a `qsub` command (either torque, SGE, or PBS). Such queuing schedulers are in general avalaible in high-performance computing facilities. They need to be installed and configured by a system administrator.
- `'msub'`: The jobs are executed on a remote execution server through independent submissions to a a queuing scheduler using a `msub` command (MOAB). This is essentially equivalent to the `qsub` mode.

Additional options are available to control the bash environment variables, as well as O/M start-up options, among others. A function called `psom_config` can be used to assess whether the configuration of PSOM is correct. This procedure includes multiple tests to validate that each stage of a job submission is working properly. It will provide some environment-specific suggestions to fix the configuration when a problem is detected. PSOM release 0.9 has been tested in a variety of platforms (Linux, windows, Mac OSX) and execution modes. More details can be found in PSOM online resources, see the discussion section for links.

5. CODING GUIDELINES FOR MODULES AND PIPELINES

The pipeline structure that is used in PSOM is very flexible, as it does not impose any constraints on the way the code executed by each job is implemented or on the way the pipeline structure itself is generated. Additional coding guidelines and tools have been developed to keep the code concise and scalable, in the sense that it can be used to deal with functions with tens or hundreds of parameters and thousands of jobs. These guidelines also facilitate the combination of multiple pipelines while keeping track of all parameters: a critical feature to ensure full

provenance of a pipeline analysis. A generic tool is available to test the presence of mandatory parameters and set up default parameter values. Another tool is the so-called "brick" function type, which can be used to run jobs. A last set of guidelines and tools have been developed to generate the pipeline structures themselves.

5.1. SETTING THE JOB PARAMETERS

There is no strict framework to set the default of the input arguments in Octave/Matlab. We developed our own guidelines, which have several advantages over a more traditional method consisting in passing each parameter one by one. As can be seen in the attributes of a job, our method consists of passing all parameters as fields of a single structure `opt`. A generic function `psom_struct_defaults` can be used to check for the presence of mandatory input arguments, set default values, and issue warnings for unkown arguments. The following example shows how to set the input arguments of a function using that approach:

```
opt.order     = [1 3 5 2 4 6];
opt.slic      = 1;
opt.timing    = [0.2,0.2];
list_fields   = { 'method' , 'order' ,
          'slice' , 'timing' , 'verb' };
list_defaults = { 'linear' , NaN , [] ,
                  NaN , true };
opt = psom_struct_defaults
      (opt,list_fields,list_defaults)
warning: The following field(s) were
    ignored in the structure :  slic
opt = {
  method = linear
  order  =  [1   3   5   2   4   6]
  slice  = [](0x0)
  timing = [0.20000   0.20000]
  verb   =  1 }
```

Note that only three lines of code are used to set all the defaults, and that a warning was automatically issued for the typo `slic` instead of `slice`. Such unlisted fields are simply ignored. Also, the default value `NaN` can be used to indicate a mandatory argument (an error will be issued if this field is absent). This approach will scale up well with a large number of parameters. It also facilitates the addition of extra parameters in future developments while maintaining backwards compatibility. As long as a new parameter is optional, a code written for old specifications will remain functional.

5.2. BUILDING MODULES FOR A PIPELINE : THE "BRICK" FUNCTION TYPE

The bricks are a special type of O/M function which take files as inputs and outputs, along with a structure to describe some options. In brief, a brick precisely mimics the structure of a job in a pipeline, except for the `files_clean` field. The command used to call a brick always follows the same syntax:

```
[files_in,files_out,opt] =
  brick_name(files_in,files_out,opt)
```

where `files_in`, `files_out` and `opt` play the same roles as the fields of a job. The key mechanism of a brick is that there will always be an option called `opt.flag_test` which allows the programmer to make a test, or dry-run. If that (boolean) option is true, the brick will not do anything but update the default parameters and file names in its three arguments. Using this mechanism, it is possible to use the brick itself to generate an exhaustive list of the brick parameters, and test if a subset of parameters are acceptable to run the brick. In addition, if a change is made to the default parameters of a brick, this change will be apparent to any piece of code that is using a test to set the parameters, without a need to change the code.

When the file names `files_in` or `files_out` are structures, a missing field will be interpreted either as a missing input which can be replaced by a default dataset, or an output that does not need to be generated. If the field is present but empty, then a default file name is generated. Note that an option `opt.folder_out` can generally be used to specify in which folder the default outputs should be generated. Finally, if a field is present and non-empty, the file names specified by the users are used to generate the outputs. These conventions allow complete control over the number of output files generated by the brick, and the flexibility to use default names. The following example is a dry-run with a real brick implemented in the neuroimaging analysis kit[17] (NIAK) (Bellec et al., 2011):

```
files_in  =
  '/database/func_motor_subject1.mnc';
files_out.filtered_data = '';
files_out.var_low = '';
opt.hp = 0.01;
opt.folder_out = '/database/filtered_data/';
opt.flag_test = true;
>>[files_in,files_out,opt] = ... niak_brick
        _time_filter(files_in,files_out,opt)
files_in  =
     /database/func_motor_subject1.mnc
files_out =
{
  filtered_data = /database/filtered_data/
              /func_motor_subject1_f.mnc
  var_high      = gb_niak_omitted
  var_low       = /database/filtered
     _data//func_motor_subject1_var_low.mnc
  beta_high     = gb_niak_omitted
  beta_low      = gb_niak_omitted
  dc_high       = gb_niak_omitted
  dc_low        = gb_niak_omitted
}
opt =
{
  hp            =  0.010000
  folder_out    = /database/filtered_data/
  flag_test     =  1
  flag_mean     =  1
```

[17] code.google.com/p/niak

```
  flag_verbose   =  1
  tr             = -Inf
  lp             = Inf
}
```

The default output names have been generated in `opt.folder_out`, and some of the outputs will not be generated (they are associated with the special tag `'gb_niak_omitted'`). A large number of other parameters that were not used in the call have been assigned some default values.

5.3. PIPELINE IMPLEMENTATION

A so-called pipeline generator is a function that, starting from a minimal description of a file collection and some options, generates a full pipeline. Because a pipeline can potentially create a very large number of outputs, it is difficult to implement a generic system that is as flexible as a brick in terms of output selection. Instead, the organization of the output of the pipeline will follow some canonical, well-structured pre-defined organization. As a consequence, the pipeline generator only takes two input arguments, `files_in` and `opt` (similar to those of a job), and does not feature `files_out`. The following example shows how to set `files_in` for `niak_pipeline_corsica`, implemented in NIAK:

```
%% Subject 1
files_in.subject1.fmri{1} =
   '/demo_niak/func_motor_subject1.mnc';
files_in.subject1.fmri{2} =
   '/demo_niak/func_rest_subject1.mnc';
files_in.subject1.transf  =
   '/demo_niak/transf_subject1.xfm';

%% Subject 2
files_in.subject2.fmri{1} =
   '/demo_niak/func_motor_subject2.mnc';
files_in.subject2.fmri{2} =
   '/demo_niak/func_rest_subject2.mnc';
files_in.subject2.transf  =
   '/demo_niak/transf_subject2.xfm';
```

The argument `opt` will include the following standard fields:

- `opt.folder_out`: Name of the folder where the outputs of the pipeline will be generated (possibly organized into subfolders).
- `opt.size_output`: This parameter can be used to vary the amount of outputs generated by the pipeline (e.g., `'all'`: generate all possible outputs; `'minimum'`, clean all intermediate outputs, etc).
- `opt.brick1`: All the parameters of the first brick used in the pipeline.
- `opt.brick2`: All the parameters of the second brick used in the pipeline.
- ...

Inside the code of the pipeline template, adding a job to the pipeline will typically involve a loop similar to the following example:

```
% Initialize the pipeline to a structure
  with no field
pipeline = struct();
% Get the list of subjects from files_in
list_subject = fieldnames(files_in);
% Loop over subjects
for num_s = 1:length(list_subject)
    % Plug the 'fmri' input files of the
      subjects in the job
    job_in  = files_in.
              (list_subject{num_s}).fmri;
    % Use the default output name
    job_out = '';
    % Force a specific folder organization
      for outputs
    opt.fmri.folder_out = [opt.folder_out
            list_subject{num_s} filesep];
    % Give a name to the jobs
    job_name =
       ['fmri_' list_subject{num_s}];
    % The name of the employed brick
    brick = 'brick_fmri';
    % Add the job to the pipeline
    pipeline = ... psom_add_job(pipeline,
    job_name,brick,job_in,job_out,opt.fmri);
    % The outputs of this brick are just
      intermediate outputs :
    % clean these up as soon as possible
    pipeline = psom_add_clean(pipeline,
           [job_name ... '_clean'],pipeline.
           (job_name).files_out);
end
```

The command `psom_add_job` first runs a test with the brick to update the default parameters and file names, and then adds the job with the updated input/output files and options. By virtue of the "test" mechanism, the brick is itself defining all the defaults. The coder of the pipeline does not actually need to know which parameters are used by the brick. Any modification made to a brick will immediately propagate to all pipelines, without changing one line in the pipeline generator. Moreover, if a mandatory parameter has been omitted by the user, or if a parameter name is not correct, an appropriate error or warning will be generated at this stage, prior to any work actually being performed by the brick. The command `psom_add_clean` adds a cleanup job to the pipeline, which deletes the specified list of files. Because the jobs can be specified in any order, it is possible to add a job and its associated cleanup at the same time. Finally, it is very simple to combine pipelines together: the command `psom_merge_pipeline` simply combines the fields of two structures `pipeline1` and `pipeline2`.

6. APPLICATIONS IN NEUROIMAGING

The PSOM project is just reaching the end of its beta testing phase, and as such it has only been adopted by a couple of laboratories as a development framework. There are still been several successful applications, including the generation of simulated fMRI (Bellec et al., 2009), clustering in resting-state fMRI (Bellec et al., 2010a,b), clustering in event-related fMRI (Orban et al., 2011), simulations in electroencephalography and optical imaging (Machado et al., 2011), reconstruction of fiber tracts (Kassis et al., 2011), as well as non-parametric permutation testing (Ganjavi et al., 2011). The PSOM framework has also been used for the development of an open-source software package called NIAK[18] (Bellec et al., 2011). This software package, which relies on the PSOM execution engine, has been used in a number of recent studies (Dansereau et al., 2011; Moeller et al., 2011; Schoemaker et al., 2011; Carbonell et al., 2012). We used the fMRI preprocessing pipeline from the NIAK package to run benchmarks of the parallelization efficiency of the PSOM execution engine. This pipeline has been integrated into the CBRAIN computing platform (Frisoni et al., 2011), where it has been used to preprocess and publicly release[19] fMRI datasets collected for about 1000 children and adolescents, as part of the ADHD-200 initiative[20] (Lavoie-Courchesne et al., 2012).

6.1. THE NIAK FMRI PREPROCESSING PIPELINE

The NIAK fMRI preprocessing pipeline applies the following operations to each functional and structural dataset in a database. The first 10 s of the acquisition are suppressed to allow the magnetization to reach equilibrium. The fMRI volumes are then corrected of inter-slice difference in acquisition time, rigid body motion, slow time drifts, and physiological noise (Perlbarg et al., 2007). For each subject, the mean motion-corrected volume of all the datasets is coregistered with a T1 individual scan using minctracc (Collins et al., 1994), which is itself non-linearly transformed to the Montreal Neurological Institute (MNI) non-linear template (Fonov et al., 2011) using the CIVET pipeline (Ad-Dab'bagh et al., 2006). The functional volumes are then re-sampled in the stereotaxic space and spatially smoothed.

Most operations are implemented through generic medical image processing modules, the MINC tools[21]. These tools are coded in a mixture of C and C++ languages, as well as some PERL scripts, and usually operate through the command line. Simple PSOM-compliant "brick" wrappers have been implemented in NIAK for the required MINC tools. Other bricks are also pure O/M implementations for original methods or a port from other O/M projects. Finally, some of the operations (motion correction, correction of physiological noise) are themselves pipelines involving several steps, see **Figure 7** for an example of a full dependency graph. The code of the individual NIAK fMRI preprocessing pipeline is 735 lines long, and only 321 lines after excluding header comments and variable initialization. The code is thus concise enough to be easily reviewed, quality-checked, and modified.

6.2. BENCHMARKS

We used the `Cambridge` resting-state fMRI database for the benchmark, which is publicly available as part of the 1000 functional connectome project[22]. This database (Liu et al., 2009) includes 198 subjects with one structural MRI and one fMRI run each (119 volumes, TR = 3 s). The processing was done in various computing environments and execution modes to test the scalability of PSOM:

- `peuplier-n`: A machine with an Intel(R) Core™ i7 CPU (four computing cores, eight threads), 16 GB of memory, a local file system and an Ubuntu operating system. For n = 1, both the pipeline manager and individual jobs were executed sequentially in a single Octave session. For n > 1, the pipeline manager and individual jobs were executed in the background in independent Octave sessions using an `at` command, with up to `n` jobs running in parallel.
- `magma-n`: a machine with four six-Core AMD Opteron™ Processor 8431 (for a total of 24 computing cores), 64 GB of memory, an NTFS mounted file system and an openSUSE operating system. For n = 1, both the pipeline manager and individual jobs were executed sequentially in a single Octave session. For n > 1, the pipeline manager ran in the background using an `at` command and individual jobs were executed in the background in independent Octave sessions using an SGE `qsub` command, with up to `n` jobs running in parallel.
- `guillimin-n`: a supercomputer with 14400 Intel Westmere-EP cores distributed across 1200 compute nodes located at the CLUMEQ-McGill data centre, Ecole de Technologie Superieure in Montreal, Canada. `guillimin` ranked 83th in the top 500 list of the most powerful supercomputers, released in November, 2011[23]. Included in the facility is nearly 2 PB of disk storage using the general parallel file system (GPFS). For n = 1, both the pipeline manager and individual jobs were executed sequentially in a single Octave session. For n > 1, the pipeline manager ran in the background using an `at` command and individual jobs were executed on distributed computing nodes in independent Octave sessions using a MOAB `msub` command, with up to `n` jobs running in parallel.

We investigated the performance of PSOM on (`peuplier-8`, `magma-{8,16,24,40}` and `guillimin-{24,50,100,200}`). For experiments on `peuplier` and `magma`, Octave release 3.2.4 was used, with PSOM release 0.8.9 and NIAK release 0.6.4.3. On `guillimin`, octave release 3.4.2 was available and some development versions of NIAK (v1270 on the subversion repository) and PSOM (v656 of the subversion repository) were used because they implemented some bug fix for this release. During the time of the experiment, the PSOM jobs were the only ones running on the execution servers for

[18] www.nitrc.org/projects/niak
[19] http://www.nitrc.org/plugins/mwiki/index.php/neurobureau:NIAKPipeline
[20] http://fcon_1000.projects.nitrc.org/indi/adhd200/
[21] http://en.wikibooks.org/wiki/MINC
[22] http://fcon_1000.projects.nitrc.org/
[23] www.top500.org/lists/2011/11

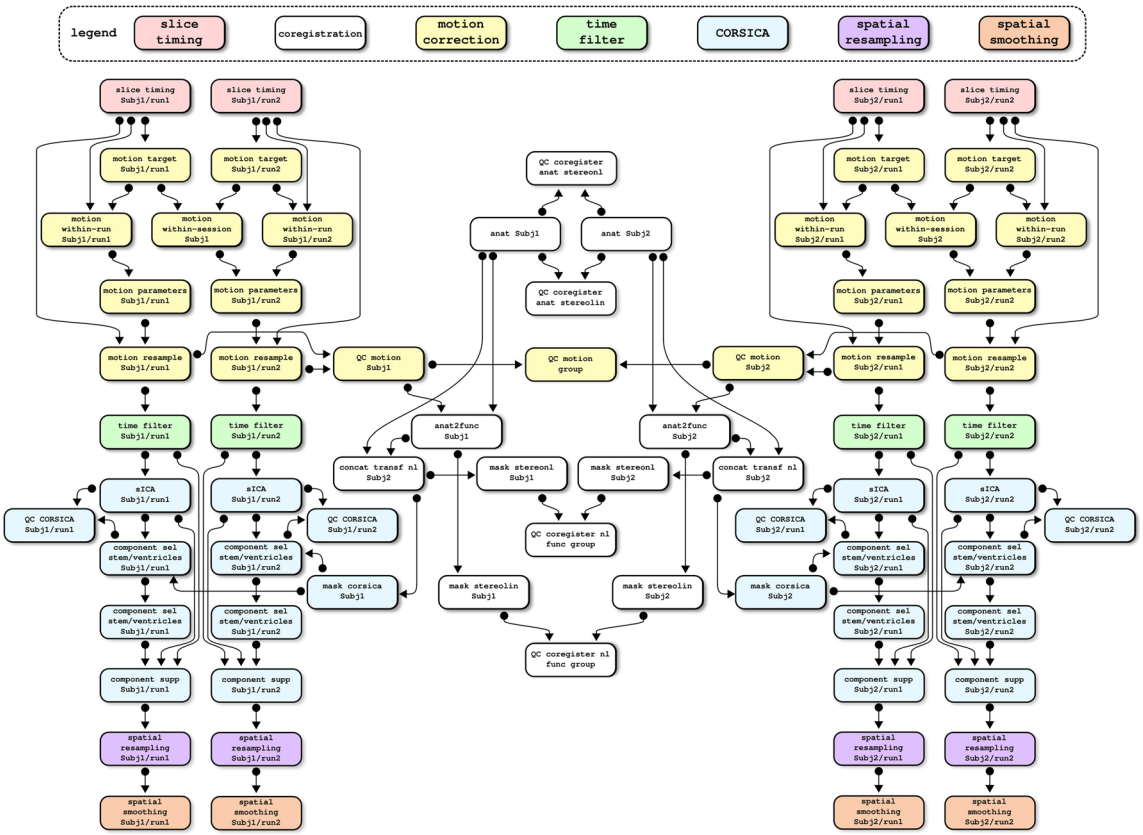


FIGURE 7 | An example of dependency graph for the NIAK fMRI preprocessing pipeline. This example includes two subjects with two fMRI datasets each. The pipeline includes close to 100 jobs, and cleanup jobs have been removed to simplify the representation. Colors have been used to code the main stages of the preprocessing.

`peuplier` and `magma`, while `guillimin` had about 75% processors in use.

6.3. RESULTS

The raw `Cambridge` database had a size of 7.7 GB, with a total of 21 GB generated by the pipeline (output/input ratio of 273%). The NIAK pipeline included 5153 jobs featuring 8348 unique input/output files (not including temporary files). **Figure 8A** shows the distribution of execution times for all jobs on `peuplier-8`. The pipeline included about 1500 "cleanup" jobs deleting intermediate files, with an execution time of less than 0.2 s. The other jobs lasted anywhere between a few seconds and 15 min, with hundreds of jobs of less than 2 min. Because of the large number of very short jobs, the pipeline manager was not able to constantly submit enough jobs to use `n` cores at all time, even when it would have been possible in theory. This effect was small on `peuplier`, `magma` or `guillimin-{24,50}` see **Figure 8B–C**. It became pronounced on `guillimin-{100,200}`, see **Figure 8D**. The serial execution time of the pipeline (sum of execution time of all jobs) varied a lot from one configuration to the other: from 120 h (5 days) on `guillimin-24` to almost double (220 h, 9 days) on `magma-8`. The serial execution time, however, increased quickly on `guillimin` with an increasing `n`, see **Figure 8E**. Despite this effect, and thanks to parallelization, the parallel execution time (time elapsed between the beginning and the end of the pipeline processing) steadily decreased with an increasing `n`, see **Figure 8F**. The speed-up factor (defined as the ratio between the serial and parallel execution time) still departed from the optimal value `n`. Consistent with our observations on the effective number of cores used on average, the departure between the speed-up factor and `n` increased with `n`, and became pronounced for `n` greater than 100, see **Figure 8G**. This result can be expressed as a parallelization efficiency, defined as the ratio between the empirical speed-up factor and `n`. Parallelization efficiency was excellent (over 90%) on `peuplier-8` and gradually decreased with an increasing `n` to reach about 80% on `peuplier-24` or `guillimin-24` and 60% on `guillimin-200`. In this last setting, the fMRI datasets and structural scans of about 200 subjects were still processed in a little bit more than 2 h.

