Research Ideas and Outcomes 3: e12346
doi: 10.3897/rio.3.e12346

Project Report

Laminar Python: tools for cortical depth-resolved analysis of high-resolution brain imaging data in Python

Julia M Huntenburg[‡,§], Konrad Wagstyl[|,¶], Christopher J Steele[‡,#], Thomas Funck[¶], Richard A.I. Bethlehem[|], Ophélie Foubet[¤], Benoit Larrat[«], Victor Borrell[»], Pierre-Louis Bazin[‡]

‡ Max Planck Institute for Human Cognitive and Brain Sciences, Leipzig, Germany
§ Free University Berlin, Berlin, Germany
| University of Cambridge, Cambridge, United Kingdom
¶ Montreal Neurological Institute, Montreal, Canada
# Douglas Mental Health University Institute of McGill University, Montreal, Canada
¤ Institut Pasteur, Paris, France
« Institut d'Imagerie Biomédicale, CEA, Paris, France
» Instituto de Neurociencias de Alicante, Alicante, Spain

Corresponding author: Julia M Huntenburg (ju.huntenburg@gmail.com)

Reviewed v1

Received: 20 Feb 2017 | Published: 23 Feb 2017

Citation: Huntenburg J, Wagstyl K, Steele C, Funck T, Bethlehem R, Foubet O, Larrat B, Borrell V, Bazin P (2017) Laminar Python: tools for cortical depth-resolved analysis of high-resolution brain imaging data in Python. Research Ideas and Outcomes 3: e12346. https://doi.org/10.3897/rio.3.e12346

Abstract

Increasingly available high-resolution brain imaging data require specialized processing tools that can leverage their anatomical detail and handle their size. Here, we present user-friendly Python tools for cortical depth resolved analysis in such data. Our implementation is based on the CBS High-Res Brain Processing framework, and aims to make high-resolution data processing tools available to the broader community.

Keywords

laminar analysis, high-resolution MRI

Introduction

Recent advances in ultra-high field and quantitative MRI facilitate non-invasive imaging of the whole brain at an unprecedented level of detail (Weiskopf et al. 2015). Standard neuroimaging software is not optimised for processing such images. Thus, there is a growing demand for dedicated tools that can take advantage of the additional information provided by the new data, and scale well with their increasing size. CBS High-Res Brain Processing Tools (CBSTools, Bazin et al. 2014) is a suite of software tools for processing MR images at submillimeter resolution. CBSTools have been developed in Java as a set of plugins for the MIPAV software package and the JIST pipeline environment (https://www.nitrc.org/projects/cbs-tools/).

In this project, we made a subset of CBSTools modules available in Python (https://github.com/juhuntenburg/laminar_python, Huntenburg 2017). The standalone package no longer requires installation of MIPAV and JIST, and allows for interactive data exploration at each processing stage. The Python interfaces also enable easy integration with other popular Python-based neuroimaging software tools such as Nibabel (Brett et al. 2016), Nipype (Gorgolewski et al. 2011) and Nilearn (Abraham et al. 2014). We focused on a set of modules that enable the analysis of multiple horizontal laminae within the cortical sheet (Waehnert et al. 2016). The package implements an equivolumetric approach for generating intracortical laminae (Waehnert et al. 2014), which accounts for the dependence of layer thickness on cortical folding (Bok 1929).

Approach

Our aim was to provide user-friendly Python interfaces to the CBSTools modules and make these available in a platform independent manner with minimal dependencies. We used the JCC package (http://lucene.apache.org/pylucene/jcc/index.html) to encapsulate the original Java classes. We then implemented a set of Python wrapper functions which convert the input data to Java data structures, initiate a Java virtual machine, call the main Java class with the specified parameters, collect, convert and return the output data.

Input and output data can either be passed as files or specific Python data structures. We chose to represent volumetric data as Nibabel SpatialImages (http://nipy.org/nibabel/reference/nibabel.spatialimages.html), in particular Nifti1Images. These standardized objects simplify data exchange with other software tools. Finding a solution to represent surface data proved to be more difficult, since neither a community standard, nor a suitable precedent solution in other Python tools exists. Here, we decided to represent a surface mesh as a dictionary with the entries *coords*, an array containing the coordinates of the mesh vertices, and *faces*, an array containing the vertex indices of the mesh faces.

