

Reproducible image-based profiling with Pycytominer

Received: 22 November 2023

Accepted: 24 January 2025

Published online: 03 March 2025

 Check for updates

Erik Serrano¹, Srinivas Niranj Chandrasekaran², Dave Bunten¹, Kenneth I. Brewer³, Jenna Tomkinson¹, Roshan Kern^{1,4}, Michael Bornholdt², Stephen J. Fleming⁵, Ruifan Pei², John Arevalo², Hillary Tsang², Vincent Rubinetti¹, Callum Tromans-Coia², Tim Becker², Erin Weisbart², Charlotte Bunne², Alexandr A. Kalinin², Rebecca Senft², Stephen J. Taylor¹, Nasim Jamali², Adeniyi Adeboye², Hamdah Shafqat Abbasi², Allen Goodman^{2,6}, Juan C. Caicedo^{2,7}, Anne E. Carpenter², Beth A. Cimini², Shantanu Singh²✉ & Gregory P. Way¹✉

Advances in high-throughput microscopy have enabled the rapid acquisition of large numbers of high-content microscopy images. Next, whether by deep learning or classical algorithms, image analysis pipelines commonly produce single-cell features. To process these single cells for downstream applications, we present Pycytominer, a user-friendly, open-source Python package that implements the bioinformatics steps key to image-based profiling. We demonstrate Pycytominer's usefulness in a machine-learning project to predict nuisance compounds that cause undesirable cell injuries.

In the past 30 years, high-content microscopy has undergone a remarkable technological transformation that has given scientists the ability to acquire thousands of single-cell measurements in high-throughput experiments¹. In turn, open-source image analysis software has proliferated, including Fiji², CellProfiler³ and others. These tools can derive biological insights from large microscopy datasets, but they lack functions for downstream bioinformatics processing of image-based features. Recently, the field of image-based profiling has emerged, which requires this kind of processing to accomplish various biological use-cases^{4,5}. Thus far, the primary applications for image-based profiling have been in drug development^{4,6}. Specifically, image-based profiling offers disease phenotype discovery, target identification, drug repurposing, toxicity assessment and the exploration of novel therapeutic hypotheses⁷. Image-based profiling also enables innovative studies into fundamental biological processes such as cell death^{8,9}, grouping functional genes¹⁰ and mitochondrial dynamics¹¹.

To generate image-based profiles, scientists first prepare cell samples that can be subjected to small molecule or genetic perturbations.

Scientists then apply microscopy to image these cells, which are often stained with fluorescent dyes to mark specific cellular compartments (Fig. 1a)⁴. During image analysis, scientists perform image correction, cell segmentation and high-content single-cell feature extraction. Feature extraction algorithms (including emerging approaches based on deep learning) quantify morphological properties per cell, such as shapes, sizes, stain intensities and more (Fig. 1a). This process generates image-based profiles, which Pycytominer then processes for downstream analyses (Fig. 1b). Users can analyze profiles at different levels: the single-cell level, which is ideal for capturing cellular heterogeneity, or the aggregate level (for example, per well), which captures morphology signatures.

Here we introduce Pycytominer, an open-source Python¹² package for designing and running image-based profiling bioinformatics pipelines. This project resulted from discussions and consensus stemming from the community's inaugural work together⁵. We distribute Pycytominer under the permissive BSD-3-Clause license, allowing for flexible use, modification and distribution (Extended Data Fig. 1).

¹Department of Biomedical Informatics, University of Colorado School of Medicine, Aurora, CO, USA. ²Imaging Platform, Broad Institute of MIT and Harvard, Cambridge, MA, USA. ³Independent Researcher, San Francisco, CA, USA. ⁴Case Western Reserve University, Cleveland, OH, USA. ⁵Data Sciences Platform, Broad Institute of MIT and Harvard, Cambridge, MA, USA. ⁶Genentech gRED, South San Francisco, CA, USA. ⁷Morgridge Institute for Research, University of Wisconsin-Madison, Madison, WI, USA. ✉e-mail: shantanu@broadinstitute.org; gregory.way@cuanschutz.edu

