OHBM2026 abstracts

Maximum numbers of characters:

- 2000ch for introduction
- 4000ch for Methods, Results & Conclusion

Source: https://www.overleaf.com/project/6890748d898a11494561f7c2

BIDSme containerized & in Neurodesk

Authors & affiliation

Bradley Spitz 1,2, Nikita Beliy 2, Fabienne Collette 2, Christophe Phillips 2

1 Télécom Physique Strasbourg, Université de Strasbourg, France 2 GIGA - CRC Human Imaging, University of Liège, Liège, Belgium

spitzbradley@gmail.com

Nikita.Beliy@uliege.be

F.Collette@uliege.be

C.Phillips@uliege.be

Title:

"*Containerizing BIDSme : A Reproducible Tool for BIDS Conversion*"

Introduction:

The Brain Imaging Data Structure (BIDS) scheme [1] has emerged as a community-driven standard for organizing and sharing neuroimaging data, helping reproducibility and interoperability across studies and software tools. However, transforming "raw"1 neuroimaging data into a valid BIDS format remains a complex, error-prone, and often manual process. BIDSme is an open-source Python application developed at the GIGA-CRC Human Imaging research unit to facilitate this "BIDS-ification" of neuroimaging data. It provides a configurable, user-guided workflow for renaming, restructuring, and enriching datasets with metadata, allowing flexibility beyond rigid BIDS-ification tools [2].

Despite its potential, BIDSme was not initially designed for broad distribution. It required local installation using Python virtual environment. This limited its accessibility for end-users, particularly those without technical expertise in Python or environment management.

To overcome these limitations, we propose a containerized version of BIDSme using Docker [3], along with a simplified interface and integration into platforms like Neurodesk [4]. This paper outlines the containerization process, design choices, usability improvements, and validation strategies, aiming to make BIDSme a robust and portable tool for the neuroimaging community.

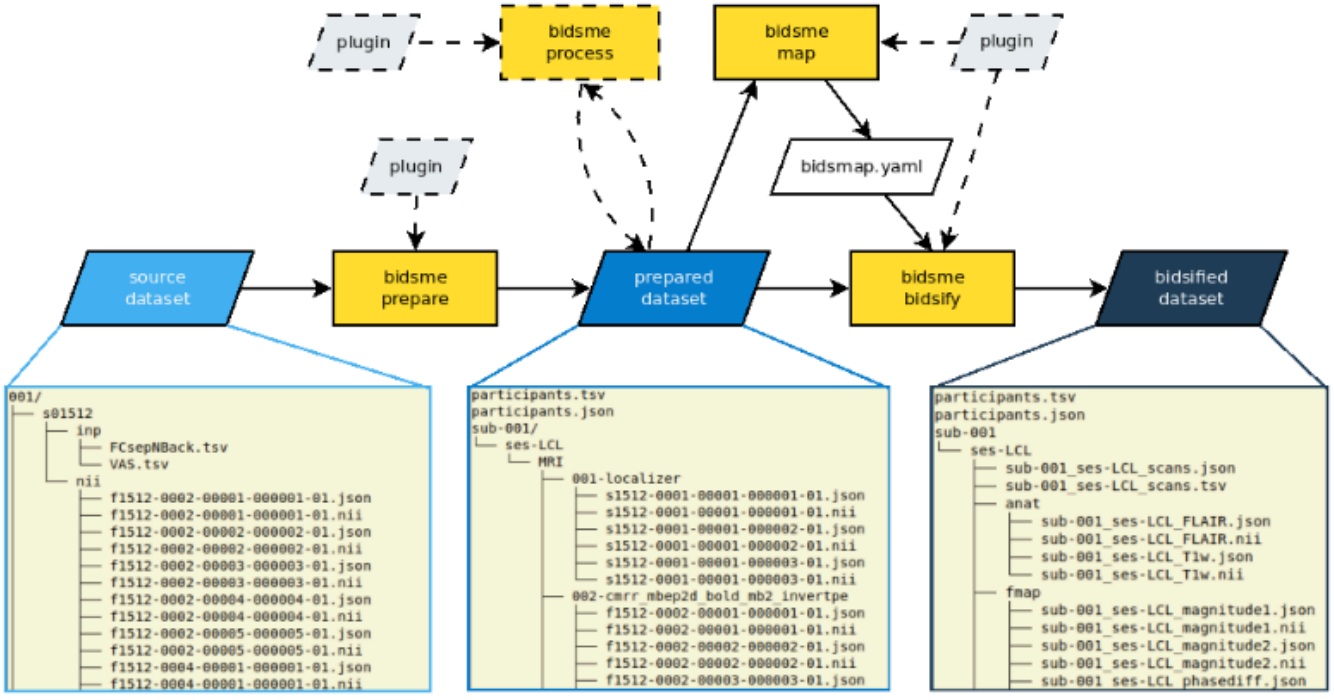
Methods:

Unlike fully automated converters, BIDSme offers a semi-automated, user-guided approach, allowing flexibility to handle complex or heterogeneous datasets typically found in research settings. Its modular design enables users to perform key steps such as renaming raw files, reorganizing directory structures, and mapping metadata fields, thereby reducing manual effort and minimizing errors. The software supports a wide range of neuroimaging modalities and formats.

The core functionalities of BIDSme are implemented as four main commands, see Fig.1:

- `prepare`: Organizes and preprocesses raw data by renaming files and arranging them into an intermediate directory structure that facilitates subsequent BIDSification.
- `map`: Maps and transfers relevant metadata and filenames from the prepared data to the final BIDS-compliant structure.
- `bidsify`: Finalizes the conversion by applying BIDS-compliant naming conventions.
- `process`: Processes the bidsified dataset prior to final BIDSification. This command can be used to produce derivatives, perform data conversion, or execute anonymization while maintaining the advantage of recording identification through the use of `bidsmap.yaml`. Essentially, it performs all processing steps similar to BIDSification but without executing the BIDSification itself.

These operations can be executed sequentially or individually, providing users with fine control over the conversion workflow.



To support BIDSme's file processing workflow and promote clarity across executions, a well-structured and consistent directory layout was established inside the container. All user-facing operations are carried out under a single shared workspace, typically mounted at `/mnt`, where input, output, and configuration files are organized. The default expected structure includes the following subdirectories:

- `/mnt/rawdata` contains the original neuroimaging files to be processed.
- `/mnt/prepared` serves as the output directory for pre-processed data, ready for BIDSification.
- `/mnt/bidsified` stores the final BIDS-compliant dataset.

- `/mnt/configuration` holds configuration files and plugins such as `bidsmap.yaml` for flexible mapping logic.

To improve usability and offer more flexibility, a custom entrypoint.sh script was added to the container. This script detects the mode of execution requested by the user and automatically launches the appropriate interface: **Command Line Interface** (**CLI**), **Interactive Shell**, or **JupyterLab**. This unified entry point simplifies container usage across different workflows and user preferences, see Fig.2. In addition, a preconfigured Jupyter notebook is included within the container. When JupyterLab is launched, this notebook serves as a starting point.

Results:

Once the container was built, functional testing was performed to ensure that BIDSme's core features operated correctly within the Dockerized environment. The validation covered the full pipeline, including the `prepare`, `map`, `bidsify`, and `process` commands. The correctness of the output directories (`prepared/`, `bidsified/`) and the absence of runtime errors were used as criteria for validation.

