A Configurable Software Pipeline for Automated Preprocessing and Analysis of NeuroImaging Data

R. Cameron Craddock[a,b], Daniel J. Lurie[a], Steven Giavasis[a], Zarrar Shehzad[b], Daniel Clark[a], Chao-Gan Yan[b,c], Joshua T. Vogelstein[d], Randal Burns[e], Stanley Colcombe[b], Maarten Mennes[f], Clare Kelly[c], Adriana Di Martino[c], F. Xavier Castellanos[c,b], Michael P. Milham[a,b]

[a]*Center for the Developing Brain, Child Mind Institute, New York, NY*
[b] *Nathan S. Kline Institute for Pyschiatric Research, Orangeburg, NY*
[c]*Phyllis Green and Randolph Cowen Institute for Pediatric Neuroscience, New York University Child Study Center New York, NY*
[d]*Departments of Mathematics and Statistical Science, Duke University, Durham, NC*
[e]*Department of Computer Science, John Hopkins University, Baltimore, MD*
[f]*Donders Institute for Brain, Cognition and Behavior, Radboud University, Nijmegen Medical Centre, Nijmegen, The Netherlands*

Abstract

Text of abstract. Text of abstract. Text of abstract. Text of abstract. Text of abstract.

Keywords: example, LaTeX, template

1. Introduction

Mapping intra-individual diversity in human behavior and physiology to variation in the brains functional architecture is a big data problem that is vital for developing biomarkers of disease state and prognosis, as well as putative targets for novel therapeutic interventions. Similar to genetics and other data types that typify big data, neuroimaging measures of brain function are characterized by very high dimensionality. In combination with the massive number of observations required to provide an adequate sampling of phenotypic variation in humans (e.g. age, sex, IQ, reading ability, sadness, etc.), this dimensionality results in daunting analytical and computational challenges to the advancement of neuroscience. In response, centralized and grass roots data sharing initiatives such as the Human Connectome Project (HCP), UK Biobank and International Data sharing Initiative (INDI) are amassing and sharing the large deeply phenotyped datasets required to perform these analyses. Unfortunately, the dearth of data analysis and computational methods required to take full advantage of these datasets, and the inaccessibility of these methods to the greater neuroscientific community, continue to hamper the elucidation of the neural correlates of phenotypic diversity in both health and disease.

C-PAC seeks to fill this void by developing methods, resources, and tools to accelerate data-driven discovery from functional neuroimaging data. In particular, C-PAC leverages the functional connectome, a graph representation of all of the functional interactions in the brain, based on its promise for transforming our understanding of the brains architecture (Craddock et al. 2013). Additionally, the functional connectome represents unique challenges for big data analyses due to the massive number of edges present in the graph, and the interdependencies between them (Varoquaux and Craddock 2013). Computationally inclined neuroscience researchers have made significant strides in bringing sophisticated analytical methods to bear to answer daunting questions in neuroscience. But harnessing the full potential of big data analytics will require the recruitment of data scientists from a variety of backgrounds who may lack the resources and the domain specific knowledge required to analyze connectome data. As new analytical techniques are developed, they must be integrated into software that is powerful enough to achieve sophisticated large-scale analyses but user-friendly enough that they are accessible to neuroscientists who lack computational backgrounds.

To ensure extensibility and facilitate participation in development by the broader community, C-PAC is based in the python programming language and developed around Nipype107, a project of the Neuroimaging in Python108 (NiPy) community initiative. Nipype facilitates automated interaction between multiple existing open-source neuroimaging packages including AFNI18, FSL19, and FreeSurfer34. Designed for use by both novice users and experts, C-PAC brings the power, flexibility and elegance of Nipype to users in a plug-and-play fashion; no programming experience required. Users can rapidly orchestrate automated pre-processing and data analysis for multiple subjects by modifying an easy to read, text-editable configuration file. Analyses can be run locally or on a compute cluster, and forkable analysis pipelines allow users to run C-PAC using multiple settings at once, allowing researchers to easily explore the impact of different processing decisions on their findings. Sun Grid Engine (SGE) support is currently available. The initial (current) Alpha release of C-PAC supports the following features:

Email addresses: `cameron.craddock@childmind.org` (R. Cameron Craddock), `michael.milham@childmind.org` (Michael P. Milham)

It will provide a modular way to perform preprocessing and analysis for the different data types and will also handle data provenance. In the context of data analysis a pipeline is an ordered sequence of various processing and analysis steps that are performed on the data. Aditionally, the pipeline must be flexible to enable a variety of different analyses (configurable pipeline builder), must be easily specified and reproduced (configuration file/hashing), must include the most advanced and up-to-date analysis methods (workflow repository), must provide methods for comparing the performance of different pipeline choices (comparing pipelines), must provide a detailed accounting of the computations applied to a dataset (provenance), must give detailed feedback on data quality (quality assessment), and must be able to quickly produce a variety of different preprocessing and analysis strategies with limited storage (caching).