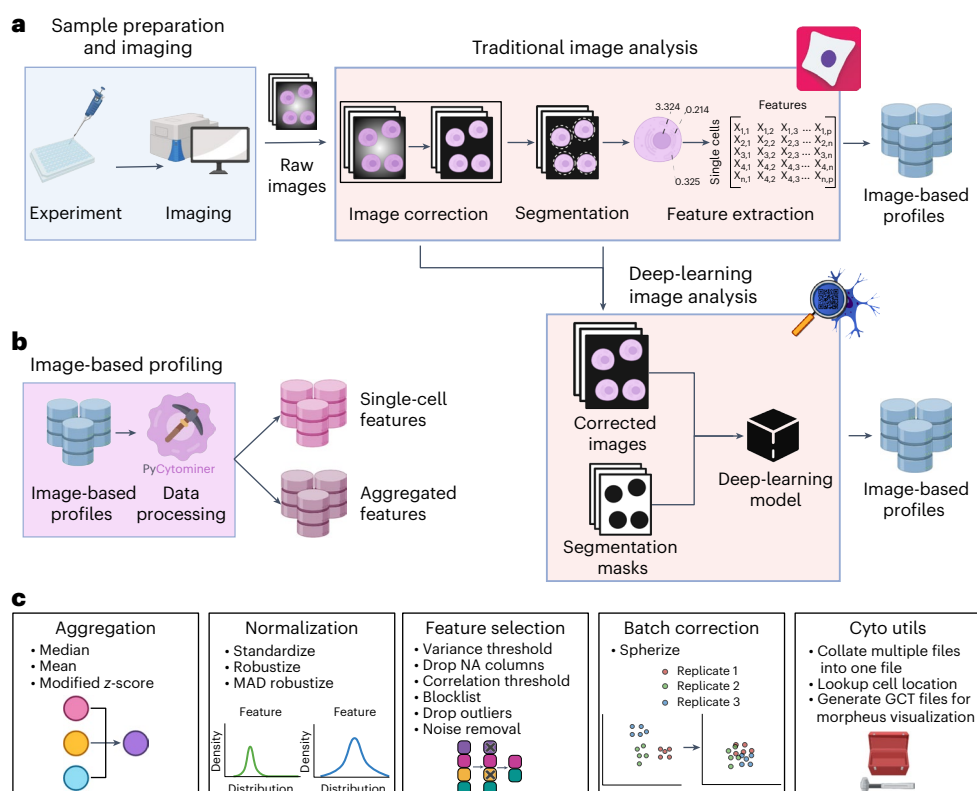


Fig. 1 | The standard image-based profiling experiment and the role of Pycytominer. a, In the experimental phase, a scientist plates cells, often perturbing them with chemical or genetic agents, and performs microscopy imaging. In image analysis, using CellProfiler for example, a scientist applies several data-processing steps to generate image-based profiles. In addition, scientists can apply deep-learning tools, such as DeepProfiler, to generate

image-based profiles. **b**, Pycytominer performs image-based profiling to process morphology features and make them ready for downstream analyses. **c**, Pycytominer performs five fundamental functions, each implemented with a simple and intuitive API. Each function enables a user to implement various methods for executing operations. Created in [BioRender.com](https://www.biorender.com).

Pycytominer is reliable, maintaining a 95% code testing coverage at the time of writing. We developed Pycytominer using Pandas¹³, Apache Parquet (Apache Software Foundation) and SQLAlchemy¹⁴, and it offers a modular application programming interface (API) to enable flexible customization. The Pycytominer API performs core functions—aggregate, annotate, normalize and feature selection—and offers several helpful utility functions such as spherize, a nonparametric batch effect correction algorithm to adjust for technical variation (Fig. 1c and Supplementary Table 1)¹⁵. This functionality and modularity make Pycytominer well suited for both small and large experiments.

Pycytominer distinguishes itself from related image-based profiling tools (see Supplementary Table 2 for a comparison) by providing a comprehensive, user-friendly, and actively maintained solution in the widely-used Python programming language, with seamless integration with popular data-processing libraries. Pycytominer has already been used in many applications. Most notably, Pycytominer processed three of the largest publicly available high-content microscopy datasets, Joint Undertaking in Morphological Profiling¹⁶, Library of Integrated Network-Based Cellular Signatures (LINCS)¹⁰ and EU-OPENSREEN¹⁷. Pycytominer has also processed the majority of the currently 31 Cell Painting datasets in the Cell Painting Gallery¹⁸. With the Pycytominer API, users can construct workflows that process image-based profiling data from a variety of entry points. The most standard entry point is raw output data from CellProfiler processed images, but Pycytominer also supports other image analysis tools such as In Carta and Harmony. Another entry point is preprocessed public image-based profiling data, which can integrate into Pycytominer functions with little modification. We demonstrate this public data entry point in the subsequent

application. This flexibility and simplicity offers new scientists easy development and customization, accelerating their research efforts. We encourage Pycytominer users to look at these existing resources for example pipelines, but we also provide full tutorials, user documentation, a handbook, an orchestration recipe and comprehensive walkthroughs (Extended Data Fig. 2; links in Methods).

To showcase Pycytominer usage, we reprocessed a publicly available high-content microscopy dataset from a study aimed at discerning nuisance compounds that induce undesirable cell injuries (Fig. 2a)¹⁹. Using the public image-based profiles, we applied Pycytominer's feature selection functionality to select the most informative and least-redundant features. Downstream of Pycytominer, we trained a multiclass logistic regression model to predict 15 cell injury categories. We split data into training and testing sets, and included three independent and different types of holdout data: whole plate, whole treatments and individual wells (Methods and Supplementary Table 3). Overall, our model demonstrated strong performance in identifying most cellular injury types including 'cytoskeletal', 'HSP90' and 'HDAC' with high F1 scores (Fig. 2b). To make these injury predictions, the model used combinations of cell morphology features, which can be directly interpreted to increase understanding of cell injury phenotypes (Supplementary Table 4).

Notably, other cell injuries such as 'tannin', 'saponin' and 'nonspecific reactive' showed relatively low performance. Overall, predictions were consistently strong, even in wells and plates held out from training (Fig. 2c) and especially when compared to a model trained with shuffled data ('shuffled model') (Extended Data Fig. 3). When holding out individual treatments, we observed high performance for predicting 'genotoxin' injuries, but poor performance for others (such as