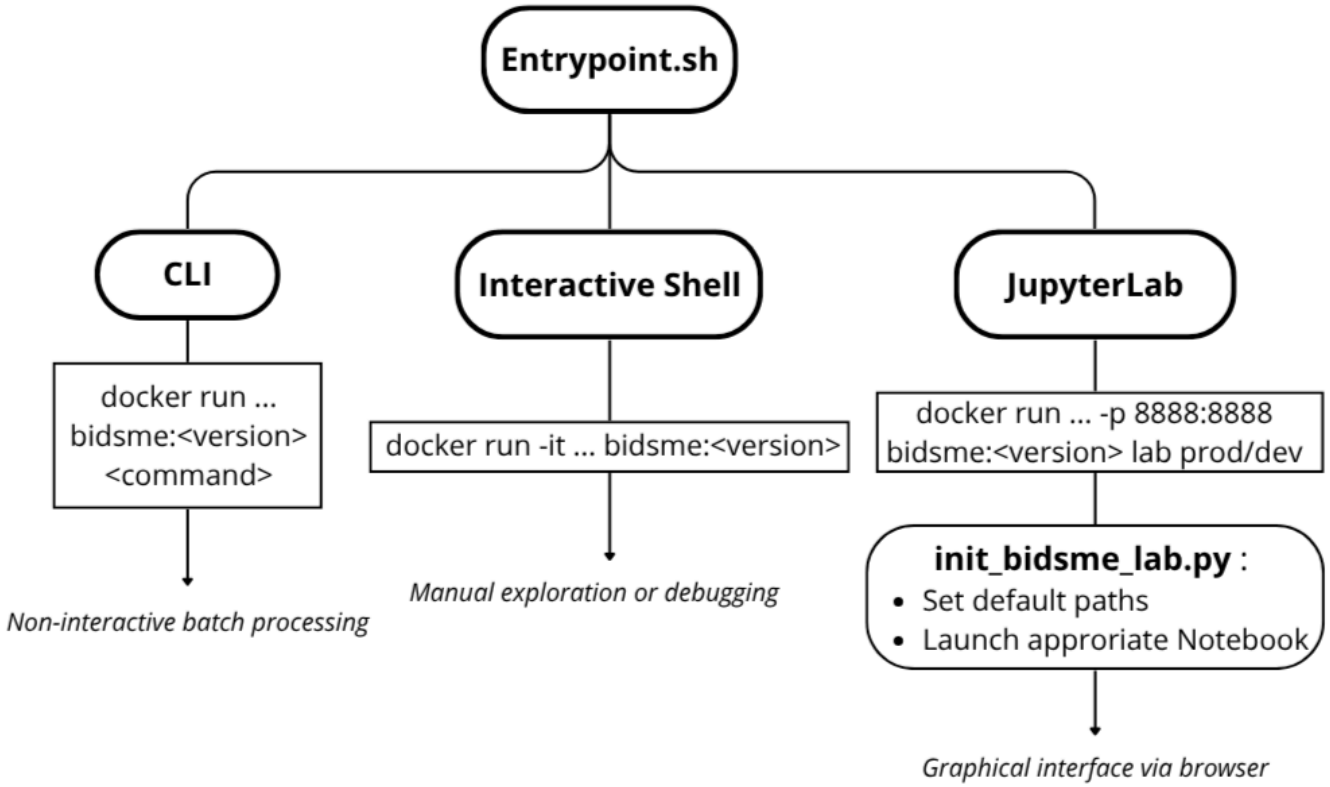
To ensure consistency and reliability across usage modes, each interface — CLI, interactive shell, and JupyterLab — was tested independently. The CLI was validated using direct docker run commands, confirming that input/output volumes were properly mounted and outputs were written as expected. The JupyterLab interface was tested to ensure that notebooks could be executed end-to-end, and that preloaded configurations and paths were properly set up.

To test reproducibility, the same dataset was processed multiple times under identical container configurations across different machines. The outputs were identical in both structure and content, demonstrating that the container provides a reproducible environment across platforms. This process also helped identify potential permission issues arising from Docker's default behavior, which often assigns root ownership to files and directories created within containers.

Depending on how the container is started (via CLI or Compose), the appropriate mode is automatically triggered by the `entrypoint.sh` logic, see Fig. 2. A dedicated configuration notebook is also embedded in the container to assist users launching via JupyterLab. For example, the JupyterLab interface in production mode can be launched using a full Docker command:

```
docker run -p 8888:8888 \
-v $(pwd)/rawdata_prod:/mnt/rawdata \
-v $(pwd)/prepared_prod:/mnt/prepared \
-v $(pwd)/bidsified_prod:/mnt/bidsified \
-v $(pwd)/configuration_prod:/mnt/configuration \
bidsme:<version> lab prod
```

Alternatively, the same setup can be started using the predefined production Compose file: `docker compose -f docker-compose.prod.yml run --service-ports bidsme lab`



Each method provides consistent behavior, with automatic initialization of environment variables and preloaded notebook paths for a smooth user experience. However, Docker Compose proves especially useful by reducing a long and error-prone multi-line Docker command to a single, concise instruction.