7. DISCUSSION

7.1. OVERVIEW

We propose a new PSOM to implement, run, and re-run pipeline analysis on large databases. Our approach is well-suited for pipelines involving heterogeneous tools that can communicate through a file system in a largely parallel fashion. This notably matches the constraints found in neuroimaging. The PSOM

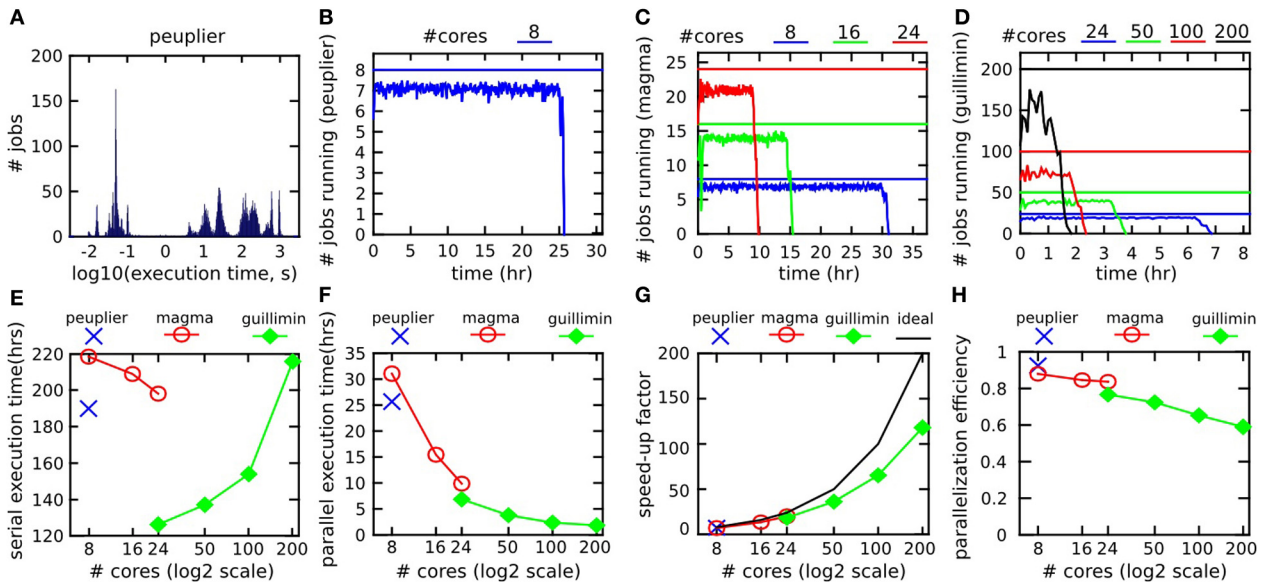


FIGURE 8 | Benchmark experiments with the NIAK fMRI preprocessing pipeline. The distribution of execution time for all jobs on one server (`peuplier`) is shown in panel **(A)**. The number of jobs running at any given time across the whole execution of the pipeline (averaged on 5 min time windows) is shown in panels **(B–D)** for servers `peuplier`, magma and `guillimin`, respectively. The user-specified maximum number of concurrent jobs is indicated by a straight line. The serial execution time of the pipeline, i.e., the sum of execution times for all jobs, is shown in panel **(E)**. The parallel execution time, i.e., the time elapsed between the beginning and the end of the pipeline processing, is shown in panel **(F)**. The speed-up factor, i.e., serial time divided by parallel time, is presented in panel **(G)**, along with the ideal speed-up, equal to the user-specified maximal number of concurrent jobs. Finally, the parallelization efficiency (i.e., the ratio between the empirical speed-up and the ideal speed-up) is presented in panel **(H).**

coding standards produce concise, readable code which in our experience is easy to maintain and develop over time. It is also highly scalable: a pipeline can incorporate thousands of jobs, each one featuring tens to hundreds of parameters. From a developer's perspective, using PSOM does not limit the scope of distribution of the software, as pipelines can be executed inside an O/M session as would any regular O/M code. The very same code can also be deployed on a multi-core machine or in a supercomputing environment simply by changing the PSOM configuration.

7.2. ONLINE DOCUMENTATION

The main body of documentation is available on a wiki hosted online by google code, see **Table 1**. This resource is updated for each new release of PSOM. It covers selected topics such as the configuration of the pipeline manager more extensively than this paper. The "short PSOM tutorial" reproduces step-by-step all the experiments reported in Section 3.

Table 1 | Online resources for PSOM.

Ressources	URL
Developer's site	code.google.com/p/psom
User's site	nitrc.org/projects/psom/
Downloads	nitrc.org/frs/?group_id = 316
Forum	nitrc.org/forum/forum.php?forum_id = 1316
Wiki overview	code.google.com/p/psom/w/list
PSOM short tutorial	code.google.com/p/psom/wiki/HowToUsePsom
Coding guidelines	code.google.com/p/psom/wiki/CodingGuidelines
PSOM configuration	code.google.com/p/psom/wiki/ConfigurationPsom
PSOM tests	code.google.com/p/psom/wiki/TestPsom

7.3. THE BENEFITS OF PIPELINE ANALYSIS

Parallel computing is a central feature of PSOM, as it allows to reduce the time necessary to complete an analysis. The pipeline system can be beneficial even when used within a single session. PSOM automatically keeps a record of all the steps and parameters of the pipeline. These logs are detailed enough to reproduce an entire analysis (as long as the production environment itself can be reproduced). This is an essential feature in the perspective of reproducible research. The pipeline logs can also be used for profiling the execution time of the whole pipeline as well as its subparts. This can be useful to run a benchmark or to identify computational bottlenecks. It is finally possible to restart the pipeline at any stage, or even to add stages or change parameters. Over multiple executions, PSOM will restart only the pipeline stages impacted by the changes. This ability to properly handle pipeline updates is critical in the development phase, and can also be useful to test alternative choices of parameter/algorithmic selection.

7.4. PARALLEL COMPUTATION CAPABILITIES

The benchmark experiments demonstrated that PSOM is able to handle pipelines featuring thousands of jobs and tens of gigabytes of data. It can also dramatically reduce the execution time: an fMRI database including almost 200 subjects could be preprocessed in less than 3 h. The parallelization efficiency was

excellent with a small or moderate number of computing cores (under 50), yet it dropped past a hundred cores. This is because the NIAK pipeline features hundreds of very short jobs (less than 0.2 s). The pipeline manager thus needs to submit jobs at a very high pace to use all resources. PSOM would behave better with longer jobs. Some alternative pipeline systems, e.g., Swift (Stef-Praun et al., 2007), scale efficiently up to thousands of cores even with short jobs (30 s long). Swift implements for this purpose a multi-level pipeline execution engine: the jobs are grouped into small sub-pipelines that are then processed independently. We are planning to add this feature to the pipeline execution engine in the next major release of PSOM.

7.5. QUALITY CONTROL

Quality control is a challenge when processing large databases. This step is, however, critical to establish the scientific credibility of the results. Quality control is too problem-specific to be implemented as a general tool in a pipeline system. It is, however, possible to integrate *ad-hoc* steps of quality control in a pipeline. The NIAK fMRI preprocessing pipeline for instance includes a group summary of the individual motion parameters as well as measures of the quality of coregistration between the structural and functional images, amongst others. This approach was found to greatly facilitate the quality control of the preprocessing of ADHD-200[24], a database including close to 1000 subjects (Lavoie-Courchesne et al., 2012).

7.6. FILES COLLECTION

Neuroimaging datasets often come as collections of files. The DICOM format for example may store individual slices as separate files. A variant of the NIFTI format (used by the SPM software) stores each brain volume of an fMRI dataset as one or two separate files. As hundreds of fMRI brain volumes are typically collected on an individual, both formats represent a large file collection. The structures used to describe input/output files in PSOM is very versatile, and can include an extensive file collection. There are, however, performance penalties for doing so. Those penalties are in part internal to PSOM, because analyzing the dependencies with DICOM or 3D NIFTI will take tens of seconds. Some operations on the file system may also slow down because of the number of files, independently of their size. This can be observed for example during the internet synchronization of file collections between sites. By contrast with the DICOM and 3D NIFTI formats, MINC or NIFTI have the ability to store a full 3D+t dataset into a single file. For computational efficiency, it is thus advisable to start a pipeline by converting the input database into such a 3D+t format.

7.7. PSOM CONFIGURATION

An important choice was made in the design of PSOM: the interactions between the pipeline manager and the execution controler (`at`, `qsub`, `msub`, etc.) are kept to a bare minimum. The main benefit of this approach is the ability of PSOM to interact easily and robustly with a variety of execution environments for the jobs. However, when a queuing scheduler is employed, PSOM has no means to interrogate the state of a particular job. It assumes that submitted jobs will be able to run, and if that assumption is met, each job will report its completion status using a file-based mechanism internal to PSOM. If this assumption is not met, the users may not get any useful feedback on the cause of failures. For this reason, a dedicated `psom_config` function is available to test each stage of job submission one by one, and will guide users when setting up their configuration.

7.8. DYNAMIC PIPELINE COMPOSITION

In its current form, PSOM supports pipelines that can be described as a static DAG. Static means that the full pipeline representation has to be generated by the user prior to execution. In alternative pipeline systems such as Taverna, a pipeline can branch or iterate depending on data-dependent conditions that are dynamically evaluated during the execution. This is not currently possible in PSOM. A future development will address this issue by allowing jobs to regenerate themselves new jobs. This will be achieved by writing a description of these new jobs in a dedicated folder constantly monitored by the pipeline manager. This generic mechanism will enable a dynamic, data-dependent composition of the pipeline.

7.9. INTEROPERABILITY

The PSOM framework fosters a modular organization of the code that is well adapted to a specific pipeline. Such organization will facilitate the subsequent implementation of the pipeline in any workflow system. Porting a pipeline from PSOM to another system may even become a largely automated task. We recently demonstrated the feasibility of this approach by building an interface between the NIAK fMRI preprocessing pipeline and CBRAIN (Lavoie-Courchesne et al., 2012). CBRAIN is a computing platform that offers transparent multipoint data transfers from various network storage nodes (file servers, S3 API, databases), transparent access to grid computing facilities, as well as a secured management of the access to a project by multiple users. This type of integration is made possible by the simplicity of the pipeline representation adopted by PSOM. This representation is moreover very similar to the ones used by Nipype and Swift and is also compatible with DAG-based representations (e.g., Soma-workflow, DAGMan, Pegasus) as long as a dependency graph is generated with PSOM. We will work in the future on a library of interfaces to allow PSOM users to select the execution engine that is the most adapted to their needs in the context of a given application.

8. CONCLUSION

In this paper we propose a PSOM. PSOM provides a solution to implement, run and re-run multi-stage processing on large databases. It automatically keeps track of the details of the pipeline in order to make the results reproducible. It also provides tools for profiling the pipeline execution. PSOM handles updates made to the pipeline: only the jobs impacted by changes will be restarted. The pipeline execution can be deployed in a variety of computing environments and can take advantage of parallel computing facilities. The same code can run in any of the supported

[24]http://fcon_1000.projects.nitrc.org/indi/adhd200/

execution environments simply by changing the PSOM configuration. On a benchmark using real neuroimaging datasets, the processing time for 198 subjects was reduced from over a week down to less than 3 h with 200 computing cores. PSOM supports a variety of operating systems (Linux, Windows, Mac OSX) and is distributed under an open-source (MIT) license. We believe that this package is a valuable resource for researchers working in the neuroimaging field, and especially those who are regular users of Octave or Matlab.

ACKNOWLEDGMENTS

The authors are grateful to the members of the 1000 functional connectome consortium for publicly releasing the "Cambridge" data sample, and would like to thank Dr. Guillaume Flandin, Dr. Salma Mesmoudi, and M. Pierre Rioux for insightful discussions. Several authors of existing pipeline solutions have reviewed the introduction section and provided very valuable feedback: Dr. Satrajit Gosh, Dr. Soizic Laguitton, Dr. Michael Wilde, Mr. Marc-Etienne Rousseau, Dr. Ewa Deelman, Dr. Ivo Dinov, and Dr. Volkmar Glauche. Google code and the "Neuroimaging Informatics Tools and Resources Clearinghouse" (NITRC) are generously hosting PSOM's developers and users site, respectively. The alpha-testers of the project (Pr. Grova's lab, Dr. Gaolong Gong, M. Christian Dansereau, and Dr. Guillaume Marrelec) also provided valuable contributions. SLC was partly supported by a pilot project grant of the Quebec bioimaging network (QBIN), "fonds de recherche en santé du Québec" (FRSQ). The computational resources used to perform the data analysis were provided by ComputeCanada[25] and CLUMEQ[26], which is funded in part by NSERC (MRS), FQRNT, and McGill University.

[25]https://computecanada.org/
[26]http://www.clumeq.mcgill.ca/

REFERENCES

Ad-Dab'bagh, Y., Einarson, D., Lyttelton, O., Muehlboeck, J. S., Mok, K., Ivanov, O., Vincent, R. D., Lepage, C., Lerch, J., Fombonne, E., and Evans, A. C. (2006). "The CIVET image-processing environment: a fully automated comprehensive pipeline for anatomical neuroimaging research," in *Proceedings of the 12th Annual Meeting of the Human Brain Mapping Organization. Neuroimage,* ed M. Corbetta (Florence, Italy).

Armstrong, T. G. (2011). *Integrating Task Parallelism into the Python Programming Language.* Master's thesis, The University of Chicago.

Ashburner, J. (2011). SPM: a history. *Neuroimage.* doi: 10.1016/j.neuroimage.2011.10.025. [Epub ahead of print].

Baker, H. G., and Hewitt, C. (1977). "The Incremental Garbage Collection of Processes. Technical Report," in *Proceedings of the 1977 symposium on Artificial intelligence and programming languages archive.* (New York, NY: ACM).

Bellec, P., Carbonell, F. M., Perlbarg, V., Lepage, C., Lyttelton, O., Fonov, V., Janke, A., Tohka, J., and Evans, A. C. (2011). "A neuroimaging analysis kit for Matlab and Octave," in *Proceedings of the 17th International Conference on Functional Mapping of the Human Brain.* (Quebec, QC, Canada).

Bellec, P., Perlbarg, V., and Evans, A. C. (2009). Bootstrap generation and evaluation of an fMRI simulation database. *Magn. Reson. Imaging* 27, 1382–1396.

Bellec, P., Petrides, M., Rosa-Neto, P., and Evans, A. C. (2010a). "Stable group clusters in resting-state fMRI at multiple scales: from systems to regions," in *Proceedings of the 16th International Conference on Functional Mapping of the Human Brain.* (Barcelona, Spain).

Bellec, P., Rosa-Neto, P., Lyttelton, O. C., Benali, H., Evans, A. C. (2010b). Multi-level bootstrap analysis of stable clusters in resting-state fMRI. *Neuroimage* 51, 1126–1139.

Biswal, B. B., Mennes, M., Zuo, X.-N. N., Gohel, S., Kelly, C., Smith, S. M., Beckmann, C. F., Adelstein, J. S., Buckner, R. L., Colcombe, S., Dogonowski, A.-M. M., Ernst, M., Fair, D., Hampson, M., Hoptman, M. J., Hyde, J. S., Kiviniemi, V. J., Kötter, R., Li, S.-J. J., Lin, C.-P. P., Lowe, M. J., Mackay, C., Madden, D. J., Madsen, K. H., Margulies, D. S., Mayberg, H. S., McMahon, K., Monk, C. S., Mostofsky, S. H., Nagel, B. J., Pekar, J. J., Peltier, S. J., Petersen, S. E., Riedl, V., Rombouts, S. A., Rypma, B., Schlaggar, B. L., Schmidt, S., Seidler, R. D., Siegle, G. J., Sorg, C., Teng, G.-J. J., Veijola, J., Villringer, A., Walter, M., Wang, L., Weng, X.-C. C., Whitfield-Gabrieli, S., Williamson, P., Windischberger, C., Zang, Y.-F. F., Zhang, H.-Y. Y., Castellanos, F. X., and Milham, M. P. (2010). Toward discovery science of human brain function. *Proc. Natl. Acad. Sci. U.S.A.* 107, 4734–4739.

Bose, R., Foster, I., and Moreau, L. (2006). *Report on the International Provenance and Annotation Workshop: (IPAW'06)*. 3–5 May 2006, Chicago, IL: SIGMOD Rec. 35, 51–53.

Burton, A. (2011). Big science for a big problem: ADNI enters its second phase. *Lancet Neurol.* 10, 206–207.

Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., Silva, C. T., and Vo, H. T. (2006). "VisTrails: visualization meets data management," in *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data.* SIGMOD '06. ACM, (New York, NY: USA), 745–747.

Carbonell, F. M., Bellec, P., and Shmuel, A. (2012). Global and system-specific resting-state BOLD fluctuations are uncorrelated: principal component analysis reveals anti-correlated networks. *Brain Connect.* doi: 10.1089/brain.2011.0065. [Epub ahead of print].

Chao-Gan, Y., and Yu-Feng, Z. (2010). DPARSF: a MATLAB Toolbox for "Pipeline" data analysis of resting-state fMRI. *Front. Syst. Neurosci.* 4:13 doi: 10.3389/fnsys.2010.00013

Collins, D. L., Neelin, P., Peters, T. M., and Evans, A. C. (1994). Automatic 3D intersubject registration of MR volumetric data in standardized Talairach space. *J. Comput. Assist. Tomogr.* 18, 192–205.

Dansereau, C. L., Pittau, F., Bellec, P., Gotman, J., and Grova, C. (2011). "Detection of abnormal resting state networks in epileptic patients," in *17th International Conference on Functional Mapping of the Human Brain.* (Quebec, QC, Canada).

Das, S., Zijdenbos, A. P., Vins, D., Harlap, J., and Evans, A. C. (2012). LORIS: a web-based data management system for multi-center studies. *Front. Neuroinform.* 5:37. doi: 10.3389/fninf.2011.00037

Deelman, E., Gannon, D., Shields, M., and Taylor, I. (2009). Workflows and e-Science: an overview of workflow system features and capabilities. *Future Generation Comput. Syst.* 25, 528–540.

Deelman, E., Singh, G., Su, M. H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G. B., Good, J., Laity, A., Jacob, J. C., and Katz, D. S. (2005). Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Sci. Program.* 13, 219–237.

Dinov, I. D., van Horn, J. D., Lozev, K. M., Magsipoc, R., Petrosyan, P., Liu, Z., Mackenzie-Graham, A., Eggert, P., Parker, D. S., and Toga, A. W. (2009). Efficient, distributed and interactive neuroimaging data analysis using the LONI pipeline. *Front. Neuroinform.* 3:22. doi: 10.3389/neuro.11.022.2009

Evans, A. C. (2006). The NIH MRI study of normal brain development. *Neuroimage* 30, 184–202.

Fonov, V., Evans, A. C., Botteron, K., Almli, C. R., McKinstry, R. C., Collins, D. L., Brain Development Cooperative Group. (2011). Unbiased average age-appropriate atlases for pediatric studies. *Neuroimage* 54, 313–327.

Frisoni, G. B., Redolfi, A., Manset, D., Rousseau, M.-E. E., Toga, A., and Evans, A. C. (2011). Virtual imaging laboratories for marker discovery in neurodegenerative diseases. *Nat. Rev. Neurol.* 7, 429–438.

Ganjavi, H., Lewis, J. D., Bellec, P., MacDonald, P. A., Waber, D. P., Evans, A. C., Karama, S., Brain Development Cooperative Group. (2011). Negative associations between corpus callosum midsagittal area and IQ in a representative sample of healthy children and adolescents. *PloS One* 6:19698. doi: 10.1371/journal.pone.0019698

Goecks, J., Nekrutenko, A., Taylor, J., Galaxy Team. (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational

research in the life sciences. *Genome Biol.* 11, R86.

Gorgolewski, K., Burns, C. D., Madison, C., Clark, D., Halchenko, Y. O., Waskom, M. L., and Ghosh, S. S. (2011). Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. *Front. Neuroinformatics* 5:13. doi: 10.3389/fninf.2011.00013

Harrison, A., Taylor, I., Wang, I., and Shields, M. (2008). WS-RF Workflow in Triana. *Int. J. High Perform. Comput. Appl.* 22, 268–283.

Kassis, N., Gong, G., Rousseau, M. E., Adalat, R., and Evans, A. C. (2011). "BrainBrowser: Web-based 3D Visualization for the maCaCC Dataset and other Surface Data," in *17th International Conference on Functional Mapping of the Human Brain*. (Quebec, QC, Canada).

Laguitton, S., Rivière, D., Vincent, T., Fischer, C., Geffroy, D., Souedet, N., Denghien, I., and Cointepas, Y. (2011). "Soma-workflow: a unified and simple interface to parallel computing resources," in MICCAI 2011 Conference, eds T. Peters, G. Fichtinger, and A. Martel (Toronto: Springer LNCS), (in press).

Lavoie-Courchesne, S., Rioux, P., Chouinard-Decorte, F., Sherif, T., Rousseau, M. E., Das, S., Adalat, R., Doyon, J., Craddock, C., Margulies, D., Chu, C., Lyttelton, O., Evans, A. C., and Bellec, P. (2012). Integration of a neuroimaging processing pipeline into a pan-canadian computing grid. *J. Phys. Conf. Ser.* 341, 012032.

Liu, H., Stufflebeam, S. M., Sepulcre, J., Hedden, T., and Buckner, R. L. (2009). Evidence from intrinsic activity that asymmetry of the human brain is controlled by multiple factors. *Proc. Natl. Acad. Sci. U.S.A.*106, 20499–20503.

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E. A., Tao, J., and Zhao, Y. (2006). Scientific workflow management and the Kepler system: research articles. *Concurr. Comput. Pract. Exper.* 18, 1039–1065.

Machado, A., Lina, J. M., Tremblay, J., Lassonde, M., Nguyen, D. K., Lesage, F., and Grova, C. (2011). Detection of hemodynamic responses to epileptic activity using simultaneous Electro-EncephaloGraphy (EEG)/Near Infra Red Spectroscopy (NIRS) acquisitions. *Neuroimage* 56, 114–125.

MacKenzie-Graham, A. J., Van Horn, J. D., Woods, R. P., Crawford, K. L., and Toga, A. W. (2008). Provenance in neuroimaging. *Neuroimage* 42, 178–195.

Marcus, D., Olsen, T., Ramaratnam, M., and Buckner, R. (2007). The extensible neuroimaging archive toolkit. *Neuroinformatics* 5, 11–33.

Mesirov, J. P. (2010). Accessible reproducible research. *Science* 327, 415–416.

Moeller, F., Maneshi, M., Pittau, F., Gholipour, T., Bellec, P., Dubeau, F., Grova, C., and Gotman, J. (2011). Functional connectivity in patients with idiopathic generalized epilepsy. *Epilepsia* 52, 515–522.

Oinn, T., Greenwood, M., Addis, M., Alpdemir, M. N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M. R., Senger, M., Stevens, R., Wipat, A., and Wroe, C. (2006). Taverna: lessons in creating a workflow environment for the life sciences. *Concurr. Comput. Pract. Exper.* 18, 1067–1100.

Orban, P., Doyon, J., Hoge, R., and Bellec, P. (2011). "Stable clusters of brain regions associated with distinct motor task-evoked hemodynamic responses," in *Proceedings of the 17th International Conference on Functional Mapping of the Human Brain.*

Perlbarg, V., Bellec, P., Anton, J.-L., Pélégrini-Issac, M., Doyon, J., and Benali, H. (2007). CORSICA: correction of structured noise in fMRI by automatic identification of ICA components. *Magn. Reson. Imaging* 25, 35–46.