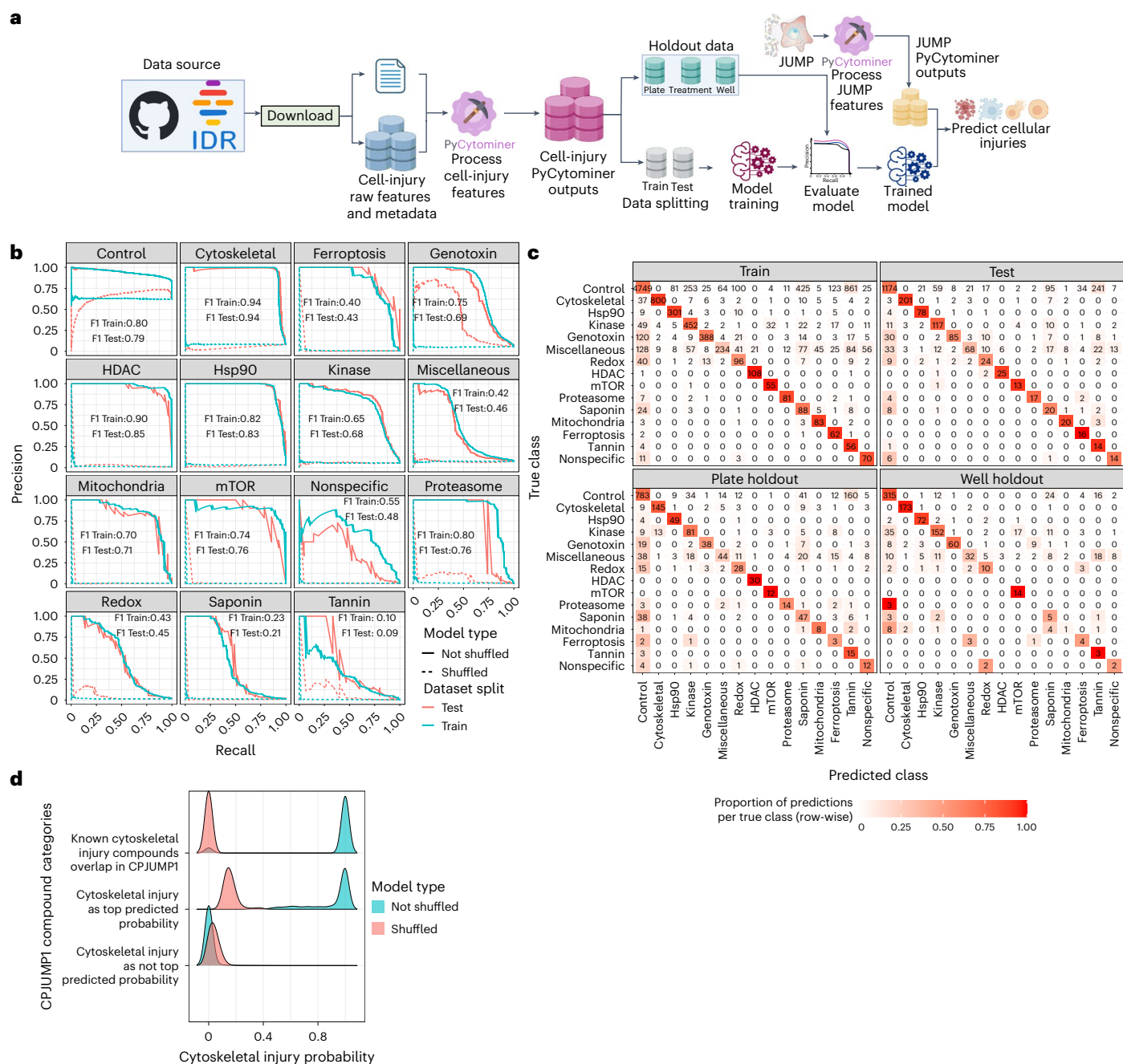


Fig. 2 | Model performance and evaluation with CPJUMP1 data. **a**, Our Pycytominer-based workflow to process publicly available data and to train a machine-learning model to predict cellular injury. We then retrained our model using only features that overlapped with the publicly available CPJUMP1 dataset. Note that CPJUMP1 was also previously processed using Pycytominer. Created in [BioRender.com](https://www.biorender.com). **b**, Precision and recall scores for predicting various cellular injuries, comparing the not-shuffled model (solid lines) with the shuffled model (dashed lines) across distinct injury types and data splits, with blue indicating the training set and red indicating the test set. The F1 scores for each injury represent

only the testing and training datasets with the not-shuffled model. **c**, Confusion matrices assessing the model's predictive performance across training, testing and holdout data. The red color gradient represents the ratio of the count over the predicted class label, with the ratios in each row summing to 1. **d**, Cytoskeletal injury probability distribution generated by the shuffled (red) and not-shuffled (blue) models (updated model trained with overlapping CPJUMP1 features). The three groups represent the ground-truth wells (top), wells predicted to have cytoskeletal injury (middle) and wells not predicted to have cytoskeletal injury (bottom).

'miscellaneous' and 'kinase'), which further demonstrates differences in generalizability across injury types (Extended Data Fig. 4).

To assess across-dataset generalizability, we retrained our model using only features that overlapped with the CPJUMP1 dataset, which comprises 19,498 well-level profiles from cell populations treated with 781 genetic and chemical perturbations (see Methods for details)¹⁶. CPJUMP1 was also previously processed using Pycytominer. We

identified 24 CPJUMP1 wells that overlapped with cytoskeletal injury compounds in the cell injury training dataset. Our model accurately predicted 22 out of these 24 CPJUMP1 ground-truth wells (Fig. 2d). The model predicted the two misclassified samples as nonspecific reactive and HSP90 injury, likely because these samples also showed substantial additional changes indicative of these injuries (Supplementary Table 4). Notably, the model predicted many CPJUMP1 compounds

without labels to cause nuisance injuries (cytoskeletal and others), which supports this approach flagging compounds in future drug screening applications (Extended Data Fig. 5 and Supplementary Table 5). We present injury type predictions for all CPJUMP1 chemical and genetic perturbations in Supplementary Table 6.

Nevertheless, Pycytominer has certain limitations. Being written in Python may exclude users proficient in other programming languages and necessitates integrating their analytical pipelines into Python. To address this, our future roadmap includes more containerization and the addition of command line interface options, which will broaden access and offer multilingual support. Second, there may be more optimal image-based profiling methods not yet discovered. In anticipation of these future developments, we have designed the Pycytominer API and testing framework with modularity in mind. This approach allows for the easy incorporation of new methods as they surpass the current state of the art. Third, it is important to note that Pycytominer focuses on a specific segment of the entire image analysis pipeline, beginning with image analysis outputs and ending with fully-processed single-cell and bulk image-based profiles. Consequently, users are required to be proficient in other software for preliminary processing steps like quality control, segmentation and feature extraction as well as downstream tools for analysis and visualization. This decision to concentrate on core image-based profiling functionality simplifies software maintenance and fosters direct innovation in this area.

Looking to the future, Pycytominer is poised to play an essential role as an integral tool for image-based profiling. With a steadfast commitment and a growing community consistently contributing new and optimized functionality, Pycytominer offers a reliable and standardized toolkit that empowers researchers to unveil new insights in multiple fields from drug discovery to fundamental cell biology research.

