



DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE

Bio-Image Data Science

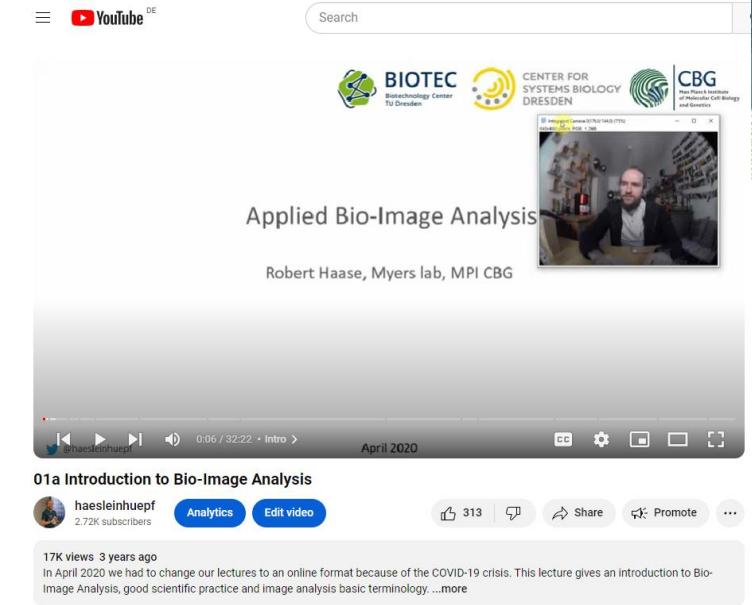
Robert Haase



Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

Hello my name is...

- Robert Haase
- Applied image data scientist, research software engineer
- Trained computer scientist (2010 Dipl-Inf(FH)) with a medical PhD (2016 Dr. rer. medic.)
- Worked 15 years on the biomedical research campus in Dresden-Johannstadt (Hospital, MPI-CBG, TU-Dresden)
- Teaching bio-image analysis to life scientists 2019-2023 @ TU Dresden, since 2024 @ Uni Leipzig
- I maintain about 50 Python and Java packages other people use to analyze microscopy image data (GPU-acceleration, machine learning, large language models)



Large parts of this
online lecture are
outdated!

Survey

What do you study?

MSc Data
Science

MSc Bio-
informatics

MSc Medical
Informatics

Something
else

Lecture overview

- Every week will follow this scheme
 - 9:15 : 90 min lecture
 - 11:15 : 90 min exercises (programming intense)
 - when you're done, enjoy the sun!
- Exam will cover the semester content accordingly
 - Bio-image Analysis / Microscopy
 - Machine Learning / Deep Learning / Large Language Models
 - Research Data / Software Management
 - “closed book exam” about lecture contents and exercises

Lecture materials

Training resources for Students at Uni Leipzig who want to dive into bio-image data science with Python. The material will develop between April and July 2024

CC-BY-4.0 license
1 star 0 forks 2 watching 1 Branch 0 Tags Activity Public repository

haesleinhuepf added first exercise
01a_setting_up_local_environment added first exercise
01b_setting_up_sc_ulei_environment added first exercise
01c_testing_environment added first exercise
.gitignore
LICENSE-CC-BY
README.md

Slides commonly available in advance

Bio-image Data Science

This repository contains training resources for Students at Uni Leipzig who want to dive into bio-image data science with Python. The material will develop between April and July 2024 and shared here in this github repository.

Teaching Goal

Students learn the full workflow of common bio-image data science projects to a degree that they can execute a scientific data analysis project in this context on their own. They will be familiar with common bio-image analysis algorithms and workflows, how to choose them according to a scientific goal, and how to measure quality of derived results. Attending the lecture and executing the practicals qualifies the students to work as bio-image data scientist in the pharmaceutical industry or basic biological research.

Course contents

- Introduction to Bio-image Data Science (Apr 2nd 2024)
 - Basics of microscopy
 - Introduction to Bio-image Analysis
 - Exercises:
 - Setting up a local environment
 - Setting up Jupyter Hub at Scientific Computing / Leipzig University
 - Execute the trailer notebook