Finally BIDSme was integrated into Neurodesk, using the existing Docker image that had already been containerized and optimized. This approach ensured a reliable and reproducible environment while reducing development overhead. Within the Neurodesk environment, BIDSme is currently accessible through the module loading mechanism. Thus BIDSme cannot be directly imported as a Python package and access is therefore limited to its CLI and JupyterLab once the module is loaded.

Conclusion:

The successful integration of BIDSme as a container and into Neurodesk delivers multiple benefits for the neuroimaging research community:

- Accessibility: BIDSme can be accessed immediately without requiring manual installation
- Reproducibility: A consistent and controlled containerized environment ensures reproducible results
- Collaboration: Researchers share the same standardized BIDS conversion tools
- Education: Provides newcomers with preconfigured access to BIDSme, lowering the entry barrier
- Maintainability: Centralized updates via Neurodesk simplify long-term maintenance.

Overall, this integration significantly improves the accessibility of BIDSme and facilitates the adoption of BIDS standards across diverse research environments. By encapsulating all dependencies and configuration, it eliminates the variability often encountered when neuroimaging pipelines are run on different systems. Providing several ways to use the containerized app gives users the freedom to perform processing according to their preferences and needs, while reducing the tediousness of this kind of work.

Despite the advantages provided by the containerized setup, the final image remains relatively large (approximately 1 GB), which may pose challenges for environments with limited storage or bandwidth. Additionally, using the container without Docker Compose can be bothersome, as it requires manually specifying long and complex volume-mounting commands. Additionally, although containerization improves portability, a basic familiarity with Docker is still necessary for effective usage, which may represent a barrier for users without prior experience in container technologies.

BIDSme has already been successfully integrated into the Neurodesk ecosystem as a containerized application. This ensures accessibility and reproducibility across platforms, in line with Neurodesk's modular container-based approach. However, since BIDSme is a pure Python tool, keeping it inside a dedicated container is not strictly necessary. A more sustainable approach would be to make BIDSme directly available within Neurodesk's shared Python environment (via a virtual environment installation). This would reduce maintenance overhead, lower image size, and simplify updates, while still providing users with the same level of accessibility.

Future work will therefore focus on discussing with the Neurodesk team whether such an installation strategy could be adopted, ensuring that BIDSme remains fully integrated while avoiding the unnecessary burden of maintaining a standalone container.

Code availability:

- main BIDSme development, https://github.com/CyclotronResearchCentre/bidsme
- BIDSme containerization, https://github.com/CyclotronResearchCentre/BIDSme_containerisation
- Neurodesk, https://neurodesk.org/
- Docker, https://www.docker.com/

References/Citations

1. Gorgolewski C. et al. "The Brain Imaging Data Structure, a Format for Organizing and Describing Outputs of Neuroimaging Experiments," Scientific Data, vol. 3, no. 1, p. 160044, Jun. 2016.
2. Beliy N. et al., "BIDSme: Expandable BIDS-ifier of Brain Imagery Datasets," Journal of Open Source Software, vol. 8, no. 92, p.5575, Dec. 2023.
3. Merkel D. , "Docker: lightweight linux containers for consistent development and deployment," Linux journal, vol. 2014, no. 239, p. 2, 2014.
4. Renton A. I. et al., "Neurodesk: an accessible, flexible and portable data analysis environment for reproducible neuroimaging," Nature Methods, vol. 21, no. 5, pp. 804–808, 2024. https://doi.org/10.1038/s41592-023-02145-x
5.

Acknowledgement

C. Phillips and F. Collette are supported by the F.R.S.-FNRS, Belgium. The original development of BIDSme as well as N. Beliy at the time were supported by the "Memodyn" EoS grant from the F.R.S.-FNRS and the FWO, Belgium.

Primary & Secondary Parent Category & Sub-Category

Primary

- Neuroinformatics and Data Sharing
    - Databasing and Data Sharing

Secondary

- Neuroinformatics and Data Sharing
    - Workflows

Keywords

(up to 10 from a list)

Behavior Diffusion MRI EEG/ERP Functional MRI PET Structural MRI MEG Neurophysiology