Powering single-cell analyses in the browser with WebAssembly

Aaron Tin Long Lun[1*] **and Jayaram Kancherla**[1*]

1 Genentech Inc., South San Francisco, United States of America * These authors contributed equally.

DOI: 10.21105/joss.05603

Software

- Review
- Repository
- Archive

Editor: Aoife Hughes

Reviewers:

- @yongrenjie
- @llewelld

Submitted: 15 May 2023
Published: 05 September 2023

Summary

We present kana, a web application for interactive single-cell 'omics data analysis in the browser. Like, literally, in the browser: kana leverages web technologies such as WebAssembly to efficiently perform the relevant computations on the user's machine, avoiding the need to provision and maintain a backend service. The application provides a streamlined one-click workflow for the main steps in a typical single-cell analysis, starting from a count matrix and finishing with marker detection. Results are presented in an intuitive web interface for further exploration and iterative analysis.

Statement of need

Single-cell 'omics is routinely used to identify cell subpopulations with distinct molecular phenotypes in heterogeneous biological samples (Stegle et al., 2015). For example, in single cell RNA sequencing (scRNA-seq), cells are clustered based on their transcriptional profiles and each cluster is characterized based on differential expression of marker genes. Changes in expression or cellular abundance for each cluster can then provide some insight into the biological processes associated with the corresponding cell type or state. This analysis is often exploratory in nature as the properties of an interesting cell subpopulation are difficult to define *a priori*. As a result, each analysis involves several iterations of computation, visualization and interpretation by biologists who may not be familiar with programming frameworks for bioinformatics.

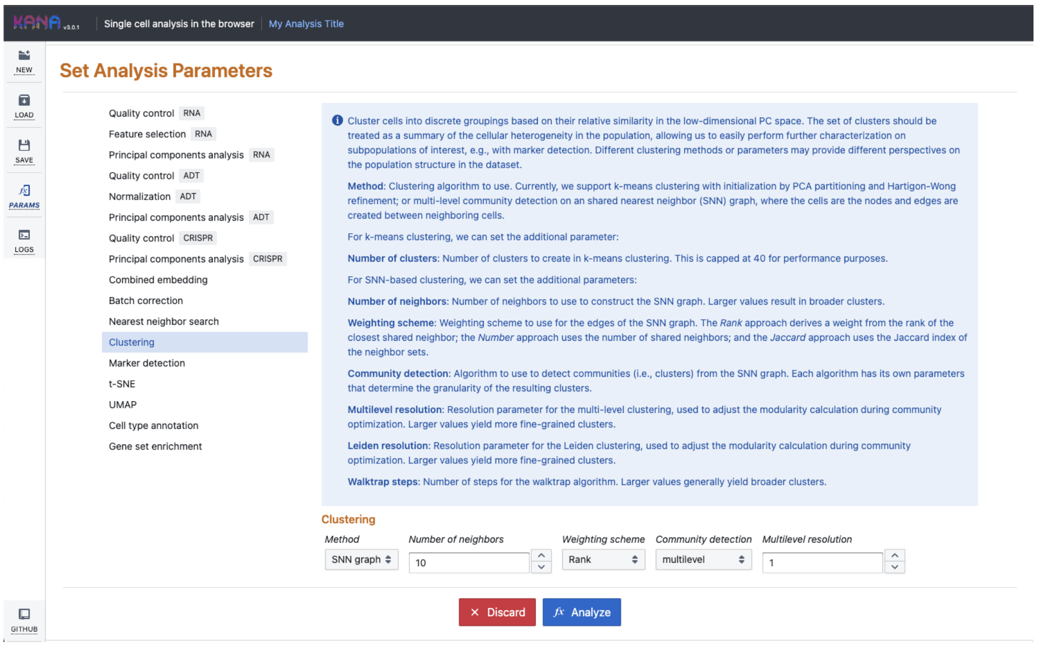
Web applications offer an ideal environment for single-cell analyses, providing a user-friendly interface to the analysis workflow without any installation of additional software (other than a browser). This strategy has been used by tools such as Cellxgene (Megill et al., 2021), Cirrocumulus (Gould et al., 2021), and a variety of R/Shiny applications (Chang et al., 2021) for processing single-cell data. The vast majority of these existing web applications use a traditional server-based architecture, where data is sent to a backend server to compute results that are returned to the client - i.e., the user's machine - for inspection. This obviously requires the deployment of a backend server, which has non-negligible cost when scaled to a large number of users. It also requires data transfer from the client to the server, which introduces latency as well as invoking concerns over data ownership and privacy.

