Chan Zuckerberg Initiative

napari: A multi-dimensional image viewer for Python

Kyle Harrington, PhD, Software Engineer/Data Scientist, CZI
Kasia Kedziora, PhD, Assistant Professor, University of Pittsburgh
Ashley Anderson, PhD, Software Engineer, CZI Imaging Tech
Dannielle McCarthy, PhD, Application Scientist, CZI Imaging Tech

Halfway to i2k Workshop, 19 October 2023

Follow along: https://bit.ly/halfway-to-i2k

Link to these slides:
https://bit.ly/halfway-to-i2k

Agenda

1. **0:00-0:10** Intro to presenters and to napari
2. **0:10–0:30** napari Basics
3. **0:30–1:30** Jupyter Notebooks and Building a Plugin
4. **1:30-2:00** plugin packaging and publishing

Workshop Team and Target Audience

Kyle Harrington, CZI
Software Engineer/Data Scientist

Katarzyna “Kasia” Kedziora
Professor, University of Pittsburgh

Ashley Anderson, CZI
Software Engineer

Dannielle McCarthy, CZI
Product Application Scientist

For folks who are:

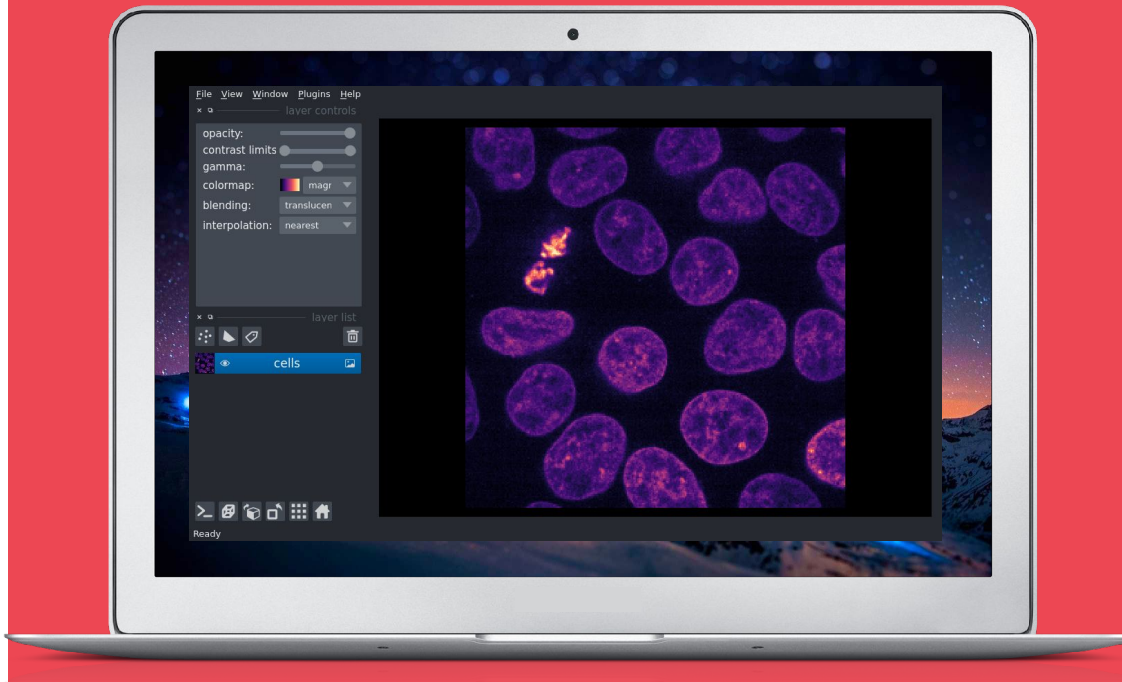
- curious about napari and adventurous about Python
- familiar with Python and want to customize their image analysis
- already creating Python tools and want to share to a broader audience

napari

Community-built (100+ contributors!), open-source tool based in Python for **browsing, annotating, and analyzing** large multi-dimensional **images**

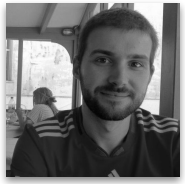
Plugin interface allows expansion into all areas of **imaging and biology**

Alpha stage of development → New features and plugins



napari is developed by an Open-source Community

Steering Council



Juan Nunez-Iglesias

Draga Doncila Pop

Kevin Yamauchi

Kyle Harrington

Contributors 145

+ 134 contributors

[Learn how to contribute!]

