SCANPY: large-scale single-cell gene expression data analysis

F. Alexander Wolf1*, Philipp Angerer1 and Fabian J. Theis1,2*

Abstract
SCANPY is a scalable toolkit for analyzing single-cell gene expression data. It includes methods for preprocessing, visualization, clustering, pseudotime and trajectory inference, differential expression testing, and simulation of gene regulatory networks. Its Python-based implementation efficiently deals with data sets of more than one million cells (https://github.com/theislab/Scanpy). Along with SCANPY, we present ANNDATA, a generic class for handling annotated data matrices (https://github.com/theislab/anndata).

Keywords: Single-cell transcriptomics, Machine learning, Scalability, Graph analysis, Clustering, Pseudotemporal ordering, Trajectory inference, Differential expression testing, Visualization, Bioinformatics

Background
Simple integrated analysis work flows for single-cell transcriptomic data [1] have been enabled by frameworks such as Seurat [2], Monocle [3], SCDE/Pagoda [4], MAST [5], Cell Ranger [6], Scater [7], and Scan [8]. However, these frameworks do not scale to the increasingly available large data sets with up to and more than one million cells. Here, we present a framework that overcomes this limitation and provides similar analysis possibilities. Moreover, in contrast to the existing R-based frameworks, SCANPY’s Python-based implementation is easy to interface with advanced machine-learning packages, such as TensorFlow [9].

Results
SCANPY integrates canonical analysis methods in a scalable way
SCANPY integrates the analysis possibilities of established R-based frameworks and provides them in a scalable and modular form. Specifically, SCANPY provides preprocessing comparable to Seurat [10] and Cell Ranger [6], visualization through TSNE [11, 12], graph-drawing [13–15] and diffusion maps [11, 16, 17], clustering similar to Phenograph [18–20], identification of marker genes for clusters via differential expression tests and pseudotemporal ordering via diffusion pseudotime [21], which compares favorably [22] with Monocle 2 [22], and Wishbone [23] (Fig. 1a).

SCANPY is benchmarked in comparisons with established packages
In a detailed clustering tutorial of 2700 peripheral blood mononuclear cells (PBMCs), adapted from one of Seurat’s tutorials (http://satijalab.org/seurat/pbmc3k_tutorial.html) [2], all steps starting from raw count data to the identification of cell types are carried out, providing speedups between 5 and 90 times in each step (https://github.com/theislab/scanpy_usage/tree/master/170505_seurat). Benchmarking against the more run-time optimized Cell Ranger R kit [6], we demonstrate a speedup of 5 to 16 times for a data set of 68,579 PBMCs (Fig. 1a,b, https://github.com/theislab/scanpy_usage/tree/master/170503_zheng17) [6]. Moreover, we demonstrate the feasibility of analyzing 1.3 million cells without subsampling in a few hours of computing time on eight cores of a small computing server (Fig. 1c, https://github.com/theislab/scanpy_usage/tree/master/170522_visualizing_one_million_cells). Thus, SCANPY provides tools with speedups that enable an analysis of data sets with more than one million cells and an interactive analysis with run times of the order of seconds for about 100,000 cells.
In addition to the mentioned standard clustering-based analyses approaches, we demonstrate the reconstruction of branching developmental processes via diffusion pseudotime [21] as in the original paper (https://github.com/theislab/scanpy_usage/tree/master/170502_haghverdi16), the simulation of single cells using literature-curated gene regulatory networks based on the ideas of [24] (https://github.com/theislab/scanpy_usage/tree/master/170430_krumsiek11), and the analysis of deep-learning results for single-cell imaging data [25] (https://github.com/theislab/scanpy_usage/tree/master/170529_images).

