Title: Nilearn for new use cases: Scaling up computational and community efforts

Authors: Yasmin Mzayek, Pierre Bellec, Paul Bogdan, Ahmad Chamma, Kun Chen, Natasha Clarke, Jelle Roelof Dalenberg, Jérôme Dockès, Elizabeth DuPre, Audrey Duran, Nicolas Gensollen, Daniel Gomez, Sami Jawhar, Connor Lane, Raphael Meudec, Dimitri Papadopoulos Orfanos, Manon Pietrantoni, Jean-Baptiste Poline, Taylor Salo, Alexis Thual, Hao-Ting Wang, Robert Williamson, Bertrand Thirion

Introduction

Nilearn (https://nilearn.github.io) is a well-established Python package that provides statistical and machine learning tools for fast and easy analysis of brain images with instructive documentation and a friendly community. This focus has led to its current position as a crucial part of the neuroimaging community's open-source software ecosystem, supporting efficient and reproducible science [1]. It has been continuously developed over the past 10 years, currently with 900 stars, 500 forks, and 176 contributors on GitHub. Nilearn leverages and builds upon other central Python machine learning packages, such as Scikit-Learn [2], that are extensively used, tested, and optimized by a large scientific and industrial community.

In recent years, efforts in Nilearn have been focused on meeting evolving community needs by increasing General Linear Model (GLM) support, interfacing with initiatives like fMRIPrep and BIDS, and improving the user documentation. Here we report on progress regarding our current priorities.

Methods

Nilearn is developed to be accessible and easy-to-use for researchers and the open-source community. It features user-focused documentation that includes a user guide and an example gallery as well as comprehensive contribution guidelines. Nilearn is also presented in tutorials and workshops throughout the year including the Montreal Artificial Intelligence and Neuroscience (MAIN) Educational Workshop, the OHBM Brainhack event, and for the Chinese Open Science Network.

The community is encouraged to ask questions, report bugs, make suggestions for improvements or new features, and make direct contributions to the source code. We use the platforms Neurostars, GitHub, and Discord to interact with contributors and users on a daily basis.

Nilearn adheres to best practices in software development including using version control, unit testing, and requiring multiple reviews of contributions. We also have a continuous integration infrastructure set up to automate many aspects of our development process and make sure our code is continuously tested and up-to-date.

Results

Nilearn supports methods such as image manipulation and processing, decoding, functional connectivity analysis, GLM, multivariate pattern analysis, along with plotting volumetric and surface data.

In the latest release, cluster-level and TFCE-based family-wise error rate (FWER) control have been added to support the mass univariate and GLM analysis modules, expanding from the already implemented voxel-level correction method (see Fig1). Optimizing Nilearn's maskers is also underway such as the recently added classes for handling multi-subject 4D image data. These also provide the option to use parallelization to speed up computation.

In addition, Nilearn has introduced a new theme to update the documentation making the website more readable and accessible (https://nilearn.github.io/). This change also sets the stage for further improvement and modernisation of several aspects of the documentation, like the user guide.

Development on Nilearn's interfaces module added a new function to write BIDS-compatible model results to disk. This and further development of the BIDS interface will facilitate interaction with other relevant community tools such as FitLins [3]. Finally, several surface plotting enhancements are in progress including improving the API for background maps (see Fig2).

Conclusion

Nilearn is extensively used by researchers of the neuroimaging community due to its implementations of well-founded methods and visualization tools which are often essential in brain imaging research for quality control and communicating results. Recent work has highlighted areas where more active work is needed to scale the project both technically and socially, including: working with large datasets, better supporting analyses on the cortical surface, and advancing standard practice in neuroimaging statistics through active community outreach.

References

[1] Poldrack, R., Gorgolewski, K., Varoquaux, G. (2019). Computational and Informatic Advances for Reproducible Data Analysis in Neuroimaging Annual Review of Biomedical Data Science 2(1), 119-138. https://dx.doi.org/10.1146/annurev-biodatasci-072018-021237

[2] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830.

[3] Markiewicz, C. J., De La Vega, A., Wagner, A., Halchenko, Y. O., Finc, K., Ciric, R., Goncalves, M., Nielson, D. M., Kent, J. D., Lee, J. A., Bansal, S., Poldrack, R. A., Gorgolewski, K. J. (2022). poldracklab/fitlins: 0.11.0 (0.11.0). Zenodo. https://doi.org/10.5281/zenodo.7217447