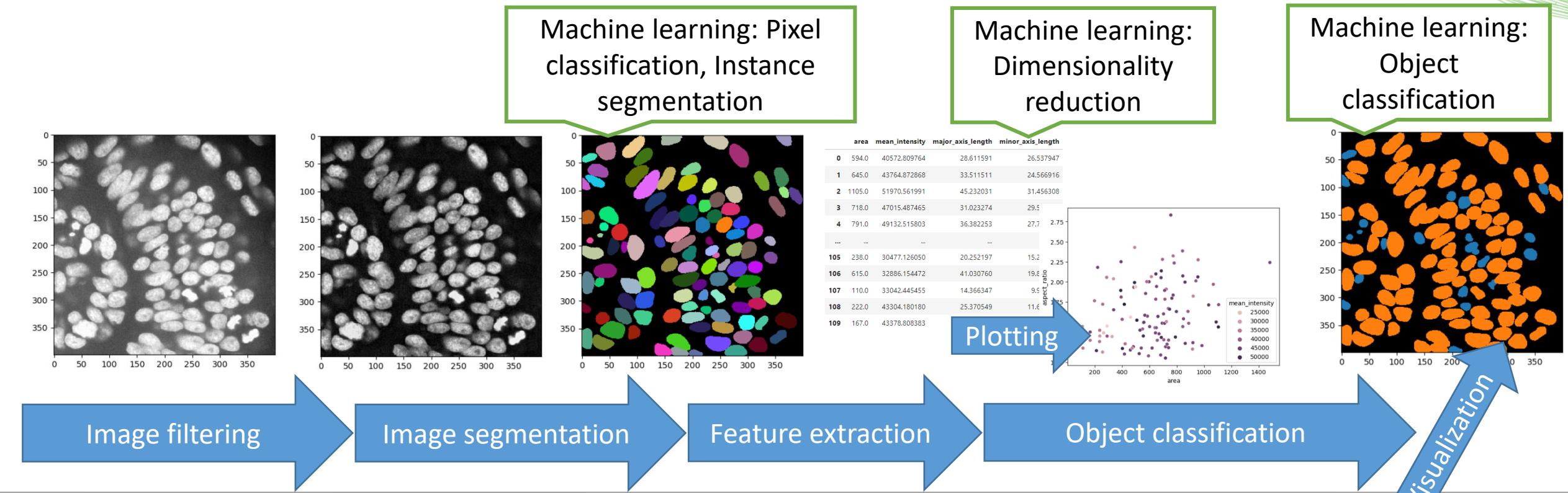


Preliminary schedule

- April 11th 2025 – Introduction Microscopy, Bioimage Analysis, Research Software Management
 - ~~April 18th 2025 – Good Friday (Karfreitag)~~
 - April 25th 2025 - Microscopy Image Processing
 - May 2nd 2025 – Segmentation of cells and nuclei
 - May 9th 2025 – Feature Extraction, Data Viz, Quality Assurance
 - May 16th 2025 – Big data, parallel processing & tiled image computing
 - May 23th 2025 – Introduction to Machine Learning for bio-image analysis
 - May 30st 2025 – Deep Learning for image denoising + segmentation
 - June 6th 2025 – Large Language Models and Prompt Engineering
 - [June 13th 2025 – DevOps Hands-on tutorial @ Data Week, optional <https://www.mathcs.uni-leipzig.de/veranstaltungsdetail/termin/data-week-leipzig-2025>]
 - June 20th 2025 – Large Language Models for Code Generation
 - June 27th 2025 – Vision Language Models and Agents
 - July 4th 2025 – Research Data Management
 - July 11nd 2024 – Summary, exam preparation
- Handout exam pre-requisite
complex exercise
- Submission deadline
complex exercise
- 

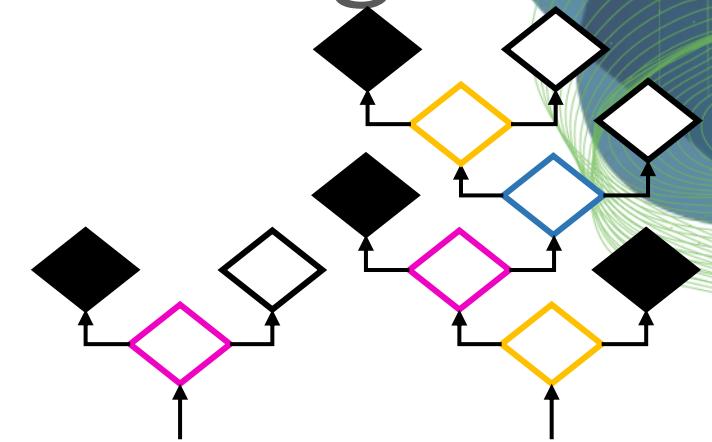
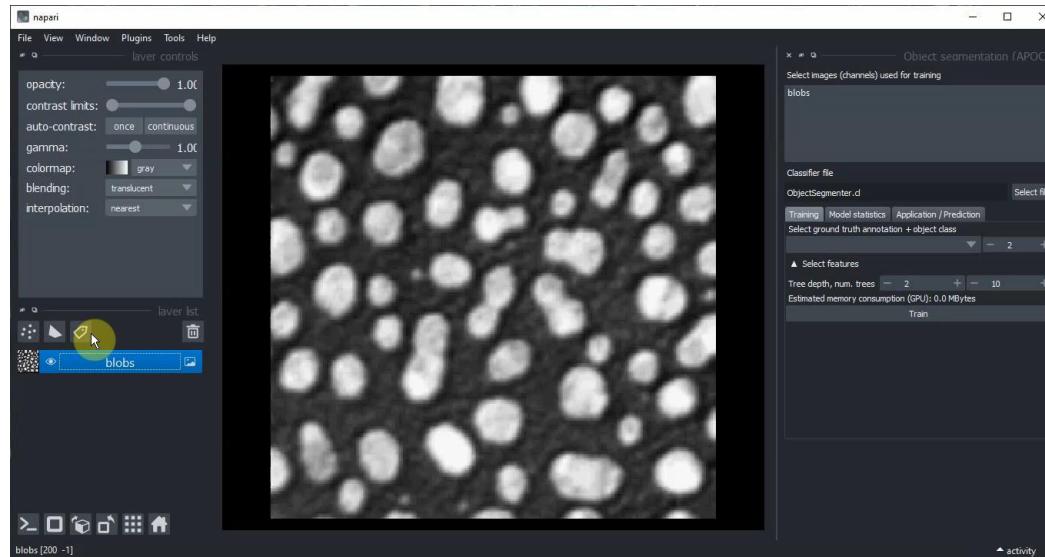
Lecture overview: Bio-image Analysis

- Image Data Analysis workflows
- Goal: **Quantify observations, substantiate conclusions with numbers**