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at <https://doi.org/10.1038/s41592-025-02611-8>.

References

- Way, G. P., Sailem, H., Shave, S., Kaspruwicz, R. & Carragher, N. O. Evolution and impact of high content imaging. *SLAS Discov.* <https://doi.org/10.1016/j.slasd.2023.08.009> (2023).
- Schindelin, J. et al. Fiji: an open-source platform for biological-image analysis. *Nat. Methods* **9**, 676–682 (2012).
- Stirling, D. R. et al. CellProfiler 4: improvements in speed, utility and usability. *BMC Bioinform.* **22**, 433 (2021).
- Scheeder, C., Heigwer, F. & Boutros, M. Machine learning and image-based profiling in drug discovery. *Curr. Opin. Syst. Biol.* **10**, 43–52 (2018).
- Caicedo, J. C. et al. Data-analysis strategies for image-based cell profiling. *Nat. Methods* **14**, 849–863 (2017).
- Krentzel, D., Shorte, S. L. & Zimmer, C. Deep learning in image-based phenotypic drug discovery. *Trends Cell Biol.* **33**, 538–554 (2023).
- Vincent, F. et al. Phenotypic drug discovery: recent successes, lessons learned and new directions. *Nat. Rev. Drug Discov.* **21**, 899–914 (2022).
- Schorpp, K. et al. CellDeathPred: a deep learning framework for ferroptosis and apoptosis prediction based on cell painting. *Cell Death Discov.* **9**, 277 (2023).
- Lippincott, M. J. et al. A morphology and secretome map of pyroptosis. Preprint at *bioRxiv* <https://doi.org/10.1101/2024.04.26.591386> (2024).
- Way, G. P. et al. Morphology and gene expression profiling provide complementary information for mapping cell state. *Cell Syst.* **13**, 911–923.e9 (2022).
- Natekar, P., Wang, Z., Arora, M., Hakozaki, H. & Schöneberg, J. Self-supervised deep learning uncovers the semantic landscape of drug-induced latent mitochondrial phenotypes. Preprint at *bioRxiv* <https://doi.org/10.1101/2023.09.13.557636> (2023).
- van Rossum, G. *Python Reference Manual* (Python, 1995); <https://docs.python.org/3/reference/index.html>
- The Pandas Development Team. Pandas-Dev/pandas: Pandas. *Zenodo* <https://doi.org/10.5281/ZENODO.3509134> (2023).
- Ramm, M. & Bayer, M. *Sqlalchemy* (Prentice Hall PTR, 2008).
- Arevalo, J. et al. Evaluating batch correction methods for image-based cell profiling. *Nat. Commun.* **15**, 6516 (2024).
- Chandrasekaran, S. N. et al. Three million images and morphological profiles of cells treated with matched chemical and genetic perturbations. *Nat. Meth.* **21**, 1114–1121 (2024).
- Wolff, C. et al. Morphological profiling dataset of EU-OPENSOURCE bioactive compounds over multiple imaging sites and cell lines. Preprint at *bioRxiv* <https://doi.org/10.1101/2024.08.27.609964> (2024).
- Weisbart, E. et al. Cell Painting Gallery. *GitHub* <https://github.com/broadinstitute/cellpainting-gallery> (2023).
- Dahlin, J. L. et al. Reference compounds for characterizing cellular injury in high-content cellular morphology assays. *Nat. Commun.* **14**, 1364 (2023).

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

© The Author(s), under exclusive licence to Springer Nature America, Inc. 2025

Methods

Pycytominer software practices

We rigorously apply open-source best practices during Pycytominer's development in four main categories: implementation, testing, release and community (Extended Data Fig. 1). (1) Implementation. Pycytominer eases the process of contributing code by providing development container specifications usable in VSCode or GitHub Codespaces that contain the full set of software dependencies needed to develop and test the codebase. When changes are ready, contributors submit pull requests, which must be reviewed to ensure adherence to best practices such as modularization, code styling and documentation. (2) Testing. Pycytominer's comprehensive testing suite, including unit tests and code coverage analysis, serves as a crucial step to ensure the correctness and functionality of the software implementation. Testing every new change against the full test suite reduces the introduction of software bugs and ensures consistent behavior across versions. (3) Release. Pycytominer follows semantic versioning and maintains a changelog to ensure users are kept informed of new features and important changes. Releases are made available directly on GitHub and are also packaged for use within Python's two major package repositories PyPI and conda. In addition, Pycytominer supports operating environment containerization (facilitated by Docker²⁰), encapsulating dependencies to enhance reproducibility. (4) Community. Pycytominer cultivates our open-source community by welcoming new contributors with clear contributing instructions and guidelines and a code of conduct to ensure that professional standards are kept. These community efforts are essential for good collaboration, maintaining quality and ensuring project sustainability. Embracing the full set of these best software practices fosters a collaborative environment that facilitates continuous improvements, encourages reproducibility, welcomes newcomers and contributes to package usability for developers and users alike.

Reprocessing the cell injury dataset

The cell injury profiles used in this study (as training data) were initially generated by Dahlin et al.¹⁹ and consisted of image-based profiles along with associated metadata. These profiles were derived from U-2OS cells treated with 218 cytotoxic compounds. The dataset was made publicly available on the Image Data Resource (accession number 0133). We downloaded only the normalized aggregated profiles that have been normalized to their negative controls.

The cell injury dataset contains 23,111 total wells, but we selected only control wells and wells with ground-truth cell injury annotations (16,701 selected wells). We labeled wells treated with dimethylsulfoxide as 'Control', while the treated wells were labeled according to their specific injuries, which included 15 types: control, cytoskeletal, Hsp90, kinase, genotoxin, miscellaneous, redox, HDAC, mTOR, proteasome, saponin, mitochondria, ferroptosis, tannin and nonspecific reactive (see Supplementary Table 3 for a complete count breakdown).