The majority of neuroimaging toolsets (i.e. SPM56, FSL57, AFNI58 were designed in an era dominated by morphometric or task-related fMRI studies with small sample sizes. While these toolsets provide outstanding implementations of the various algorithms necessary for imaging analysis, none of them excel at high through-put analysis. Different pipelining software has emerged to perform batch processing on very large datasets, most of which exploit parallel computing to achieve high throughput on multi-Core or cluster computer systems.

Nipype14 (see Figure 13) is an open-source pipeline library for python that supports a variety of neuroimaging toolsets (i.e. SPM, FSL, AFNI, Freesurfer27, Ants59) and high performance computer infrastructures (multi-processor, sun grid engine, openpbs, condor, and slurm). Using Nipype, a user describes a pipeline in python that specifies the interconnection of various nodes, each of which represent an operation or calculation that is performed on the data. Then, the Nipype scheduler builds a graph representation of the pipeline representing execution dependencies in the pipeline, and exploits parallelism that it identifies from the graph to efficiently execute the pipeline on a computational infrastructure. To enable users to explore the impact of different processing parameters, while minimizing the amount of duplicated computation, Nipype includes iterables, which replicates a node of the pipeline, and all of the nodes that depend on the output of that node, for a variety of settings of one or more parameters. Nipype enable restarting a pipeline, after a previous failure or reconfiguration, and only re-calculates derivatives that did not previously complete, or that will change based on the new configuration. Additionally, Nipype includes provenance, which is a human and machine - readable file that catalogues all of the operations performed on a dataset over the course of executing the pipeline.

Configurable Pipeline for the Analysis of Connectome (C-PAC): Although Nipype provides a powerful interface for building and executing pipelines, it still requires a proficiency in python programming and substantial domain specific knowledge in neuroimaging data preprocessing and analysis to specify a pipeline. The configurable pipeline for the analysis of Connectomes (C-PAC) is a Nipype application designed for use by both novice users and experts. C-PAC brings the power, flexibility and elegance of Nipype to users in a plug-and-play fashion; no programming experience is required. C-PAC incorporates a pipeline builder that generates python code to implement Nipype pipelines from an easy to read, text-editable configuration file or using a graphical user interface. A pipeline can be reproduced from the configuration file, or from a unique configuration fingerprint. A C-PAC pipeline is executed using the standard Nipype scheduling interfaces, and the ongoing execution is tracked using C-PACs status interface. C-PAC provides a web-based quality assessment interface that calculates a myriad of quality metrics for various stages of pipeline. C-PAC offers a variety of preprocessing steps and analysis workflows for structural and functional MRI, and provides support for a variety of group-level analyses, including flexible ANOVA and ANCOVA, as well as multivariate multiple-regression models using the connectome-wide association studies (CWAS) framework, and unsupervised analysis using the bootstrap analysis of stable clusters (BASC).

To provide an initial test and demonstration of usability, the FCP/INDI team used C-PAC to support the feasibility analyses carried out by the ABIDE consortium60. Following the success of C-PAC in supporting the ABIDE initiative, we have released an alpha version via github and NITRC, and identified nearly a dozen alpha users to support in their testing efforts and provided extensive web-based documentation.

1.1. Flexible pipeline generation

At the heart of the COINSTAC pipeline system is the pipeline builder that is capable of generating a computationally efficient processing pipeline from an abstract representation. A pipeline architect will use a graphical user interface to design a pipeline, the pipeline configuration will be provided to the pipelining system, along with the location of raw and/or cached preprocessed data. The pipeline builder will use this information to build a pipeline that is customized to the local configuration, and that integrates cached data to minimize duplicated computation. The builder will then use a data-dependency graph generated from the pipeline, along with information from the local system configuration, to schedule and execute the pipelines. Abstracting the specification of the pipeline configuration from the computing configuration at the local site frees the pipeline architect from worries about these details during pipeline specification. Building the pipeline locally further enables this abstraction while preserving the ability to customize the execution of the pipeline to maximize its computational efficiency on the local computational resources. A key aspect of the pipeline configuration is that it has to be transportable and easily replicable. The primary method of enabling this level of portability will be through the use of a standardized markup language for specifying a pipeline configuration. Additionally, a unique identifier will be generated for each configuration that will allow it to be able to be quickly regenerated at a different site. This identifier will be paired with a revision number that indicates the versions of the various software packages that were used in the pipeline.