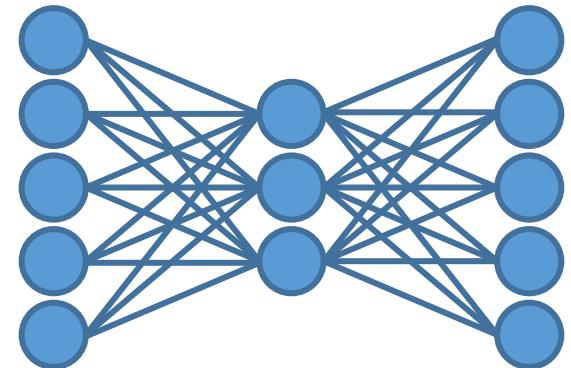


Lecture overview: Machine learning

- Machine learning
 - in the context of bioimage analysis
- Computers can *learn* from data, potentially revealing relationships that are not obvious to humans
- Goal: **Give you an insight into state-of-the-art methods**



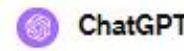
Random forest classifiers



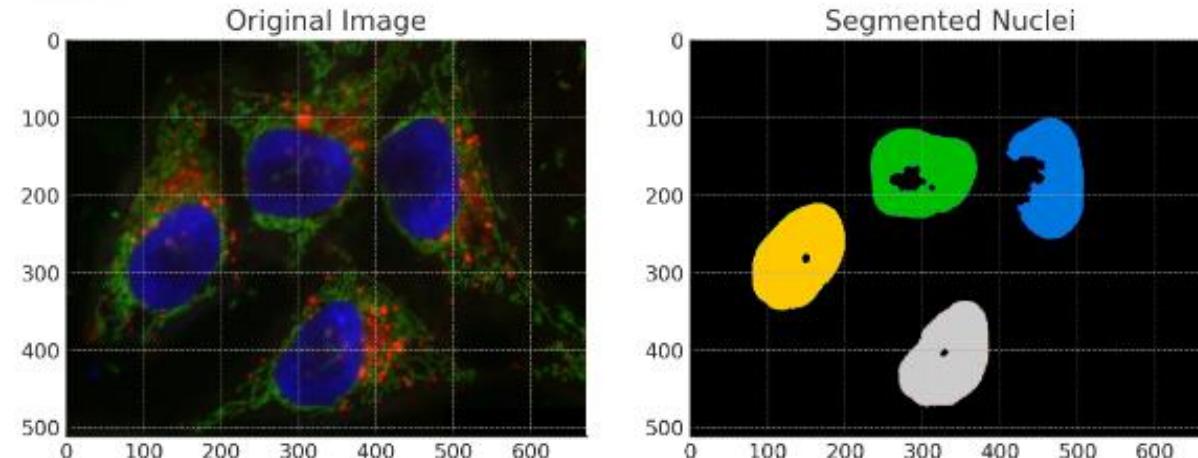
Neural networks

Lecture Overview: Large Language Models

Prompt:
“Please segment
the nuclei in the
blue channel of
the image”



ChatGPT



The segmentation of the nuclei has been performed on the blue channel of the image. The right side of the figure shows the segmented nuclei in different colors, each corresponding to a different nucleus in the field of view. Each color represents a unique label assigned to the individual nuclei detected in the image.

```
Analysis

from skimage import io, filters, measure, morphology
import matplotlib.pyplot as plt
import numpy as np

# Load the image
image = io.imread('mnt/data/hela-cells-8bit.png')

# Since it's a color image, we'll just take the blue channel
blue_channel = image[:, :, 2]

# Threshold the image to get the binary image for nuclei
thresh = filters.threshold_otsu(blue_channel)
binary = blue_channel > thresh

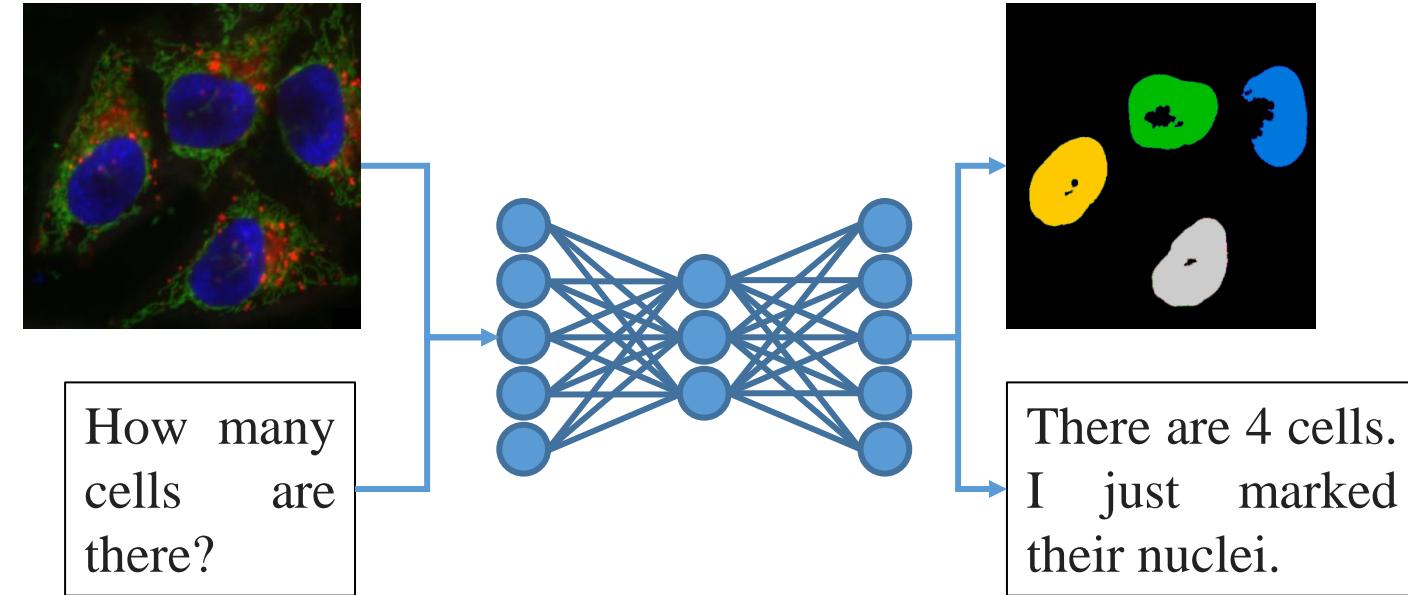
# Remove small objects and perform morphological closing to
cleaned = morphology.remove_small_objects(binary, 50)
closed = morphology.closing(cleaned, morphology.disk(3))

# Label the image
label_image = measure.label(closed)

# Plot the original image and the binary image
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
ax[0].imshow(image)
ax[0].set_title('Original Image')
ax[1].imshow(label_image, cmap='nipy_spectral')
ax[1].set_title('Segmented Nuclei')
plt.show()
```