We used Pycytominer's feature selection function to select informative morphological features to include during model training. We applied the variance threshold operation to filter out features with low variance. Additionally, we incorporated an 'na_cutoff' parameter to drop features if the proportion of missing values (NaN) exceeded 5%. Last, we applied a correlation threshold of 0.9, removing features that exhibited high correlation with other features to ensure a more diverse and representative feature set. If two features had a Pearson correlation higher than 0.9, we removed the one feature that had the highest combined correlation with all other features. This process removed 20 morphology features, which left us with 352 features for model training.

Machine-learning application: generating data splits

We generated three holdout datasets in a specific sequence: plate, treatment and well holdouts. First, we created plate holdouts by randomly selecting ten plates from the cell injury dataset. For treatment holdouts, we identified specific cell injuries with ten or more unique treatments

(we excluded injuries from the holdout set if they had fewer than ten unique treatments). From the remaining categories, we selected the one treatment with the fewest wells as our holdout (see Extended Data Figure 4). This process enabled us to retain a high diversity of different treatments in the training set, while enabling us to evaluate performance on never-before-seen treatments. We generated well holdouts by selecting all nonholdout plates and randomly choosing 15 wells (5 controls and 10 wells with each cell injury) from each plate. All holdout sets contained independent samples (we did not include a sample in more than one holdout set). The remaining data (13,502 wells) constituted our training dataset, which we further split into 80% (10,801 wells) for training and 20% (2,701 wells) for testing, maintaining the same class-label proportions to address dataset imbalance.

Machine-learning application: model training

We trained and hyperparameterized our multiclass logistic regression models using scikit-learn's RandomizedSearchCV. To address label imbalance, we configured our logistic regression model to automatically adjust class weights based on their frequencies. This adjustment helps to mitigate the impact of imbalanced labels during model training. For the hyperparameter tuning, we defined a parameter grid for RandomizedSearchCV. This grid included exploration of three different penalties: Lasso (L1), Ridge (L2) and Elastic Net (combination of L1 and L2). We varied the regularization strength across a range of values (0.0001, 0.01, 0.1, 1, 10 and 100) to assess its impact on model performance. Additionally, we experimented with different tolerance values (1×10^{-6} and 1×10^{-3}) to determine the threshold for stopping the hyperparameter search process.

To explore the Elastic Net regularization further, we searched different ratios (0.1, 0.3, 0.5, 0.7 and 0.9) between L1 and L2 penalties. Finally, we evaluated various solvers (newton-cg, lbfgs, liblinear, sag and saga) to optimize the logistic regression model parameters. Each solver offers distinct optimization techniques, allowing us to assess their effectiveness in our model's context. Results of the hyperparameter tuning and cross validation can be found in Supplementary Table 7. We selected the model parameters that optimized this cross-validation procedure. In the supplementary tables, we call this the 'fs_model'.

We applied this training approach to two models: the standard multiclass logistic regression model (not-shuffled model type) and a second multiclass logistic regression model trained on a shuffled dataset (shuffled model type), where we randomized feature values. The shuffled model serves as a baseline to evaluate the performance of our standard model. By comparing the performance of the standard model against the shuffled model, we are able to determine whether the results from the standard model reflected meaningful patterns in the data or are simply due to random chance.

CPJUMP1 application

We applied the trained cell injury model to the CPJUMP1 dataset (cpg0000-jump-pilot), which we accessed via the Cell Painting Gallery website, specifically from the path 'cpg0000-jump-pilot/source_4/workspace/profiles/2020_11_04_CPJUMP1'. Because both the CPJUMP1 and the cell injury dataset were processed using different image analysis and image-based profiling pipelines, they each had a different feature space. We identified 221 morphology features in common (total features before feature selection).

We then subset the normalized cell injury dataset to this common set of features and applied the Pycytominer feature selection procedure, as previously described, with one exception: we excluded the correlation threshold step to retain more features. In this case, Pycytominer did not drop any features. We then applied the machine-learning procedure as described in the 'Machine-learning application: model training' section to the cell injury dataset with the 221 common feature space. In the supplementary tables, we call this the 'aligned_model'.

We used International Chemical Identifiers (InChIKeys) to identify chemical perturbations common to both datasets. We identified 24 CPJUMP1 wells that contained chemical perturbations labeled as causing cytoskeletal injuries in the cell injury study. We set these wells as our ground truth when conducting our evaluation.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

We accessed the images and resulting profiles that we analyzed in Fig. 2 from IDR0133 (ref. 19). We make all intermediate data for this analysis of the IDR0133 dataset available at <https://github.com/WayScience/predicting-cell-injury-compounds/tree/v2.1>.

Code availability

- Pycytominer is an open-source project and its source code can be viewed and downloaded from <https://github.com/cytomining/pycytominer>.

- Pycytominer's installation and usage documentation is available at <https://Pycytominer.readthedocs.io/>.

- A tutorial on how to conduct single-cell image-based profiling is available at https://Pycytominer.readthedocs.io/en/latest/walkthroughs/single_cell_usage.html.

- The repository containing the code used to conduct analysis and generate results is available at <https://github.com/WayScience/predicting-cell-injury-compounds/tree/v2.1.1>.

- The repository containing profiling recipe written in Pycytominer is available at <https://github.com/cytomining/profiling-recipe>.

- The image-based profiling handbook is available at <https://cytominer.github.io/profiling-handbook11/22/2023>.

References

20. Merkel, D. Docker: lightweight Linux containers for consistent development and deployment. *Linux J.* <https://doi.org/10.5555/2600239.2600241> (2014).