Poinot, M. (2010). Five good reasons to use the hierarchical data format. *Comput. Sci. Eng.* 12, 84–90.

Schoemaker, D., Soder, R. B., Sziklas, V., Rowley, J., Carbonnell, F., Mohades, S., Fonov, V., Bellec, P., Dagher, A., Schmuel, A., Gauthier, S., and Rosa Neto, P. (2011). "Hippocampal resting connectivity lateralize memory function in aMCI brain networks," in *Alzheimer's Association 2011 International Conference on Alzheimer's Disease (ICAD)*. (Paris, France).

Stef-Praun, T., Clifford, B., Foster, I., Hasson, U., Hategan, M., Small, S. L., Wilde, M., and Zhao, Y. (2007). Accelerating medical research using the swift workflow system. *Stud. Health Technol. Informatics* 126, 207–216.

Tadel, F., Baillet, S., Mosher, J. C., Pantazis, D., and Leahy, R. M. (2011). Brainstorm: a user-friendly application for MEG/EEG analysis. *Comput. Intell. Neurosci.* 2011, 1–13.

Wilde, M., Hategan, M., Wozniak, J. M., Clifford, B., Katz, D. S., and Foster, I. (2011). Swift: a language for distributed parallel scripting. *Parallel Comput.* 37, 633–652.

Worsley, K. J., Liao, C. H., Aston, J., Petre, V., Duncan, G. H., Morales, F., and Evans, A. C. (2002). A general statistical analysis for fMRI data. *Neuroimage* 15, 1–15.

Zijdenbos, A. P., Jimenez, A., and Evans, A. C. (1998). "Pipelines: large scale automatic analysis of 3d brain data sets," in *Proceedings of the 4th International Conference on Functional Mapping of the Human Brain.*

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 07 November 2011; accepted: 05 March 2012; published online: 03 April 2012.

Citation: Bellec P, Lavoie-Courchesne S, Dickinson P, Lerch JP, Zijdenbos AP and Evans AC (2012) The pipeline system for Octave and Matlab (PSOM): a lightweight scripting framework and execution engine for scientific workflows. Front. Neuroinform. ***6****:7. doi: 10.3389/fninf.2012.00007*

frontiers in
NEUROINFORMATICS

TECHNOLOGY REPORT ARTICLE
published: 15 January 2015
doi: 10.3389/fninf.2014.00090

Automatic analysis (aa): efficient neuroimaging workflows and parallel processing using Matlab and XML

Rhodri Cusack[1]*, Alejandro Vicente-Grabovetsky[2], Daniel J. Mitchell[3], Conor J. Wild[1], Tibor Auer[3], Annika C. Linke[1] and Jonathan E. Peelle[4]

[1] *Brain and Mind Institute, Western University, London, ON, Canada*
[2] *Donders Institute for Brain, Cognition and Behaviour, Nijmegen, Netherlands*
[3] *MRC Cognition and Brain Sciences Unit, Cambridge, UK*
[4] *Department of Otolaryngology, Washington University in St. Louis, St. Louis, MO, USA*

Edited by:
John Van Horn, University of California, Los Angeles, USA

Reviewed by:
Daniel Gardner, Weill Cornell Medical College, USA
Andrei Irimia, University of Southern California, USA

****Correspondence:***
Rhodri Cusack, Brain and Mind Institute, Western University, London, ON N6A 1W8, Canada
e-mail: rhodri@cusacklab.org

Recent years have seen neuroimaging data sets becoming richer, with larger cohorts of participants, a greater variety of acquisition techniques, and increasingly complex analyses. These advances have made data analysis pipelines complicated to set up and run (increasing the risk of human error) and time consuming to execute (restricting what analyses are attempted). Here we present an open-source framework, automatic analysis (*aa*), to address these concerns. Human efficiency is increased by making code modular and reusable, and managing its execution with a processing engine that tracks what has been completed and what needs to be (re)done. Analysis is accelerated by optional parallel processing of independent tasks on cluster or cloud computing resources. A pipeline comprises a series of modules that each perform a specific task. The processing engine keeps track of the data, calculating a map of upstream and downstream dependencies for each module. Existing modules are available for many analysis tasks, such as SPM-based fMRI preprocessing, individual and group level statistics, voxel-based morphometry, tractography, and multi-voxel pattern analyses (MVPA). However, *aa* also allows for full customization, and encourages efficient management of code: new modules may be written with only a small code overhead. *aa* has been used by more than 50 researchers in hundreds of neuroimaging studies comprising thousands of subjects. It has been found to be robust, fast, and efficient, for simple-single subject studies up to multimodal pipelines on hundreds of subjects. It is attractive to both novice and experienced users. *aa* can reduce the amount of time neuroimaging laboratories spend performing analyses and reduce errors, expanding the range of scientific questions it is practical to address.

Keywords: neuroimaging, functional magnetic resonance imaging (fMRI), diffusion tensor imaging (DTI), diffusion weighted imaging (DWI), multi-voxel pattern analysis (MVPA), software, pipeline

THE NEED FOR EFFICIENT WORKFLOWS

The last two decades have seen enormous growth in the use of magnetic resonance imaging (MRI) as a tool to understand brain function, and in the size and complexity of the datasets acquired. The number of participants in individual studies has grown for many reasons, including: the increasing availability of MRI scanners; a move from fixed- to random-effects designs (Friston et al., 1999; Mumford and Nichols, 2008); a demand for greater replication in neuroimaging ("The dilemma of weak neuroimaging papers," http://www.danielbor.com/dilemma-weak-neuroimaging); the need to overcome statistical noise in studies of individual differences, genetics, aging, development or disease; large scale investments such as the Human Connectome Project (Van Essen et al., 2012), Alzheimer's Disease Neuroimaging Initiative (Mueller et al., 2005) or Cambridge Centre for Aging and Neuroscience (http://www.cam-can.org); and a growth in open data sharing (Van Horn et al., 2001; Biswal et al., 2010; Poldrack et al., 2013; http://www.xnat.org).

Furthermore, the neuroimaging data acquired from each participant have become richer. Whereas in the past, researchers frequently collected data using a single method, many now acquire diverse MRI protocols, including structural (e.g., T1, T2, PD), functional (echoplanar imaging; EPI), connectivity (diffusion-weighted imaging; DWI), fieldmaps (multi-echo; gradient echo) and myelination (magnetization transfer ratio; MTR) measurements in single studies. Accelerated sequences using parallel imaging (SENSE, GRAPPA, and multiband EPI) have allowed for finer temporal or spatial resolution and increased the size of datasets by up to an order of magnitude.

Alongside the increasing quantity of data, the palette of analysis methods has also grown. In functional MRI (fMRI), in addition to the standard preprocessing stages of motion correction, slice-timing correction, warping-to-template (normalization) and smoothing, denoising is now possible using tools based upon independent components analysis (Calhoun et al.,

2009; Kundu et al., 2012; http://fsl.fmrib.ox.ac.uk/fslcourse/graduate/icaprac/artdata/dim33.ica/report; http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FIX), modeling of noise components (Kay et al., 2013), and image rejection (Power et al., 2012). Statistical analyses are now often conducted both using standard univariate methods and multi-voxel pattern analysis (MVPA) (Haynes and Rees, 2006; Kriegeskorte et al., 2006; Norman et al., 2006). Brain structure is often analyzed using voxel- (Ashburner, 2009) and surface-based (Winkler et al., 2012) morphometry, and gyrification indices (Schaer et al., 2008). Registration between individuals can use relatively low-dimensional warping to a template, or higher dimensional registration (Ashburner, 2007, 2009). Diffusion data can be analyzed with probabilistic or deterministic methods, by summarizing parameters such as the fractional anisotropy (FA) on a skeleton (Smith et al., 2006) or by tracing tracts (Behrens et al., 2007). In addition to the sheer number of useful analysis methods now available, many methods are highly computationally intensive, such as searchlight MVPA (Kriegeskorte et al., 2006), probabilistic tractography, and high-dimensional image warping (Ashburner, 2007). Implementing these complementary approaches commonly requires a combination of software packages, which follow diverse concepts and may even use different file formats. The integration of results from these different software packages (e.g., using fMRI activation clusters as seeds for diffusion tractography) further increases the complexity of an analysis workflow.

The increasing quantity of raw data and greater number of computationally intensive analysis methods have led to two challenges. The first is an increase in the complexity of the workflows required: There are a greater number of individual "chunks" of processing, and more complex dependencies between these chunks. Furthermore, even the best-run neuroimaging study does not always proceed exactly according to plan, and there are often idiosyncrasies that result from technical glitches, operator error, or participant non-compliance. Manual intervention in this complex workflow leads to the potential for human error.

The second challenge is an increase in computation time per study. Many neuroimagers are already stretched by the need to become multidisciplinary experts in the physics of neuroimaging, the mathematics for analysis, the psychology of cognitive function, and the biology of the brain. They do not all necessarily relish the additional challenge of becoming a programmer and computer scientist so that they can make the most efficient use of computing resources.

The many stages of analysis required to draw conclusions from MRI data were once almost universally accomplished using point-and-click interfaces, a practice many continue. However, as the field matures, this sort of "manual" analysis is becoming increasingly impractical and unattractive. Here, we present a software package, automatic analysis (*aa*) (http://automaticanalysis.org), which provides a simple but flexible way to specify complex workflows, keep track of what needs to be done, and facilitate parallel computing. *aa* is engineered so that even when used by a "lazy" operator precise records are kept. It is easily extendable, and code naturally becomes re-useable and shareable.

EXISTING SOFTWARE

Once the decision is made to use a processing pipeline, there are a number of options. Although the best solution depends a great deal on individual preferences and priorities, we have engineered *aa* to fill needs not met by other processing pipelines.

Neuroimaging benefits enormously from a dynamic software development community, with new analysis tools frequently disseminated by large teams. However, these packages focus primarily on implementing specific tools, rather than managing efficient workflows. *aa* provides access to many (though not all) functions in the major neuroimaging packages of SPM, FSL, and Freesurfer; other tools such as the Advanced Normalization Tools (ANTs); and our own implementation of searchlight- or ROI- based MVPA. In addition, although not discussed in this manuscript, it also includes growing support for other modalities including MEG, EEG, and ECoG.

DESIGN GOALS

EFFICIENT AND EASY-TO-READ SPECIFICATION OF COMPLEX PIPELINES

As neuroimaging pipelines become increasingly complicated, it becomes important to develop elegant ways of describing them. With *aa*, we aimed to separate a high-level description of what needs to be done (e.g., motion correction followed by slice-timing correction) from the individual parameters that control each stage. Furthermore, wherever possible, sensible default values are available for each stage, so that an analysis can be specified as leanly and efficiently as possible, without the need to re-invent the wheel each time. We make extensive use of XML markup language to provide easy-to-read descriptions of tasklists (i.e., the list of processing stages) and settings.

MODULAR DESIGN

To make it easier to identify the code that is responsible for a given task, and to facilitate parallel computing, each stage of processing is described by an encapsulated "module."

SEPARATION OF METHOD AND DATA

A separation is enforced between the algorithms that should be applied and the data (i.e., participants and sessions) on which they should operate. This separation ensures that modules are re-useable: once written in the context of one analysis, modules may usually be re-used without modification in another analysis of different data.

ONLY DO WHAT NEEDS TO BE DONE

Modules are never called directly by the user; instead, their execution is handled by the *aa* scheduling engine (*aa_doprocessing*). The scheduling engine identifies whether a module has already been run on a given piece of data, and whether the inputs to a module have changed (e.g., a subject has been added) since it was last run. If a module has already been run, it is not repeated. Although simple, checking for completed stages provides three important practical benefits. First, it saves computational resources. Second, it makes debugging quicker: If an analysis crashes partway through, then the next time it is re-run,

all of the stages that lead up to the crashing stage will not be executed. Third, it stops the user from needing to "comment out" lines that have already completed when rerunning just one later part again. As a result, in practice the final *aa* script will typically recreate an analysis in its entirety.

Checking for previously-completed stages also facilitates complex pipelines with multiple analysis pathways. For example, in the case where all processing stages save one are identical (e.g., to compare preprocessing with and without slice-timing correction), *aa* can be informed about a branched tasklist and re-use inputs that are common to both branches.

FACILITATE PARALLEL PROCESSING

As analyses become more computationally intensive, being able to easily accelerate them across a cluster of machines is increasingly important. Often, execution time determines what analyses a user can bear. For example, even if an analysis runs in a single-threaded manner in a practical amount of time (say 5 days), a user will be highly discouraged from running it again to fix some small issue.

aa uses coarse-grained parallelization, meaning that where possible, multiple modules, different EPI sessions, subjects, or even analyses (e.g., groups of searchlights in an MVPA analysis for a single module) are run in parallel. Modules themselves are not written differently for parallel or single-threaded execution: parallelization is achieved entirely in the scheduling engine (although individual modules can in principle be parallelized at a finer-grained level).

KEEP TRACK OF WHAT HAS HAPPENED

A precise record of everything that has happened in an *aa* analysis is saved and can be referred to in the future. It is stored as a Matlab structure, which can be read back in to recreate the analysis, or probed for parameter settings.

DIAGNOSTICS AND QUALITY CONTROL

One of the drawbacks of batch analysis is that a user may be tempted to only look at the final results, and not inspect the data at each stage of processing. However, complex analysis pipelines can fail in a greater number of ways than simpler pipelines. Some failures can be obvious (e.g., activation outside the brain due to imperfect registration), while others are harder to track down (e.g., weaker group activation detected due to high between-subject variability caused by motion). Consequently, inspection of data is as important as ever. Several existing solutions generate some diagnostic data during the analysis (e.g., FSL's FEAT Pre-stats and Registration reports); however, the information provided is limited, sometimes complicated to reach, and almost never submitted to between-subject analysis (important for the measurement of between-subject variance and outlier detection).

To address this problem, many *aa* modules create diagnostic results (e.g., plots of motions to be corrected, registration overlays, thresholded statistical parameter maps for first-level contrasts). In addition, *aa* also implements various quality control tools (mostly SPM- and FSL-based). A dedicated module for low-level quality control (*tsdiffana*) is also bundled with *aa*, which—thanks to the flexible modular concept—can be employed before or after various stages or even multiple times, which allows a user to follow how the data change during the analysis. Conveniently, these diagnostic results are collected into a central place in a multi-level fashion, allowing a user to browse both vertically (within-subject) and horizontally (between-subject). Where applicable (e.g., motion correction), between-subject visual comparison and/or statistics are also provided.

SYSTEM AND SOFTWARE REQUIREMENTS

- *aa* is developed in a *nix environment and actively used on machines running Ubunto, RedHat, and Mac OS X. It is not currently supported on Windows.
- *aa* is Matlab-based and requires a base installation of Matlab. Some functions may require additional toolboxes; for example the Image Processing Toolbox. In general, though, *aa* is written with the goal of minimizing use of Matlab toolboxes by using versions of functions included in the base Matlab installation or by recreating these functions.
- As a processing pipeline, *aa* does not include external software (such as SPM, FSL, etc.), which must be installed separately and placed in a user's path.

SOFTWARE ARCHITECTURE

This manuscript describes *aa* version 4.2. Not all components apply to earlier versions. The latest version is available from: http://automaticanalysis.org/getting-started/download-installation/. Here, we describe the components in the order a typical user might encounter them, providing a description of each and the motivation for the architecture. The earlier topics will be needed by any *aa* user, while the later ones are likely to be of more interest to experienced users.

USER SCRIPT

The core of an *aa* analysis is the user script, which describes what processing should happen, and what data it should be applied to. Almost all analyses will require the user to create a user script in Matlab, typically by modifying an example script (found in the "examples" folder distributed with *aa*). An example user script is shown below:

```
% Example aa version 4 user script
%
% Note: For an example of a complete user script,
please see:
%
% http://automaticanalysis.org/getting-started/worked-
example/
%

% Define study specific parameters
aap=aarecipe('aap_tasklist_typical_fmri.xml');

% Directory for analyzed data
aap.acq_details.root='/imaging/rhodri/camcan/cc_movie';

% Sub-directory for analyzed data
aap.directory_conventions.analysisid='data';

% Define subjects, and EPI series number ordered as
aas_addsession lines
aap=aas_addsubject(aap,'CBU110000_*',{6});
% One or more sessions
```

```
aap=aas_addsession(aap,'movie');

% How many dummies?
aap.acq_details.numdummies=3;

% (Note: for a full analysis events and contrasts need
to be added)

% Do processing
aa_doprocessing(aap);
```

This script executes a typical fMRI processing pipeline (discussed more in the next section) on a single subject (CBU110000) for a single session (imaging series 6, labeled "movie").

The user script can set parameters, such as output paths, or settings for modules. Here, three dummy scans are specified to be ignored in the analysis by the line:

```
aap.acq_details.numdummies=3
```

Note that the entire analysis—comprising the set of tasks to be run and the data they are to be run on—is described in a single structure (the "aap" variable). It is initially constructed by the *aarecipe* command. Because the analysis is fully specified by a single structure (along with the codebase), it is trivial to keep a record of the analysis, or to re-run it at a later date.

BASIC TASKLISTS

The tasklist is an XML format file that describes what should be done. A number of tasklists are available, many of which are useful without modification (**Table 1**).

Each tasklist describes a series of modules that should be executed. In the example user script given above, the tasklist specified was *aap_tasklist _typical_fmri.xml*. **Figure 1** shows the processing that will be run for this tasklist. Note that a subject's structural (T1) and fMRI (EPI) data go through a number of processing stages, and some modules operate on the data together. The XML code that underlies this tasklist is below.

```
<?xml version="1.0" encoding="utf-8"?>
<aap>
  <tasklist>
    <initialisation>
      <module><name>aamod_checkparameters</name>
         </module>
      <module><name>aamod_evaluatesubjectnames</name>
         </module>
      <module><name>aamod_study_init</name></module>
      <module><name>aamod_newsubj_init</name></module>
    </initialisation>
    <main>
      <module><name>aamod_autoidentifyseries_timtrio
         </name></module>
      <module><name>aamod_get_dicom_structural</name>
         </module>
      <module><name>aamod_get_dicom_epi</name></module>
      <module><name>aamod_convert_structural</name>
         </module>
      <module><name>aamod_convert_epis</name></module>
      <module><name>aamod_realign</name></module>
      <module><name>aamod_tsdiffana</name></module>
      <module><name>aamod_slicetiming</name></module>
      <module><name>aamod_coreg_noss</name></module>
      <module><name>aamod_norm_noss</name></module>
      <module><name>aamod_norm_write</name></module>
      <module><name>aamod_smooth</name></module>
      <module><name>aamod_firstlevel_model</name>
        </module>
      <module><name>aamod_firstlevel_contrasts</name>
        </module>
      <module><name>aamod_secondlevel_model</name>
        </module>
    </main>
  </tasklist>
</aap>
```

There are two sections to this simple tasklist. The "initialisation"[1] modules are run every time, for tasks such as checking the input parameters, or expanding wildcards in the subject names. The "main" modules are only run once on each piece of data, unless an explicit re-run is requested.

Note also that the dependencies (that is, which pieces of data act as the input to each module) are not usually explicitly specified in the tasklist. Instead, the pipeline is automatically connected up at the start of processing using information in each module's interface. This simplifies specification of tasklists, and allows modules to be reordered with reduced potential for error. The dependencies are reported at the start of an analysis.

OUTPUT FILE STRUCTURE

The example of an output file tree for an *aa* analysis is shown in **Figure 2**. The path to which this structure gets written is determined by the *aa* setting

```
aap.acq_details.root='/imaging/rhodri/mypath';
```

The name of the directory for the analysis is specified in:

```
aap.directory_conventions.analysisid='myanalysis';
```

Each module operates on data stored in a separate directory (e.g., *aamod_realign_00001, aamod_smooth_00001*). This differs from the conventions with packages such as SPM where all analysis stages are written to a single directory, often with different prefixes or suffixes to distinguish the stages. There are a number of practical benefits to *aa*'s directory separation. First, it reduces the number of files within subdirectories, which makes them more manageable, particularly for fMRI or DTI with a 3D data format

[1] British spellings are used throughout *aa*, reflecting its country of origin.

Table 1 | Example tasklists.