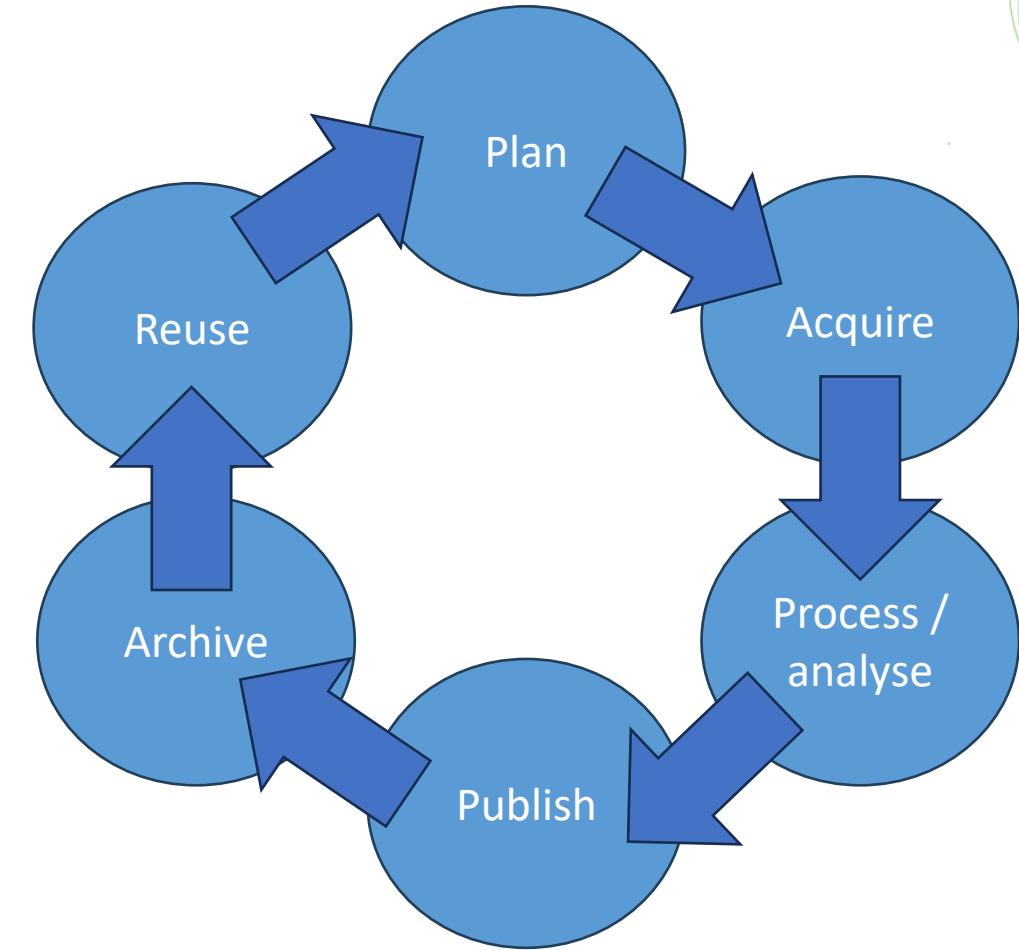
Lecture Overview: Large Language Models

After we learned how analyze images, we will teach an AI-System to do it.



Lecture overview: Research Data Management

- FAIR Principles
- Sharing / licensing
- Open Source code
- Data Management Plans
- Big-Data
- Distributed computing





DRESDEN LEIPZIG

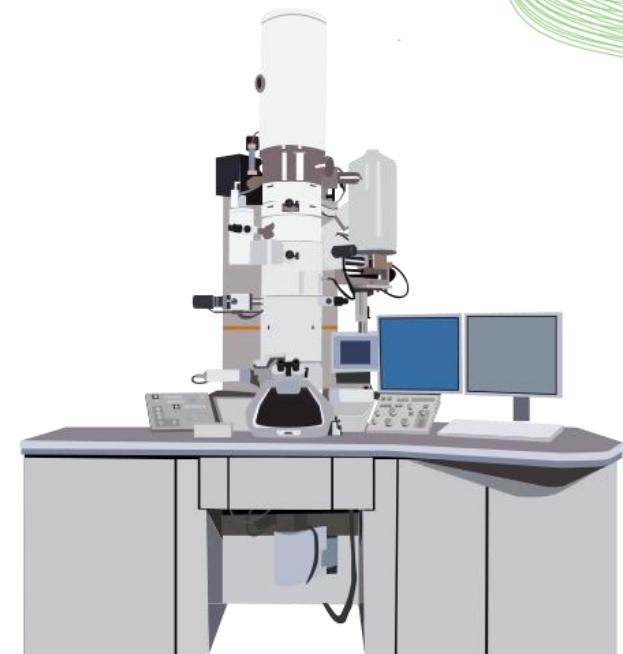
CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE

Basics of Microscopy

Robert Haase

Microscopy

- Common tool to answer biological questions



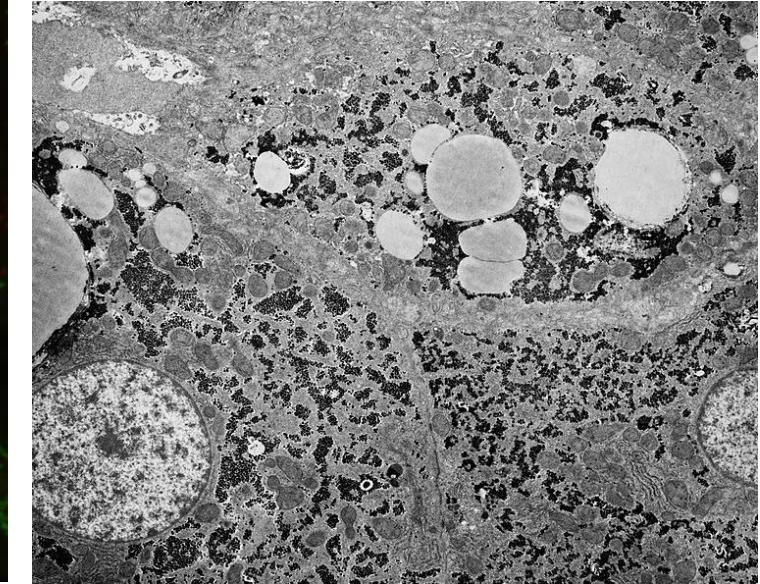
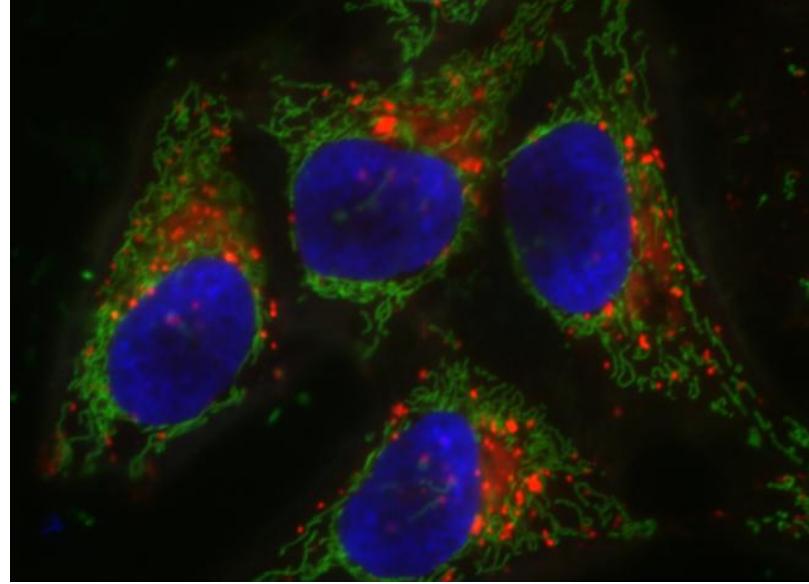
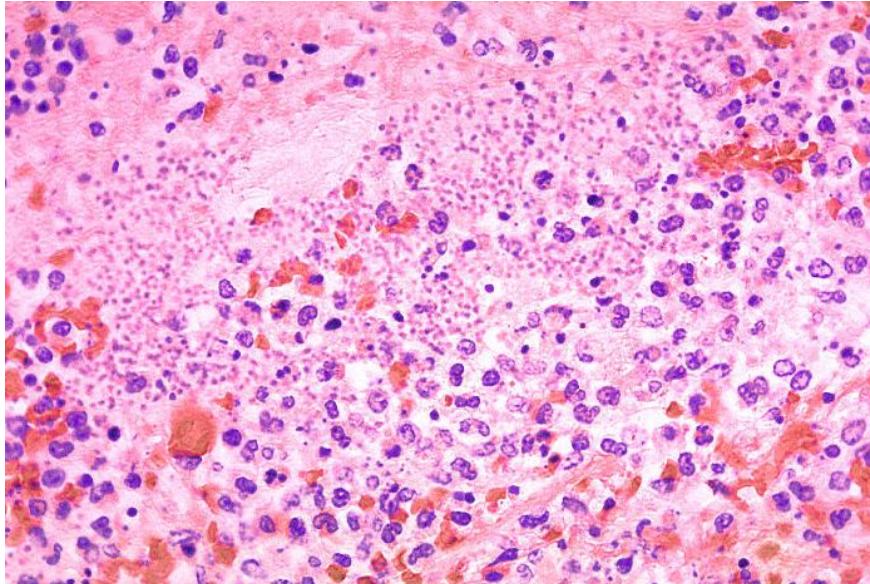
Transmitted light microscope