1.2. QuickPACs

Caching preprocessing, intermediate results, and derivatives: There is no consensus on the best pre-processing strategies for neuroimaging data, or their implementation. As a result, the pipelining system needs to be highly flexible, and incorporate several different processing strategies. When trying to provide customizable preprocessing strategies, a trade-off emerges between computation time and storage requirements. Re-computing every preprocessing step and derivative for every different analysis is expensive, especially when considering the amount of overlap between different preprocessing strategies. Retaining data processed from a variety of strategies will provide significant improvements to computation time, but at the high cost of data storage. C-PAC incorporates intelligent caching strategies to optimize the computational and storage requirements of a variety of sites.

To accomplish, this we will use Quick-PAC, which is a compressed archive file containing semi-preprocessed imaging data and the necessary files required to quickly produce results for a variety of prepro-cessing strategies and data derivatives. The preprocessing steps used to generate a Quick-PAC are those that require substantial computational time and/or manual intervention i.e., the rate limiting steps. For example, calculation of the non-linear transformation from a sMRI image to a template space (e.g. MNI152) requires about an hour of calculation time, and if it fails can be improved by manual intervention. Retaining the resulting transformation file, as well as transformations calculated using different strategies, takes very little space, when compared to the number of computation hours saved. On the other hand, spatial smoothing is consistently applied to both structural and functional data, although using a variety of different parameters. Applying spatial smoothing to data can be accomplished in under a minute on modern workstations, but the data storage requirement for a derivative doubles for each additional smoothing size retained, therefore data smoothed with different size filters are not a part of a Quick-PAC. Quick-PACs are created during the execution of the first analysis of the data. Each file in the Quick-PAC is annotated with a fingerprint that indicates the processing strategy used to produce the file, and a version number. This will allow the Quick-PACs to be updated to keep up with new developments in analysis strategies.

1.3. Quality assessment interface

The integration of data between multiple sites without ever directly accessing all of the data requires detailed information about data quality. In this context, quality refers to the quality of the raw data, as well as the quality of the results of the various preprocessing steps that were applied to the data. The biggest challenge for automatic assessment of data quality is that there is no consensus on what constitutes good quality data, nor the best metrics for assessing quality. To address this issue the COINSTAC pipeline will calculate a variety of different quality metrics (see Table 2) and will provide an interface for the manual assessment of the data and processing. The assessment interface will be a series of web pages that provides a variety of visualizations of individual data at various levels through the pipeline, as well as, group level comparisons of summary metrics. There is often a tradeoff between dataset quality and size. It is not clear whether the benefits of using the highest quality data are worth the cost of reducing the sample size, and vice versa. To accommodate the various attitudes about this topic, preprocessed data and derivatives will be retained regardless of its quality, as long as it is of sufficient quality to be processed. This will additionally provide a resource for methodological researchers to use for developing methods that are robust to noise.

1.4. Comparing Pipelines

A pipeline architect will have a large variety of processing and analysis workflows at their disposal when implementing a pipeline for HARKiN. Rather than being prescriptive and limiting the pipe-lines to a few choices, our policy will be to implement many different strategies, and provide tools for their comparison. The intelligent pipelining system permits the rapid generation of preprocessed data, derivatives, and group-level analysis results. As different pipelines are executed, their results will be compared to the cached results from executing other pipelines. To further facilitate these comparisons, a spectrum of similarity metrics will be provided, including (from least to most stringent): Dice coefficient63, mutual information64, Pearsons correlation, and concordance correlation coefficient65. Workflows will be provided to automate these comparisons, along with more rigorous methods for evaluating pipelines, such as the NPAIRS framework66,67.