Core Developers

Juan Nunez-Iglesias jni

Ahmet Can Solak AhmetCanSolak

Kevin Yamauchi kevinyamauchi

Draga Doncila Pop DragaDoncila

Talley Lambert tlambert03

Wouter-Michiel Vierdag melonora

Loic A. Royer royerloic

Lorenzo Gaifas brisvag

Andy Sweet andy-sweet

Kira Evans kne42

Grzegorz Bokota Czaki

Genevieve Buckley GenevieveBuckley

Nicholas Sofroniew sofroniewn

Peter Sobolewski psobolewskiPhD

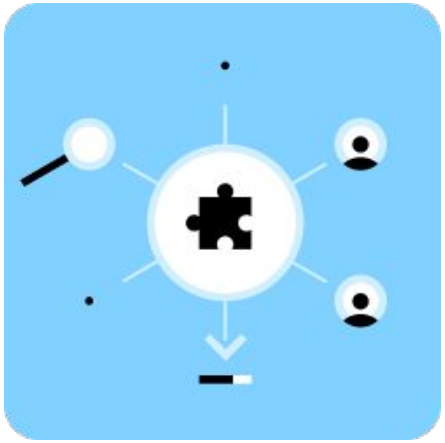
alisterburt

CZI Imaging Tech

How CZI is involved in the napari Community

napari viewer
visualization & analysis
interface

CZI: Fund core developers and
direct additions to repo

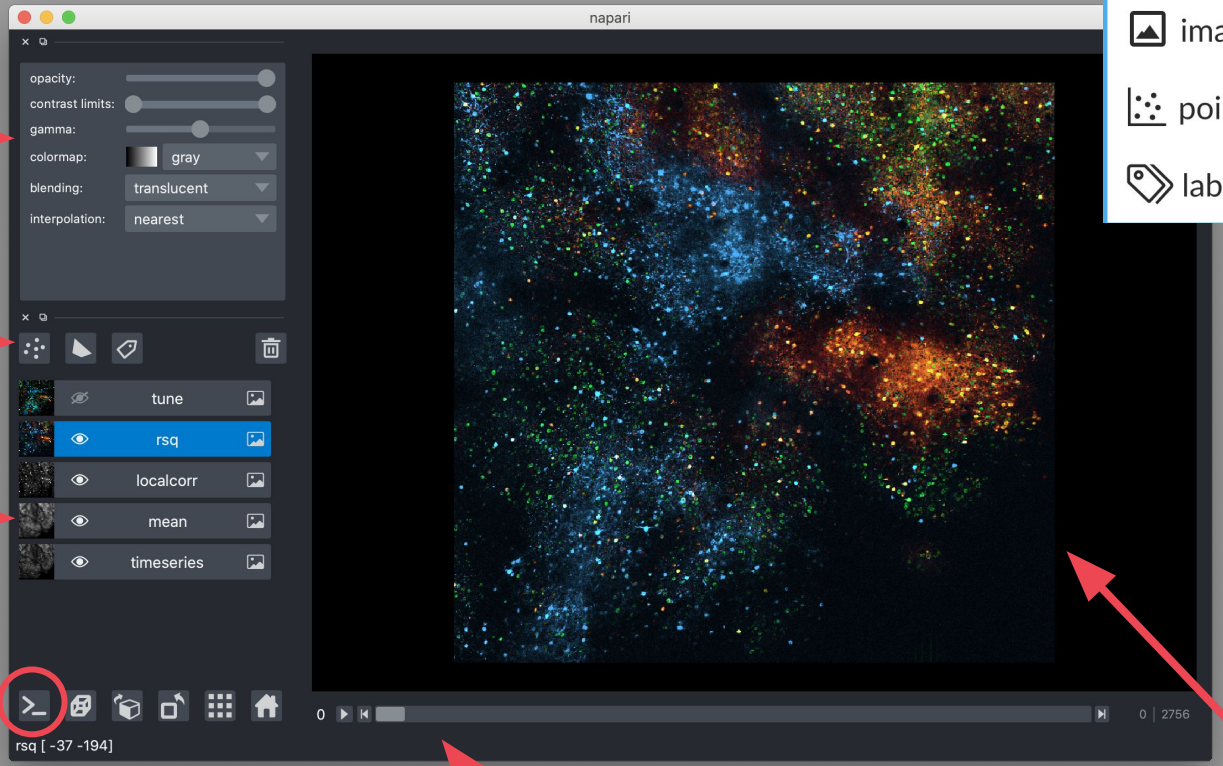


napari hub
discovery & sharing
of analysis plugins

CZI: Develop in service to the
community and fund plugin grants

Give researchers access to reproducible, quantitative image analysis.