An alternative paradigm is to perform all the compute on the client device. This "client-side compute" circumvents all of the aforementioned issues with a backend server as data remains local to the user's machine and is analyzed wholly within the browser. Some existing web applications have employed this approach for bioinformatics data analysis (Fan et al., 2017; Gómez et al., 2013; Schmid-Burgk & Hornung, 2015), but these represent a minority in the application ecosystem. This is not surprising given the paucity of efficient, browser-compatible implementations of various algorithms required for such analyses. However, new web technologies such as WebAssembly (Haas et al., 2017) have greatly enhanced browsers' capabilities for intensive computation. If we could generate a WebAssembly (Wasm) binary containing single-cell analysis functionality, we could feasibly repurpose the browser as a self-contained interactive data analysis tool for single-cell 'omics.

To this end, we present kana, a web application for analyzing single-cell 'omics data inside the browser (https://www.kanaverse.org/kana). kana provides a streamlined one-click workflow for the main steps in a typical single-cell analysis (Amezquita et al., 2020), starting from a count matrix (or multiple such matrices, for datasets with multiple modalities and/or samples) and finishing with marker detection. Users can interactively explore the low-dimensional embeddings, clusterings and marker genes in an intuitive graphical interface that encourages iterative re-analysis. Once finished, users can save their analysis and results for later examination or sharing with collaborators. By using technologies like WebAssembly and web workers, we achieve high-performance client-side compute for datasets containing hundreds of thousands of cells.

Usage

Given a single-cell dataset - typically for gene expression, but possibly with protein and/or CRISPR counts - kana executes a routine analysis based on A. T. L. Lun et al. (2020). Users can supply data in several formats, such as Matrix Market files produced by the Cellranger pipeline; HDF5 files, using either the 10X HDF5 feature barcode matrix format or as H5AD files; SummarizedExperiment or SingleCellExperiment objects (Amezquita et al., 2020) saved to RDS files; or datasets from public repositories like Bioconductor's ExperimentHub (Morgan & Shepherd, n.d.). The entire analysis can then be executed with a single click, though users can easily customize key parameters if desired (Figure 1).



Set Analysis Parameters

Quality control RNA
Feature selection RNA
Principal components analysis RNA
Quality control ADT
Normalization ADT
Principal components analysis ADT
Quality control CRISPR
Principal components analysis CRISPR
Combined embedding
Batch correction
Nearest neighbor search
Clustering
Marker detection
t-SNE
UMAP
Cell type annotation
Gene set enrichment

Cluster cells into discrete groupings based on their relative similarity in the low-dimensional PC space. The set of clusters should be treated as a summary of the cellular heterogeneity in the population, allowing us to easily perform further characterization on subpopulations of interest, e.g., with marker detection. Different clustering methods or parameters may provide different perspectives on the population structure in the dataset.

Method: Clustering algorithm to use. Currently, we support k-means clustering with initialization by PCA partitioning and Hartigon-Wong refinement; or multi-level community detection on an shared nearest neighbor (SNN) graph, where the cells are the nodes and edges are created between neighboring cells.

For k-means clustering, we can set the additional parameter:

Number of clusters: Number of clusters to create in k-means clustering. This is capped at 40 for performance purposes.

For SNN-based clustering, we can set the additional parameters:

Number of neighbors: Number of neighbors to use to construct the SNN graph. Larger values result in broader clusters.

Weighting scheme: Weighting scheme to use for the edges of the SNN graph. The *Rank* approach derives a weight from the rank of the closest shared neighbor; the *Number* approach uses the number of shared neighbors; and the *Jaccard* approach uses the Jaccard index of the neighbor sets.

