


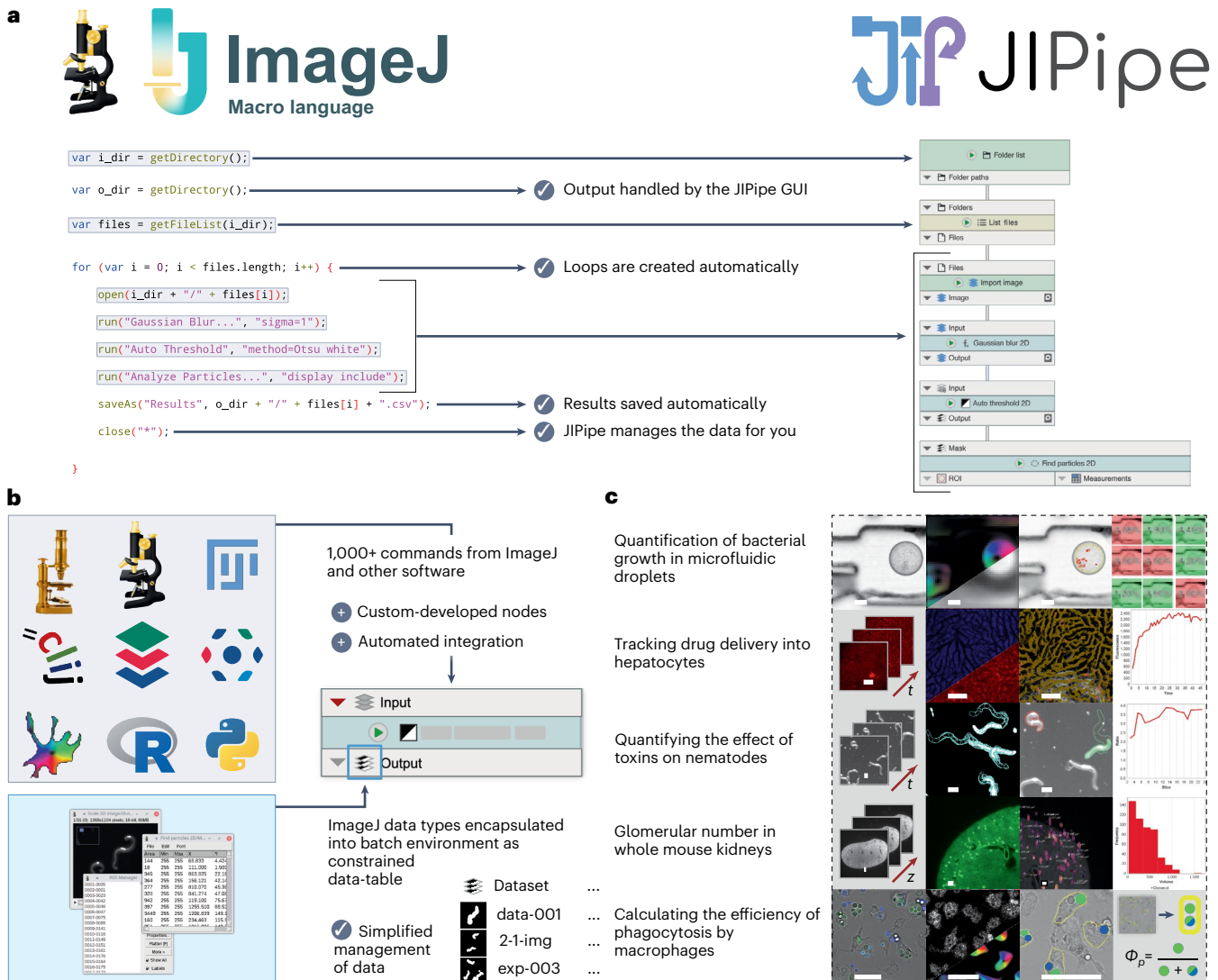
# JIPipe: visual batch processing for ImageJ