napari Overview



layer controls

add layer buttons

layer list

integrated console

dimension sliders

canvas

layer types

image

shapes

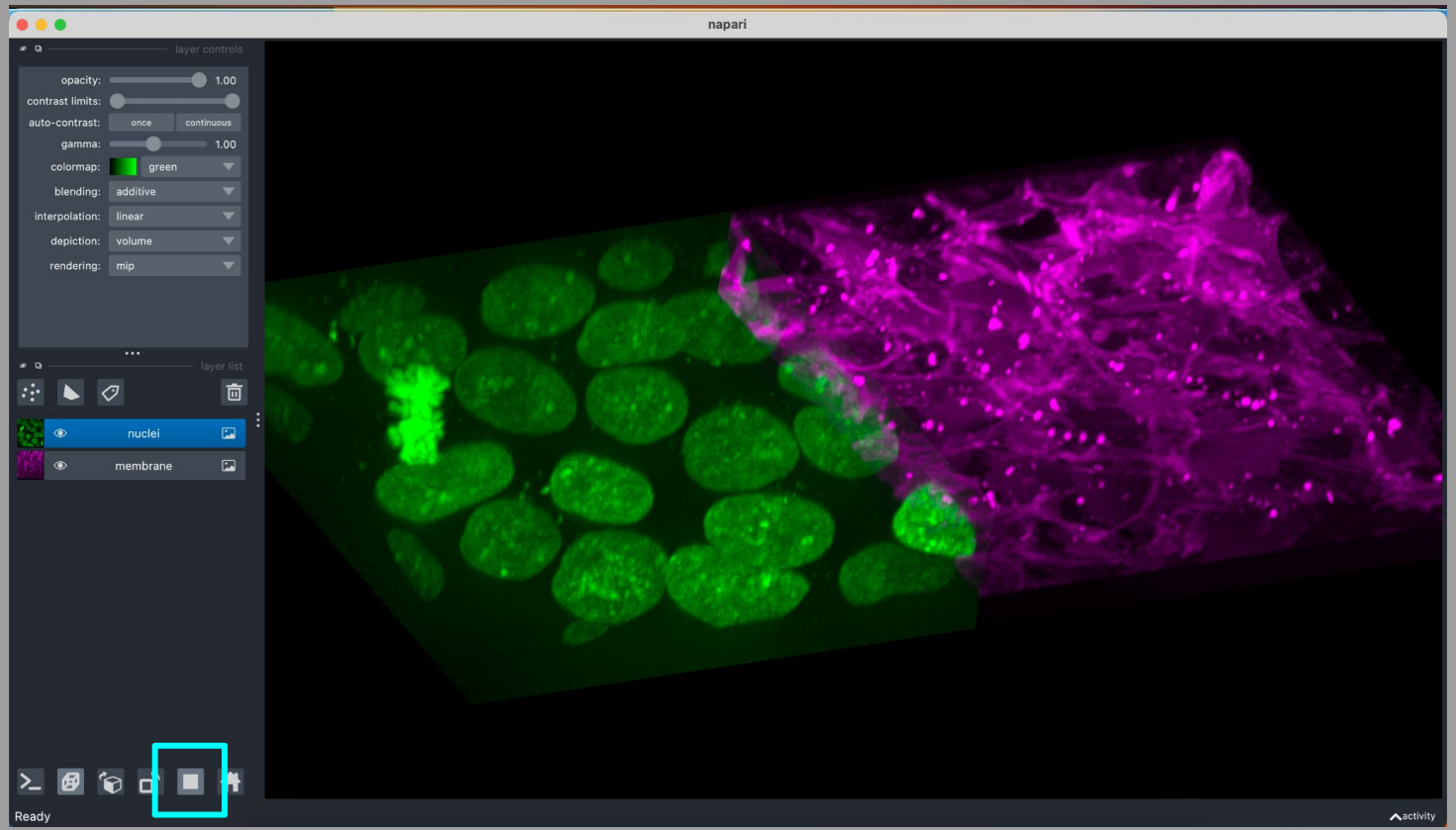
points

vectors

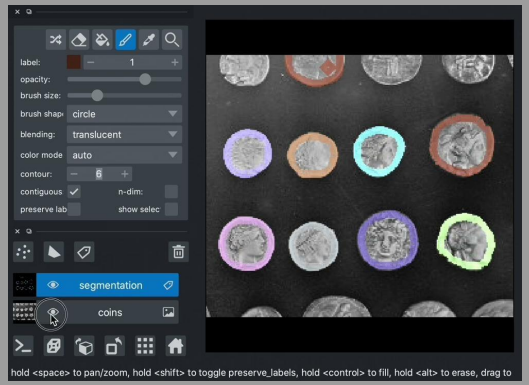
labels

surface

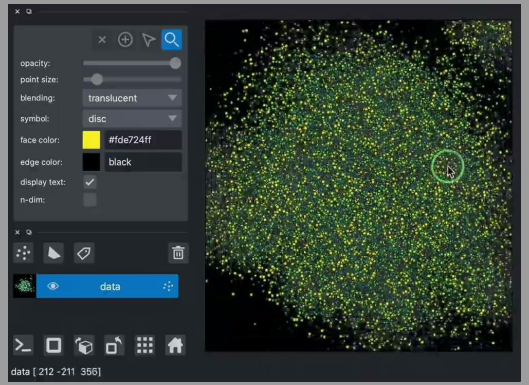
napari for 3D Image Visualization



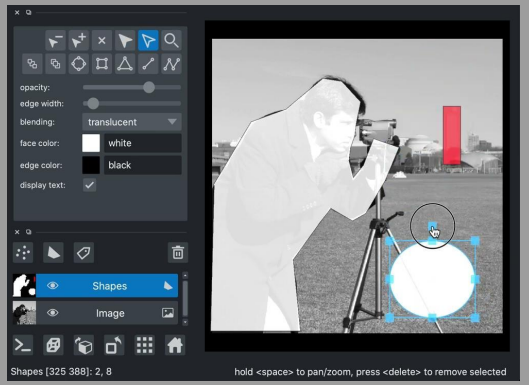
Examples of Layer Types in napari