Fluorescence microscope

Electron microscope

Microscopy

- Common tool to answer biological questions

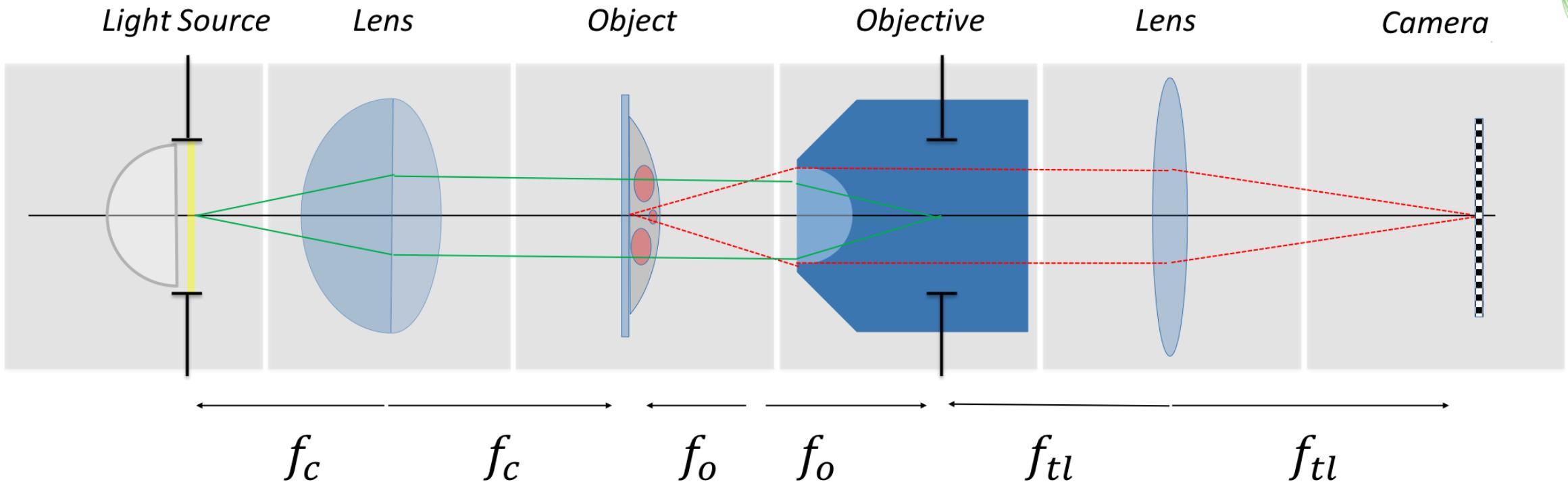


Transmitted light microscope

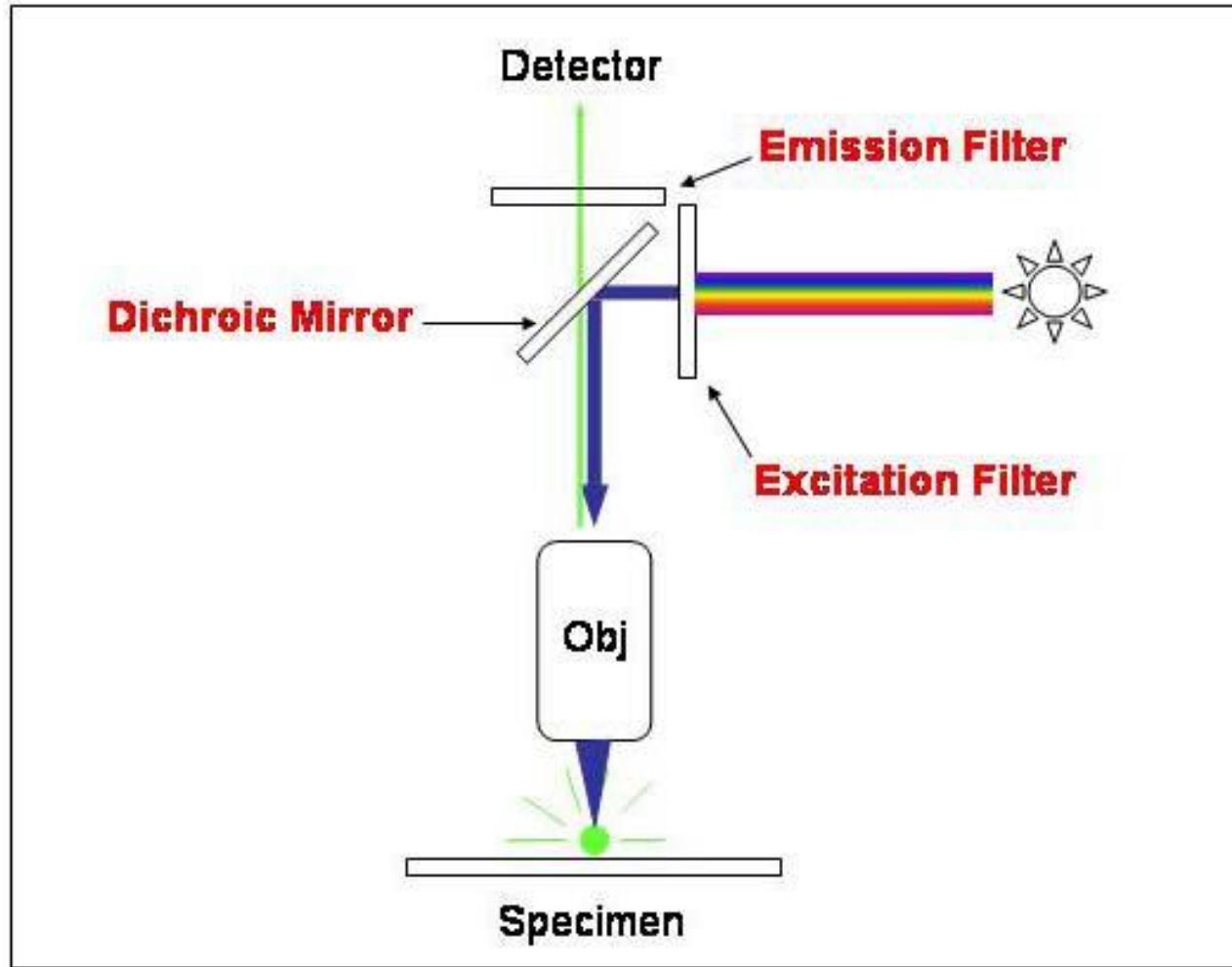
Fluorescence microscope