Community detection: Algorithm to use to detect communities (i.e., clusters) from the SNN graph. Each algorithm has its own parameters that determine the granularity of the resulting clusters.

Multilevel resolution: Resolution parameter for the multi-level clustering, used to adjust the modularity calculation during community optimization. Larger values yield more fine-grained clusters.

Leiden resolution: Resolution parameter for the Leiden clustering, used to adjust the modularity calculation during community optimization. Larger values yield more fine-grained clusters.

Walktrap steps: Number of steps for the walktrap algorithm. Larger values generally yield broader clusters.

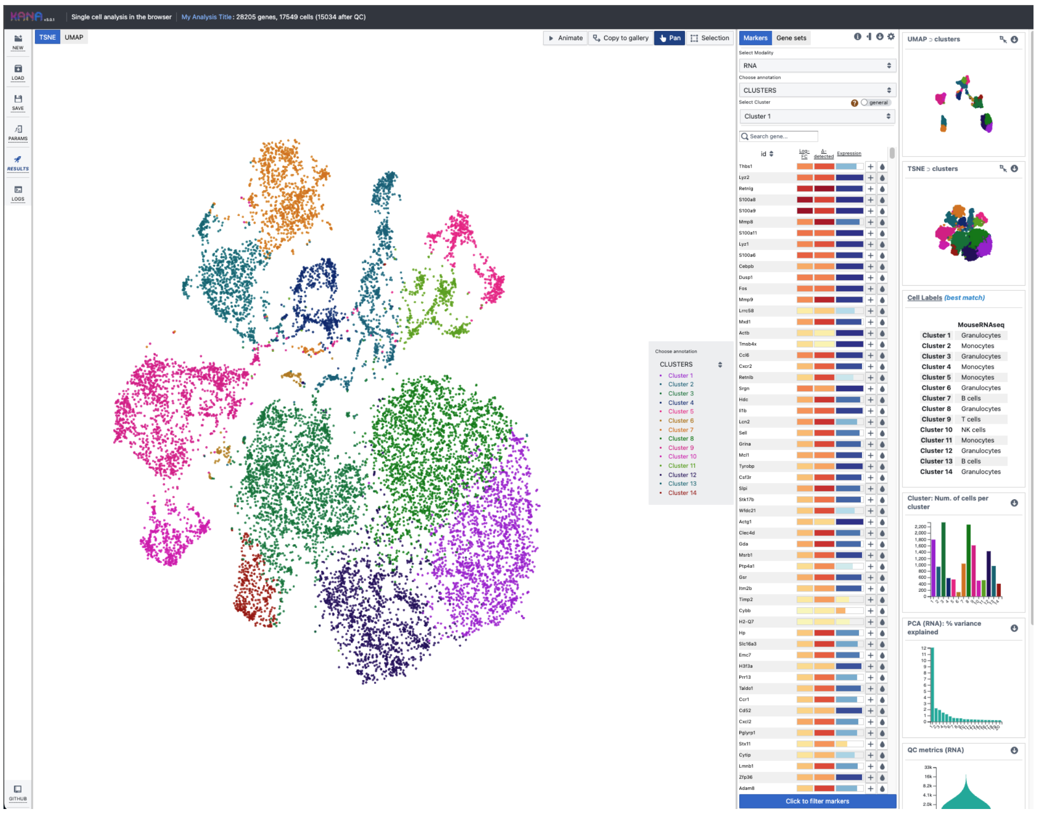
Clustering

Method	*Number of neighbors*	*Weighting scheme*	*Community detection*	*Multilevel resolution*
SNN graph	10	Rank	multilevel	1

× Discard   Analyze

Figure 1: Screenshot showing the analysis configuration panel in the kana application. Clicking "Analyze" will perform the entire analysis.