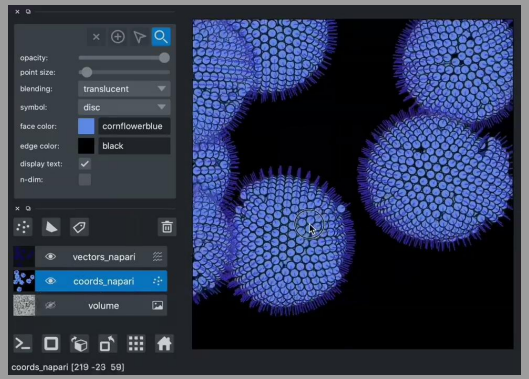
Labels: nDarray of Integers >= 0



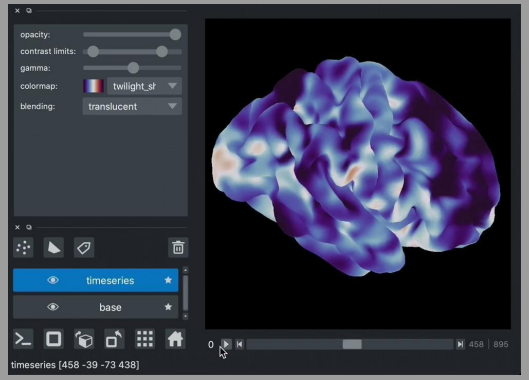
Points: List of Coordinates



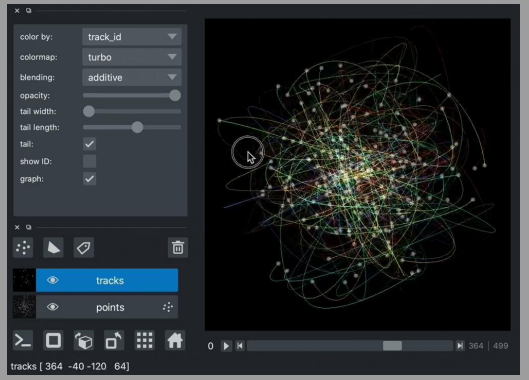
Shapes:



Vectors: Direction from A to B



Surface: Tuple of [Vertices, Faces, Values]



Tracks:

Thanks to Talley Lambert from Harvard Medical School for the videos!

Python image & data analysis tools are powerful!