SCANPY introduces efficient modular implementation choices

With SCANPY, we introduce the class ANNDATA—with a corresponding package ANNDATA—which stores a data matrix with the most general annotations possible: annotations of observations (samples, cells) and variables (features, genes), and unstructured annotations. As SCANPY is built around that class, it is easy to add new functionality to the toolkit. All statistics and machine-learning tools extract information from a data matrix, which can be added to an ANNDATA object while leaving the structure of ANNDATA unaffected. ANNDATA is similar to R’s EXPRESSIONSET [26], but supports sparse data and allows HDF5-based backing of ANNDATA objects on disk, a format independent of platform, framework, and language. This allows operating on an ANNDATA object without fully loading it into memory—the functionality is offered via ANNDATA’s backed mode as opposed to its memory mode. To simplify memory-efficient pipelines, SCANPY’s functions operate in-place by default but allow the optional non-destructive transformation of objects. Pipelines written this way can then also be run in backed mode to exploit online-learning formulations of algorithms. Almost all of SCANPY’s tools are parallelized.

Conclusions

SCANPY’s scalability directly addresses the strongly increasing need for aggregating larger and larger data...
sets [30] across different experimental setups, for example within challenges such as the Human Cell Atlas [31]. Moreover, being implemented in a highly modular fashion, SCANPY can be easily developed further and maintained by a community. The transfer of the results obtained with different tools used within the community is simple, as SCANPY’s data storage formats and objects are language independent and cross-platform. SCANPY integrates well into the existing Python ecosystem, in which no comparable toolkit yet exists.

During the revision of this article, the loom file format (https://github.com/linnarsson-lab/loompy) was proposed for HDF5-based storage of annotated data. Within a joint effort of facilitating data exchange across different labs, ANNDATA now supports importing and exporting to loom (https://github.com/linnarsson-lab/loompy). In this context, we acknowledge the discussions with S. Linnarson, which motivated us to extend ANNDATA’s previously static to a dynamic HDF5 backing. Just before submission of this manuscript, a C++ library that provides simple interfacing of HDF5-backed matrices in R was made available as a preprint [32].

Methods
SCANPY’s technological foundations
SCANPY’s core relies on NUMPY [33], SCIPY [34], MATPLOTLIB [35], PANDAS [36], and H5PY [37]. Parts of the toolkit rely on scikit-learn [27], STATSMODELS [38], SEABORN [39], NETWORKX [28], iGRAPH [14], the TSNE package of [40], and the Louvain clustering package of [41]. The ANNDATA class—available within the package ANNDATA—relies only on NUMPY, SCIPY, PANDAS, and H5PY.

SCANPY’s Python-based implementation allows easy interfacing to advanced machine-learning packages such as TENSORFLOW [9] for deep learning [42], LIMIX for linear mixed models [43], and GPY/GPFLOW for Gaussian processes [44, 45]. However, we note that the Python ecosystem comes with less possibilities for classical statistical analyses compared to R.

Comparison with existing Python packages for single-cell analysis
Aside from the highly popular SCLVM (https://github.com/PMBio/sclVM) [46, 47], which uses Gaussian process latent variable models for inferring hidden sources of variation, there are, among others, the visualization frameworks FASTPROJECT (https://github.com/YosefLab/FastProject) [48], ACCENSE (http://www.cellaccense.com/) [49], and SPRING (https://github.com/AllonKleinLab/SPRING) [15]—the latter uses the JavaScript package (http://d3js.org) D3.js for the actual visualization and Python only for preprocessing—the trajectory inference tool SCIMITAR (https://github.com/dimenwarper/scimitar), the clustering tool PHENOGRAPH (https://github.com/jacoblevine/PhenoGraph) [19], the single-cell experiment design tool MIMOSCA (https://github.com/asncd/MIMOSCA) [50], UMISS (https://github.com/vals/umiss) for handling raw read data [51], the tree-inference tool ECLAIR (https://github.com/GGiecold/ECLAIR) [52], and the framework FLOTILLA (https://github.com/yeolab/flotilla), which comes with modules for simple visualization, simple clustering, and differential expression testing. Hence, only the latter provides a data analysis framework that solves more than one specific task. In contrast to SCANPY, however, FLOTILLA is neither targeted at single-cell nor at large-scale data and does not provide any graph-based methods, which are the core of SCANPY. Also, FLOTILLA is built around a complicated class STUDY, which contains data, tools, and plotting functions. SCANPY, by contrast, is built around a simple HDF5-backed class ANNDATA, which makes SCANPY both scalable and extendable (law of Demeter).