Once each step of the analysis is complete, kana visualizes its results in a multi-panel layout (Figure 2). One panel contains a scatter plot for the low-dimensional embeddings, where each cell is a point that is colored by cluster identity or gene expression. Another panel contains a table of marker statistics for a selected cluster, where potential marker genes are ranked and filtered according to the magnitude of upregulation over other clusters. Scientists can use this interface to examine the cellular heterogeneity in the dataset and to determine which biological processes are most active in each subpopulation. We also provide a gallery to visualize miscellaneous details such as the distribution of QC metrics.



Select Modality: RNA

Choose annotation: CLUSTERS

Select Cluster: Cluster 1 (general)

id
Thbs1
Lyz2
Retnlg
S100a8
S100a9
Mmp8
S100a11
Lyz1
S100a6
Cebpb
Dusp1
Fos
Mmp9
Lrrc58
Mxd1
Actb
Tmsb4x
Ccl6
Cxcr2
Retnlb
Srgn
Hdc
Il1b
Lcn2
Sell
Grina
Mcl1
Tyrobp
Csf3r
Slpi
Stk17b
Wfdc21
Actg1
Clec4d
Gda
Msrb1
Ptp4a1
Gsr
Itm2b
Timp2
Cybb
H2-Q7
Hp
Slc16a3
Emc7
H3f3a
Prr13
Taldo1
Ccr1
Cd52
Cxcl2
Pglyrp1
Stx11
Cytip
Lmnb1
Zfp36
Adam8

Cell Labels *(best match)*

	MouseRNAseq
Cluster 1	Granulocytes
Cluster 2	Monocytes
Cluster 3	Granulocytes
Cluster 4	Monocytes
Cluster 5	Monocytes
Cluster 6	Granulocytes
Cluster 7	B cells
Cluster 8	Granulocytes
Cluster 9	T cells
Cluster 10	NK cells
Cluster 11	Monocytes
Cluster 12	Granulocytes
Cluster 13	B cells
Cluster 14	Granulocytes

Figure 2: Screenshot showing the multi-panel layout for results in the kana application. The left panel is used for the low-dimensional embeddings, the middle panel contains the marker table for a selected cluster, and the right panel contains a gallery of miscellaneous plots.

Once the analysis is complete, users can export the analysis configuration and results for later inspection. The exported results can be quickly reloaded in a new browser session, allowing users or their collaborators to explore existing results without repeating the computation. Indeed, kana's "explore-only" mode can be used more generally for single-cell analysis results in other formats, e.g., to load RDS or H5AD files containing pre-computed clusterings and low-dimensional embeddings. Similarly, the exported configuration can be used to restore the analysis session for further parameter tuning and iteration.

From a user perspective, kana's interface is quite similiar to those of other single-cell web applications like Cellxgene. In fact, most single-cell user interfaces have very similar layouts and functionality due to the commonality of the analysis steps and the standardized nature of the results. kana's novelty comes from how the analysis is performed - unlike most other applications, we do not rely on a backend server, but perform the analysis directly on the user's machine. Importantly, this is achieved via the browser and does not require installation of additional software like, e.g., the 10X Genomics Loupe browser.

Implementation details

We convert the browser into a compute engine by compiling scientific libraries for bioinformatics data analysis to Wasm - see the biowasm project (Aboukhalil, 2019) for previous efforts in this direction. For kana, we created C++ implementations of the data representations or algorithms required for each analysis step:

- tatami (A. Lun, 2021a) provides an abstract interface to different matrix classes, based on ideas in the beachmat (A. T. L. Lun et al., 2018) and DelayedArray (Pagès et al., 2021) packages.
- CppWeightedLowess (A. T. L. Lun, 2021h) contains a C++ port of the weightedLowess function from the limma package (Ritchie et al., 2015), itself derived from R's LOWESS implementation (Cleveland, 1979).
- CppIrlba (A. T. L. Lun, 2021d) contains a C++ implementation of the IRLBA algorithm (James Baglama & Reichel, 2005), based on the C code in the irlba R package (Jim Baglama et al., 2019).
- CppKmeans (A. T. L. Lun, 2021e) implements the several algorithms for k-means clustering Su & Dy (2007), based on R's Fortran code.
- knncolle (A. T. L. Lun, 2021g) wraps several nearest neighbor detection algorithms (Bernhardsson, 2021; Yianilos, 1993) in a consistent interface, using the same design as the BiocNeighbors package (A. T. L. Lun, 2021c).
- libscran (A. T. L. Lun, 2021a) implements high-level methods for scRNA-seq data analysis based on code from the scran, scuttle and scater packages (Aaron T. L. Lun et al., 2016; McCarthy et al., 2017).
- qdtsne (A. T. L. Lun, 2021f) contains a C++ implementation of the Barnes-Hut t-SNE algorithm (Maaten, 2014), refactored and optimized from code in the Rtsne package (Krijthe, 2015).
- umappp (A. T. L. Lun, 2021b) contains a C++ implementation of the UMAP algorithm (McInnes et al., 2018), derived from code in the uwot R package (Melville, 2021).
- SinglePP (A. T. L. Lun, 2022c) contains a C++ implementation of the SingleR algorithm for cell type annotation (Aran et al., 2019).
- CppMnnCorrect (A. T. L. Lun, 2022a) implements the MNN method for batch correction (Haghverdi et al., 2018), loosely based on an existing method from the batchelor package (A. T. L. Lun & Haghverdi, 2018).
- rds2cpp (A. T. L. Lun, 2022b) provides a RDS file parser/writer as a standalone C++ library.

We compiled these libraries to a Wasm binary using the Emscripten toolchain (Zakai, 2011), using PThreads to enable parallelization via web workers. We then wrapped the binary in the scran.js library (A. Lun & Kancherla, 2021) to provide JavaScript bindings for use in web applications. kana itself was developed using React with extensive use of WebGL for efficient plotting. Testing indicates that kana is only 25-50% slower than an equivalent native executable (Table 1), indicating that Wasm's promise of near-native execution is feasible.

Table 1: Performance of kana for analyzing different scRNA-seq datasets, compared to a native executable created from the same C++ libraries. Measurements were taken on a laptop with an Intel Core i7-8850H CPU (2.60GHz, 6 cores) and 32 GB memory running Manjaro Linux. Runtimes are reported in seconds with the mean and standard error across 3 runs. See A. Lun & Kancherla (2023) for more details.

Dataset	Number of cells	kana runtime	Native runtime
Zeisel et al. (2015)	3005	7.00 ± 0.10	5.60 ± 0.05
Paul et al. (2015)	10368	17.59 ± 0.20	13.52 ± 0.38
Bach et al. (2017)	25806	54.96 ± 1.13	43.33 ± 0.39
Ernst et al. (2019)	68937	157.15 ± 7.39	114.67 ± 1.86
Bacher et al. (2020)	104417	228.02 ± 2.85	170.32 ± 1.34

Dataset	Number of cells	kana runtime	Native runtime
Zilionis et al. (2019)	173954	272.26 ± 4.22	183.77 ± 2.46

Further comments

kana's key innovation lies in its use of modern web technologies to perform the analysis directly in the browser. This eliminates the difficulties of software installation and makes the analysis accessible to a non-programming audience. At the same time, we retain all the benefits of client-side operation, namely:

- No dependency on a backend server, which greatly simplifies application deployment and maintenance for developers. For example, we do not need any monitoring, scaling, hardening, or other DevOps processes typically associated with a backend architecture.
- No latency from transfer of data and results to/from the server, which yields a more responsive user experience. In particular, uploads of large data files (up to 1 GB, depending on the number of cells and sequencing depth) would cause a significant delay.
- No issues with data ownership, enabling users to process sensitive datasets from the privacy of their own machine. Users are not forced to trust the application maintainers to correctly handle and secure their datasets on the latter's server.
- Effectively free compute, as each client brings its own computing power to the application. No funding is required for a centralized backend, allowing us to pass on those savings to our users, i.e., kana can be used by anyone for free.

Client-side compute has interesting scalability characteristics compared to a traditional backend approach. Most obviously, we are constrained by the computational resources available on the client machine, which limits the size of any single dataset that can be analyzed by a particular client. From another perspective, though, client-side compute is more scalable as it automatically distributes analyses of many datasets across any number of machines at no cost and with no configuration. This is especially relevant for web applications like kana where the maintainers would otherwise be responsible for provisioning more computing resources to match user demand.