Figures

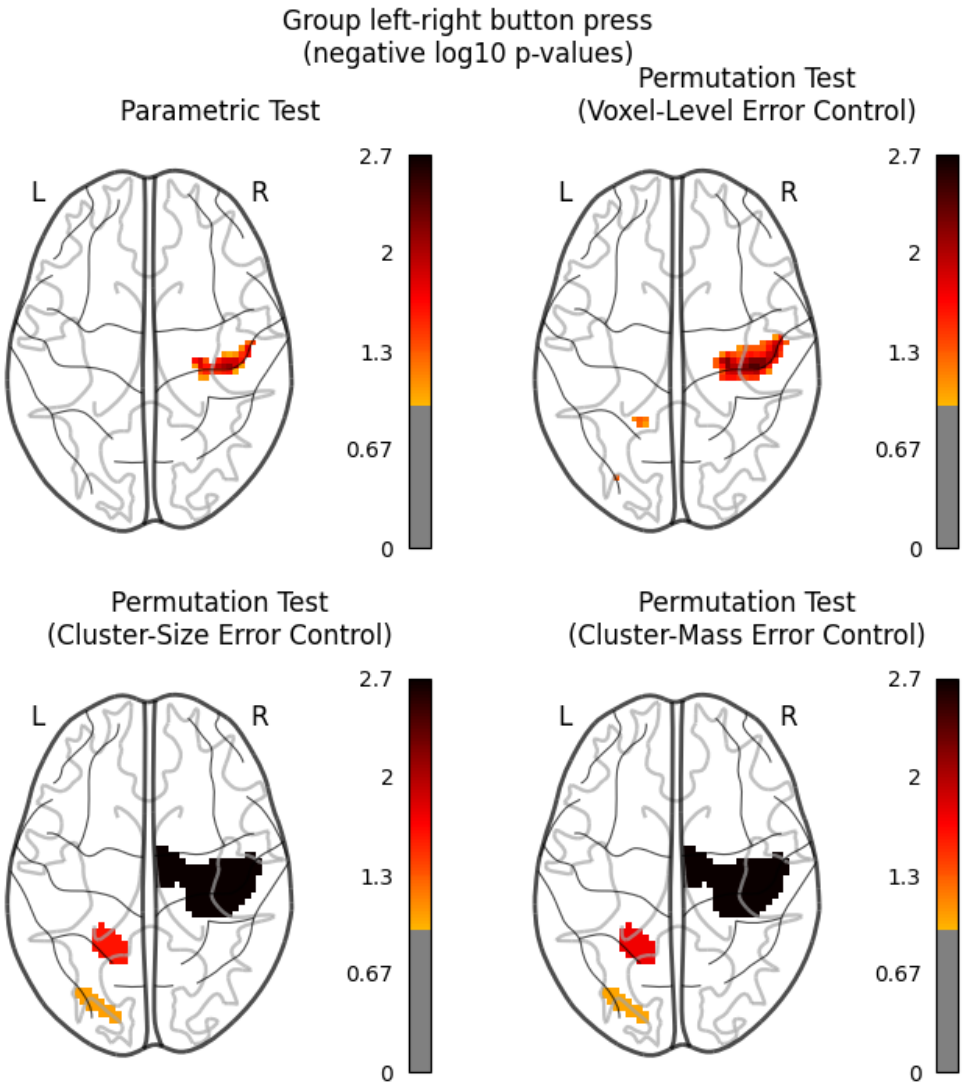

Fig1. Comparison of voxel-level and cluster-level error control of results from a second-level GLM analysis.

Motor Cortex Activation on Surface

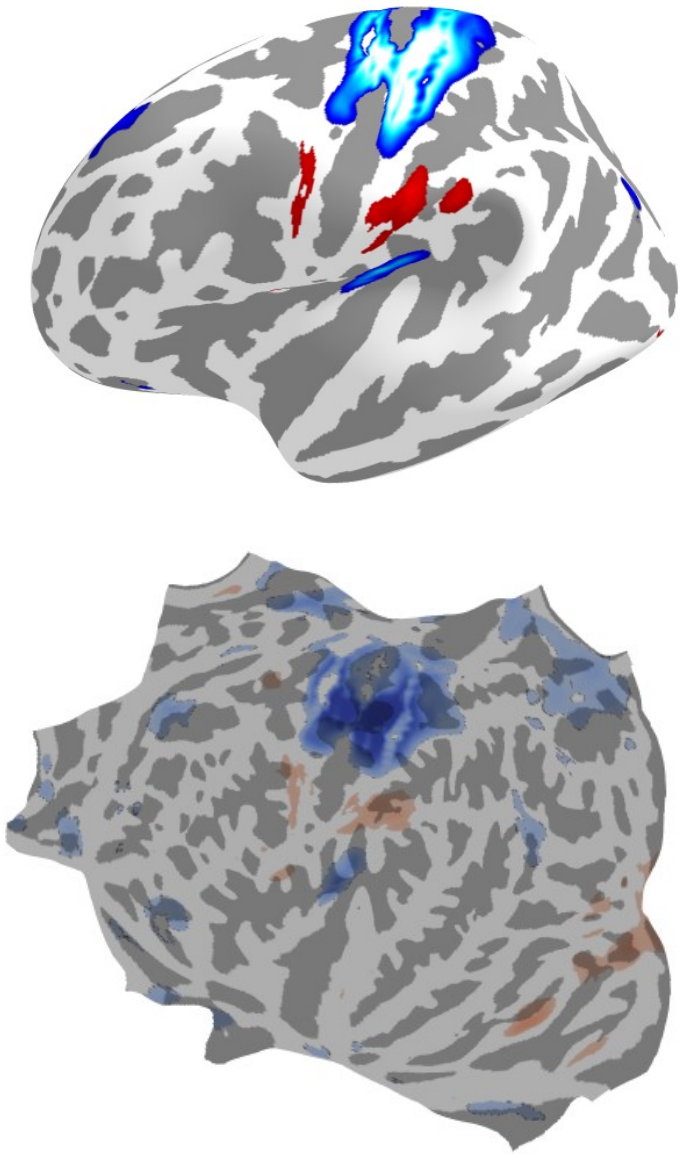

Fig2. Motor cortex activation map plotted on an inflated surface with plotly (top) and on a flat surface (bottom), both using curvature sign as a background map.