Python scientific image computing toolbox
https://github.com/scikit-image

Python machine-learning toolbox
https://github.com/scikit-learn

Cell segmentation
https://github.com/MouseLand/cellpose

Data visualization & exploration
https://github.com/matplotlib
https://github.com/seaborn

Object detection with Stardist
https://github.com/stardist

cle._

GPU-accelerated image processing
https://github.com/clEsperanto

Napari

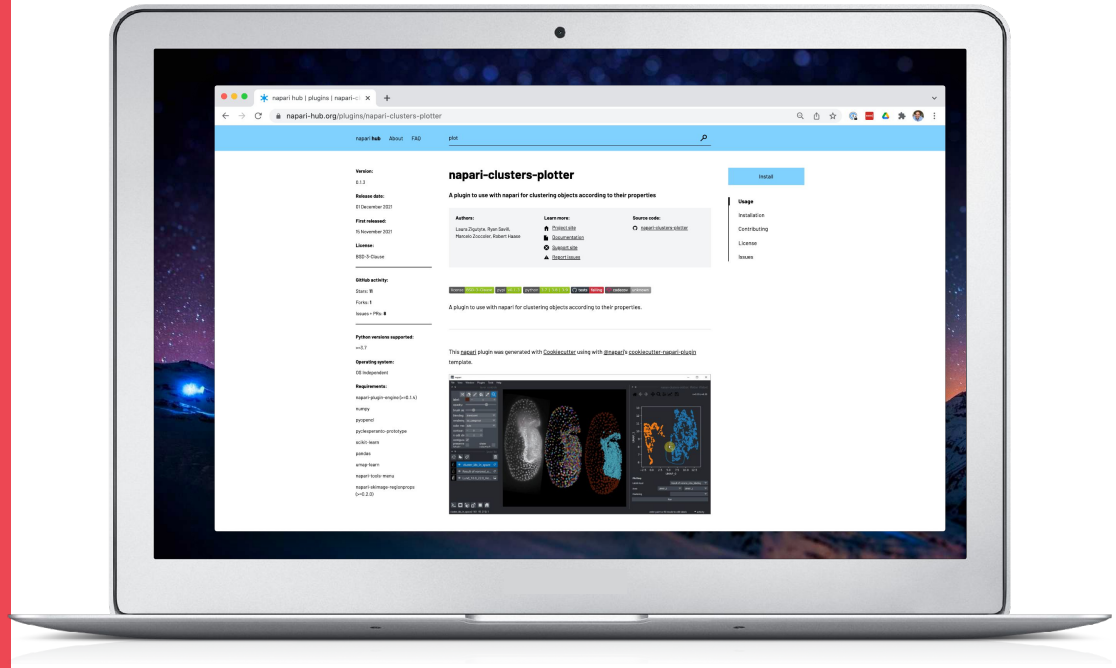
Existing functionality can be turned into plugins!
→ Interactivity
→ Automatic GUI generation

Slide courtesy of Johannes Soltwedel (full presentation)