That said, how do we deal with large datasets? Our C++ implementations mean that we are not limited to computation in the browser. We can easily provide wrappers to the same underlying libraries in any client-side framework, e.g., as a command-line tool or as an extension to existing data science ecosystems (A. Lun, 2021b). Indeed, one could use the wrapped C++ libraries to run large analyses on a sufficiently well-resourced backend, export the results in a kana-compatible file format, and then serve them to clients for use in kana's exploration mode.

Acknowledgements

We thank Michael Lawrence, Hector Corrada Bravo and Adrian Waddell for their suggestions to improve this manuscript.

References

Aboukhalil, R. (2019). *Biowasm*. https://github.com/biowasm/biowasm

Amezquita, R. A., Lun, A. T. L., Becht, E., Carey, V. J., Carpp, L. N., Geistlinger, L., Marini, F., Rue-Albrecht, K., Risso, D., Soneson, C., Waldron, L., Pagès, H., Smith, M. L., Huber, W., Morgan, M., Gottardo, R., & Hicks, S. C. (2020). Orchestrating single-cell analysis with Bioconductor. *Nat Methods*, *17*(2), 137–145. https://doi.org/10.1038/s41592-019-0654-x

Aran, D., Looney, A. P., Liu, L., Wu, E., Fong, V., Hsu, A., Chak, S., Naikawadi, R. P., Wolters, P. J., Abate, A. R., & others. (2019). Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. *Nature Immunology*, *20*(2), 163–172. https://doi.org/10.1038/s41590-018-0276-y

Bach, K., Pensa, S., Grzelak, M., Hadfield, J., Adams, D. J., Marioni, J. C., & Khaled, W. T. (2017). Differentiation dynamics of mammary epithelial cells revealed by single-cell RNA sequencing. *Nature Communications*, *8*(1), 1–11. https://doi.org/10.1038/s41467-017-02001-5

Bacher, P., Rosati, E., Esser, D., Martini, G. R., Saggau, C., Schiminsky, E., Dargvainiene, J., Schröder, I., Wieters, I., Khodamoradi, Y., & others. (2020). Low-avidity CD4+ T cell responses to SARS-CoV-2 in unexposed individuals and humans with severe COVID-19. *Immunity*, *53*(6), 1258–1271.

Baglama, James, & Reichel, L. (2005). Augmented implicitly restarted lanczos bidiagonalization methods. *SIAM Journal on Scientific Computing*, *27*(1), 19–42. https://doi.org/10.1137/04060593X

Baglama, Jim, Reichel, L., & Lewis, B. W. (2019). *Irlba: Fast truncated singular value decomposition and principal components analysis for large dense and sparse matrices*. https://CRAN.R-project.org/package=irlba

Bernhardsson, E. (2021). *Annoy*. https://github.com/spotify/annoy

Chang, W., Cheng, J., Allaire, J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., & Borges, B. (2021). *Shiny: Web application framework for r*. https://CRAN.R-project.org/package=shiny

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, *74*(368), 829–836. https://doi.org/10.1080/01621459.1979.10481038

Ernst, C., Eling, N., Martinez-Jimenez, C. P., Marioni, J. C., & Odom, D. T. (2019). Staged developmental mapping and X chromosome transcriptional dynamics during mouse spermatogenesis. *Nature Communications*, *10*(1), 1–20. https://doi.org/10.1038/s41467-019-09182-1

Fan, J., Fan, D., Slowikowski, K., Gehlenborg, N., & Kharchenko, P. (2017). UBiT2: A client-side web-application for gene expression data analysis. *bioRxiv*, 118992. https://doi.org/10.1101/118992

Gómez, J., García, L. J., Salazar, G. A., Villaveces, J., Gore, S., García, A., Martín, M. J., Launay, G., Alcántara, R., Del-Toro, N., & others. (2013). BioJS: An open source JavaScript framework for biological data visualization. *Bioinformatics*, *29*(8), 1103–1104. https://doi.org/10.1093/bioinformatics/btt100