Acknowledgements

We thank M. Lippincott and J. Tomkinson for performing code review. This software was created with the support of National Institutes of Health (NIH) grant R35 GM122547 to A.E.C. and institutional startup funding to G.P.W. B.A.C. was supported by grant 2020-225720 (<https://doi.org/10.37921/977328pjbvca>) from the Chan Zuckerberg Initiative DAF, an advised fund of Silicon Valley Community Foundation (funder <https://doi.org/10.13039/100014989>). Research reported in this

publication was supported by the National Library of Medicine of the NIH under award number T15LM009451 to E.S. The authors gratefully acknowledge funding from the Massachusetts Life Sciences Center Bits to Bytes Capital Call program (to A.E.C.) and the Data Science Internship Program (supporting R.P. and A.A.), partners of the JUMP Cell Painting Consortium, Starr Cancer Consortium (112-0039 to A.E.C.), Schmidt Fellowship program of the Broad Institute (to J.C.C.) and the Human Frontier Science Program (RGY0081/2019 to S.S.).

Author contributions

Conceptualization: S.N.C., A.G., J.C.C., A.E.C., B.A.C., S.S. and G.P.W. Data curation: E.S., B.A.C., S.S. and G.P.W. Formal analysis: E.S., S.N.C., B.A.C., S.S. and G.P.W. Funding acquisition: E.S., J.C.C., A.E.C., B.A.C., S.S. and G.P.W. Investigation: E.S., S.N.C., B.A.C., S.S. and G.P.W. Methodology: E.S., S.N.C., D.B., K.I.B., J.T., R.K., M.B., S.J.F., R.P., J.A., H.T., C.T., T.B., E.W., C.B., A.A.K., R.S., S.J.T., N.J., A.A., H.S., A.G., J.C.C., A.E.C., B.A.C., S.S. and G.P.W. Project administration: A.E.C., S.S. and G.P.W. Resources: A.E.C., S.S. and G.P.W. Software: E.S., S.N.C., D.B., K.I.B., J.T., R.K., M.B., S.J.F., R.P., J.A., H.T., E.W., C.B., A.A.K., R.S., S.J.T., N.J., A.A., B.A.C., S.S. and G.P.W. Supervision: A.E.C., B.A.C., S.S. and G.P.W. Validation: E.S., S.N.C., B.A.C., S.S. and G.P.W. Visualization: E.S., V.R., A.E.C., S.S. and G.P.W. Writing - original draft: E.S. and G.P.W. Writing - review and editing: E.S., S.N.C., D.B., K.I.B., J.T., R.K., M.B., S.J.F., R.P., J.A., H.T., V.R., C.T., T.B., E.W., C.B., A.A.K., R.S., S.J.T., N.J., A.A., H.S., A.G., J.C.C., A.E.C., B.A.C., S.S. and G.P.W.

Competing interests

The authors declare no competing interests.

Additional information

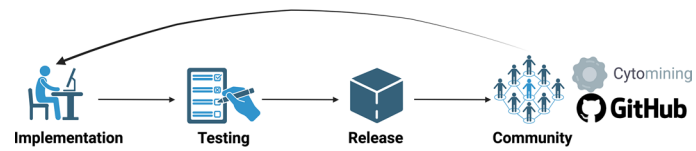
Extended data is available for this paper at <https://doi.org/10.1038/s41592-025-02611-8>.

Supplementary information The online version contains supplementary material available at <https://doi.org/10.1038/s41592-025-02611-8>.

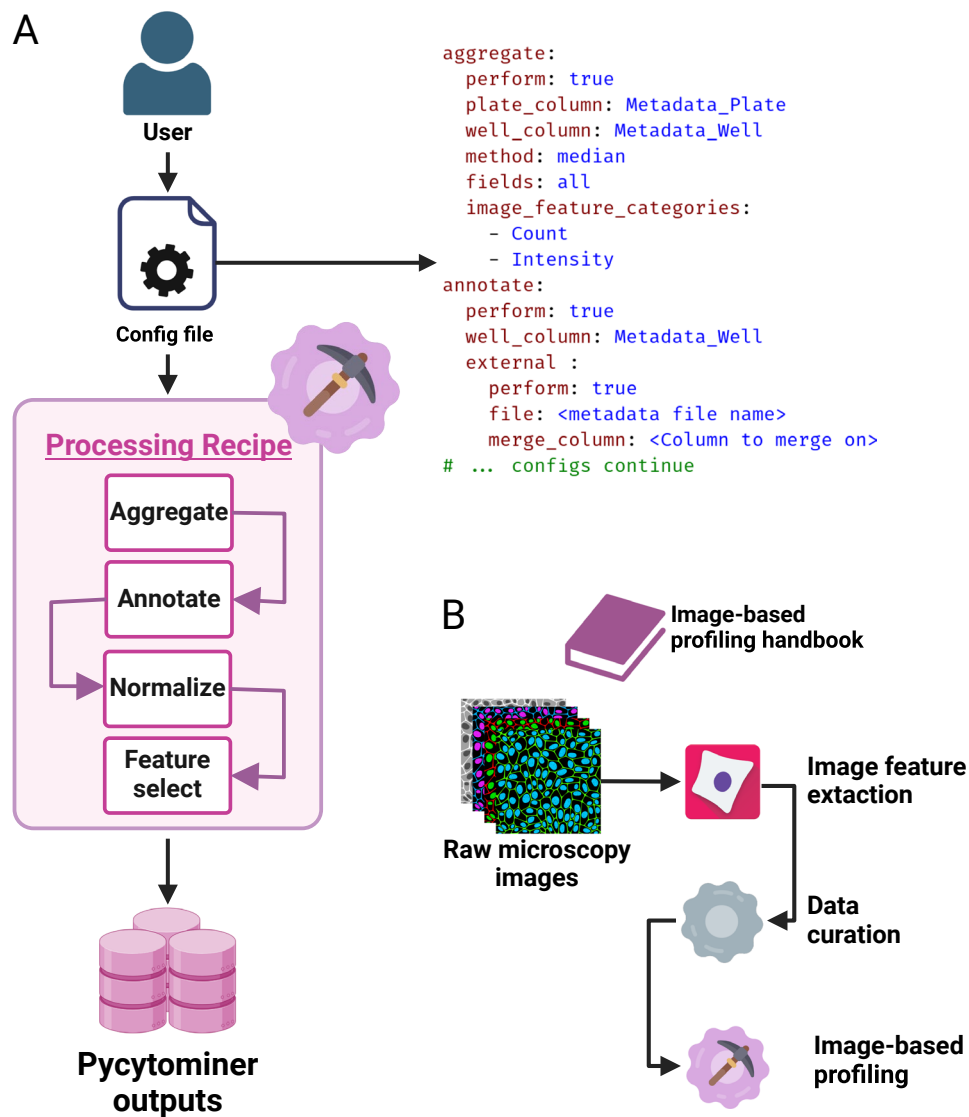
Correspondence and requests for materials should be addressed to Shantanu Singh or Gregory P. Way.