Functions for loading and saving of volumetric and surface mesh data in various file formats (currently nifti, gifti, ply, vtk, obj and Freesurfer formats) can be called directly by the user, but are also employed by the main processing functions. The loading functions

automatically determine the input type: supported file formats are loaded and Python data structures are tested for compliance with the expected pattern. This approach is inspired by the input and output management in Nilearn. It makes it easy for the user to call the main functions directly on their data files, without further specifications. At the same time, it is flexible to accommodate non-standard data formats, which the user can load into the appropriate Python data structure with custom scripts.

Results

The set of functions implemented in this package enables sampling of a given intensity image on multiple intracortical laminae, starting from a simple tissue classification. We illustrated their usage in an example workflow (https://github.com/juhuntenburg/laminar_python/blob/master/examples/laminar_python_demo.ipynb). Here, the initial inputs are two binary images demarcating the inner and outer boundary of the cortical grey matter of a ferret (*Mustela putorius furo)* brain (Fig. 1a). Both images are converted into levelset representations using the *create_levelsets* function (Fig. 1b). The levelsets are passed to the *layering* function, which subdivides the intracortical space between the two boundaries in equivolumetric laminae. This function outputs three images: a continuous (Fig. 1c) and a discrete (Fig. 1d) representation of equivolumetric intracortical depth, and levelset representations of each of the intracortical surfaces. In the example, the latter output is passed to the *profile_sampling* function, together with an aligned T2 contrast image. T2 values are then sampled at different cortical depths (Fig. 1e). Importantly, the equivolumetric laminae do not represent architectonic layers, but provide an anatomically meaningful coordinate system of cortical depth.

The example data is taken from a 7 Tesla MR scan of an adult ferret (voxel size = 120 µm isotropic). With no additional manipulation, the package was readily applied to the animal data, testifying that it can also be used for cross-species analysis. Nilearn plotting functions were used for visualization, demonstrating the straightforward integration between the two packages.

Limitations and future directions

The current stage of the project faces several limitations, which might be overcome in future work. First, we focused on a subset of CBSTools modules. A more complete migration of CBSTools functionality to Python is a logical next step. Second, platform independence has not yet been achieved and requires pre-compilation of the JCC wrappers on different platforms. Third, atlases, lookup tables and example data are currently located within the GitHub repository. Better solutions for providing these files and other relevant datasets to the user should be found in the long term. Fourth, while our approach ensures general compatibility with other Python-based neuroimaging software, we aim for a closer integration, for instance by providing Nipype interfaces. Fifth, CBSTools are mainly used for processing MRI data, but are generally applicable to other types, such

as histological data. It would be interesting to expand usability to different data types and provide respective examples.

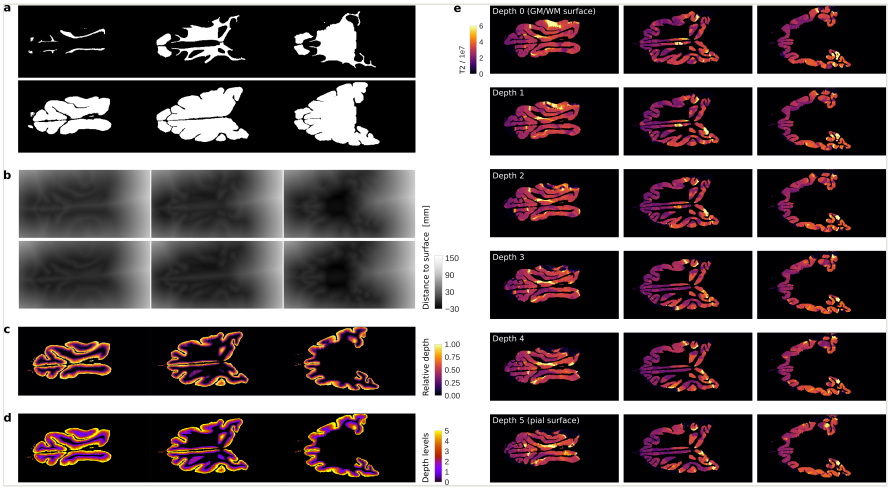

Figure 1.