napari **hub**

Discover, install, and share napari plugins

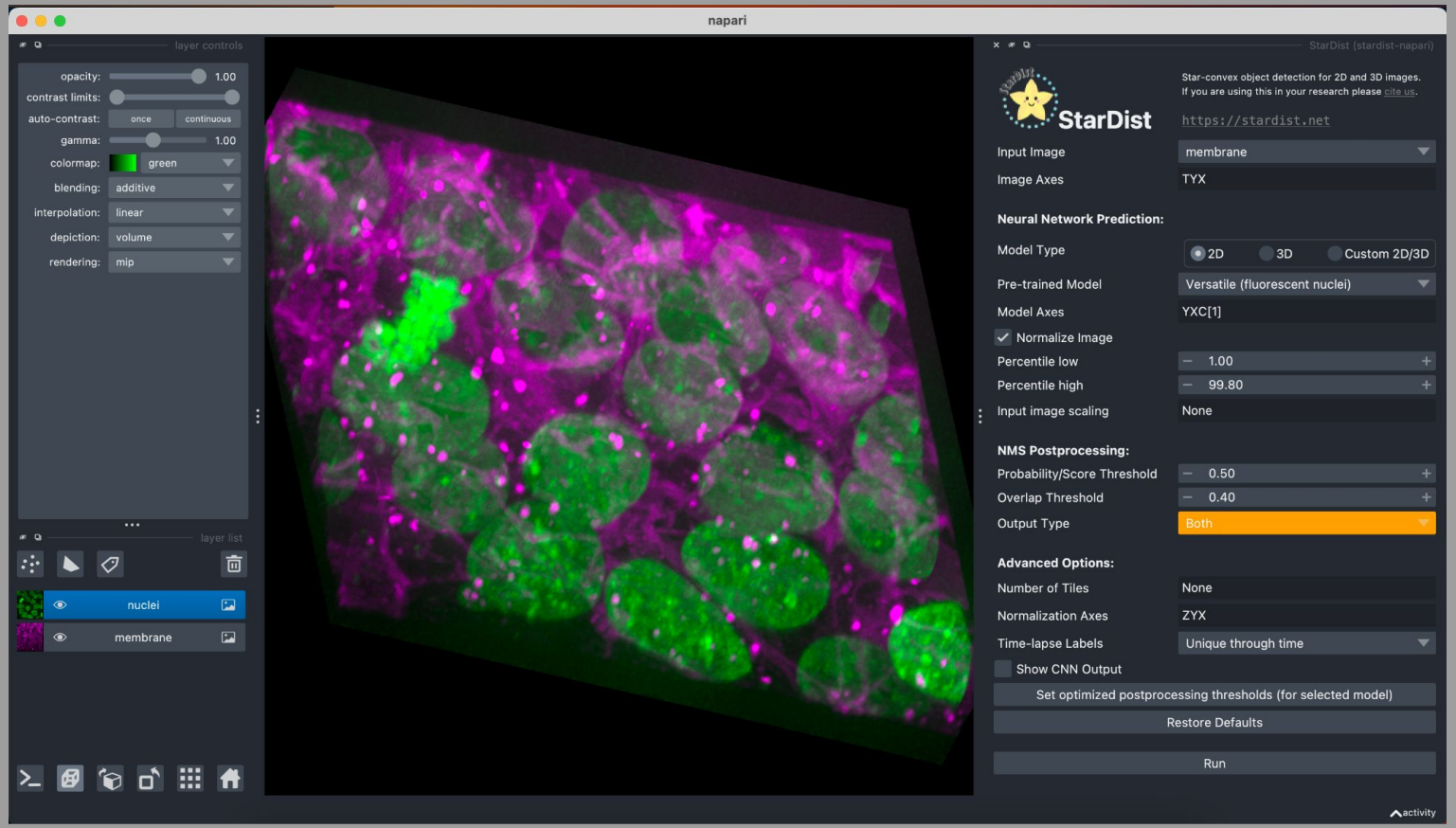
- Discover plugins that solve your image analysis challenges
- Learn how to install into napari
- Share your image analysis tools with napari's growing community



Built and hosted by the CZI Imaging Tech Team as a service to the napari community

Explore napari-hub.org

Example Plugin: StarDist Widget Plugin



FYI: Plugins for Getting Started

Chan
Zuckerberg
Initiative

Up Next: Work through Modules

Get involved @

napari.org
Getting Started and Documentation

napari-hub.org
All things plugins

forum.image.sc/tag/napari
Image analysis and plugin development questions

github.com/napari/napari
Source code and reporting bugs & features

@napari_imaging
The latest news from the community

napari.zulipchat.com
More synchronous chat

(Bi)weekly community meetings
Discuss with us live

Chan
Zuckerberg
Initiative

Some information about packaging

"There should be one-- and preferably only one --obvious way to do it."

Tim Peters, The Zen of Python

Definitions

what is she building in there?

- *script* or *notebook*
  - run specific analysis or transformation
  - intended for one/few-time use
  - maybe additional materials for a paper
- *library*
  - something others can build on or integrate with their own code
  - loosely couple with dependencies (write tests!)
- *application*
  - an executable, typically with some user interface (GUI, TUI, CLI)
  - may want a plugin system, bundled app, etc.

Write portable code

great news for Python users

- Pick a minimum Python version to support
- Consider dependencies (portability, availability, licensing)
- Don't assume things about the file system
- GUIs can be tricky
  - Consider writing a plugin (*napari* or some other application)
  - Use abstractions (pyapp-kit, qtpy)
  - Consider web-based technologies

Managing dependencies

use pip or conda (+ also pip)

pip is kind of the standard, comes with Python

- mostly Python packages mostly from the Python Package Index (PyPI)
- good support for binary packages (*wheels*) these days
- more prevalent in broader Python community