Peer review information *Nature Methods* thanks Ana Stojiljkovic and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Primary Handling Editor: Rita Strack, in collaboration with the *Nature Methods* team.

Reprints and permissions information is available at www.nature.com/reprints.

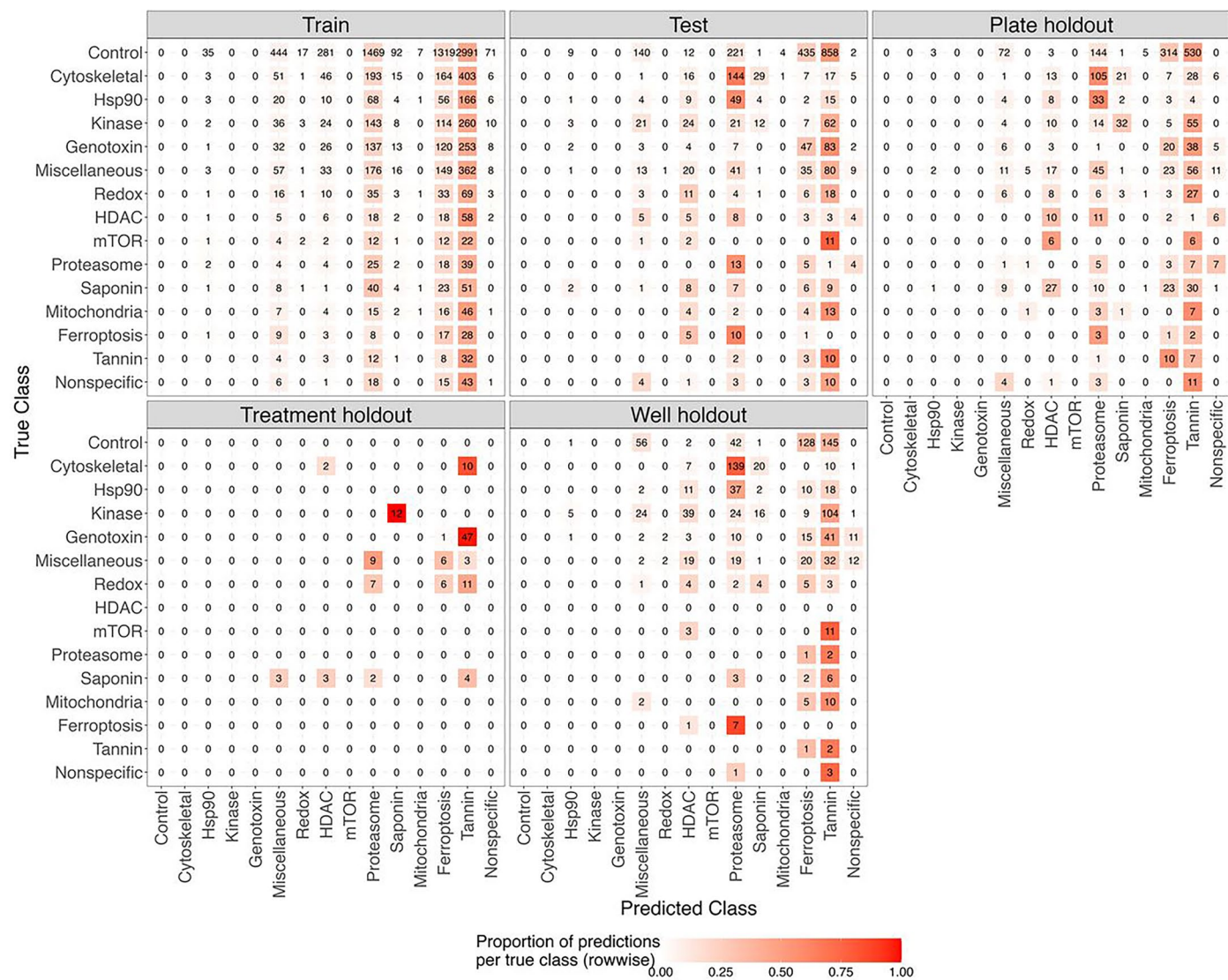


Extended Data Fig. 1 | Overview of Pycytominer’s open-source development practices. Pycytominer’s open-source development starts with implementing new features, followed by rigorous testing to ensure reliability. After a formal release, the community engages with the tool, providing feedback and suggestions. Created in <https://BioRender.com>.