Laminar python pipeline, demonstrated using high-resolution MR data of a ferret brain. a) Binary images demarcating inner (grey-white matter interface, top) and outer (pial surface, bottom) boundaries of the cortex. b) Levelset representations of the same surfaces, where positive values are assigned to voxels outside of the volume deliminated by the surface, and negative values to voxels inside, each increasing in value with euclidean distance from the surface. c) Continuous equivolumetric intracortical depth, which models the positions of laminae relative to cortical morphology. d) Discrete representations of equivolumetric depth levels. e) T2 values, sampled at the six equivolumetric intracortical depths. Note that the equivolumetric laminae do not represent architectonic layers, but provide an anatomically meaningful coordinate system of cortical depth.

Conclusion

We encapsulated a subset of CBSTools in Python and implemented user-friendly interfaces for the laminar analysis of high-resolution MR images. This is a first step to making high-resolution data processing tools available to the broader community, which also aims to encourage other scientists to contribute with their own code.

Acknowledgements

This work was completed during OHBM Hackathon Lausanne 2016 and Brainhack Anatomy Paris 2016.

Author contributions

JMH and PLB conceived the project. JMH, KW, CJS, TF and PLB contributed to the code. RAIB and OF tested the code and gave feedback for revision. JMH wrote the initial draft of the manuscript. KW, CJS, TF, RB, OF and PLB revised the manuscript. OF, BL and VB provided the example data.

Conflicts of interest

None declared.

References

- Abraham A, Pedregosa F, Eickenberg M, Gervais P, Mueller A, Kossaifi J, Gramfort A, Thirion B, Varoquaux G (2014) Machine learning for neuroimaging with scikit-learn. Frontiers in Neuroinformatics 8 https://doi.org/10.3389/fninf.2014.00014
- Bazin P, Weiss M, Dinse J, Schäfer A, Trampel R, Turner R (2014) A computational framework for ultra-high resolution cortical segmentation at 7Tesla. NeuroImage 93: 201-209. https://doi.org/10.1016/j.neuroimage.2013.03.077
- Bok ST (1929) Der Einfluss der in den Furchen und Windungen auftretenden Kruemmungen der Grosshirnrinde auf die Rindenarchitektur. Zeitschrift für die gesamte Neurologie und Psychiatrie 12: 682-750. https://doi.org/10.1007/bf02864437
- Brett M, Hanke M, Cipollini B, Côté M, Markiewicz C, Gerhard S, Larson E, Lee G, Halchenko Y, Kastman E, cindeem, Morency F, moloney, Millman J, Rokem A, jaeilepp, Gramfort A, den Bosch JFv, Subramaniam K, Nichols N, embaker, bpinsard, chaselgrove, Oosterhof N, St-Jean S, Amirbekian B, Nimmo-Smith I, Ghosh S, Varoquaux G, Garyfallidis E (2016) nibabel: 2.1.0. Zenodo https://doi.org/10.5281/ZENODO.60808
- Gorgolewski K, Burns C, Madison C, Clark D, Halchenko Y, Waskom M, Ghosh S (2011) Nipype: A Flexible, Lightweight and Extensible Neuroimaging Data Processing Framework in Python. Frontiers in Neuroinformatics 5 https://doi.org/10.3389/fninf.2011.00013
- Huntenburg J (2017) juhuntenburg/laminar_python: initial release. v1.0. Zenodo. Release date: 2017 2 02. URL: http://doi.org/10.5281/zenodo.268021
- Waehnert M, Dinse J, Schäfer A, Geyer S, Bazin P, Turner R, Tardif CL (2016) A subject-specific framework for in vivo myeloarchitectonic analysis using high resolution quantitative MRI. NeuroImage 125: 94-107. https://doi.org/10.1016/j.neuroimage.2015.10.001
- Waehnert MD, Dinse J, Weiss M, Streicher MN, Waehnert P, Geyer S, Turner R, Bazin P- (2014) Anatomically motivated modeling of cortical laminae. NeuroImage 93: 210-220. https://doi.org/10.1016/j.neuroimage.2013.03.078
- Weiskopf N, Mohammadi S, Lutti A, Callaghan M (2015) Advances in MRI-based computational neuroanatomy. Current Opinion in Neurology 28 (4): 313-322. https://doi.org/10.1097/wco.0000000000000222