Electron microscope

Transmitted light microscopy



Fluorescence Microscopy



Fluorescence Microscope

- NOT Fluorescent **microscope**



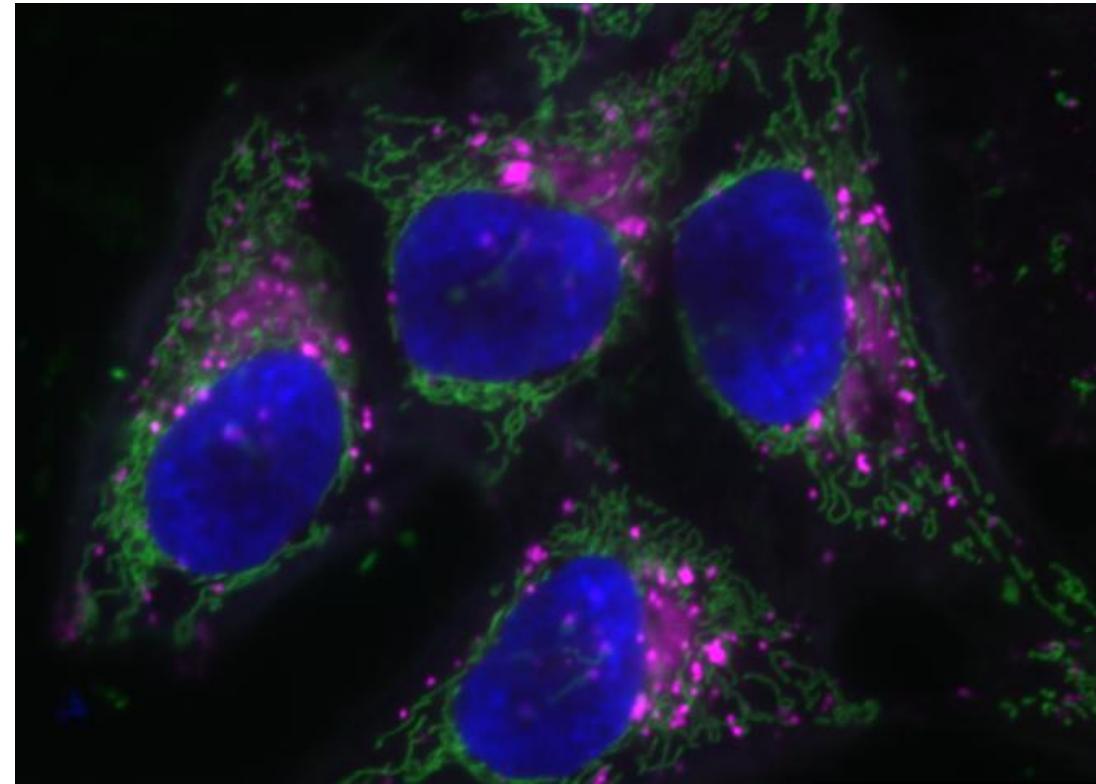
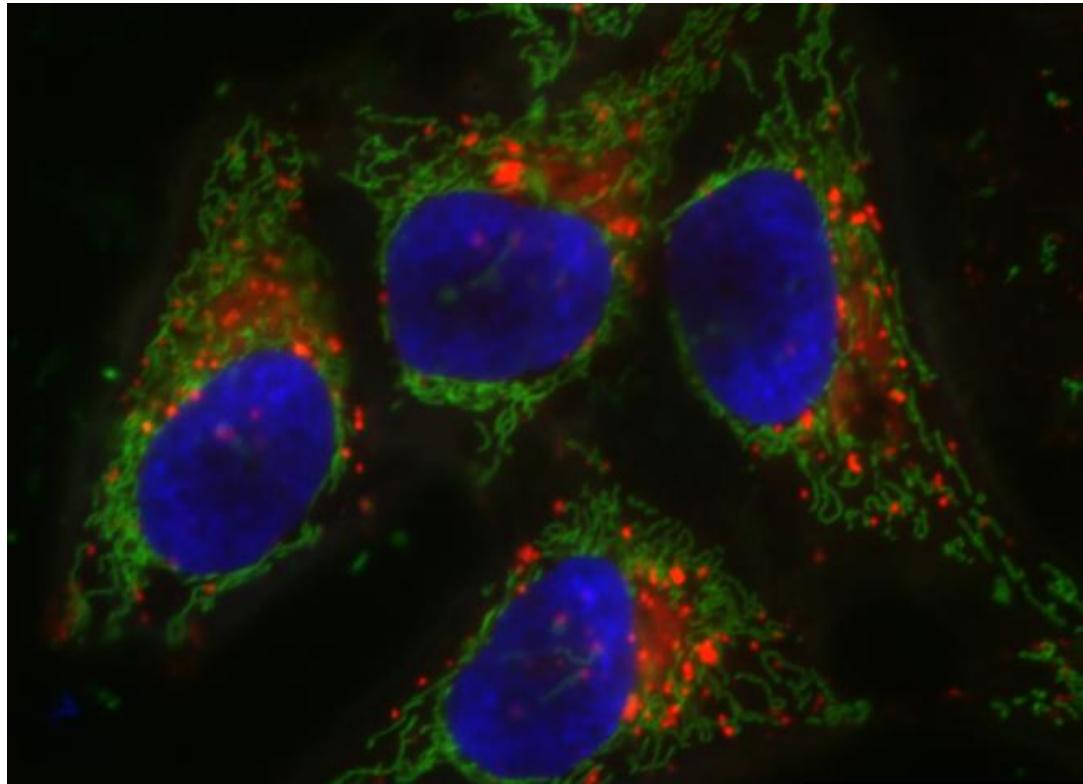
Fluorescence microscope