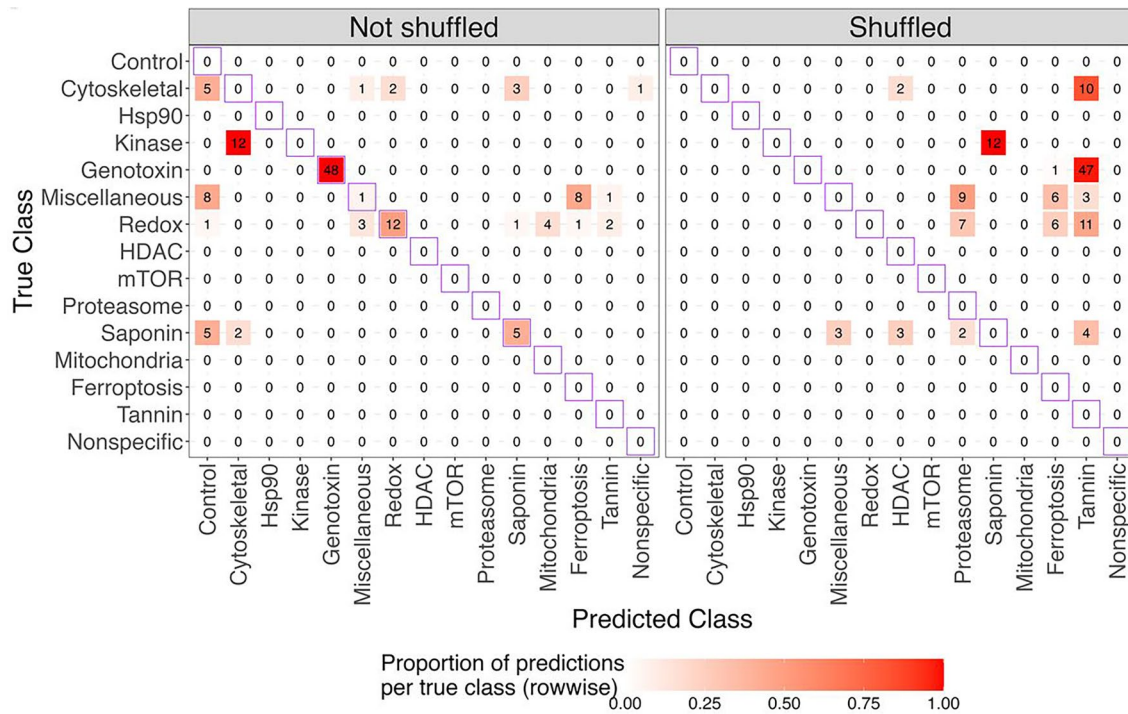


Extended Data Fig. 2 | Pycytominer recipe and handbook for image-based profiling. (a) Users can configure a profiling recipe to customize Pycytominer implementation of image-based profiling steps. Users interact with the profiling recipe through a configuration file provided in yaml format, to parameterize each function within the Pycytominer workflow. The profiling recipe can be

found at: <https://github.com/cytomining/profiling-recipe>. (b) We have also written an image-based profiling handbook available at <https://cytomining.github.io/profiling-handbook/>, which documents all steps in a full image-based profiling workflow. Created in <https://BioRender.com>.



Extended Data Fig. 3 | Shuffled multi-class logistic regression model predictive performance. The confusion matrix demonstrates the model’s ability to predict cell injuries across various data splits and holdouts. The red color gradient represents the ratio of the count over the true class label, where the ratios in each row summing to 1.



Extended Data Fig. 4 | Comparison of multi-class logistic regression model predictive performance for treatment holdouts. The confusion matrix evaluates the predictive performance of the Not Shuffled model (left) versus the

Shuffled model (right) using only the treatment holdout dataset. The red color gradient corresponds to the count over the predicted class label, with the ratios in each row summing to 1.

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our [Editorial Policies](#) and the [Editorial Policy Checklist](#).

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a Confirmed

- The exact sample size (n) for each experimental group/condition, given as a discrete number and unit of measurement
- A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
- The statistical test(s) used AND whether they are one- or two-sided
Only common tests should be described solely by name; describe more complex techniques in the Methods section.
- A description of all covariates tested
- A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
- A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
- For null hypothesis testing, the test statistic (e.g. F , t , r) with confidence intervals, effect sizes, degrees of freedom and P value noted
Give P values as exact values whenever suitable.
- For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
- For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
- Estimates of effect sizes (e.g. Cohen's d , Pearson's r), indicating how they were calculated

Our web collection on [statistics for biologists](#) contains articles on many of the points above.

Software and code

Policy information about [availability of computer code](#)

Data collection

NA

Data analysis

To process and analyze the cell injury dataset, we used Python version 3.11, pycytominer version 1.1, scikit-learn version 1.5.1, and pandas version 2.1.4. This work also presents our custom, open-source software, called pycytominer.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio [guidelines for submitting code & software](#) for further information.

Data

Policy information about [availability of data](#)

All manuscripts must include a [data availability statement](#). This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our [policy](#)

We accessed the images and resulting profiles that we analyzed in Figure 2 from IDR013320. We make all intermediate data for this analysis of the IDR0133 dataset available at <https://github.com/WayScience/predicting-cell-injury-compounds/tree/v2.1>.

Human research participants

Policy information about [studies involving human research participants and Sex and Gender in Research](#).

Reporting on sex and gender	NA
Population characteristics	NA
Recruitment	NA
Ethics oversight	NA

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

Life sciences Behavioural & social sciences Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	We are using publicly-available data and cannot modify sample size.
Data exclusions	NA
Replication	Our GitHub repository can be fully reproduced, and contains all instructions to do so.
Randomization	NA
Blinding	NA

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
<input checked="" type="checkbox"/>	<input type="checkbox"/> Antibodies
<input checked="" type="checkbox"/>	<input type="checkbox"/> Eukaryotic cell lines
<input checked="" type="checkbox"/>	<input type="checkbox"/> Palaeontology and archaeology
<input checked="" type="checkbox"/>	<input type="checkbox"/> Animals and other organisms
<input checked="" type="checkbox"/>	<input type="checkbox"/> Clinical data
<input checked="" type="checkbox"/>	<input type="checkbox"/> Dual use research of concern

Methods

n/a	Involved in the study
<input checked="" type="checkbox"/>	<input type="checkbox"/> ChIP-seq
<input checked="" type="checkbox"/>	<input type="checkbox"/> Flow cytometry
<input checked="" type="checkbox"/>	<input type="checkbox"/> MRI-based neuroimaging