Tasklist	Purpose
aap_tasklist_typical_fmri.xml	fMRI preprocessing and first/second level statistics
aap_tasklist_fmri.xml	fMRI preprocessing and first/second level statistics—variant using fieldmaps, realignunwarp.
aap_tasklist_dartelvbm8.xml	VBM with SPM8 and DARTEL
aap_tasklist_diffusion.xml	Diffusion tractography with FSL
aap_tasklist_diffusion2.xml	Nonlinear DTI and DKI
aap_tasklist_freesurfer.xml	Structural processing with Freesurfer

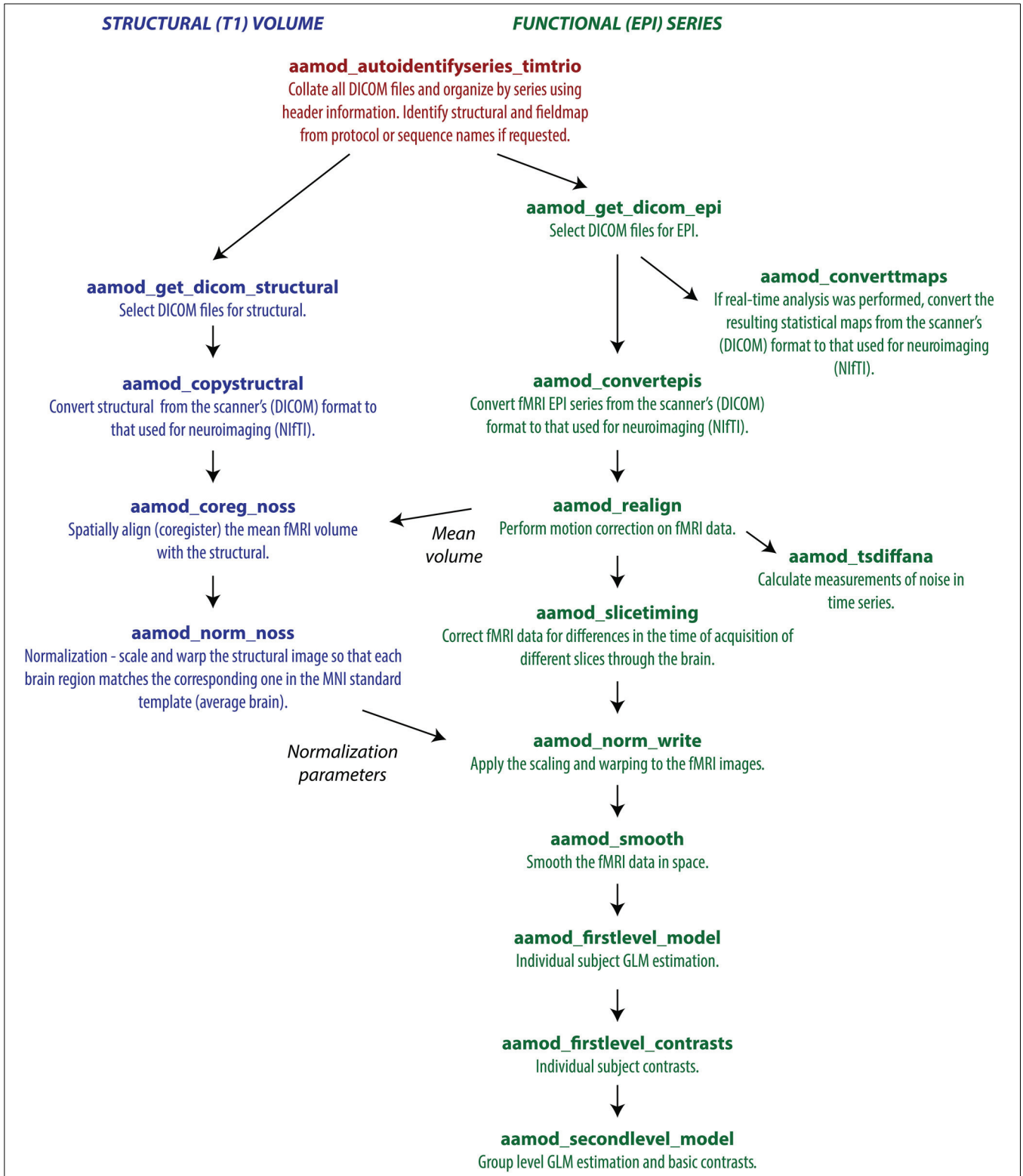


FIGURE 1 | A typical fMRI pipeline comprising a set of *aa* modules (filenames prefixed with aamod_). Blue colors refer to modules processing the structural, green colors processing the EPI, and red are general. This pipeline does preprocessing and first-level (individual) and second-level (group) statistics.

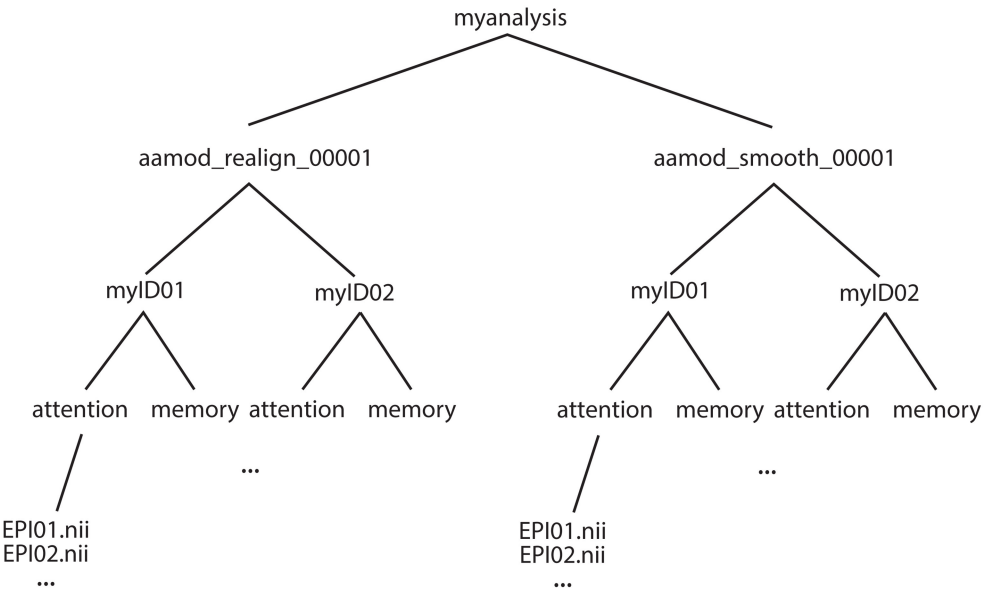


FIGURE 2 | Example file structure for *aa* output. Each analysis comprises output directories organized by processing stage (here, for example, realignment and smoothing) which are then each subdivided by subject, then session.

(e.g., one image per timepoint). Second, it makes it easier to see at a glance what processing has happened, and to find a file when browsing. Finally, it makes maintenance easier when, for example, a user wishes to delete intermediate stages of analysis to save disk space.

These ease-of-use and aesthetic advantages come along with more fundamental benefits. Partitioning the workspace of modules into separate directories facilitates the encapsulation of data. The *aa* engine is responsible for putting a module's input data into the directory in which it will execute. If a module does not request a piece of data, it will not be there, and it cannot accidentally be used. Similarly, the *aa* engine is responsible for picking up outputs and passing them along the pipeline. If a module does not explicitly declare an output, it will not be passed. Thus, directory separation allows the *aa* scheduling engine to maintain tight control of data dependencies. This has a number of benefits. It permits parallel processing with a reduced potential for conflicts due to unexpected module behaviors. When executing on a cluster, data transfer demands are reduced as a compute node does not need to receive the whole analysis, but only the specific data it is working on. Finally, the one-directory-per-module structure facilitates branched tasklists, where an analysis forks, and is continued in two different ways (e.g., with a smoothing kernel of 8 or 12 mm).

Here, both modules had the suffix _00001. If either module were present more than once in a tasklist (e.g., *tsdiffana* run before and after a processing stage), this index would be incremented by one for each subsequent entry.

Note that this architecture does not restrict the level at which a module can operate. That is, if data for all sessions and subjects are needed to complete an analysis, they will all be copied to the appropriate directory. However, as this is more often the exception than the rule, on the whole *aa*'s limited copying approach saves bandwidth and reduces opportunities for error.

MODULES

At the heart of every *aa* analysis are the modules. A module performs a single task, such as motion correction or smoothing. Some examples are given in **Table 2**.

Each module requires two files: an XML interface (e.g., *aamod_smooth.xml)*, and the corresponding Matlab source (e.g., *aamod_smooth.m)*. Occasionally, an interface file may specify a Matlab file with a different name to its source (e.g., *aamod_autoidentifyseries_ge.xml* points to *aamod_autoidentifyseries.m)* using an *mfile_alias*='…' attribute.

One of a module's most important properties, specified in this XML interface, is the "domain" at which it operates. Modules with a domain of "study" are called just once (i.e., a single instance is created each time the module occurs in the processing pipeline). Modules with a domain of "subject" are called once for each subject, while modules with a domain of "session" are called once for each session of each subject. These are the three most common module domains; others include diffusion_session, meg_session, and hyperalignment_searchlight_package. However, new domains can be easily added to the *aa* engine, and user-written modules can make use of new domains.

Instances of a module should restrict their processing to a particular set of input data (i.e., for a given session-domain module, there might be an instance for subject 3, session 2). This instance should take care to only attempt to process this portion of the data, and should never attempt to write data outside its domain (in this example, to another session).

Table 2 | Example *aa* modules.

Input data sorting

aamod_autoidentifyseries_timtrio
Scan input DICOM files to get series and acquisitions irrespective of filenames, which are typically site-specific. Identify structural and fieldmap series numbers.

Anatomy

Basic structural

aamod_get_dicom_structural
Find all DICOM files corresponding to the structural acquisition.

aamod_coreg_extended_1
Coregister an individual's structural to a standard space template using a rigid body transformation, which improves robustness of later normalization stage.

aamod_norm_noss
Estimate nonlinear warp that will transform an individual subject's space into a standard template space (SPM normalization).

aamod_norm_write
Apply normalization parameters derived from structural to EPIs.

DARTEL-VBM

aamod_biascorrect_segment8
Run New Segment (introduced in SPM 8) and save bias-corrected image (e.g., for segmenting).

aamod_segment8
Tissue class segmentation using New Segment (SPM 8).

aamod_structuralstats
Retrieve total tissue class volume and TIV from segmented images.

aamod_dartel_createtemplate
Use DARTEL to create a template.

aamod_dartel_normmni
Write DARTEL-warped images to MNI space.

aamod_normalizebytotalgray
Divide segmented images by total gray matter (proportional scaling).

aamod_norm_write_dartel
Apply normalization parameters derived using DARTEL to other modalities (e.g., EPI, contrasts, DWI, ROIs).

aamod_dartel_denorm
Transform images in standard MNI space (e.g., atlas labels) into native space based on normalization parameters derived using DARTEL (multimodal).

Freesurfer surface extraction

aamod_freesurfer_initialise
Prepare for a Freesurfer analysis.

aamod_freesurfer_deface
Defaces structural (T1) and produces a mask.

aamod_freesurfer_deface_apply
Apply defacing mask to a co-registered image.

aamod_freesurfer_autorecon_all
Runs a Freesurfer pipeline with recon-all.

Anatomical processing from FSL

aamod_fsl_FAST
Use FAST (FSL) for segmentation.

aamod_fsl_FIRST
Use FIRST (FSL) to characterize structure shape.

(Continued)

Table 2 | Continued

ANTS software

aamod_ANTS_epi2template
Create transformation matrix for ANTS normalization to study template.

aamod_ANTS_warp_ROIs
Apply inverse warp to ROIs.

aamod_ANTS_warp_cons
Apply warp to first level contrasts.

fMRI activation studies

fMRI preprocessing

aamod_get_dicom_epi
Find all DICOM files corresponding to the EPI acquisitions.

aamod_convert_epi
Convert the DICOM files to NIfTI format. Handles with multi-echo EPI with various weighting schemes.

aamod_realign
Perform motion correction with SPM.

aamod_slicetiming
Slice timing correction with SPM.

aamod_coreg_extended_2epi
Applies to the EPIs the transformation derived from coregistering the structural to a standard-space template (in aamod_coreg_extended_1). Then, fine-tunes the registration of the EPI to the structural with a further coregistration.

aamod_coreg_noss
Coregisters structural to mean EPI using SPM.

aamod_smooth
Smooth data.

Distortion correction

aamod_fieldmap_undistort
Use fieldmap (with phase and magnitude) to correct EPI distortions.

aamod_realignunwarp
Realign and unwarp from SPM.

aamod_pewarp_estimate

aamod_pewarp_write
Constrained nonlinear coregistration.

Statistics

aamod_firstlevel_model
Run first level statistical model. Simple specification of events in user script.

aamod_firstlevel_contrasts
Run first level contrasts. Simple specification of contrasts.

aamod_secondlevel_model
Run a *t*-test across subjects for every first level contrast.

aamod_OneWay_ANOVA
Run repeated measures (across subjects) one-way ANOVA.

Networks

Connectivity matrices

aamod_fconnmatrix_seedseed
Calculate seed-to-seed connectivity matrix from relationship of time-courses across seed regions.

PPI

aamod_vois_extract
Extract ROI timeseries after first level analysis.

(Continued)

Table 2 | Continued

aamod_ppi_prepare
Prepare PPI regressors based on ROI timeseries.

ICA

aamod_tensor_ica
Run individual or group tensor ICA.

Movie inter-subject correlation analysis

aamod_highpassfilter_epi
High-pass filter fMRI time series using discrete cosine model, like SPM.

aamod_meantimecourse
Calculate mean time course for each voxel across subjects.

aamod_moviecorr_meantimecourse
Calculate correlation of each subject's timecourse with mean.

aamod_moviecorr_summary
Statistics to find which correlations are significant across subjects.

Diffusion

Basic processing

aamod_get_dicom_diffusion
Get a list of all of the DICOM files that correspond to the diffusion series (typically, as identified by aamod_autoidentifyseries_timtrio).

aamod_convert_diffusion
Convert diffusion images from DICOM to NIfTI

aamod_3dto4d_diffusion
Convert diffusion images from 3D to 4D. The XML file is 'aamod_3dto4d_diffusion.xml' which refers to the matlab file (using mfile_alias) 'aamod_3dto4d.m'.

aamod_diffusion_eddycorrect
Use eddy_correct (FSL) to correct image distortions, head movements using affine registration to a reference volume (T2 image).

aamod_diffusion_extractnodif
Use FSL to extract the reference(s) image(s) (T2 image with *b*-value of 0), called nodif.

aamod_bet_diffusion
Use FSL to extract the brain of the nodif image. Brain extraction toolbox. Its "mfile" is aamod_bet.

Diffusion tensors

aamod_diffusion_dtifit
Use FSL to fit a diffusion tensor model at each voxel. Note that dtifit is not necessary in order to run probabilistic tractography (which depends on the output of BEDPOSTX).

aamod_diffusion_dkifit
Fit diffusion kurtosis parameters using linear model.

aamod_diffusion_dtinlfit
Fit diffusion tensor parameters using nonlinear model.

aamod_coreg_structural2fa
Coregister structural to diffusion image (dti_FA).

Probabalistic tractography

aamod_unnormalize_seeds
Use SPM to "unnormalize" the seeds (i.e., apply the inverse matrix to transform the seed (MNI space) to diffusion space).

aamod_unnormalize_targets
Use SPM to "unnormalize" the targets (i.e., apply the inverse matrix to transform the targets (MNI space) to diffusion space).

aamod_diffusion_bedpostx
Use FSL to apply bedpostx Monte Carlo modeling of PDFs of diffusion parameters.

(Continued)

Table 2 | Continued

aamod_diffusion_probtrackx
Use FSL to apply probtrackx, which repetitively samples from the distributions on voxel-wise principal diffusion directions, each time computing a streamline through these local samples to generate a probabilistic streamline or a sample from the distribution on the location of the true streamline.

aamod_diffusion_probtrackxsummarize_indv
Get the results of probtrackx (diffusion space) of each participant, merge the different splits and transform them to the MNI space.

aamod_diffusion_probtrackxsummarize_group
Averages the seed-to-target connectivity images across subjects, which we have used for visualization.

MVPA

aamod_MVPaa_brain_1st
Runs an MVPA searchlight on a set of beta or *t*-values (typically in native space).

aamod_MVPaa_brain_SPM
Convert results from searchlight into NIfTI images readable in SPM.

aamod_unnormalize_rois
Set ROIs from standard space into subject space.

aamod_MVPaa_roi_1st
Runs an MVPA analysis within an ROI, using a set of beta or *t*-values (typically in native space).

Other important properties of a module are the type of data (e.g., *epi* or *structural*) it requires as an input, and the type of data it produces as an output.

An example interface file, *aamod_smooth.xml*, is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
 <aap>
  <tasklist>
   <currenttask domain='session' desc='SPM
        smooth' modality='MRI'>
    <qsub>
     <timeBase>0.5</timeBase>
     <memoryBase>1</memoryBase>
    </qsub>
    <permanenceofoutput>2</permanenceofoutput>

    <FWHM>10</FWHM>
    <inputstreams>
     <stream ismodified='0'>epi</stream>
    </inputstreams>
    <outputstreams>
     <stream>epi</stream>
    </outputstreams>
   </currenttask>
  </tasklist>
</aap>
```

The domain is specified in the attributes of the "currenttask" line, along with a description (which is displayed to the user) and the modality of the data—here "MRI."

The next two sections are of less focus here. The "qsub" fields are estimates of the resources used by this module, for use by some parallel schedulers. The "permanenceofoutput" field is used by the garbage collection tool to delete less important, intermediate data prior to archiving. Higher numbers correspond to more important data.

More central to the function of this particular module, the "FWHM" field describes a setting of this module—in this case, the full-width half maximum of the smoothing kernel, in millimeters. There is then a description of the sorts of input data (or "streams") that this module requires, here only "epi" data, and the output data, again just "epi" for this module. The operation of these is discussed more in the next section. The Matlab code for a module implements the function.

CUSTOMIZING ANALYSIS PARAMETERS

In the *aa* user script, the *aarecipe* command sets the initial state of the *aap* structure that describes the analysis:

```
aap=aarecipe('aap_parameters_defaults.xml',
             'aap_tasklist_typical_fmri.xml');
```

The values in this *aap* structure come from three sources:

1. The file *aap_parameters_defaults.xml*, which contains general settings;
2. The tasklist XML file (here *aap_tasklist_typical_fmri.xml*);
3. The XML interface files for each of the modules in the tasklist.

The values returned by the *aarecipe* command are often customized in the user script. Any parameter in *aap* may by modified. Examples are:

```
aap.acq_details.numdummies=3;
aap.tasksettings.aamod_smooth.FWHM=8;
```

Alternatively, it is sometimes more convenient to create modified XML files. XML tasklists may set parameters for an individual instance of a module, with syntax like this:

```
<module>
  <name>aamod_smooth</name>
  <extraparameters>
    <aap><tasklist><currenttask><settings>
      <FWHM>8</FWHM>
    </settings></currenttask></tasklist></aap>
  </extraparameters>
</module>
```

It is also possible to create XML files that inherit the parameters from the standard files, and override a few of them. For example, one can create a site/study/specific version of *aap_parameters_defaults.xml*, such as *aap_parameters_defaults_CBSU.xml* (specific for the MRC Cognition and Brain Sciences Unit):

```
<?xml version="1.0" encoding="utf-8"?>
<aap xmlns:xi="http://www.w3.org/2001/XInclude">
  <xi:include href="aap_parameters_defaults.xml"
              parse="xml"/>
  <local>
        <directory_conventions>
              ...
        </directory_conventions>
        <options>
              ...
        </options>
  </local>
</aap>
```

in which most of the settings are imported from *aap_parameters_defaults.xml* using XML Inclusion (http://www.w3.org/TR/xinclude) and only the path-related settings are redefined in the <*local*/> section.

SPM defaults are a special case. These can be modified in the *aap.spm.defaults* structure.

SPECIFICATION OF STATISTICAL MODELS FOR fMRI

For users who wish to analyze fMRI data with *aa*, a simple set of commands is available for the specification of first-level statistical models. The format is:

```
aap=aas_addevent(aap,modulename,subject,session,
    eventname,ons,dur,parametric);
```

where:

```
modulename=module(e.g.,'aamod_firstlevel_model')
    for which this event applies
subject=subject for whom this model applies
session=session for which this applies
eventname=name of the stimulus or response event
ons=event onset times (in scans). Does not need
    to be sorted
dur=event durations (in scans), either a single
    element (if all occurrences have the same
    duration) or in order that corresponds to ons
parametric=parametric modulator (optional - can
    omit)
```

For example,

```
aap=aas_addevent(aap,'aamod_firstlevel_model','*','*',
   'VisualStimulus',[0:15:75],7.5);
```

specifies that every session of every subject was a block design, with a regressor titled "VisualStimulus" with onsets every 15 scans and a duration of 7.5 scans.

Using the "subject" and "session" fields, customized designs for each subject and/or session may be specified.

A contrast may then be specified with

```
aap=aas_addcontrast(aap,modulename,subject,format,
    vector,contype,automatic_movesandmeans)
```

where:

```
modulename= module (e.g.,'aamod_firstlevel_
contrasts') for which this contrast applies
subject=subject for whom this model applies
format=format for contrast specification, one of:
  * "sameforallsessions" - vector contains contrast
  to be applied to all sessions
  * "singlesession:[sessionname]" - vector contains
  contrast for just one session, all other sessions
  will be set to 0. [sessionname] should be
  replaced with name of that session.
  * "uniquebysession" - long contrast string that
  separately specifies contrast for every session
contype="T" or "F" (defaults to "T")
automatic_movesandmeans=1 or 0, add means & moves
to contrast automatically?
```

For example,

```
aap=aas_addcontrast(aap,'aamod_firstlevel_contrasts',
    '*','sameforallsessions',[1 -1]);
```

to contrast the first vs. the second column of every session in every subject.

If the desired second level model is to run a simple *t*-test for every contrast run in every subject at the first level, then the module *aamod_secondlevel_model* may be added to the tasklist. It does not require customization.

STREAMS

All data into and out of an instance of a module are managed by the *aa* engine. Each type of data is referred to as a "stream." Common streams are "epi," "structural," and "dicom_header." Note that these descriptions are deliberately unspecific about the state of the data—e.g., the data in the *epi* stream may be normalized, or not—as subsequent modules (e.g., first level statistics) often do not need to change their behavior to work on one kind of data or another.

A module's interface (XML file) describes the data streams that it requires wants as an input:

```
<inputstreams>
    <stream>epi</stream>
</inputstreams>
```

and what it produces as an output:

```
<outputstreams>
    <stream>realignment_parameter</stream>
    <stream>meanepi</stream>
    <stream>epi</stream>
</outputstreams>
```

This information is then used to connect up the pipelines of data from one module to the next. So, for example, if a tasklist contains:

```
<module><name>aamod_realign</name></module>
<module><name>aamod_tsdiffana</name></module>
<module><name>aamod_slicetiming</name></module>
```

The module *aamod_slicetiming* requests an *epi* input. The quality control module *aamod_tsdiffana* does not produce an *epi* output, so *aa* looks further back up the tasklist (see **Figure 1**). It finds that *aamod_realign* produces an *epi* ouput, and so it will pass the *epi* output of *aamod_realign* to *aamod_slicetiming*. This automatic connection of pipelines makes it straightforward to rearrange modules.

A complexity that is largely hidden from the user is that dependencies are calculated at the level of particular instances of a module, and are affected by the domains at which the source and target modules operate. Consider this fragment of a tasklist:

```
<module><name>aamod_norm_write</name></module>
<module><name>aamod_smooth</name></module>
```

Both *aamod_norm_write* and *aamod_smooth* operate on the domain of single EPI sessions for single subjects. The instance of the module *aamod_smooth* that processes subject 4, session 2, only needs the data from the instance of the module *aamod_norm_write* that has processed subject 4, session 2, and so only the corresponding data are is passed to the module instance. Furthermore, when executing in parallel, each *aamod_smooth* instance may execute as soon as the corresponding *aamod_norm_write* module has completed, and it does not need to wait for any others to finish. Although transparent to the user, dependencies become more complicated when the domain of a module that is the source of a given stream is different from the domain of a module that is the target of that stream. The restriction that is enforced is that any module may only write data at the level of its domain or lower (i.e., not sideways or above in **Figure 2**). However, modules may read from levels up toward the trunk, but never sideways.

THE SCHEDULING ENGINE AND PARALLEL PROCESSING

The *scheduling engine* executes all analyses described within the *aap* structure. The command included in every user script is:

```
aap=aa_doprocessing(aap);
```

This executes an *aa* analysis. To do this, it builds a map of all the instances of all the modules that need to be executed, and the data dependencies between them.

To test whether an instance of a module needs to be executed, *aa* looks for a file named *done_aamod_[modulename]_[index]*. This file will be stored in the root directory of the instance: for a session domain module, in the session directory. If it exists, that instance is considered to have been completed, and will not be re-run. The exception to this rule is an earlier module instance in the pipeline needing to be rerun, on which this module instance is dependent. This will cause the *done_* flag to be deleted, and the module will be re-run.

aa_doprocessing examines the field *aap.options.wheretoprocess* to decide how to run these modules. If the field has a value "localsingle" it will step through these modules one at a time, in the current Matlab process (as implemented in the object *@aaq_localsingle*). If it has the value "qsub" it will use the parallel computing toolbox component "createTask" to submit a job. If it has the value "condor" it will compile the job and submit it to a condor queuing system, using the shell script specified in *aap.directory_conventions.condor_wrapper*. *@aaq_matlab_pct* uses Matlab's parallel computing toolbox.

Ultimately, regardless of the scheduling mechanism, instances of modules are run by calls to the *aa_doprocessing_onetask* function.

BRANCHED TASKLISTS

Neuroimaging studies frequently require data to be analyzed in different ways. This might be because there is some uncertainty on the ideal parameters or analysis strategy (for example, whether motion correction should be performed before or after slice timing correction, or what smoothing kernel should be used). Alternatively, it might be because the data are to be analyzed in a number of different ways—with ICA, with

conventional univariate fMRI, with MVPA, and with functional connectivity[2].

Traditionally, these scenarios would probably involve either creating entirely independent pipelines, or processing to the branch point, making a copy of the analyzed data in a different directory, and then taking the new analysis forwards. By contrast, *aa* provides a straightforward way of specifying branched tasklists, as in the following fragment:

```
...
<module>
    <branch>
        <analysisid_suffix>_realign_then_slicetime
        </analysisid_suffix>
        <module><name>aamod_realign</name></module>
        <module><name>aamod_slicetiming</name></module>
    </branch>
    <branch>
        <analysisid_suffix>_slicetime_then_realign
        </analysisid_suffix>
        <module><name>aamod_slicetiming</name></module>
        <module><name>aamod_realign</name></module>
    </branch>
</module>
...
```

In this command, *<analysisid_suffix>* is included within each branch, so that the two branches get separated into different directories. Although tidy, this is not strictly necessary, as the duplicated modules will be suffixed with different indices—e.g., in the first branch realignment will be output to *aamod_realign_00001* and the second to *aamod_realign_00002*.

FULLY QUALIFIED STREAM REFERENCES

By default, the input for a stream to a module comes from the last module in the tasklist that outputs that kind of data. Often, this is the desirable behavior. However, sometimes, an explicit earlier reference may be desired. This can be achieved with a fully qualified stream reference comprising *[module-name].[stream-name]* as in this example:

```
<inputstreams>
    <stream>aamod_realign_00001.epi</stream>
</inputstreams>
```

ADJUSTING DEFAULTS, AND SITE-SPECIFIC CONFIGURATION

There are at least two ways a user may customize *aa* for a particular site. One way is to have a site-specific configuration file, conventionally called *aas_localconfig_[sitename]*. This is then inserted into the user script, soon after the recipe command, with the line:

```
aap=aas_localconfig_[sitename](aap);
```

Another way is to create a customized *aap_parameters_defaults.xml* file, typically by including the existing *aap_parameters_defaults.xml* file and then overriding some parameters for this local installation, like this:

```
<?xml version="1.0" encoding="utf-8"?>
<aap xmlns:xi="http://www.w3.org/2001/XInclude">
    <xi:include href="aap_parameters_defaults.xml"
    parse="xml"/>
    <local>
        <directory_conventions>
            <rawdatadir desc='Subdirectories to
            find raw MRI data'
ui='dir_list'>/mridata/cbu:/mridata/csl:/mridata/
camcan</rawdatadir>
        </directory_conventions>
    </local>
</aap>
```

INPUT DATA FORMAT

A user must prepare raw data in a form acceptable for input to *aa*. The easiest starting point is typically the raw DICOM data, exported as a set of files from the scanner. One challenge we faced in porting *aa* between sites was that the dumping of the raw data out of DICOM database (PACS) systems led to idiosyncratic filename and directory structures. *aa* will automatically scan the data and structure it into acquisition series for Siemens and GE scanners, provided all of the files from each subject can be isolated into one directory (or a directory with subdirectories). No particular naming convention is required, other than a consistent filename extension for the DICOM files. The DICOM headers are used to organize the files. The system may work also on data from other scanner manufacturers, but we have not tested it.

In a user's tasklist (or later, as a site-specific configuration) the dicomfilter can be set, typically to one of:

```
aap.directory_conventions.dicomfilter='*.dcm';
    % if DICOM files end in.dcm
aap.directory_conventions.dicomfilter='*.ima';
    % if DICOM files end in.ima
aap.directory_conventions.dicomfilter='*';
    % if only DICOM in raw data directories
```

For any tasklist, setting the first main module to *aamod_autoidentifyseries_timtrio* for data from Siemens scanners, or *aamod_autoidentifyseries_ge* with GE scanners, will identify the DICOM files.

Provided researchers use a consistent name for their structural scans, these scans can be automatically identified by setting:

```
aap.options.autoidentifystructural=true;
aap.directory_conventions.protocol_structural='MPRAGE';
```

The first line requests automatic scanning for the structural (the default), and the second, which protocol should be sought. If a user sometimes acquires more than one structural (for example, if a subject moves) but always stops once they have a good one, it is possible to specify that in this circumstance the last structural is the one to be used:

```
aap.options.autoidentifystructural_chooselast=true;
```

A second alternative is to use data already converted into NIfTI format. This is possible, either by using the *aas_addinitialstream*

[2]Of course, care must be taken when trying out multiple analysis options, and exploration is best done on independent data so as not to bias the results. Our point is that there are many instances in which researchers might reasonably want to compare analysis strategies in a systematic way, which *aa* facilitates.

command in the user script, or the *aamod_epifromnifti* module. However, detailed instructions for doing so are beyond the scope of this overview.

CONNECTING PIPELINES

It is often the case that a researcher will want to analyze a subset of data from a larger database, or continue an analysis that exists in a different location (i.e., a remote location). For example, a lab might store and preprocess all their subject MRI data—fMRI, structural images, and diffusion images—on a central server, but one user might want to only analyze the fMRI data from a few subjects on their local machine. *aa* allows a user to easily accomplish this by creating an analysis script that connects to the *aa* pipeline on the central server; the user does not have to manually copy and import any data. The new analysis does not replicate any of the modules or data on the central server, but instead connects the input streams of the local analysis to the data output streams in the remote location. By default, the connection is made to the terminal end of the remote pipeline (i.e., the final instance of each output stream), but the user can easily specify a connection to an earlier stage of processing (e.g., to take the EPI stream before the normalization stage). Furthermore, every time the local analysis is executed, *aa* will check to see if the remote data have changed, and re-run any local modules that depend on those data. The ability to connect pipelines facilitates data sharing within and between labs, promotes good practices for organizing and storing data, reduces data duplication, and simplifies the process of starting new analyses on existing data sets. Detailed examples of this feature are provided in the *aa* documentation.

COMMUNITY

aa has been used for hundreds of analyses covering many thousands of participants. It is currently supported by a small but active base of coders.

BRAIN AND MIND INSTITUTE, WESTERN UNIVERSITY, LONDON, CANADA

Authors Rhodri Cusack, Annika C. Linke, Conor J. Wild and colleagues at the Brain and Mind Institute are actively developing for *aa*, and use it for fMRI, DTI and structural data from a variety of MRI scanners—Siemens 3T (Trio, Prisma), Siemens 7T, and GE 1.5 T (MR450w)—and EEG (EGI, Grass).

MRC COGNITION AND BRAIN SCIENCES UNIT, CAMBRIDGE, UNITED KINGDOM

In addition to authors Tibor Auer and Daniel J. Mitchell a handful of other coders in the Unit also actively participate in developing *aa* modules. In the Unit, *aa* is the backbone of analysing fMRI, DTI, MTR and structural data from Siemens 3T (Trio, Prisma) MRI scanner, Elekta Neuromag Vectorview MEG scanner and Brain Products BrainAmp EEG. New colleagues are introduced to *aa* right from the start by means of workshops, which allow them to perform analysis quite early on. A highlighted project, the Cambridge Centre for Aging and Neuroscience, involving multiple sessions of hundreds of subjects, also employs *aa*, which ensures both high consistency via standardized user scripts and tasklists and high processing speed via parallelization. The Unit also hosts a wiki (http://imaging.mrc-cbu.cam.ac.uk/imaging/AA) complementing the *aa* documentation.

DONDERS CENTER FOR COGNITIVE NEUROSCIENCE, NIJMEGEN, THE NETHERLANDS

Author Alejandro Vicente-Grabovetsky and colleagues in the Doeller laboratory are actively developing for *aa*, and use it for Siemens 3T and 7T fMRI analyses.

WASHINGTON UNIVERSITY IN ST LOUIS

Author Jonathan E. Peelle and his laboratory are developing structural and functional MRI analysis for Siemens 3T data.

GITHUB SOURCE CONTROL, SUPPORT, AND DOCUMENTATION

The codebase is maintained at: https://github.com/rhodricusack/automaticanalysis.

There are two main branches: the *master* branch, which contains a recent stable release, and the *devel-share* branch, which contains the latest versions of the code published by each of our sites. There are also occasional releases, under "tags," which contain frozen past versions of the code.

A website (http://automaticanalysis.org) contains the latest documentation for the code, and an issues discussion forum is used to report bugs or ask questions (https://github.com/rhodricusack/automaticanalysis/issues).

OTHER DESIGN DECISIONS

Our software provides access to most functions of SPM, one of the most commonly used neuroimaging tools worldwide, for analyses such as fMRI modeling and voxel-based morphometry. For several diagnostics in general and DWI analysis we use the well-established FSL functions, and for cortical-surface based measures, Freesurfer.

LIMITATIONS

Every processing approach has limitations, and *aa* is no different. Perhaps the biggest hurdle for novices is the requirement of knowing enough Matlab to organize analyses. The choice of Matlab as a programming language grew out of the origins of *aa* as a pipeline for SPM. There are clearly advantages and disadvantages to this choice. Matlab is widely used in neuroimaging, other areas of neuroscience, engineering and finance, and Matlab programming is a skill that is transferrable to other disciplines. The language provides an enormous library of high-level mathematical functions that are well tested, and in most cases highly optimized. It provides compact and elegant syntax for matrix math. It has a mature integrated-development environment (IDE) including line-by-line debugging, workspace inspection, computation time profiling, and 2D/3D graphics. It is a well-supported product, with regular updates and new features. A disadvantage is that as a commercial product, it comes with substantial costs, and is not open-source, reducing the potential for quality assurance and innovation directly from the community. However, Matlab does come with a compiler, allowing functions to be redistributed freely (but not to be changed), and it has an active user software exchange.

Like most pipelines that serve as interfaces to other tools, *aa* can be a bottleneck: one can only incorporate into a pipeline those tools that are already "wrapped" into *aa*. For example, there are currently no *aa* modules for *AFNI* tools. However, *aa*'s open source nature and its easy extendibility allow the user to implement the corresponding functionality and even to make it available to others.

Another consequence of automated pipelines such as *aa* is that they facilitate the processing of large datasets, in turn producing more data and increasing demands for file storage. Although *aa* attempts to keep only necessary files through garbage collection, analyses can quickly take up large amounts of disk space if not kept in check, which may prove to be a limitation in some contexts.

Finally, there is always the danger when using automated batch analysis pipelines that the researcher might try every possible combination of analysis tools and parameters —so-called "experimenter degrees of freedom"—to obtain the desired results. This is not a new problem in neuroimaging, but *aa* at least provides a way for researchers to keep track of different analysis approaches through branched tasklists and detailed analysis logs.

Despite these possible limitations, we believe that *aa* is successful in balancing the diverse needs of neuroimagers, and facilitating open, reproducible science on datasets of many sizes and complexities.

ACKNOWLEDGMENTS

The *aa* team would like to thank the many people who have contributed code that has been incorporated into, or inspired, *aa* modules. These include Matthew Brett, Rik Henson, Jason Taylor, and Adam Hampshire. Work reported here was supported in part by NSERC/CIHR CHRP (201110CPG), NSERC Discovery and the Canada Excellence Research Chair (CERC) in Cognitive Neuroimaging, grants R01AG038490 and R01DC013063 from the US National Institutes of Health, and The Dana Foundation. We are grateful to the Organization for Human Brain Mapping and the Seattle Local Organizing Committee from the 2013 conference for their support.

REFERENCES

Ashburner, J. (2007). A fast diffeomorphic image registration algorithm. *Neuroimage* 38, 95–113. doi: 10.1016/j.neuroimage.2007.07.007

Ashburner, J. (2009). Computational anatomy with the SPM software. *Magnetic Resonance Imaging*, 27, 1163–1174. doi: 10.1016/j.mri.2009.01.006

Behrens, T. E. J., Berg, H. J., Jbabdi, S., Rushworth, M. F. S., and Woolrich, M. W. (2007). Probabilistic diffusion tractography with multiple fibre orientations: what can we gain? *Neuroimage* 34, 144–155. doi: 10.1016/j.neuroimage.2006.09.018

Biswal, B. B., Mennes, M., Zuo, X.-N., Gohel, S., Kelly, C., Smith, S. M., et al. (2010). Toward discovery science of human brain function. *Proc. Natl. Acad. Sci. U.S.A.* 107, 4734–4739. doi: 10.1073/pnas.0911855107

Calhoun, V. D., Liu, J., and Adali, T. (2009). A review of group ICA for fMRI data and ICA for joint inference of imaging, genetic, and ERP data. *Neuroimage* 45, S163–S172. doi: 10.1016/j.neuroimage.2008.10.057

Friston, K. J., Holmes, A. P., and Worsley, K. J. (1999). How many subjects constitute a study? *Neuroimage* 10, 1–5. doi: 10.1006/nimg.1999.0439

Haynes, J.-D., and Rees, G. (2006). Decoding mental states from brain activity in humans. *Nat. Rev. Neurosci.* 7, 523–534. doi: 10.1038/nrn1931

Kay, K. N., Rokem, A., Winawer, J., Dougherty, R. F., and Wandell, B. A. (2013). GLMdenoise: a fast, automated technique for denoising task-based fMRI data. *Front. Neurosci.* 7:247. doi: 10.3389/fnins.2013.00247

Kriegeskorte, N., Goebel, R., and Bandettini, P. (2006). Information-based functional brain mapping. *Proc. Natl. Acad. Sci. U.S.A.* 103, 3863–3868. doi: 10.1073/pnas.0600244103

Kundu, P., Inati, S. J., Evans, J. W., Luh, W.-M., and Bandettini, P. A. (2012). Differentiating BOLD and non-BOLD signals in fMRI time series using multi-echo EPI. *Neuroimage* 60, 1759–1770. doi: 10.1016/j.neuroimage.2011.12.028

Mueller, S. G., Weiner, M. W., Thal, L. J., Petersen, R. C., Jack, C. R., Jagust, W., et al. (2005). Ways toward an early diagnosis in Alzheimer's disease: the Alzheimer's disease neuroimaging initiative (ADNI). *Alzheimer Dement.* 1, 55–66. doi: 10.1016/j.jalz.2005.06.003

Mumford, J. A, and Nichols, T. E. (2008). Power calculation for group fMRI studies accounting for arbitrary design and temporal autocorrelation. *Neuroimage* 39, 261–268. doi: 10.1016/j.neuroimage.2007.07.061

Norman, K. A., Polyn, S. M., Detre, G. J., and Haxby, J. V. (2006). Beyond mind-reading: multi-voxel pattern analysis of fMRI data. *Trends Cognit. Sci.* 10, 424–430. doi: 10.1016/j.tics.2006.07.005

Poldrack, R. A., Barch, D. M., Mitchell, J. P., Wager, T. D., Wagner, A. D., Devlin, J. T., et al. (2013). Toward open sharing of task-based fMRI data: the openfMRI project. *Front. Neuroinform.* 7:12. doi: 10.3389/fninf.2013.00012

Power, J. D., Barnes, K. A., Snyder, A. Z., Schlaggar, B. L., and Petersen, S. E. (2012). Spurious but systematic correlations in functional connectivity MRI networks arise from subject motion. *Neuroimage* 59, 2142–2154. doi: 10.1016/j.neuroimage.2011.10.018

Schaer, M., Cuadra, M. B., Tamarit, L., Lazeyras, F., Eliez, S., and Thiran, J.-P. (2008). A surface-based approach to quantify local cortical gyrification. *IEEE Trans. Med. Imaging* 27, 161–170. doi: 10.1109/TMI.2007.903576

Smith, S. M., Jenkinson, M., Johansen-Berg, H., Rueckert, D., Nichols, T. E., Mackay, C. E., et al. (2006). Tract-based spatial statistics: voxelwise analysis of multi-subject diffusion data. *Neuroimage* 31, 1487–1505. doi: 10.1016/j.neuroimage.2006.02.024

Van Essen, D. C., Ugurbil, K., Auerbach, E., Barch, D., Behrens, T. E. J., Bucholz, R., et al. (2012). The human connectome project: a data acquisition perspective. *Neuroimage* 62, 2222–2231. doi: 10.1016/j.neuroimage.2012.02.018

Van Horn, J. D., Grethe, J. S., Kostelec, P., Woodward, J. B., Aslam, J. A., Rus, D., et al. (2001). The functional magnetic resonance imaging data center (fMRIDC): the challenges and rewards of large-scale databasing of neuroimaging studies. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 356, 1323–1339. doi: 10.1098/rstb.2001.0916

Winkler, A. M., Sabuncu, M. R., Yeo, B. T. T., Fischl, B., Greve, D. N., Kochunov, P., et al. (2012). Measuring and comparing brain cortical surface area and other areal quantities. *Neuroimage* 61, 1428–1443. doi: 10.1016/j.neuroimage.2012.03.026

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 18 September 2014; accepted: 22 December 2014; published online: 15 January 2015.

*Citation: Cusack R, Vicente-Grabovetsky A, Mitchell DJ, Wild CJ, Auer T, Linke AC and Peelle JE (2015) Automatic analysis (aa): efficient neuroimaging workflows and parallel processing using Matlab and XML. Front. Neuroinform. **8**:90. doi: 10.3389/fninf.2014.00090*

This article was submitted to the journal Frontiers in Neuroinformatics.

frontiers in Neuroinformatics

ORIGINAL RESEARCH
published: 24 April 2015
doi: 10.3389/fninf.2015.00012

Reproducibility of neuroimaging analyses across operating systems

*Tristan Glatard[1,2], Lindsay B. Lewis[1], Rafael Ferreira da Silva[3], Reza Adalat[1], Natacha Beck[1], Claude Lepage[1], Pierre Rioux[1], Marc-Etienne Rousseau[1], Tarek Sherif[1], Ewa Deelman[3], Najmeh Khalili-Mahani[1] and Alan C. Evans[1]**

[1] McConnell Brain Imaging Centre, Montreal Neurological Institute, McGill University, Montreal, QC, Canada, [2] Centre National de la Recherche Scientifique, University of Lyon, INSERM, CREATIS, Villeurbanne, France, [3] Information Sciences Institute, University of Southern California, Marina del Rey, CA, USA

Neuroimaging pipelines are known to generate different results depending on the computing platform where they are compiled and executed. We quantify these differences for brain tissue classification, fMRI analysis, and cortical thickness (CT) extraction, using three of the main neuroimaging packages (FSL, Freesurfer and CIVET) and different versions of GNU/Linux. We also identify some causes of these differences using library and system call interception. We find that these packages use mathematical functions based on single-precision floating-point arithmetic whose implementations in operating systems continue to evolve. While these differences have little or no impact on simple analysis pipelines such as brain extraction and cortical tissue classification, their accumulation creates important differences in longer pipelines such as subcortical tissue classification, fMRI analysis, and cortical thickness extraction. With FSL, most Dice coefficients between subcortical classifications obtained on different operating systems remain above 0.9, but values as low as 0.59 are observed. Independent component analyses (ICA) of fMRI data differ between operating systems in one third of the tested subjects, due to differences in motion correction. With Freesurfer and CIVET, in some brain regions we find an effect of build or operating system on cortical thickness. A first step to correct these reproducibility issues would be to use more precise representations of floating-point numbers in the critical sections of the pipelines. The numerical stability of pipelines should also be reviewed.

Keywords: reproducibility, operating systems, Freesurfer, CIVET, FSL

OPEN ACCESS

Edited by:
Andrew P. Davison, Centre National de la Recherche Scientifique, France

Reviewed by:
Michael Hanke, Otto-von-Guericke University, Germany
Marc De Kamps, University of Leeds, UK

***Correspondence:**
Alan C. Evans, ACElab, McConnell Brain Imaging Centre, Montreal Neurological Institute, McGill University, 3801 University Street, Webster 2B #208, Montreal, QC H3A 2B4, Canada
alan.evans@mcgill.ca

Received: *23 January 2015*
Accepted: *08 April 2015*
Published: *24 April 2015*

Citation:
Glatard T, Lewis LB, Ferreira da Silva R, Adalat R, Beck N, Lepage C, Rioux P, Rousseau M-E, Sherif T, Deelman E, Khalili-Mahani N and Evans AC (2015) Reproducibility of neuroimaging analyses across operating systems. Front. Neuroinform. 9:12. doi: 10.3389/fninf.2015.00012

1. Introduction

Neuroimaging pipelines are known to generate different results depending on the computing platform where they are compiled and executed (Krefting et al., 2011; Gronenschild et al., 2012). Such reproducibility issues, also known as computing noise, arise from variations in hardware architectures and software versions. The state-of-the-art solution to deal with these issues is to restrict studies to a single computing platform (hardware and software), which has several drawbacks: (i) results may not be reproducible over time, when the computing platform used to produce them becomes obsolete; (ii) the use of High-Performance Computing (HPC) is limited to homogeneous sets of platforms, while available platforms are increasingly versatile; (iii) in some cases, homogenizing computing platforms is not even feasible, for instance when shared databases are processed in different institutions. Before such reproducibility issues can be resolved, a first step is

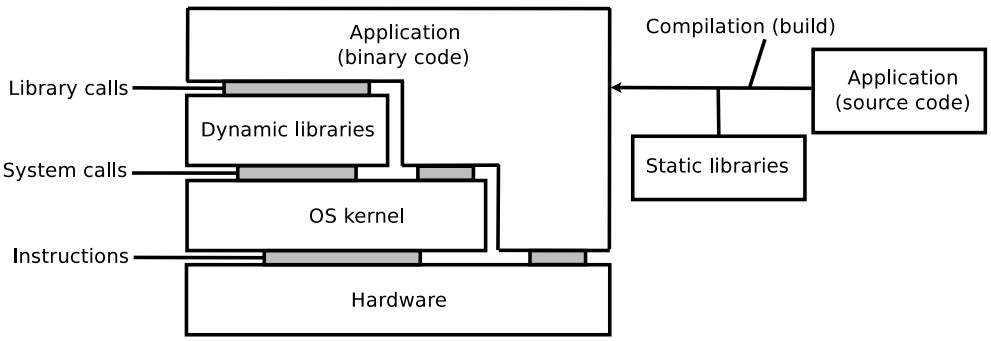


FIGURE 1 | Source code, compilation, libraries, kernel and hardware.

to properly quantify and explain them in various use-cases, which is the objective of this paper.

As illustrated on **Figure 1**, the execution of an application depends on its source code, on the compilation process, on software libraries, on an operating system (OS) kernel, and on a hardware processor. Libraries may be embedded in the application, i.e., statically linked, or loaded from the OS, i.e., dynamically linked. The reproducibility of results may be influenced by any variation in these elements, in particular: versions of the source code, compilation options, versions of the dynamic and static libraries (in particular when these libraries implement mathematical functions), or architecture of hardware systems. Some programming languages, for instance MATLAB, Java, Python, Perl, and other scripting languages, additionally rely on a specific runtime software, which can further influence the results.

On GNU/Linux, a dominant OS in neuroimaging (Hanke and Halchenko, 2011) and in HPC[1], applications rely on the GNU C library, `glibc`[2], which includes a mathematical library, `libmath`. New versions of `glibc` are released regularly, and subsequently adopted by distributions of the GNU/Linux OS, sometimes several years later. We focus on the differences generated by different library versions, which we call *inter-OS* differences for dynamically-linked applications, and *inter-build* differences for statically-linked applications. *Inter-run* differences, that is, differences between runs of the same application on the same platform may also occur, for instance when applications use pseudo-random numbers (this particular case can be addressed by forcing the seed number used to initialize the pseudo-random number generator).

This paper reports on our experiments with three of the main neuroimaging tools: the FMRIB Software Library (FSL, Jenkinson et al., 2012), Freesurfer (Fischl, 2012), and CIVET (Ad-Dabbagh et al., 2006). We quantify the reproducibility of tissue classification (cortical and subcortical), resting-state fMRI analysis, and cortical thickness extraction, using different builds of the tools, deployed on different versions of GNU/Linux. We also identify some causes of these differences, using library-call and system-call interception. The paper closes with a discussion suggesting directions to address the identified reproducibility issues.

[1]http://www.top500.org
[2]http://www.gnu.org/software/libc

TABLE 1 | Operating systems and analysis software.

	Cluster A	Cluster B
Applications	Freesurfer 5.3.0, build **1**	Freesurfer 5.3.0, build **1** and **2**
	FSL 5.0.6, build **1**	FSL 5.0.6, build **1** and **2**
	CIVET 1.1.12-UCSF, build **1**	CIVET 1.1.12-UCSF, build **1**
Interpreters	`Python` 2.4.3, `bash` 3.2.25,	`Python` 2.7.5, `bash` 4.2.47,
	`Perl` 5.8.8, `tcsh` 6.14.00	`Perl` 5.18.2, `tcsh` 6.18.01
`glibc` version	2.5	2.18
OS	CentOS 5.10	Fedora 20
Hardware	x86_64 CPUs (Intel Xeon)	x86_64 CPUs (Intel Xeon)

2. Materials and Methods

2.1. Operating Systems and Applications

Table 1 summarizes the platforms used in our experiments. We used two HPC clusters with Red-Hat-like Linux distributions: (**A**) CentOS release 5.10, running `glibc` 2.5 released in 2006, and (**B**) Fedora release 20, running `glibc` 2.18 released in 2013. We installed Freesurfer 5.3.0 and FSL 5.0.6 on these clusters using the 64-bit binaries released on their respective websites[3,4]. We used the Freesurfer CentOS 4 (**1**) and CentOS 6 (**2**) builds[5], and the FSL CentOS 5 (**1**) and CentOS 6 (**2**) builds[6]. We compiled and installed CIVET version 1.1.12-UCSF on cluster **A**, and used the same build on cluster **B**.

Freesurfer releases mainly consist of statically-linked executables and `tcsh` scripts. Dynamically-linked executables and `Perl` scripts are also present, in the `mni` directory where the `minc` tools are installed. The main differences between the CentOS 4 and CentOS 6 builds are the version of the `gcc` compiler potentially used to compile them (`gcc` 3.x on CentOS 4 vs. `gcc` 4.y on CentOS 6), and the `glibc` versions embedded in the executables (`glibc` 2.3 on CentOS 4 vs. 2.12 on CentOS 6). FSL and CIVET consist of dynamically linked executables which depend

[3]http://freesurfer.net/fswiki/Download
[4]http://fsl.fmrib.ox.ac.uk/fsldownloads/fsldownloadmain.html
[5]md5sum: 084d341cdf98305127aaeee48a6f4e0b and 6642289df823ebc27de52af57e9b3989.
[6]md5sum: 4d3a170d2311fa1c7e3cf6efd13f51a5 and 6cf9e3e58b35948416f833a21f495bd8.

on `libmath` and other libraries. FSL also contains `Tcl` (provided with the FSL release), `bash` and `Python` scripts, while CIVET has `Perl` and `bash` scripts.

All data movements and task executions on the clusters were performed with the CBRAIN platform for High-Performance Computing (Sherif et al., 2014).

2.2. FSL: Tissue Classification

We used 1.5T T1-weighted MR images from 150 subjects of the International Consortium for Brain Mapping (ICBM, Mazziotta et al., 2001). First, non-brain tissue was removed from the images with FSL BET (Brain Extraction Tool, Smith, 2002), using the default parameters and no options. Next, for cortical and subcortical tissue classification, we used FSL FAST (FMRIB's Automated Segmentation Tool, Zhang et al., 2001) and FSL FIRST (FMRIB's Linear Image Registration Tool, Patenaude et al., 2011) with the default parameters and no options. The experiment was repeated twice in each execution condition to ensure that no inter-run differences were present. Differences were first identified from file checksums. When checksums did not match, classification results were compared using the Dice similarity index (Dice, 1945) (global measure), and the sum of binarized differences across subjects (local measure).

2.3. FSL: Resting-state fMRI

We used 37 resting-state fMRI (RSfMRI) data arbitrarily selected from an ADNI-GO[7] dataset (site 130). All fMRI volumes were collected on a 3T Achieva Philips Medical Systems scanner with a gradient echo EPI (TR/TE = 3000/30 ms; Flip Angle = 80.0°; 64.0 × 64.0 inplane isotropic resolution of 3.3125 mm and slice thickness of 3.313 mm). Each RSfMRI dataset contained 140 volumes. Structural images were obtained using a manufacturer T1W MPRAGE sequence.

RSfMRI analysis was carried out using Probabilistic Independent Component Analysis (ICA, Beckmann and Smith, 2004) as implemented in MELODIC (Multivariate Exploratory Linear Decomposition into Independent Components) Version 3.14. We executed MELODIC with FSL build **1**, with the default parameters and different initializations of the random seed: (**a**) fixed, and (**b**) variable (time-based), which is the default. We also varied the dimension of the space of independent components: (**c**) dimension set to 20, and (**d**) automatic dimension detection using the Laplace approximation to the Bayesian evidence of the model order (Minka, 2000; Beckmann and Smith, 2004), which is the default. For variable random seeds, we re-executed MELODIC twice on each cluster to measure the inter-run variability.

We compared results between clusters **A** and **B** by computing the Dice coefficient between their binarized thresholded components, distinguishing the negative and positive parts of the components. As components may not be ordered consistently between **A** and **B**, each component in **A** was matched to the maximally correlated component in **B** using FSL's `fslcc`. Because this operation is not symmetric, we included Dice coefficients for both **A**–**B** and **B**–**A**. In case **d**, we also compared the number of dimensions detected on cluster **A** vs. cluster **B**.

Then, we analyzed the inter-OS differences between fMRI pre-processing steps. Using `fslmaths` and `fslstats`, we computed the mean absolute difference after motion correction, thresholding, spatial smoothing, intensity normalization, and temporal filtering. For motion correction, we also determined the residual rigid transformation $T_1 \circ T_2^{-1}$ at each timepoint, where T_1 and T_2 are the transformations obtained on the different clusters. We measured the norm of the translation vector and the absolute value of the rotation angle of this residual transformation.

2.4. Freesurfer and CIVET: Surface Segmentation and Cortical Thickness Extraction

Cortical thickness maps were generated with Freesurfer and CIVET from the same ICBM dataset used in Section 2.2. In our Freesurfer analysis, we performed all stages of cortical reconstruction using the `recon-all` pipeline, with `qcache` option enabled. In our CIVET analysis, we used the default options with the following additional specifications: an N3 spline distance of 200 mm, 12° of freedom for the linear registration, and the `tlink` metric with a smoothing kernel size of 20 mm FWHM (full-width at half maximum) for the cortical thickness.

Cortical thickness maps were computed in each subject's native space. For Freesurfer, these thickness maps were then resampled to Freesurfer's default `fsaverage` surface template as a common space, while cortical thickness maps for CIVET were resampled to CIVET 1.1.12's default MNI152 surface template. Resampled thickness files from both Freesurfer and CIVET were imported to the SurfStat MATLAB toolbox (Worsley et al., 2009) for statistical analyses.

To directly compare the effect of build and OS on cortical thickness, a difference score between processing conditions (cluster **A**–**B** or build **1**–**2**) was calculated with SurfStat for the cortical thickness of every subject at every vertex, and a Generalized Linear Model (GLM) was computed consisting simply of the formula $Y = 1$.

2.5. Library and System Call Interception

We recorded calls to `libmath` performed by dynamically-linked applications using `ltrace`[8] version 0.7.91, patched to facilitate output formatting, and configured to trace children processes created by `fork()` and `clone()`. We first completely re-executed a task on each cluster using `ltrace`'s summary mode to list the mathematical functions called by the application. Next, we configured `ltrace` to record and print the input and output values used in these function calls. In order to avoid excessively large log files, we limited the analysis to a few hours per task, which covered the first few million calls. We also recorded system calls made by applications using `strace`[9].

To compare two `ltrace` traces, we assumed that two executions producing identical results perform the same calls to mathematical functions, in the same order. Traces can then be

[7] http://www.adni-info.org

[8] http://ltrace.org

[9] http://strace.sourceforge.net

compared line by line. We classified differences between trace lines in four types. Type-1 differences correspond to functions called on different arguments that produce identical results. They are likely to occur in non-injective functions such as `floor()` and `ceil()`. They have little impact on the execution, but are a sign of other differences. Type-2 differences correspond to functions called on different arguments that produce different results. Type-3 differences correspond to functions called on identical arguments that produce different results. They are a sign of implementation differences in the mathematical functions. Type-3 differences usually trigger cascading type-2 and type-3 differences. Mismatches correspond to trace lines where different functions are called. They are a sign that the control flow of the compared conditions differed, for instance due to different numbers of iterations in loops.

3. Results

3.1. FSL: Brain Extraction

FSL BET produced identical results for all subjects on clusters **A** and **B**, as well as for builds **1** and **2**.

TABLE 2 | Dice coefficients between cortical tissue classifications on cluster A vs. cluster B (FSL FAST, build 1, *n* = 150 subjects).

Tissue	Average dice	Standard deviation
Global	0.99973	0.00013
Gray matter	0.99971	0.00014
White matter	0.99973	0.00013
CSF	0.99977	0.00012

3.2. FSL: Cortical Tissue Classification

FSL FAST cortical tissue classification produced identical results for builds **1** and **2**, but differences between cluster **A** and cluster **B** were found in the classifications of all 150 tested subjects. **Table 2** shows the Dice coefficients comparing results obtained on clusters **A** and **B** with FSL FAST, using build **1**. Dice coefficients are very high, indicating very minor differences. **Figure 2** shows the sum of binarized differences across segmented subjects. Differences are mostly localized at the interfaces between tissues.

Library call interception reveals the cause of these differences. **Figure 3** plots a trace of the first 22 million calls to `libmath` made by FSL FAST to process a randomly-chosen subject of the study. Only `log()` and `expf()` were called.

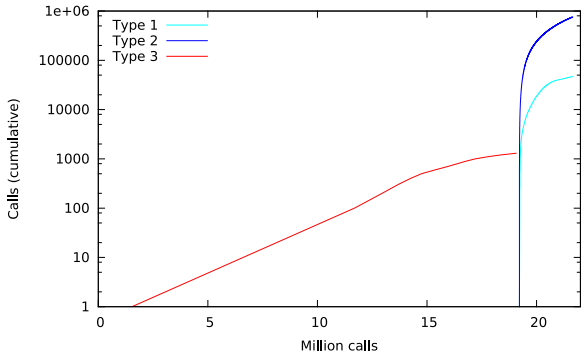


FIGURE 3 | Cumulative inter-OS differences in FSL FAST.

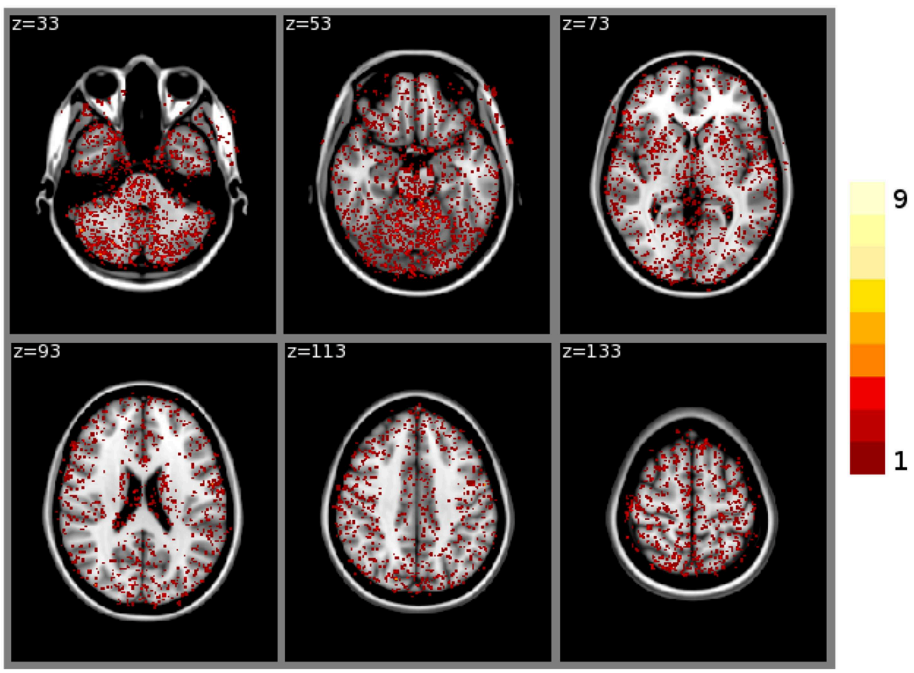


FIGURE 2 | Sum of binarized differences between cortical tissue classifications obtained on cluster A and cluster B (FSL FAST, build 1, *n* = 150 subjects). All binarized differences were resampled to the default MNI152 volume template.

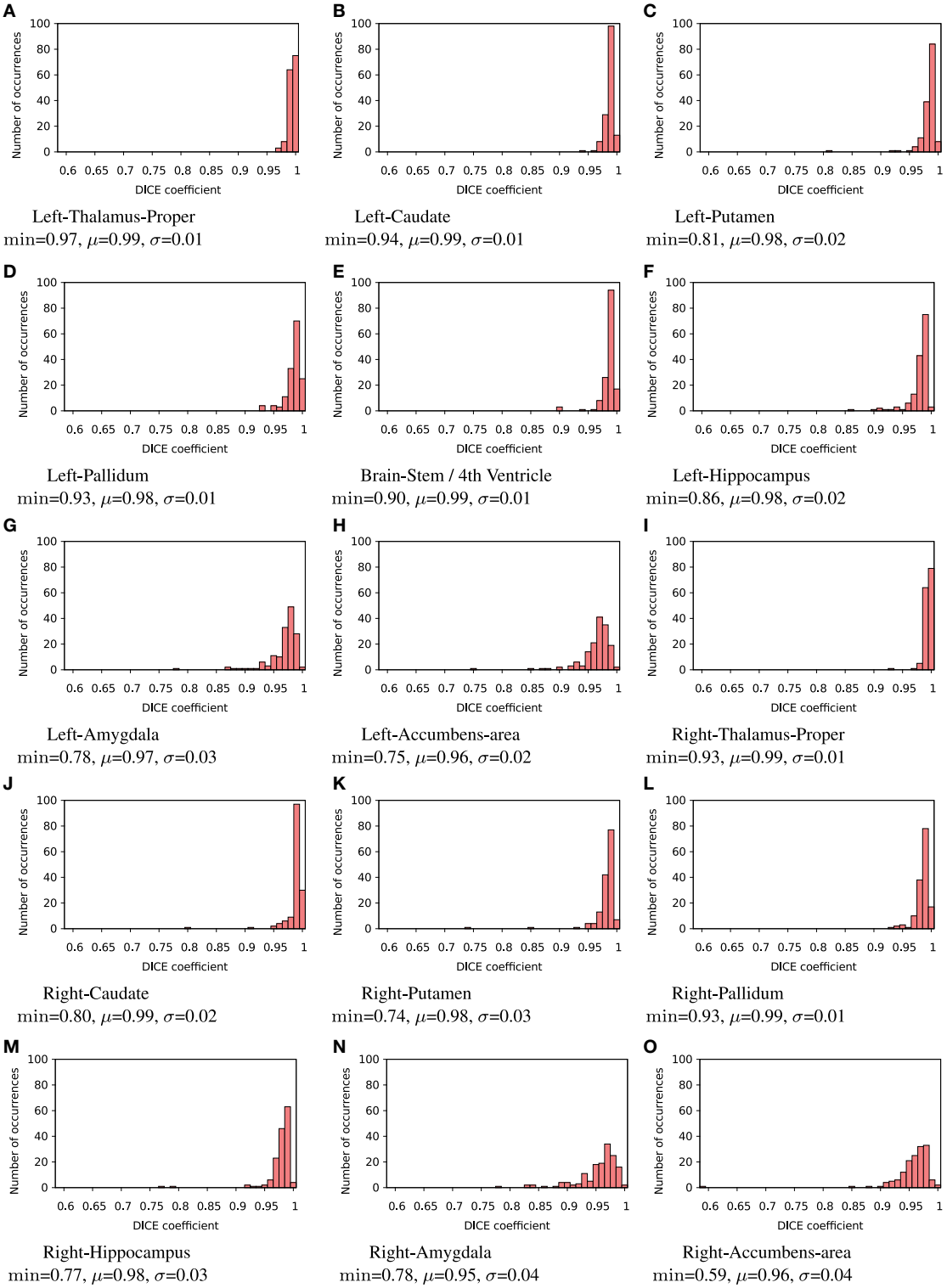


FIGURE 4 | Histograms of Dice coefficients between classifications obtained on cluster A vs. cluster B with FSL FIRST. All bins are of size 0.1. min, μ, and σ are the minimum, mean and standard deviation Dice coefficient, respectively.

The first differences appear at 1.5 million calls: they are type-3 differences in function `expf()` which manipulates single-precision floating-point representations. Type-1 and type-2 differences appear at 19.2 million calls, both in `log()` and `expf()`. No mismatch was found. The following C program excerpt reproduces the first observed type-3 difference:

```
float a=1.5405185;
float b=expf(a);
printf("expf(%.30f)=%.30f\n",a,b);
```

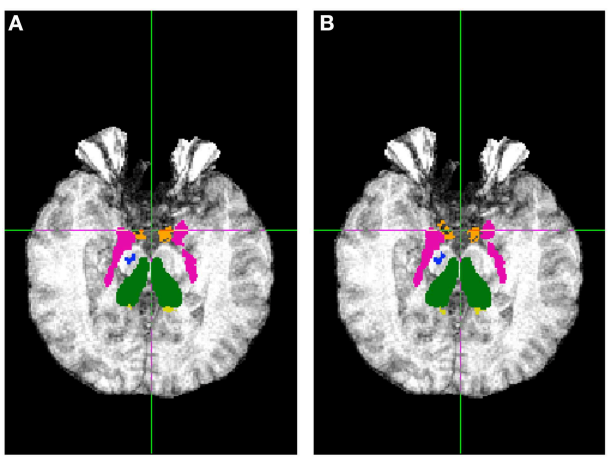
This program prints 30 decimals to display the complete representation of the floating-point numbers. When this representation has less than 30 decimals, `printf()` pads the displayed string with zeros. With `glibc` 2.5, this program prints:

```
expf(1.540518522262573242187500000000)
         =4.667009353637695312500000
```

The result produced by `expf()`, stored in variable `b`, is encoded as `24 58 95 40` in hexadecimal (little-endian convention). On the other hand, with `glibc` 2.18, the program prints:

```
expf(1.540518522262573242187500000000)
     =4.667009830474853515625000
```

The result produced by `expf()`, stored in variable `b`, is encoded as `25 58 95 40` in hexadecimal (little-endian convention): 1 bit is flipped compared to the result obtained with `glibc` 2.5. These numerical differences, which originate in changing implementation of `expf()` between `glibc` 2.5 and 2.18, are a cause of the inter-OS differences in FSL FAST.



Cluster **A** Cluster **B**

Structure	Color	Dice
L. and R. putamen	Magenta	0.92 and 0.95
R. pallidum	Dark blue	0.93
L. and R. thalamus	Green	0.97 and 0.93
L. and R. accumbens area	Orange	0.75 and 0.59
L. and R. hippocampus	Yellow	0.92 and 0.77

FIGURE 5 | Sample subcortical classifications with FSL FIRST: subject 260, $Z = 114$.

3.3. FSL: Subcortical Tissue Classification

FSL FIRST subcortical tissue classification produced identical results for builds **1** and **2**, but differences between cluster **A** and cluster **B** were found in the classifications of all 150 tested subjects. **Figure 4** plots the histograms of Dice coefficients for the 15 structures segmented with FSL FIRST, using build **1**. All histograms have a main mode around 0.99, but overall, only 12.7% of the classifications are identical on cluster **A** and cluster **B** (286 classifications out of 2250). Some Dice coefficients are very low, down to 0.59, in particular for small structures such as the amygdalae and the accumbens areas. **Figure 5** shows a result sample with Dice coefficients ranging from 0.75 to 0.95.

Figure 6 plots a trace of the first 53 million calls to `libmath` made by FSL FIRST to process a randomly-chosen subject. The trace shows no inter-OS difference until 43 million calls, where type-3 differences start to appear in function `cosf()`, soon followed by type-1 differences in `ceilf()` and `floorf()`, and type-2 differences in `cosf()`, `sinf()`, `ceilf()`, `floorf()`, and `logf()`. Mismatches appear at 43.9 million calls, indicating that inter-OS differences have an impact on the control flow of the program. An inspection of the source code shows that the bounds of a few loops are determined from `floorf()` and `ceilf()`[10], which is a plausible explanation for these mismatches.

Type-3 differences come exclusively from function `cosf()` which manipulates single-precision floating-point representations. The following C program excerpt reproduces the first one:

```
float a=0.523598790;
float b=cosf(a);
printf("cosf(%.30f)=%.30f\n",a,b);
```

With `glibc` 2.5, this program prints:

```
cosf(0.523598790168762207031250000000)
      =0.866025447845458984375000000
```

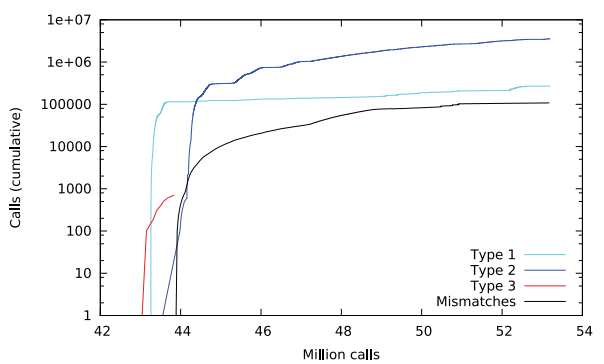


FIGURE 6 | Cumulative inter-OS differences in FSL FIRST.

[10]See for instance the `for` loops in method `intensity_hist` in `first.cc`, called from `do_work`

The result produced by `cosf()`, stored in variable `b`, is encoded as `d8 b3 5d 3f` in hexadecimal (little-endian convention). With `glibc` 2.18, this program prints:

```
cosf(0.523598790168762207031250000000)
     =0.866025388240814208984375 0
```

The result produced by `cosf()`, stored in variable `b`, is encoded as `d7 b3 5d 3f` in hexadecimal (little-endian convention): again, 1 bit is flipped compared to the result obtained with `glibc` 2.5. These numerical differences, which originate in changing implementation of `cosf()` between `glibc` 2.5 and 2.18, are a cause of the inter-OS differences in FSL FIRST.

3.4. FSL: Resting-state fMRI

3.4.1. Variable Random Seeds

In case **d** (automatic dimension detection), we observed no inter-run differences in the number of detected dimensions, but we found inter-OS differences in 2 subjects out of 37 (47 vs. 48 components and 55 vs. 57 components, respectively).

For the remaining 35 subjects, inter-run and inter-OS differences obtained with variable random seeds are shown in **Figure 7** for case **d** (automatic dimension detection), and in **Figure 8** for case **c** (dimension fixed to 20). All histograms appear bimodal, with a first mode at Dice = 0, and a second around Dice = 0.9. The modes at Dice = 0 correspond to situations where the positive and negative components are inverted, or one of the two compared components has very few voxels. Inter-run and inter-OS differences are significant, and they are of similar magnitude (see modes *m* reported above the graphs).

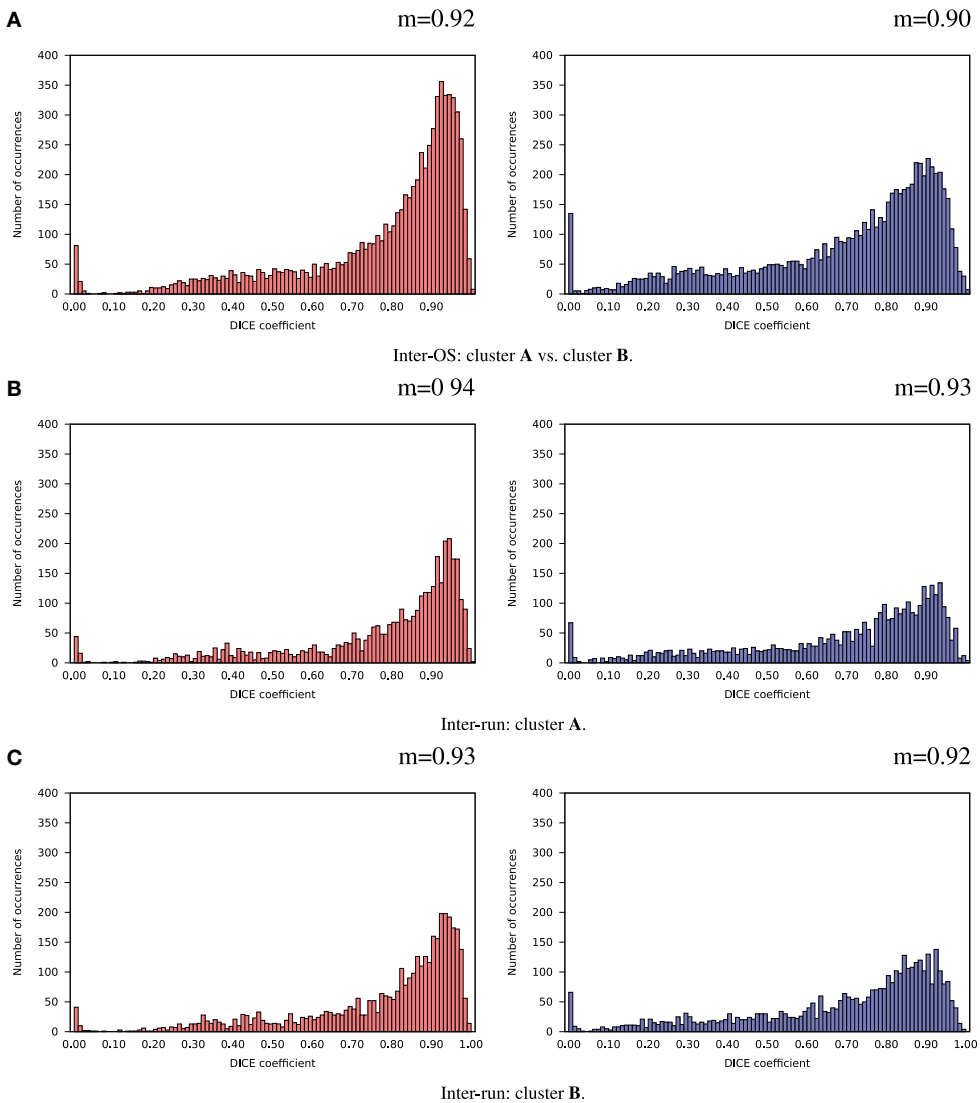


FIGURE 7 | Histograms of Dice coefficients between matched ICA components. Variable random seed initialization (case **b**), automatic dimension detection (case **d**). Red histograms, positive components; Blue histograms, negative components; m, mode of the histogram.

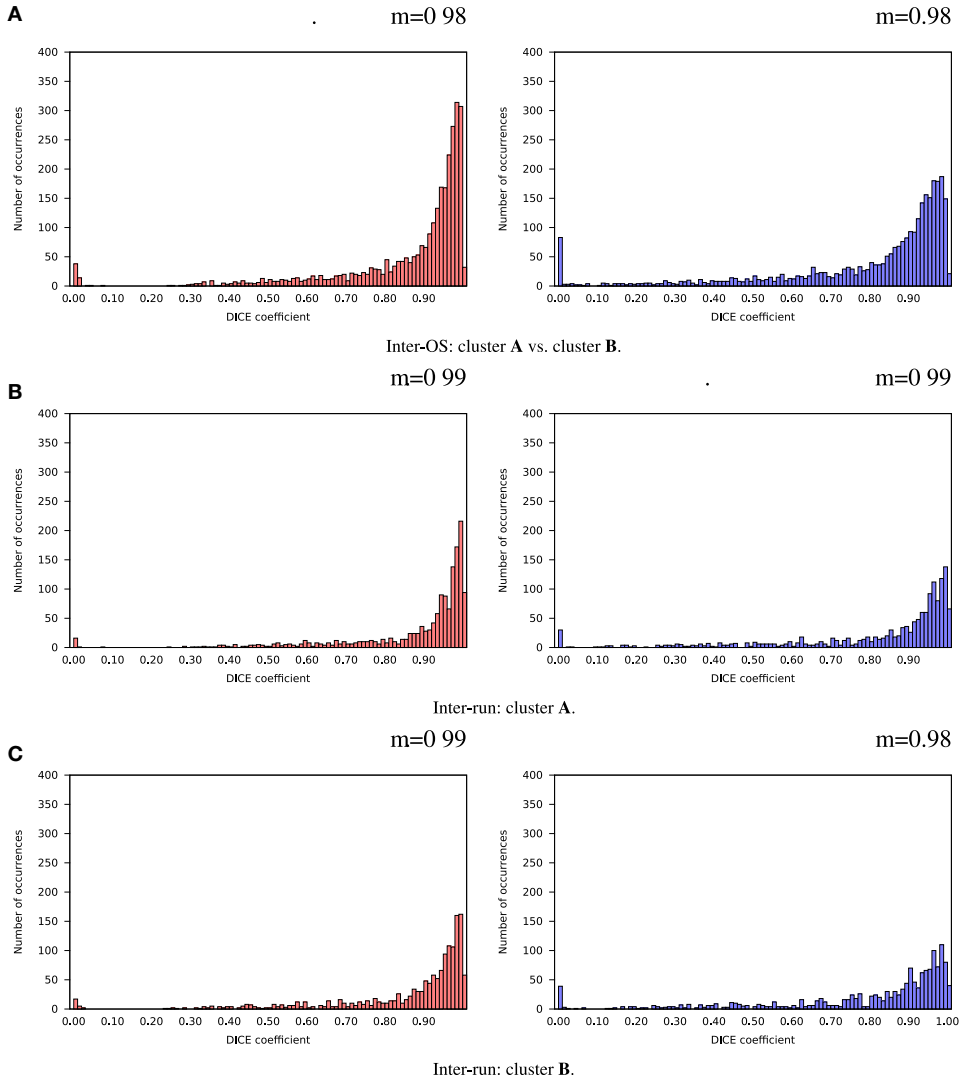


FIGURE 8 | Histograms of Dice coefficients between matched ICA components. Variable random seed initialization (case **b**), fixed dimension (case **c**). Red histograms, positive components; Blue histograms, negative components; m, mode of the histogram.

3.4.2. Fixed Random Seeds

Inter-OS differences in the number of detected dimensions were found in the same 2 subjects as for variable seeds. For the remaining 35 subjects, inter-OS differences obtained with fixed random seeds are shown on **Figure 9** for fixed (case **c**) and automatically detected dimensions (case **d**). Inter-OS differences are substantial in both cases, with Dice values lower than 0.9.

We found that inter-OS differences appear if and only if pre-processed data are different, which occurs in 12 out of 37 subjects. More precisely, inter-OS differences appear if and only if motion-corrected data are different. **Figure 10** plots the measured inter-OS mean absolute difference after each main pre-processing step, normalized with the mean absolute difference after all pre-processing steps. We can see that motion correction generates only slight differences, less than 20% of the total difference created by pre-processing. These differences are reduced by spatial smoothing but largely amplified by intensity normalization. Thresholding and temporal filtering have only a minor impact on the global error. Differences in motion correction are quite subtle: residual transformations all have a norm of translation vector below 10^{-5} mm, and rotation angle under 0.096°.

Figure 11 shows a trace of the first 14 million calls to `libmath` made by `mcflirt` to process a randomly-chosen subject. The first inter-OS difference is a type-3, observed at 1.6 million calls in function `sinf()` which manipulates single-precision floating-point representations. Another type-3 difference in the same function appears at 11.6 million calls, soon followed by type-1 and type-2 differences in `ceilf()`, `cosf()`, `logf()`, `sinf()`, and `floorf()`.

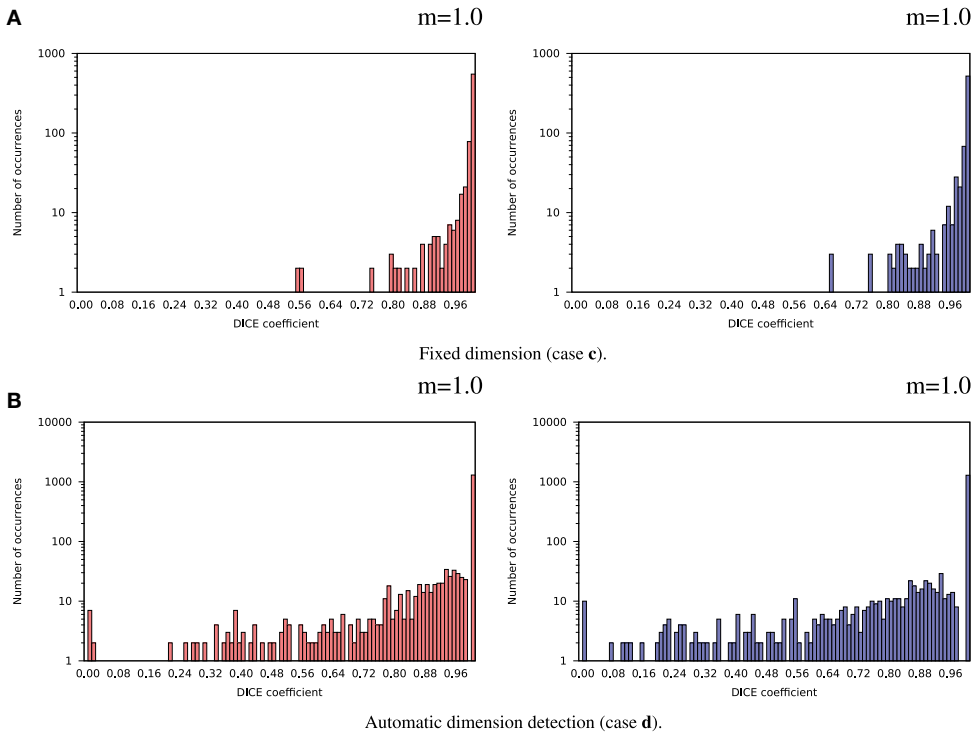


FIGURE 9 | Histograms of Dice coefficients between matched ICA components on cluster A vs. cluster B (logarithmic scale). Fixed random seed initialization (case **a**). Red histograms, positive components; Blue histograms, negative components; m, mode of the histogram.

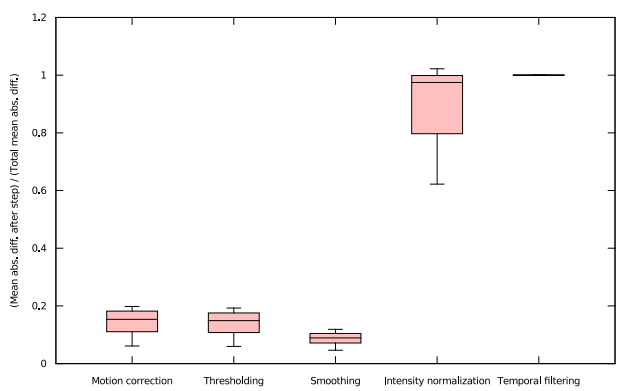


FIGURE 10 | Mean absolute differences after successive steps of pre-processing, normalized by the mean absolute difference after all pre-processing steps (all 37 subjects).

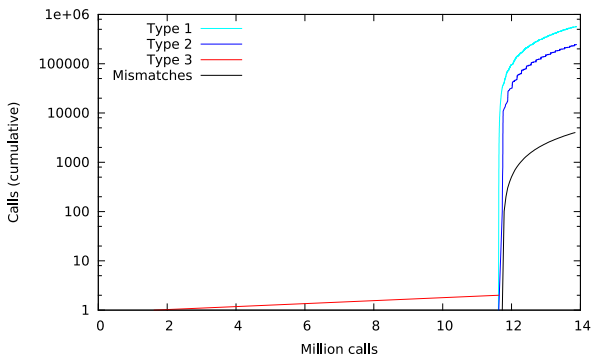


FIGURE 11 | Cumulative inter-OS differences in FSL mcflirt.

Mismatches appear at 11.7 million calls, indicating that inter-OS differences have an impact on the control flow of the program. The two observed type-3 differences come from function `sinf()`. The following C program excerpt reproduces the first one:

```
float a=0.042260922;
float b=sinf(a);
printf("sinf(%.30f)=%.30f\n",a,b);
```

With `glibc` 2.5, this program prints:

```
sinf(0.042260922491550445556640625000)
      =0.042248345911502838134765625000
```

The result produced by `sinf()`, stored in variable `b`, is encoded as `9a 0c 2d 3d` in hexadecimal (little-endian convention). With `glibc` 2.18, the program prints:

```
sinf(0.042260922491550445556640625000)
      =0.042248342186212539672851562500
```

The result produced by `sinf()`, stored in variable `b`, is encoded as `99 0c 2d 3d` in hexadecimal (little-endian convention):

again, 1 bit is flipped compared to the result obtained with `glibc` 2.5. These numerical differences, which originate in changing implementation of `sinf()` between `glibc` 2.5 and 2.18, are a cause of the inter-OS differences in `mcflirt`.

3.5. Freesurfer and CIVET: Surface Segmentation and Cortical Thickness Extraction

Four subjects were dropped from the results for the following reasons: Freesurfer analysis failed to reach completion ($n = 3$), and missing age information ($n = 1$).

3.5.1. Freesurfer: Inter-Build Differences

Some localized regions of differences were found for Freesurfer build **1** vs. **2** on cluster **B**. **Figure 12** shows surface maps of mean absolute difference, standard deviation of absolute difference, *t*-statistics and whole-brain random field theory (RFT) corrections (peaks and clusters) for $n = 146$ subjects at significance value of $p < 0.01$, comparing the cortical thickness values extracted by Freesurfer build **1** and build **2** on cluster **B**. Areas in shades of blue on the RFT map are significant at the cluster (but not peak) level. The cortical thickness values extracted with build **1** are significantly different than with build **2** in the left inferior frontal gyrus at an initial cluster threshold of $p < 0.01$ (family-wise error (FWE) of $p < 0.05$).

3.5.2. Freesurfer: Inter-OS Differences

Despite the static linking of Freesurfer's main executables, we still found small inter-OS differences. **Figure 13** shows surface maps of mean absolute difference, standard deviation of absolute difference, *t*-statistics and whole-brain random field theory (RFT) corrections for $n = 146$ subjects at a significance value of $p < 0.05$, comparing the cortical thickness values extracted by Freesurfer build **1** on cluster **A** and cluster **B**. Note the different scales compared to **Figure 12**. Although no values on the RFT map reach significant levels, the t values do reach upwards of ± 2. These residual differences, present in 6 subjects, are introduced by statically-linked executables `mri_em_register` (2 subjects) and `mri_surf2surf` (4 subjects). Using `strace`, we found that these tools open a few libraries from the operating system, including `libmath`. Differences in these libraries are very likely to create the observed inter-OS differences, although `ltrace` cannot be used on statically-linked tools to confirm this hypothesis.

3.5.3. CIVET: Inter-OS Differences

We also found some localized regions of differences for CIVET cluster **A** vs. **B**. **Figure 14** shows surface maps of mean absolute difference, standard deviation of absolute difference, *t*-statistics and random field theory (RFT) for $n = 146$ subjects at a significance value of $p < 0.05$, comparing the cortical thickness values

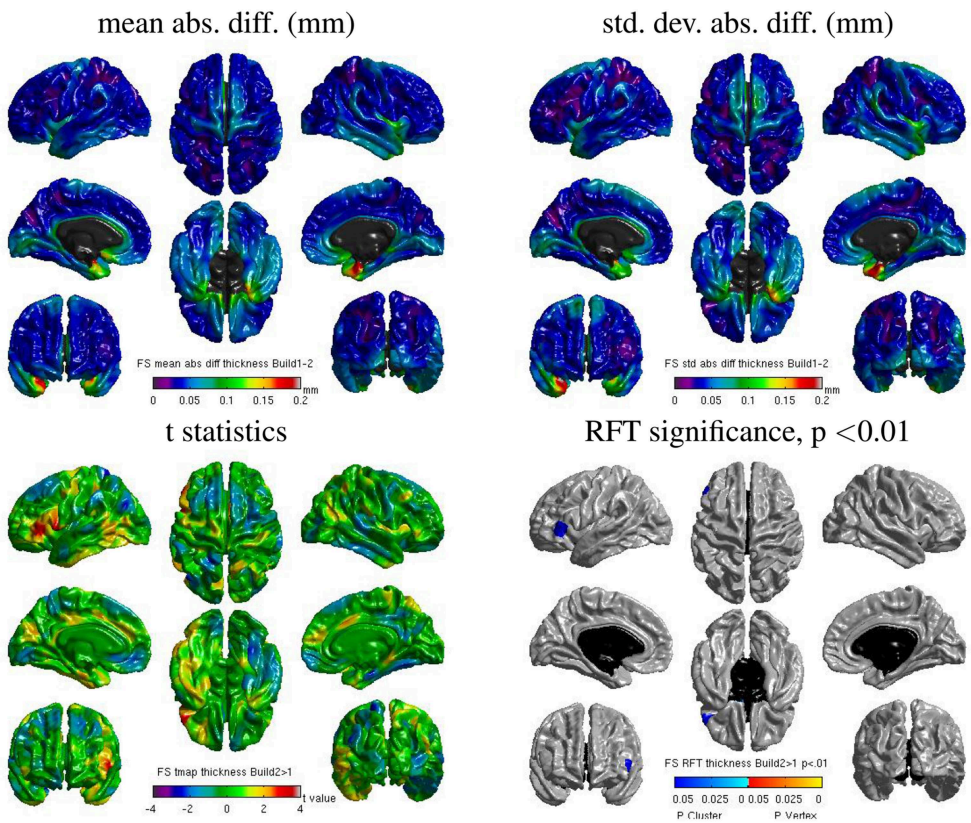


FIGURE 12 | Surface maps of mean absolute difference, standard-deviation of absolute difference, *t*-statistics and RFT significance values showing regions where the cortical thickness extracted with Freesurfer differs for build 1 and build 2 (both executed on cluster B).

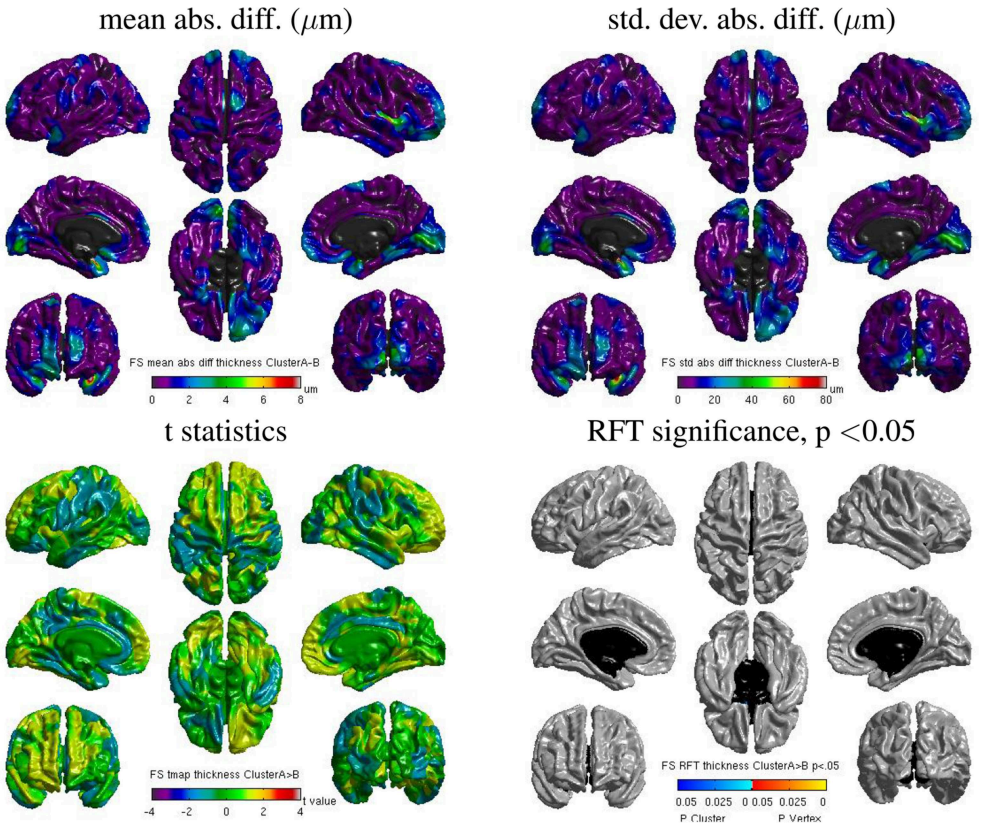


FIGURE 13 | Surface maps of mean absolute difference, standard-deviation of absolute difference, *t*-statistics and RFT significance values showing regions where the cortical thickness extracted with Freesurfer differs for cluster A and cluster B (both executed with build 1).

extracted by CIVET on cluster **A** and **B**. The cortical thickness values extracted on cluster **A** are significantly different than on cluster **B** at an initial cluster threshold of $p < 0.05$ (FWE of $p < 0.0005$ in the right paracentral lobule and FWE of $p < 0.04$ in the left middle temporal region). No significant difference between clusters **A** and **B** was found at a stricter initial cluster threshold of $p < 0.01$.

4. Discussion

4.1. General Conclusions

The implementation of mathematical functions manipulating single-precision floating-point numbers in `libmath` has evolved during the last years, leading to numerical differences in computational results. While these differences have little or no impact on simple analysis pipelines such as brain extraction and cortical tissue classification, their accumulation creates important differences in longer pipelines such as the subcortical tissue classification, RSfMRI analysis, and cortical thickness extraction.

For cortical tissue classification with FSL, Dice values as low as 0.59 were found between OSes. In RSfMRI, different numbers of components were occasionally found in the two OSes, and the identified components had important differences. Differences in cortical thickness were found for some brain regions as a function of build or OS.

Statically building programs improves reproducibility across OSes, but small differences may still remain when dynamic libraries are loaded by static executables, as observed with Freesurfer. When static builds are not an option, software heterogeneity might be addressed using virtual machines (VMs) as tested in CBRAIN (Glatard et al., 2014), or lighter container environments such as Docker[11]. Specific Linux distributions such as Neurodebian (Halchenko and Hanke, 2012) could be used with these environments to guarantee a wide reproducibility within the community. However, such solutions are only workarounds: differences may still arise between static executables built on different OSes (as seen in our Freesurfer study), or between dynamic executables executed in different VMs.

Although it would not improve numerical stability, a more rigorous way to address reproducibility issues would be to use higher-precision representations of floating-point numbers, and to avoid using functions operating on single-precision numbers (e.g., `expf()`, `cosf()`, ...). Using double precision would probably address most issues, and the remaining ones could be tackled with quadruple or even arbitrary precision as discussed in Bailey et al. (2012). To limit the resulting performance reduction,

[11] http://www.docker.com

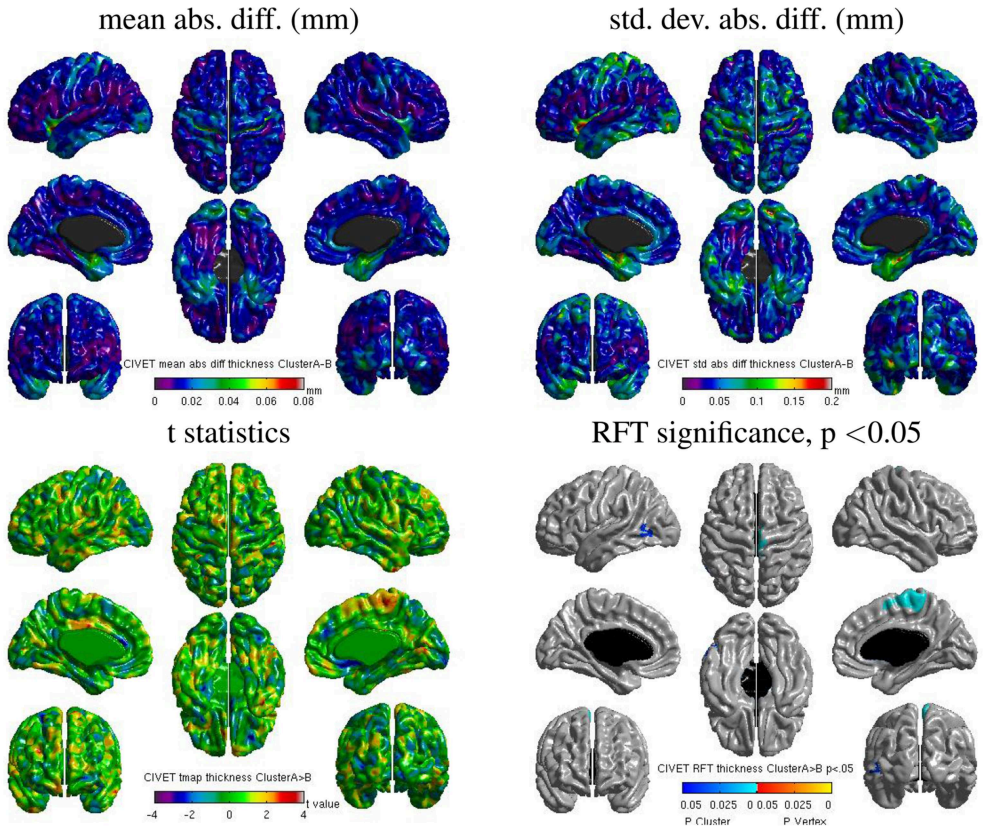


FIGURE 14 | Surface maps of mean absolute difference, standard-deviation of absolute difference, *t*-statistics and RFT significance values showing regions where the cortical thickness extracted with CIVET differs for cluster A and cluster B (both executed with build 1).

precision could be increased only in the code sections creating reproducibility issues.

Identifying such code sections is not trivial though, in particular when pipelines result from a long development process. We showed that library call interception yields accurate information about the functions that are responsible for reproducibility issues in dynamically-linked programs. This technique is, however, extremely heavy in terms of computational overhead and size of the generated traces, and therefore could not be used systematically.

When pipelines produce intermediary result files, a more efficient way to identify suspicious code sections is to compare these intermediary files using some data-specific distance. For instance, using the mean absolute difference between intermediary results produced by FSL pipelines, we were able to quantify the effect of fMRI pre-processing steps on inter-OS reproducibility and to narrow-down the investigation to motion correction. We were also able to identify the tools creating inter-OS differences in Freesurfer.

To conclude, it is clear to us that developers should carefully review the numerical reproducibility and stability of their pipelines using quantitative tests conducted in different execution conditions. However, this could not be done systematically unless a proper platform is available to run such tests and interpret the results. Such a platform could provide benchmarks, virtual execution environments, and analysis tools to help developers identify the cause of observed differences. Frameworks such as testkraut[12] could be useful in this context.

4.2. Limitations

Our results cover some of the main neuroimaging analysis tools (Freesurfer, FSL and CIVET), executed on RedHat-like Linux operating systems which are widely used in neurosciences. To cover a large spectrum of OSes, we used the oldest still-supported version of CentOS and the latest version of Fedora which anticipates on the coming CentOS versions. This encompasses 7 years of `glibc` development, from version 2.5 in 2006 to 2.18 in 2013, and a much longer range of Linux distributions. For instance, our study gives an idea of reproducibility issues that will arise when upgrading platforms to the recently-released CentOS 7 distribution, which is based on `glibc` 2.17.

The range of operating systems tested in this study remains, of course, limited. We expect that comparing intermediate `glibc` versions would only reduce the magnitude of the reported effects. Other Linux distributions, for instance Debian and Ubuntu, are very likely to suffer the same reproducibility issues as long as they are based on `glibc`. Similar issues are also very likely to

[12]https://testkraut.readthedocs.org/en/latest/index.html

occur on non-Linux operating systems, see for instance differences observed between Mac OS 10.5 and 10.6 by Gronenschild et al. (2012).

Our study is limited to compiled application programs. Applications written with interpreted languages such as MATLAB and Python would most likely behave differently. Compilation options were also not considered in this study and are likely to impact the reproducibility of results. For instance, the `gcc` C compiler has several options that speed-up floating-point operations at the cost of numerical correctness. Using such options to compile programs that are sensitive to small numerical differences is very likely to compromise inter-OS reproducibility, too. Some of the differences observed between Freesurfer builds are likely to originate from the use of different versions of `gcc` to compile these builds.

4.3. Related Work

Gronenschild et al. (2012) report the effects of Freesurfer version, workstation type, and OS version on anatomical volume and cortical thickness measurements. Their study was conducted with different versions of Freesurfer (4.3.1, 4.5.0, and 5.0.0). We deliberately chose not to compare different versions of the tested pipelines. Instead, we focused on differences that originate in the system libraries. The Freesurfer versions used by Gronenschild et al. (2012) were dynamically linked (version 5.0.0 was linked statically on Linux, but dynamically on Mac), while the current one (5.3) is statically linked. Thus, the difference reported by Gronenschild et al. (2012) between Mac OS 10.5 and Mac OS 10.6, and between HP and Mac, most likely comes from the use of different system libraries in these platforms. Statically building executables might be seen as a way to address the issues shown by Gronenschild et al. (2012); our study shows that it is only a workaround since different builds unsurprisingly yield different results. We also show that these problems are not specific to Freesurfer, but generalize to FSL and to some extent CIVET; it suggests that several other analysis packages are likely to be impacted. Besides, our choice of operating systems (CentOS 5.10 and Fedora 20) encompasses 7 years of `glibc` development; this gives an idea of how results may evolve in the coming upgrades of HPC clusters to CentOS 7. Finally, we provide an explanation of the causes for inter-OS reproducibility issues; this suggests that these issues may be addressed by using more precise representations of floating-point numbers in some sections of the pipelines.

Krefting et al. (2011) studied the reproducibility of Freesurfer 5.0.0 on Mac OS 10.6, CentOS 4, and SUSE Linux 10.1. They report that the CentOS 5 and CentOS 4 Freesurfer builds gave identical results, but that results obtained with the same build were different across operating systems. This seems in contradiction with our results (we found that different Freesurfer builds give different results). A possible explanation for these differences is that the authors used a dynamically-linked version of Freesurfer 5.0.0, as suggested when they report that different implementations of dynamically linked libraries may explain their findings.

Acknowledgments

We thank Fabrice Bellet for his rigorous administration of the cluster at Creatis used in this work (cluster **B**). We also thank Compute Canada and Calcul[13] Québec[14] for providing the infrastructure (cluster **A**) to perform the experiments presented in this paper. This work is in the scope of the LABEX PRIMES (ANR-11- LABX-0063) of Université de Lyon, within the program "Investissements dAvenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR). This paper also acknowledges the support of the National Science Foundation under grant #ACI-1148515. Data collection and sharing for the fMRI part of this study was funded by the Alzheimer's Disease Neuroimaging Initiative (ADNI) (National Institutes of Health Grant U01 AG024904) and DOD ADNI (Department of Defense award number W81XWH-12-2-0012). ADNI is funded by the National Institute on Aging, the National Institute of Biomedical Imaging and Bioengineering, and through generous contributions from the following: Alzheimers Association; Alzheimers Drug Discovery Foundation; Araclon Biotech; BioClinica, Inc.; Biogen Idec Inc.; Bristol-Myers Squibb Company; Eisai Inc.; Elan Pharmaceuticals, Inc.; Eli Lilly and Company; EuroImmun; F. Hoffmann-La Roche Ltd and its affiliated company Genentech, Inc.; Fujirebio; GE Healthcare; ; IXICO Ltd.; Janssen Alzheimer Immunotherapy Research & Development, LLC.; Johnson & Johnson Pharmaceutical Research & Development LLC.; Medpace, Inc.; Merck & Co., Inc.; Meso Scale Diagnostics, LLC.; NeuroRx Research; Neurotrack Technologies; Novartis Pharmaceuticals Corporation; Pfizer Inc.; Piramal Imaging; Servier; Synarc Inc.; and Takeda Pharmaceutical Company. The Canadian Institutes of Health Research is providing funds to support ADNI clinical sites in Canada. Private sector contributions are facilitated by the Foundation for the National Institutes of Health (www.fnih.org). The grantee organization is the Northern California Institute for Research and Education, and the study is coordinated by the Alzheimer's Disease Cooperative Study at the University of California, San Diego. ADNI data are disseminated by the Laboratory for Neuro Imaging at the University of Southern California.

References

Ad-Dabbagh, Y., Einarson, D., Lyttelton, O., Muehlboeck, J.-S., Mok, K., Ivanov, O., et al. (2006). "The CIVET image-processing environment: a fully automated comprehensive pipeline for anatomical neuroimaging research," in *Proceedings of the 12th Annual Meeting of the Organization for Human Brain Mapping*. (Florence).

Bailey, D., Barrio, R., and Borwein, J. (2012). High-precision computation: mathematical physics and dynamics. *Appl. Math. Comput.* 218, 10106–10121. doi: 10.1016/j.amc.2012.03.087

Beckmann, C. F., and Smith, S. M. (2004). Probabilistic independent component analysis for functional magnetic resonance imaging. *IEEE Trans. Med. Imaging* 23, 137–152. doi: 10.1109/TMI.2003.822821

Dice, L. (1945). Measures of the amount of ecologic association between species. *Ecology* 26, 297–302. doi: 10.2307/1932409

Fischl, B. (2012). FreeSurfer. *Neuroimage* 62, 774–781. doi: 10.1016/j.neuroimage.2012.01.021

[13]http://www.computecanada.ca

[14]http://www.calculquebec.ca

Glatard, T., Rousseau, M., Rioux, P., Adalat, R., and Evans, A. C. (2014). "Controlling the deployment of virtual machines on clusters and clouds for scientific computing in CBRAIN," in *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (Chicago), 384–393.

Gronenschild, E. H. B. M., Habets, P., Jacobs, H. I. L., Mengelers, R., Rozendaal, N., van Os, J., et al. (2012). The effects of FreeSurfer version, workstation type, and Macintosh operating system version on anatomical volume and cortical thickness measurements. *PLoS ONE* 7:e38234. doi: 10.1371/journal.pone.0038234

Halchenko, Y. O., and Hanke, M. (2012). Open is not enough. Let's take the next step: an integrated, community-driven computing platform for neuroscience. *Front. Neuroinform.* 6:22. doi: 10.3389/fninf.2012.00022

Hanke, M., and Halchenko, Y. O. (2011). Neuroscience Runs on GNU/Linux. *Front. Neuroinform.* 5:8. doi: 10.3389/fninf.2011.00008

Jenkinson, M., Beckmann, C. F., Behrens, T. E. J., Woolrich, M. W., and Smith, S. M. (2012). FSL. *Neuroimage* 62, 782–790. doi: 10.1016/j.neuroimage.2011.09.015

Krefting, D., Scheel, M., Freing, A., Specovius, S., Paul, F., and Brandt, A. (2011). "Reliability of quantitative neuroimage analysis using freesurfer in distributed environments," in *MICCAI Workshop on High-Performance and Distributed Computing for Medical Imaging*. (Toronto, ON).

Mazziotta, J., Toga, A., Evans, A., Fox, P., Lancaster, J., Zilles, K., et al. (2001). A probabilistic atlas and reference system for the human brain: international Consortium for Brain Mapping (ICBM). *Philos. Trans. R. Soc. Lond. Ser. B Biol. Sci.* 356, 1293–1322. doi: 10.1098/rstb.2001.0915

Minka, T. (2000). *Automatic Choice of Dimensionality for PCA*. Technical report, Technical Report 514, MIT Media Lab Vision and Modeling Group.

Patenaude, B., Smith, S. M., Kennedy, D. N., and Jenkinson, M. (2011). A Bayesian model of shape and appearance for subcortical brain segmentation. *Neuroimage* 56, 907–922. doi: 10.1016/j.neuroimage.2011.02.046

Sherif, T., Rioux, P., Rousseau, M.-E., Kassis, N., Beck, N., Glatard, T., et al. (2014). CBRAIN: a web-based, distributed computing platform for collaborative neuroimaging research. *Front. Neurosci.* 8:54. doi: 10.3389/fninf.2014.00054

Smith, S. M. (2002). Fast robust automated brain extraction. *Hum. Brain Mapp.* 17, 143–155. doi: 10.1002/hbm.10062

Worsley, K., Taylor, J., Carbonell, F., Chung, M., Duerden, E., Bernhardt, B., et al. (2009). SurfStat: a matlab toolbox for the statistical analysis of univariate and multivariate surface and volumetric data using linear mixed effects models and random field theory. *Neuroimage* 47, S102. doi: 10.1016/S1053-8119(09)70882-1

Zhang, Y., Brady, M., and Smith, S. (2001). Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. *IEEE Trans. Med. Imaging* 20, 45–57. doi: 10.1109/42.906424

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.