Availability and requirements
SCANPY’s and ANNDATA’s open-source code are maintained on GitHub (https://github.com/theislab/scanpy, https://github.com/theislab/anndata) and published under the BSD3 license.

SCANPY and ANNDATA are released via the Python packaging index: https://pypi.python.org/pypi/scanpy and https://pypi.python.org/pypi/anndata.

Demonstrations and benchmarks discussed in the main text are all stored at https://github.com/theislab/scanpy_usage and summarized here:

- Analyzing 68,579 PBMCs (Fig. 1) in a comparison with the CELL RANGER R kit [6]: https://github.com/theislab/scanpy_usage/tree/master/170503_zheng17.
- Clustering and identifying cell types, adapted from and benchmarked with http://satijalab.org/seurat/pbmc3k_tutorial.html and one of SEURAT’s tutorials [2]: https://github.com/theislab/scanpy_usage/tree/master/170505_seurat.
- Visualizing and clustering 1.3 million cells (Fig. 1c): https://github.com/theislab/scanpy_usage/tree/master/170522_visualizing_one_million_cells.
- Reconstructing branching processes via diffusion pseudotime [21]: https://github.com/theislab/scanpy_usage/tree/master/170502_haghverdi16.
- Simulating single cells using gene regulatory networks [24]: https://github.com/theislab/scanpy_usage/tree/master/170430_krumsiek11.
- Analyzing deep-learning results for single-cell images [25]: https://github.com/theislab/scanpy_usage/tree/master/170529_images.
The data sets used in demonstrations and benchmarks.

Programming language: Python

Operating system: Linux, Mac OS and Windows

Acknowledgements

We thank the authors of Seurat, Cell Ranger, and Scanpy for sharing their great tutorials. We are grateful to Sten Linnarson for discussions on HDF5-backing of data on disk. We thank S. Tritschler, L. Simon, D. S. Fischer, and M. Büttner for commenting on the software package. We thank M. Lotfollahi for clustering the 1.3-million-cell data set and N. K. Chlis for setting up installation instructions for Windows.

Funding

FAW acknowledges the support of the Helmholtz Postdoc Programme, Initiative and Networking Fund of the Helmholtz Association. FJT acknowledges support from the German Research Foundation (DFG) within the Collaborative Research Centre 1243, Subproject A17.

Authors’ contributions

FAW conceived the project and developed the software. PA co-developed the software, mainly in regard to architecture and maintainability. FJT supervised the project and helped interpret and present the results. FAW wrote the manuscript with the help of PA and FJT. All authors read and approved the final manuscript.

Ethics approval and consent to participate

Ethics approval was not applicable for this study.

Competing interests

None of the authors declare competing interests.

