CytoTable: High Performance and Scalable Single-cell Morphology Feature Engineering

Dave Bunten[1], Erik Serrano[1], Jenna Tomkinson[1], Michael J. Lippincott,[1], Faisal Alquaddoomi[1], Vince Rubinetti[1], Gregory P. Way[1]

[1] Department of Biomedical Informatics, University of Colorado Anschutz

Department of Biomedical Informatics
SCHOOL OF MEDICINE
UNIVERSITY OF COLORADO **ANSCHUTZ MEDICAL CAMPUS**

Introduction

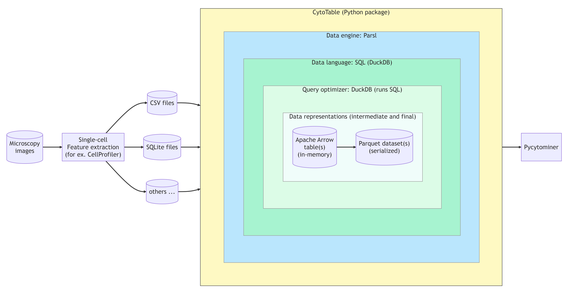

Figure 1. Diagram showing high-dimensional single-cell morphology data flow with relationship to CytoTable modular data stack components.

High-dimensional single-cell morphology data, such as those extracted from CellProfiler[1] from large-scale microscopy drug screening applications help prioritize effective treatments for patients suffering from a variety of diseases and enable the discovery of new biological mechanisms. We are solving significant scalability and replicability challenges with these data by implementing novel and effective capabilities as a modular, portable, and cross-language single-cell data stack[2]: (a) language frontend: SQL (DuckDB[3]), (b) intermediate representation: Apache Arrow[4] and Apache Parquet[5], (c) query optimizer: DuckDB[3], (d) execution engine: Parsl[6] with Pythonic MapReduce design patterns[7], (e) execution runtime, Python package (PyPI, source)(Figure 1).

Morphology Feature Data Scalability

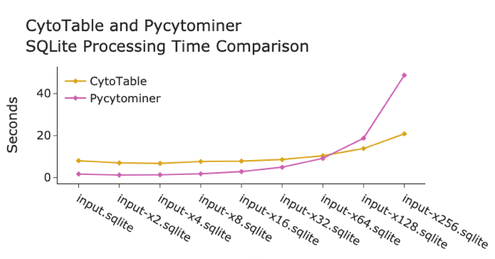

Figure 2. Plot showing processing time duration for CytoTable and Pycytominer for various datasets.

CytoTable builds upon the shoulders of Pycytominer, helping to streamline the `SingleCells.merge_single_cells(...)` method. We improve the processsing completion time for large amounts of data by leveraging composable data stack elements.

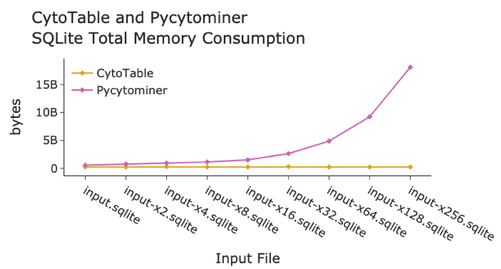

Figure 3. Plot showing total memory consumption for CytoTable and Pycytominer for various datasets.

CytoTable also decreases memory consumption when compared to Pycytominer's `SingleCells.merge_single_cells(...)` method. Decreased memory footprints mean that CytoTable may be used on anything from local laptop to high-performance computing (HPC) environments.

Empowering the Cytomining Ecosystem

Orchestration: CytoSnake

something

References

1. Carpenter, A. E., Jones, T. R., Lamprecht, M. R., Clarke, C., Kang, I., Friman, O., Guertin, D. A., Chang, J., Lindquist, R. A., Moffat, J., Golland, P., & Sabatini, D. M. (2006). CellProfiler: Image analysis software for identifying and quantifying cell phenotypes. Genome Biology, 7(10), R100. https://doi.org/10.1186/gb-2006-7-10-r100
2. Pedreira, P., Erling, O., Karanasos, K., Schneider, S., McKinney, W., Valluri, S. R., Zait, M., & Nadeau, J. (2023). The Composable Data Management System Manifesto. Proceedings of the VLDB Endowment, 16(10), 2679–2685. https://doi.org/10.14778/3603581.3603604
3. Raasveldt, M., & Mühleisen, H. (2019). DuckDB: An Embeddable Analytical Database. Proceedings of the 2019 International Conference on Management of Data, 1981–1984. https://doi.org/10.1145/3299869.3320212
4. Apache Arrow. (n.d.). [Computer software]. Apache Software Foundation. https://arrow.apache.org/docs/
5. Apache Parquet. (n.d.). [Computer software]. Apache Software Foundation. https://parquet.apache.org/docs/
6. Babuji, Y., Woodard, A., Li, Z., Katz, D. S., Clifford, B., Kumar, R., Lacinski, L., Chard, R., Wozniak, J. M., Foster, I., Wilde, M., & Chard, K. (2019). Parsl: Pervasive Parallel Programming in Python. Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing, 25–36. https://doi.org/10.1145/3307681.3325400
7. Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1), 107–113. https://doi.org/10.1145/1327452.1327492

Acknowledgements

Special thanks goes to the following for their help in contributing to CytoTable design and development or related work.

- The Way Lab: Cameron Mattson
- The Broad Institute: Shantanu Singh, Beth Cimini, Sam Chen