 Check for updates

The growth in microscopy adoption has led to a concomitant upsurge in the development of software tools for the automated analysis of image data. Pillars among these tools are ImageJ<sup>1</sup> and its Fiji<sup>2</sup> distribution, which have been serving the imaging community for decades and continue to gain public support to keep up with the quantification

needs of the newest and most-demanding microscopy techniques. The hallmark of ImageJ is its intuitive graphical user interface, which provides access to its many tools. On the other hand, the creation of reproducible batch-processing workflows is only possible using a macro language. As programming skills are uncommon among experimentalists<sup>3</sup>, the need for scripting contributes to an

already-existing communication gap between life and computer scientists. Visual programming languages that replace the writing of text commands with the design of a flowchart offer a solution. Existing tools contribute to this effort by providing a visual way to build pipelines or by simplifying the scripting procedure (Supplementary Information, section 1). Our newly developed visual programming



**Fig. 1 | Overview of JIPipe. a**, JIPipe is a visual programming language to realize code-free workflow building for ImageJ-based image analyses. GUI, graphical user interface. **b**, Currently, JIPipe unifies the functionality of over 1,000 ImageJ commands into a standardized interface, represented as nodes in the pipeline

flow chart. The window-based data management implemented in ImageJ is replaced with a table-based model designed for batch processing. **c**, A range of applications showcases the versatility of JIPipe. Scale bars, 100  $\mu$ m.

language, which we term Java image processing pipeline (JIPipe) (<https://www.jipipe.org>), provides a macro programming alternative that is particularly designed for ImageJ and supports the transition from interactive single-image manipulation to multi-image algorithmic batch processing (Fig. 1a).

The close relationship between JIPipe and ImageJ is symbiotic in nature. JIPipe already encapsulates over 1,000 commands from the default ImageJ installation as well as from numerous plugins, in form of easily deployable nodes. ImageJ is capable of accessing all JIPipe-based algorithms from within its graphical user interface and macro language. This mutual exchange of functionalities will advance the way in which image quantification and visualization pipelines are built using either of the two platforms. Current ImageJ users will find it natural to work with JIPipe functions owing to their familiar behavior, while taking advantage of the project-based design of workflows, features to organize larger pipelines and easy scaling to batch analysis, as well as the standardized interface to parameters and documentation.

The assortment of nodes covers functionalities of ImageJ and various plugins that are captured automatically by our ImageJ2 integration or by dedicated JIPipe extensions (Fig. 1b). Aside from ImageJ functionalities, we also provide R and Python-based scripting, and an integration of cellpose<sup>4</sup>. A full list of all dependency libraries and external tools is given in Supplementary Information, section 2. JIPipe imposes a strict standardization of nodes and their parameters, as well as their inputs and outputs, and facilitates the adoption of the FAIR ('findability, accessibility, interoperability and reusability') principles<sup>5</sup> by the implementation of standardized storage formats for pipeline projects, data and metadata. Our software organizes all data through a constrained table model that enforces the existence of a primary data column of a specific suitable type. The data model enables an intuitive solution for batch processing with multiple inputs without the reliance on structural loop nodes. This is achieved by automatically assigning the data to appropriate text-metadata columns that are generated from the input files or by user input. The result is a highly flexible node model that unambiguously communicates the expected data types to the users. The range of JIPipe functionalities

is extendable through plugins that are built upon the existing Java-based library ecosystem of ImageJ, thus opening our platform to all related ongoing community-driven and open-source efforts. This allows JIPipe to keep pace with the increasing complexity of new image-analysis tasks arising from the continuously improved set of imaging techniques.

To highlight the abilities of JIPipe, we show examples from a wide variety of applications in which JIPipe was successfully used to quantify image data (Fig. 1c). High-throughput quantification of bacterial growth inside droplets (first row, Fig. 1c) was carried out both with a classical analysis pipeline<sup>6</sup> and using our cellpose<sup>4</sup> nodes. Additionally, JIPipe workflows were written to investigate the drug delivery efficiency of nanocarriers in the liver<sup>7</sup> (second row, Fig. 1c); to test the survival ratio of nematodes that digested toxin-producing bacteria<sup>8</sup> (third row, Fig. 1c); to count glomeruli in healthy and pathogenic kidneys<sup>9</sup> (fourth row, Fig. 1c); and to analyze confrontation assays between macrophages and fungal spores<sup>10</sup> (fifth row, Fig. 1c).