Publisher’s Note

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

Received: 16 August 2017 Accepted: 20 December 2017

Published online: 06 February 2018

References

1. Wagner A, Regev A, Yosef N. Revealing the vectors of cellular identity with single-cell genomics. Nat Biotechnol. 2016;34:1145–60.
2. Satija R, Farrell JA, Gennert D, Schier AF, Regev A. Spatial reconstruction of single-cell gene expression data. Nat Biotechnol. 2015;33:495–502.
3. Trapnell C, et al. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. Nat Biotechnol. 2014;32:381–6.
4. Balooch AM, et al. Diffusion maps for high-dimensional single-cell expression data. Bioinformatics. 2015;31:2989–98.
5. Buitkert F, Theis FJ. Diffusion maps for high-dimensional single-cell analysis of differentiated data. Bioinformatics. 2015;31:1974–80.
6. Xu C, Su Z. Identification of cell types from single-cell transcriptomes using a novel clustering method. Bioinformatics. 2015;31:1974–80.
7. Setty, M, et al. Wishbone identifies bifurcating developmental trajectories from single-cell data. Nat Biotechnol. 2016;34:637–45.
8. Wittmann, DM, et al. Transforming Boolean models to continuous models: methodology and application to T-cell receptor signaling. BMC Syst Biol. 2009;3:98.
9. Eulenberg P, et al. Reconstructing cell cycle and disease progression using deep learning. Nat Commun. 2017;8:463.
10. Huber, W, et al. Orchestrating high-throughput genomic analysis with Bioconductor. Nat Methods. 2015;12:115–21.
11. Pedregosa F, et al. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
12. Hagberg AA, Schult DA, Swart PJ. Exploring network structure, dynamics, and function using NetworkX. In: Proceedings of the 7th Python in Science Conference (SciPy2008). Pasadena; 2008. p. 11–15.
13. Bastian M, Heymann S, Jacomy M. Gephi: an open source software for exploring and manipulating networks. International AAAI Conference on Weblogs and Social Media. 2009.
14. Angerer, P, et al. Single cells make big data: new challenges and opportunities in transcriptomics. Curr Opin Syst Biol. 2017;4:85–91.
15. Regev A, et al. Science forum: the human cell atlas. eLife. 2017;6:e27041.
16. Lut AL, Pagé H, Smith ML. BEACON: a BIOCONDUCTOR C++ API for accessing single-cell genomics data from a variety of R matrix types. bioRxiv. 2017. https://doi.org/10.1101/167445.
17. van der Walt S, Colbert SC, Varoquaux G. The NumPy array: a structure for efficient numerical computation. Comput Sci Eng. 2011;13:22–30.
18. Jones E, Oliphant T, Peterson P, et al. SciPy: source scientific tools for Python. 2001. https://www.scipy.org/citing.html.
19. Hunter JD. Matplotlib: a 2D graphics environment. Comput Sci Eng. 2007;9:90–9.
20. McKinney W. Data structures for statistical computing in Python. In: van der Walt S, Millman J, editors. Proceedings of the 9th Python in Science Conference, 2010. p. 51–6.
21. Collette A. Python and HDF5. Sebato pol: O’Reilly; 2013.
22. Seabold S, Perktold J. Statsmodels: econometric and statistical modeling with Python. 9th Python in Science Conference. 2014.
23. Waskom, M, et al. In: Varoquaux G, Vaught T, Millman J, editors. S
library using TensorFlow. J Mach Learn Res. 2017;18(40):1–6. http://jmlr.org/papers/v18/16-537.html.

45. Matthews de, G, Alexander G, et al. GPFLOW: a Gaussian process library using TensorFLow. J Mach Learn Res. 2017;18:1–6. https://github.com/SheffieldML/GPFlow.

46. Buettner F, et al. Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. Nat Biotechnol. 2015;33:155.

47. Buettner F, Pratanwanich N, McCarthy DJ, Marioni JC, Stegle O. F-SCLVM: scalable and versatile factor analysis for single-cell RNA-seq. Genome Biol. 2017;18:212.

48. DeTomaso D, Yosef N. FASTPROJECT: a tool for low-dimensional analysis of single-cell RNA-seq data. BMC Bioinform. 2016;17:315.

49. Shekhar K, Brodin P, Davis MM, Chakraborty AK. Automatic classification of cellular expression by nonlinear stochastic embedding (ACCENSE). 2013. p 202–7.

50. Dixit A, et al. PERTURB-SEQ: dissecting molecular circuits with scalable single-cell RNA profiling of pooled genetic screens. Cell. 2016;167:1853–66.e17.

51. Svensson V, et al. Power analysis of single cell RNA-sequencing experiments. Nat Methods. 2017;14:381.

52. Giecold G, Marco E, Garcia SP, Trippa L, Yuan G-C. Robust lineage reconstruction from high-dimensional single-cell data. Nucleic Acids Res. 2016;44:e122.