doi:10.1017/S1431927619001399
Microsc. Microanal. 25 (Suppl 2), 2019

Scientific Python: A Mature Computational Ecosystem for Microscopy

Stéfan van der Walt[1*]

[1.] Berkeley Institute for Data Science, University of California, Berkeley USA.
[*] Corresponding author: stefanv@berkeley.edu

Over the past decade, the Scientific Python ecosystem has matured into a fully fledged data analysis environment. Python, as a language for *science*, draws strength from being a *general purpose programming language*: its ecosystem extends far beyond that of a scientific-only language. For example, Python has high-quality tools that can be used to ingest data from the internet or to control acquisition hardware, process the resulting data, store it in a database, and present it to collaborators via a web application or through interactive notebooks.

But Python has also seen rapid adoption in scientific computing itself, and there now exists a variety of libraries for handling N-dimensional array computation, processing table data, calculating scientific functions, plotting results, and disseminating results as programs accompanied by rich, interactive documents. Python is also prominent in data science [1] and *the* key player in machine learning [2]—specifically, deep learning, which has seen an explosion of applications in microscopy.

Assessing the needs of a typical data analyst in microscopy, we identify key challenges along with the libraries that address them:

- **Numerical data processing**: Regardless of whether experiments deal with images or other input formats, they all have associated numerical data—often best represented as arrays of numbers. Examples are 1-dimensional arrays for time-series, and 3-dimensional arrays for single-band volumetric data. The `NumPy` library [3] provides N-dimensional arrays, with methods for performing associated calculations. `NumPy` is under active development, both by its community, and a team funded by the Moore & Sloan Foundations at the Berkeley Institute for Data Science. Other prominent packages in this space include `xarray` [4], which provides labeled axes for N-dimensional arrays, and zarr, which implements chunked, compressed, N-dimensional arrays.

- **Image processing**: Almost all forms of microscopy involve image processing, such as segmentation, denoising, morphological analysis, region analysis, feature detection, and registration. `scikit-image` is a library with roots in two-dimensional image processing that has since been developed to include many N-dimensional capabilities. The term `scikit` (SciPy toolkit) indicates a package that is developed separately from the `SciPy` library, but according to similarly rigorous standards, often using the same processes, documentation formats, and so forth.

- **Visualization**: Matplotlib has long been the de-facto standard in Python for producing high quality, 2D publication figures. More recently, a number of web-enabled packages have emerged, such as Bokeh for interactive 2D display, and ipyvolume and ITK Jupyter Widgets for interactive 3D visualization in Jupyter notebooks. `spimagine` provides accelerated 3D visualization of time lapsed volumetric data, and Mayavi [5] delivers general 3D visualization through VTK. This space is still actively developing.

- **Parallel processing / batch pipelines**: Acquisition, especially on modern devices, produce large volumes of data, which often cannot be processed on a single laptop. The `dask` library provides mechanisms for developing code on a laptop, that can subsequently be executed on multiple cores, or multiple machines (nodes). It provides integration with cluster batch submissions systems such as SLURM, and can launch workers in the cloud through, e.g., Kubernetes. Task progress can be monitored via a browser dashboard.

Using these libraries, several additional specialist libraries have been developed for solving problems pertinent to microscopy. These include CellProfiler [6] for quantifying cell phenotypes, skan [7] for the analysis of fine structure skeletons through graphs, Trackpy [8] for particle tracking in any number of dimensions, PyCroscopy [9] for analysis of nanoscale materials imaging data, and Ilastik [10] for interactive image classification and segmentation.

A large amount of microscopy is currently done in the Java ecosystem. It is encouraging to note that one of the explicit goals of ImageJ2 [11] is better inter-operability with libraries such as OpenCV [12] and scikit-image [13].

We are also watching the building of online communities and infrastructure, such as Pangeo [14], with interest: these are excellent platforms through which to evaluate the suitability of various libraries for domain-specific, large data analysis tasks—knowledge that can then be fed back into tool development.

In this talk, we examine the scientific Python ecosystem, explore various of its key components, and argue that it is a mature and powerful toolset with application across a wide array of microscopy-related tasks.

References:

[1] D. Robinson, "Why is python growing so quickly?" *Stack Overflow blog*. https://stackoverflow.blog/2017/09/14/python-growing-quickly, 2017.
[2] T. Elliott, "The state of the octoverse: Machine learning." https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/, 2019.
[3] S. J. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, p. 22, 2011.
[4] S. Hoyer and J. Hamman, "Xarray: N-D labeled arrays and datasets in Python," *Journal of Open Research Software*, vol. 5, no. 1, 2017 [Online]. Available: http://doi.org/10.5334/jors.148
[5] P. Ramachandran and G. Varoquaux, "Mayavi: 3D Visualization of Scientific Data," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 40–51, 2011.
[6] A. E. Carpenter *et al.*, "CellProfiler: Image analysis software for identifying and quantifying cell phenotypes," *Genome Biology*, vol. 7, no. 10, p. R100, Oct. 2006 [Online]. Available: https://doi.org/10.1186/gb-2006-7-10-r100
[7] J. Nunez-Iglesias, A. J. Blanch, O. Looker, M. W. Dixon, and L. Tilley, "A new python library to analyse skeleton images confirms malaria parasite remodelling of the red blood cell membrane skeleton," in *PeerJ*, 2018.
[8] D. B. Allan, T. Caswell, N. C. Keim, and C. M. van der Wel, "Trackpy," Apr. 2018 [Online]. Available: https://doi.org/10.5281/zenodo.1213240
[9] S. Somnath, C. R. Smith, Nouamane Laanait, and S. Jesse, "Pycroscopy. Computer software. Oak ridge national laboratory" [Online]. Available: https://pycroscopy.github.io/pycroscopy/about.html
[10] C. Sommer, C. Straehle, U. Koethe, and F. A. Hamprecht, "Ilastik: Interactive learning and segmentation toolkit," in *Biomedical imaging: From nano to macro, 2011 ieee international symposium on*, 2011, pp. 230–233.
[11] C. T. Rueden *et al.*, "ImageJ2: ImageJ for the next generation of scientific image data," *BMC bioinformatics*, vol. 18, no. 1, p. 529, 2017.
[12] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
[13] S. van der Walt *et al.*, "Scikit-image: Image processing in Python," *PeerJ*, vol. 2, p. e453, Jun. 2014 [Online]. Available: https://doi.org/10.7717/peerj.453
[14] R. Abernathey *et al.*, "Pangeo NSF Earthcube Proposal," Aug. 2017 [Online]. Available: https://figshare.com/articles/Pangeo_NSF_Earthcube_Proposal/5361094

https://doi.org/10.1017/S1431927619001399 Published online by Cambridge University Press