Gould, J., Yang, Y., & Li, B. (2021). *Cirrocumulus*. https://cirrocumulus.readthedocs.io/en/latest/

Haas, A., Rossberg, A., Schuff, D. L., Titzer, B. L., Holman, M., Gohman, D., Wagner, L., Zakai, A., & Bastien, J. (2017). Bringing the web up to speed with WebAssembly. *SIGPLAN Not.*, *52*(6), 185–200. https://doi.org/10.1145/3140587.3062363

Haghverdi, L., Lun, A. T., Morgan, M. D., & Marioni, J. C. (2018). Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nature Biotechnology*, *36*(5), 421–427. https://doi.org/10.1038/nbt.4091

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100–108. https://doi.org/10.2307/2346830

Krijthe, J. H. (2015). *Rtsne: T-distributed stochastic neighbor embedding using barnes-hut implementation*. https://github.com/jkrijthe/Rtsne

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137. https://doi.org/10.1109/TIT.1982.1056489

Lun, A. (2021a). *A C++ API for all sorts of matrices*. https://github.com/LTLA/tatami

Lun, A. (2021b). *A slimmed-down version of scran*. https://github.com/LTLA/scran.chan

Lun, A. T. L. (2021a). *A C++ library for single-cell data analysis*. https://github.com/LTLA/libscran

Lun, A. T. L. (2021b). *A C++ library for UMAP*. https://github.com/LTLA/umappp

Lun, A. T. L. (2021c). *BiocNeighbors: Nearest neighbor detection for bioconductor packages*. https://bioconductor.org/packages/BiocNeighbors

Lun, A. T. L. (2021d). *C++ library for IRLBA*. https://github.com/LTLA/CppIrlba

Lun, A. T. L. (2021e). *C++ library for k-means*. https://github.com/LTLA/CppKmeans

Lun, A. T. L. (2021f). *C++ library for t-SNE*. https://github.com/LTLA/qdtsne

Lun, A. T. L. (2021g). *Collection of KNN algorithms*. https://github.com/LTLA/knncolle

Lun, A. T. L. (2021h). *Weighted LOWESS for C++*. https://github.com/LTLA/CppWeightedLowess

Lun, A. T. L. (2022a). *A C++ implementation of the MNN correction algorithm*. https://github.com/LTLA/CppMnnCorrect

Lun, A. T. L. (2022b). *A C++ library to read and write RDS files*. https://github.com/LTLA/rds2cpp

Lun, A. T. L. (2022c). *C++ port of the SingleR method for cell type annotation*. https://github.com/LTLA/SinglePP

Lun, A. T. L., Amezquita, R. A., Gottardo, R., & Hicks, S. C. (2020). *Orchestrating single-cell analysis with Bioconductor*. Bioconductor. https://doi.org/10.1038/s41592-019-0654-x

Lun, A. T. L., & Haghverdi, L. (2018). *Single-cell batch correction methods*. https://bioconductor.org/packages/release/bioc/html/batchelor.html

Lun, Aaron T. L., McCarthy, D. J., & Marioni, J. C. (2016). A step-by-step workflow for low-level analysis of single-cell RNA-seq data with bioconductor. *F1000Res.*, *5*, 2122. https://doi.org/10.12688/f1000research.9501.2

Lun, A. T. L., Pagès, H., & Smith, M. L. (2018). beachmat: A Bioconductor C++ API for accessing high-throughput biological data from a variety of R matrix types. *PLoS Comput Biol*, *14*(5), e1006135. https://doi.org/10.1371/journal.pcbi.1006135

Lun, A., & Kancherla, J. (2021). *Single cell RNA-seq analysis in Javascript*. https://github.com/jkanche/scran.js

Lun, A., & Kancherla, J. (2023). Powering single-cell analyses in the browser with WebAssembly. *bioRxiv*. https://doi.org/10.1101/2022.03.02.482701

Maaten, L. van der. (2014). Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, *15*(93), 3221–3245. http://jmlr.org/papers/v15/vandermaaten14a.html