1.5. Provenance

Another key requirement for performing a large-scale data analysis across various sites is having a complete record of all of the computations that have been performed on a dataset and the corre-sponding parameters. The NIPYPE library currently provides a mechanism for tracking data provenance using the World Wide Web Consortium (W3C) PROV Recommendation and stored internally as RDF. Core 1 and 5 will interact to ensure that this information will be stored in the COINSTAC database to permit the easy identifi-cation of data that has been processed using identical pipelines. Pipelines will additionally be assigned unique identifiers and versioning information to simplify these comparisons.

1.6. Plurality

Operating in a Community Without Consensus or Ground Truths: The C-PAC Solution to Preprocessing Decision-Making. Lack of consensus in the selection of preprocessing steps and parameters (i.e., preprocessing strategy) represents a major obstacle to the advancement of the neuroimaging field. Statistical issues alone will result in more than 50% of the findings in the current literature to be irreplicable due to small sample size105. Compounding this unfortunate reality is the myriad of preprocessing strategies employed in the literature, precluding fair comparison of findings across studies or understanding failure to replicate1. The availability of publicly shared data allows researchers to make progress in this regard by applying multiple differing approaches to the same data106. Unfortunately this tends to happen too late (i.e., after publication) and such comparisons are rarely reported. A key reason for researchers not determining the extent to which findings will generalize from one preprocessing strategy to another is the difficulties associated with exploring different strategies in particular the tendency of software packages to be relatively rigid, running a single pipeline with a circumscribed set of options. Even packages capable of running a broader set of options tend to be designed for single execution, requiring complete reruns from scratch for different strategies.

C-PAC is designed to not only allow users to bring together and select from the broader array of available preprocessing methods, but to enable them to generate and execute a hierarchical tree of pipelines, with each branch point representing a different preprocessing step and each leaf being a different option. While a user must still identify their intended primary pipeline strategy, specification of an array of alternative pipelines is relatively trivial for the user (i.e., from a single text readable a configuration file) and execution is optimized for running time and space utilization. These abilities allow users to determine the extent to which their findings will extend to other preprocessing strategies with little effort. Pipeline optimization studies can also be carried out with relative ease using the C-PAC. Importantly, the C-PAC offers a broad array of data reduction and time series extraction options capable of outputting to python and .csv formats for those not familiar with more traditional imaging formats. Additionally, C-PAC features one of the broadest arrays of connectome-specific derivatives currently available in a neuroimaging analysis package to date.

1.7. Reproducibility

Here we will talk about methods that improve the reproducible research.

Anatomical Preprocessing

Table 1: Anatomical preprocessing pipeline.

Step	Method
Deoblique	3drefit [3]
Re-orient to RPI	3dresample [3]
Skull Strip	3dSkullStrip [3]
Denormalize	3dcalc [3]
Register to Template	FLIRT [5] + FNIRT [1]
(nonlinear)	ANTS [2]
Tissue Segmentation	FAST [6]

Functional Preprocessing

Table 2: Functional preprocessing pipeline.

Step	Software	Function
Drop Volumes	AFNI	3dcalc [3]
Slice Timing Correction	AFNI	3dTshift [3]
Deoblique	AFNI	3drefit [3]
Re-orient to RPI	AFNI	3dresample [3]
Volume Realignment	AFNI	3dvolreg [3]
Remove Skull	AFNI	3dcalc [3]
Intensity Normalization	FSL	fslmaths
Register to Anatomical	FSL	FLIRT with or without BBR (lin
Register to MNI	FSL	applywarp [1]
	ANTS	WarpTimeSeriesImageMultiTra

References

References

[1] J.L.R. Andersson, M. Jenkinson, S. Smith, Non-linear optimisation, FMRIB Technical Report TR07JA1 (2007).

[2] B. Avants, A. Khan, L. McCluskey, L. Elman, M. Grossman, Longitudinal cortical atrophy in amyotrophic lateral sclerosis with frontotemporal dementia, Arch Neurol 66 (2009) 138–9.

[3] R.W. Cox, AFNI: software for analysis and visualization of functional magnetic resonance neuroimages., Computers and biomedical research, an international journal 29 (1996) 162–73.

[4] D.N. Greve, B. Fischl, Accurate and robust brain image alignment using boundary-based registration, Neuroimage 48 (2009) 63–72.

[5] M. Jenkinson, P. Bannister, M. Brady, S. Smith, Improved optimization for the robust and accurate linear registration and motion correction of brain images, Neuroimage 17 (2002) 825–41.

[6] Y. Zhang, M. Brady, S. Smith, Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm, IEEE Trans Med Imaging 20 (2001) 45–57.