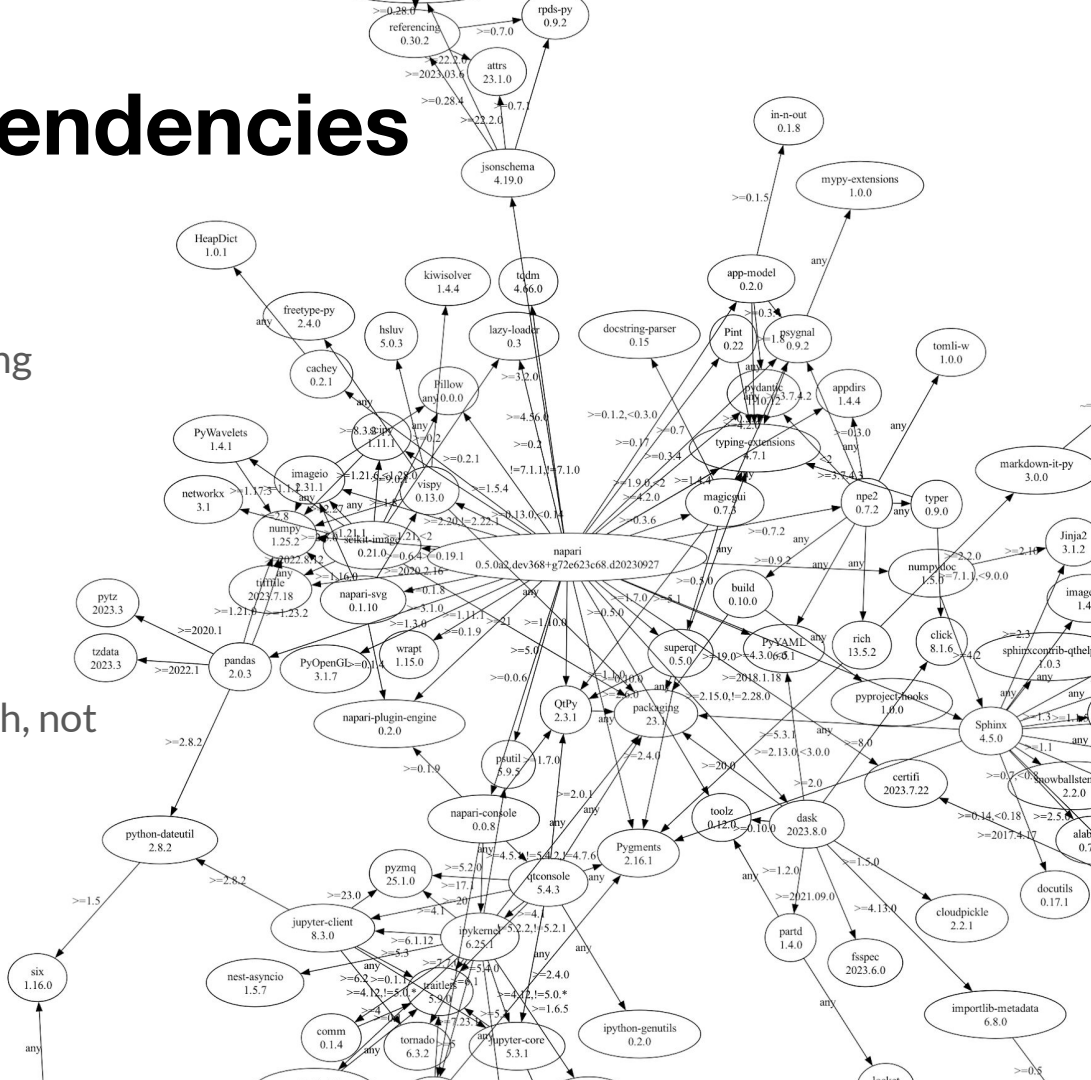
conda solves more problems, more focus on scientific computing

- not just Python packages
- can distribute "system" packages (OpenSSL) and complicated dependencies (CUDA)
- *conda-forge* - orchestrates coordinated compilation with shared tooling (and it works!)

Keep track of dependencies

to pin or not to pin

- Definitely use virtual environments, using *venv* or *conda*
- Create a list of your dependencies
  - *pip freeze > requirements.txt*
  - *conda env export > environment.yml*
- Sometimes too strict or not strict enough, not cross-platform; tools can help
  - pip-tools
  - conda-lock



What about Docker?

ship your code and all its dependencies together

- Generally not practical for libraries, not great for GUIs
- Still requires a learning curve
- Performance penalty on macOS and Windows
- Accessing hardware acceleration (GPU) can be difficult
- Images can be pretty big
- Sometimes still be a reasonable choice!
  - Web apps (or apps designed to run in the cloud, possibly with web-based GUI)
  - Certain headless applications with complex dependencies
  - Immutable once baked, may be good as demo or appendix to a paper

What makes a package?