McCarthy, D. J., Campbell, K. R., Lun, A. T. L., & Willis, Q. F. (2017). Scater: Pre-processing, quality control, normalisation and visualisation of single-cell RNA-seq data in R. *Bioinformatics*, *33*, 1179–1186. https://doi.org/10.1093/bioinformatics/btw777

McInnes, L., Healy, J., Saul, N., & Grossberger, L. (2018). UMAP: Uniform manifold approximation and projection. *The Journal of Open Source Software*, *3*(29), 861. https://doi.org/10.21105/joss.00861

Megill, C., Martin, B., Weaver, C., Bell, S., Prins, L., Badajoz, S., McCandless, B., Pisco, A. O., Kinsella, M., Griffin, F., Kiggins, J., Haliburton, G., Mani, A., Weiden, M., Dunitz, M., Lombardo, M., Huang, T., Smith, T., Chambers, S., … Carr, A. (2021). Cellxgene: A performant, scalable exploration platform for high dimensional sparse matrices. *bioRxiv*. https://doi.org/10.1101/2021.04.05.438318

Melville, J. (2021). *Uwot: The uniform manifold approximation and projection (UMAP) method for dimensionality reduction*. https://github.com/jlmelville/uwot

Morgan, M., & Shepherd, L. (n.d.). *ExperimentHub: Client to access ExperimentHub resources*. https://bioconductor.org/packages/release/bioc/html/ExperimentHub.html

Pagès, H., Hickey, P., & Lun, A. T. L. (2021). *DelayedArray: A unified framework for working transparently with on-disk and -memory array-like datasets*. https://bioconductor.org/packages/DelayedArray

Paul, F., Arkin, Y., Giladi, A., Jaitin, D. A., Kenigsberg, E., Keren-Shaul, H., Winter, D., Lara-Astiaso, D., Gury, M., Weiner, A., & others. (2015). Transcriptional heterogeneity and lineage commitment in myeloid progenitors. *Cell*, *163*(7), 1663–1677. https://doi.org/10.1016/j.cell.2015.11.013

Ritchie, M. E., Phipson, B., Wu, D., Hu, Y., Law, C. W., Shi, W., & Smyth, G. K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research*, *43*(7), e47. https://doi.org/10.1093/nar/gkv007

Schmid-Burgk, J. L., & Hornung, V. (2015). BrowserGenome.org: Web-based RNA-seq data analysis and visualization. *Nature Methods*, *12*(11), 1001–1001. https://doi.org/10.1038/nmeth.3615

Stegle, O., Teichmann, S. A., & Marioni, J. C. (2015). Computational and analytical challenges in single-cell transcriptomics. *Nature Reviews Genetics*, *16*(3), 133–145. https://doi.org/10.1038/nrg3833

Su, T., & Dy, J. G. (2007). In search of deterministic methods for initializing k-means and gaussian mixture clustering. *Intelligent Data Analysis*, *11*(4), 319–338. https://doi.org/10.3233/ida-2007-11402

Vassilvitskii, S., & Arthur, D. (2007). K-means++: The advantages of careful seeding. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035.

Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, 311–321. ISBN: 0898713137

Zakai, A. (2011). Emscripten: An LLVM-to-JavaScript compiler. *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion*, 301–312. https://doi.org/10.1145/2048147.2048224

Zeisel, A., Muñoz-Manchado, A. B., Codeluppi, S., Lönnerberg, P., La Manno, G., Juréus, A., Marques, S., Munguba, H., He, L., Betsholtz, C., & others. (2015). Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science*, *347*(6226), 1138–1142. https://doi.org/10.1126/science.aaa1934

Zilionis, R., Engblom, C., Pfirschke, C., Savova, V., Zemmour, D., Saatcioglu, H. D., Krishnan, I., Maroni, G., Meyerovitz, C. V., Kerwin, C. M., & others. (2019). Single-cell transcriptomics of human and mouse lung cancers reveals conserved myeloid populations across individuals and species. *Immunity*, *50*(5), 1317–1334. https://doi.org/10.1016/j.immuni.2019.03.009