In summary, JIPipe fills a niche by contributing a visual alternative to macro programming that is particularly designed for ImageJ, thus simplifying the development of advanced automated image-processing methods, bridging the gap between experimentalists and computer scientists and facilitating the adoption of the FAIR principles. JIPipe is fully open source and continues to be developed in close collaboration with the wider ImageJ community (<https://image.sc/>).

## Data availability

The project files and example data used to demonstrate JIPipe, as well as example pipelines, are available at <https://doi.org/10.6084/m9.figshare.19733320.v3>.

## Code availability

The JIPipe software, training material, full documentation and source code are available at <https://www.jipipe.org/> and <https://www.github.com/applied-systems-biology/JIPipe/> (<https://doi.org/10.5281/zenodo.6532719>). JIPipe is also available from within the ImageJ update service.

Ruman Gerst<sup>1,2,4</sup>, Zoltán Cseresnyés<sup>1,4</sup> & Marc Thilo Figge<sup>1,3</sup> ✉

<sup>1</sup>Applied Systems Biology, Leibniz Institute

for Natural Product Research and Infection Biology – Hans Knöll Institute (HKI), Jena, Germany. <sup>2</sup>Faculty of Biological Sciences, Friedrich Schiller University Jena, Jena, Germany. <sup>3</sup>Institute of Microbiology, Faculty of Biological Sciences, Friedrich Schiller University Jena, Jena, Germany. <sup>4</sup>These authors contributed equally: Ruman Gerst, Zoltán Cseresnyés.

✉ e-mail: [thilo.figge@leibniz-hki.de](mailto:thilo.figge@leibniz-hki.de)

Published online: 10 January 2023

## References

1. Rueden, C. T. et al. *BMC Bioinformatics* **18**, 529 (2017).
2. Schindelin, J. et al. *Nat. Methods* **9**, 676–682 (2012).
3. Martins, G. G. et al. *F1000 Res.* **10**, 334 (2021).
4. Stringer, C., Wang, T., Michaelos, M. & Pachitariu, M. *Nat. Methods* **18**, 100–106 (2021).
5. Wilkinson, M. D. et al. *Sci. Data* **3**, 160018 (2016).
6. Svensson, C.-M. et al. *Small* **15**, 1802384 (2019).
7. Muljajew, I. et al. *ACS Nano* **15**, 12298–12313 (2021).
8. Büttner, H. et al. *Proc. Natl. Acad. Sci. USA* **118**, e2110669118 (2021).
9. Klingberg, A. et al. *J. Am. Soc. Nephrol.* **28**, 452–459 (2017).
10. Cseresnyés, Z., Kraibooj, K. & Figge, M. T. *Cytometry A* **93**, 346–356 (2018).

## Acknowledgements

This work was financially supported by the International Leibniz Research School for Microbial and Biomolecular Interactions Jena – ILRS Jena (R.G.). The German Research Foundation (DFG) funded this project through the Collaborative Research Center PolyTarget 1278 – project number 316213987, subproject Z01 (Z.C.). This work was also supported by the Collaborative Research Center Funginet 124 – project number 210879364, subproject B4 (M.T.F.), by the Cluster of Excellence 'Balance of the Microverse' under Germany's Excellence Strategy (EXC 2051 – project ID 390713860) (M.T.F.) and by the Leibniz ScienceCampus InfectoOptics Jena (M.T.F.), which is financed by the funding line Strategic Networking of the Leibniz Association. We received support from the Federal Ministry of Education and Research, Germany (grant number 13GW0456B) in the context of the InfectoGnostics Research Campus Jena (M.T.F.). We are particularly thankful to M. Roth (microfluidic droplets data), C. Hertweck (nematode imaging), K. Voigt (confrontation assay images), A. Press (intravital liver microscopy data) and M. Gunzer (kidney light-sheet images) for kindly providing image data.

## Author contributions

R.G. developed the software. Z.C. designed the bioimage analysis pipelines and tested the software. M.T.F. and Z.C. conceived the idea. M.T.F. directed and supervised the project. All authors wrote the initial draft, read and contributed to the paper, and approved the content.

## Competing interests

The authors declare no competing interests.

## Additional information

**Supplementary information** The online version contains supplementary material available at <https://doi.org/10.1038/s41592-022-01744-4>.

**Peer review information** *Nature Methods* thanks Christopher Schmedt, Mark Willett and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.