it's all about the metadata

- In the end, a package is usually just an archive (you can unzip a wheel)
- Essential metadata
  - package name
  - version
  - [dependencies (+version pins)]
- Consider starting from a [cookiecutter template]
- Tooling exists for copying pip packages to conda-forge

pip	conda
pyproject.toml setup.cfg setup.py MANIFEST.in this is usually all you need now	**recipe/** **meta.yaml** build.sh (macOS, Linux) bld.bat (Windows)
upload to PyPI (website, *twine*, ...)	open a PR to conda-forge/staged-recipes "feedstock" repository contains your recipe

cibuildwheel

packaging compiled extension modules into wheels

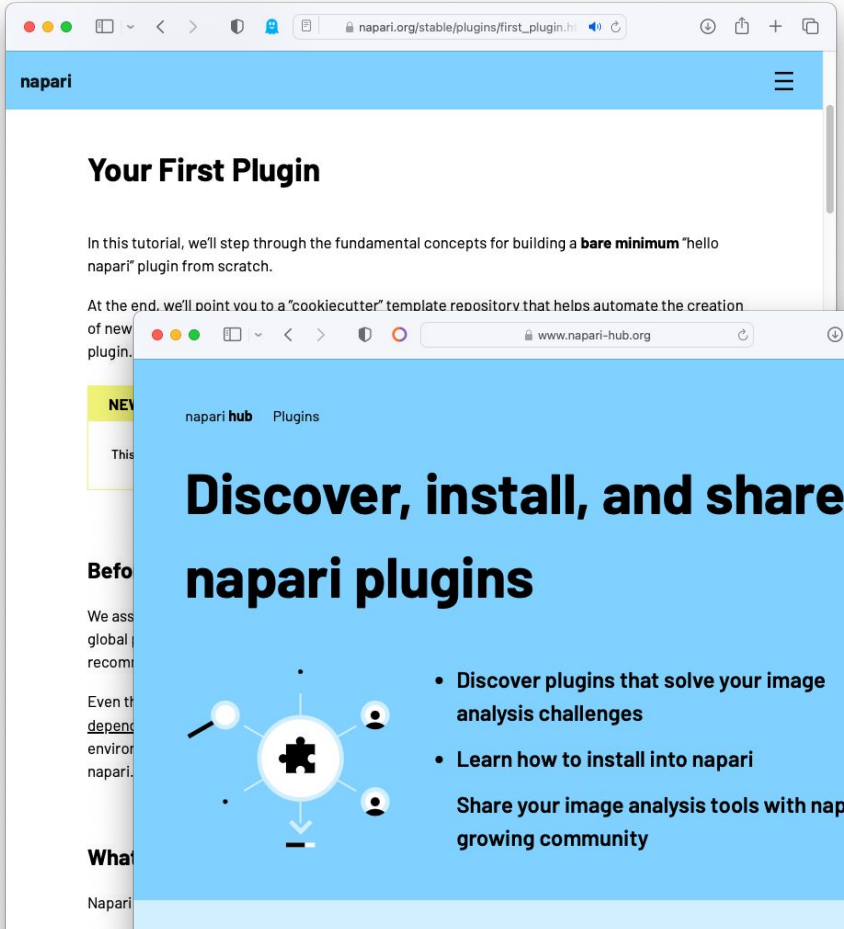
- Use Docker to build, not distribute
  - create wheels with *manylinux* to be portable across, well *many* Linux distros
  - *manylinux* is just an old version of Linux in a Docker container, takes advantage of ABI stability of *glibc* for compiled C-extensions
- Dynamic libraries are copied into the wheel with relative paths

- Select a build backend based on how the code is compiled
  - scikit-build (CMake)
  - maturin (Rust/PyO3)
  - meson-python (Meson)

napari plugins

they're just Python packages

- Installed into a virtual environment with napari via pip, conda, or napari plugin manager (GUI)
- napari uses *npe2* to parse metadata for:
  - readers/writers
  - widgets
    - magicgui makes this usually very easy
    - use qtpy to be compatible with Qt backends
  - sample data
  - themes
- Start with the cookiecutter template
- Find more best practices in the napari docs



Making a bundled app

easy installation for end-users

- *conda + menuinst + Constructor*
- Creates platform-specific installers for macOS, Linux, and Windows (*Constructor)*
- Installs a *miniconda* distribution
- Creates "shortcuts" to run apps (*menuinst*)
  - Windows Start Menu
  - Linux .desktop file
  - macOS .app bundle