Fluorescent microscope

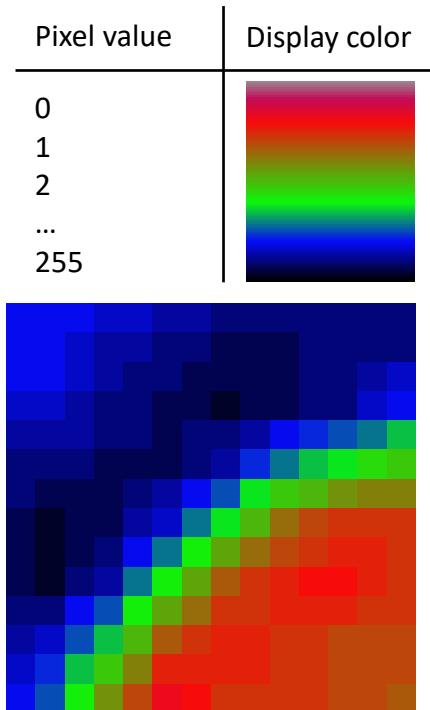
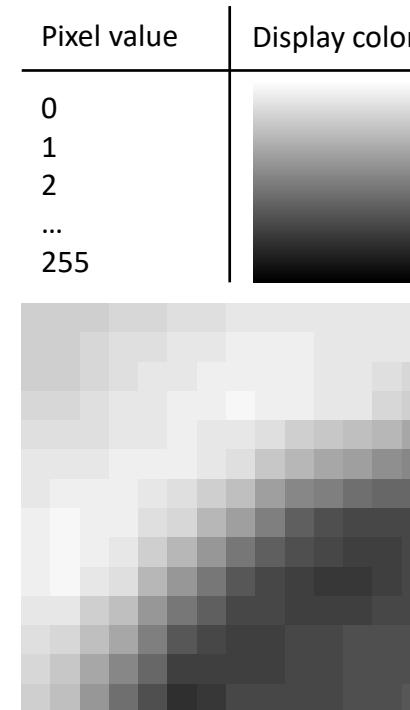
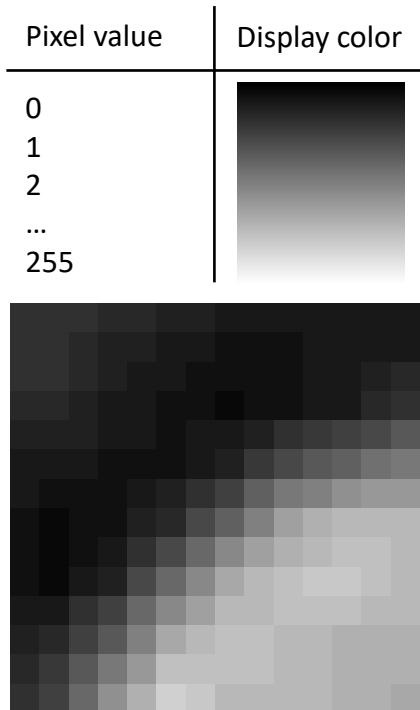
Color maps / lookup tables

- Just because you see something in red on your screen, doesn't mean it was imaged emitting red light.



Color maps / lookup tables

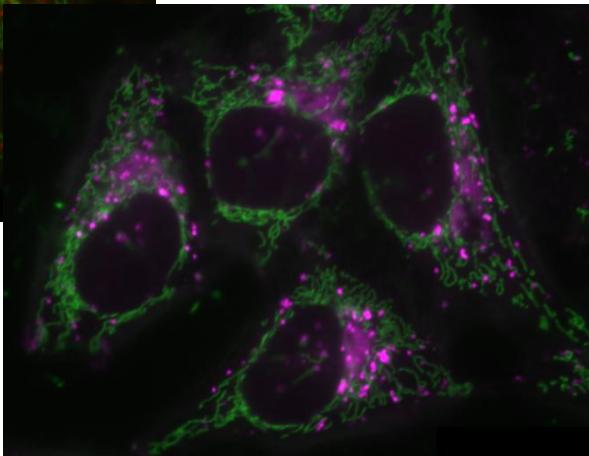
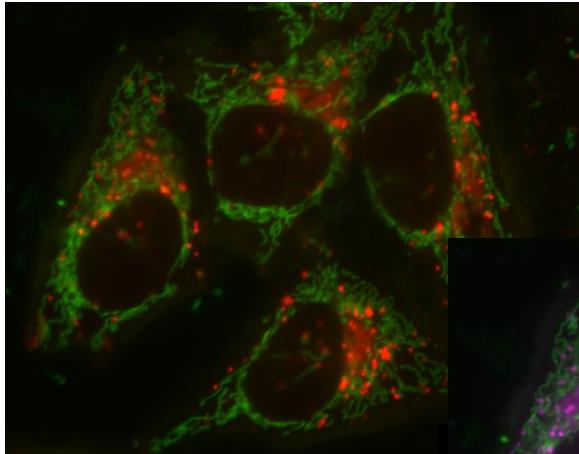
- The lookup table decides how the image is displayed on screen.
- Applying a different lookup table doesn't change the image. All pixel values stay the same, they just appear differently



Color maps / lookup tables

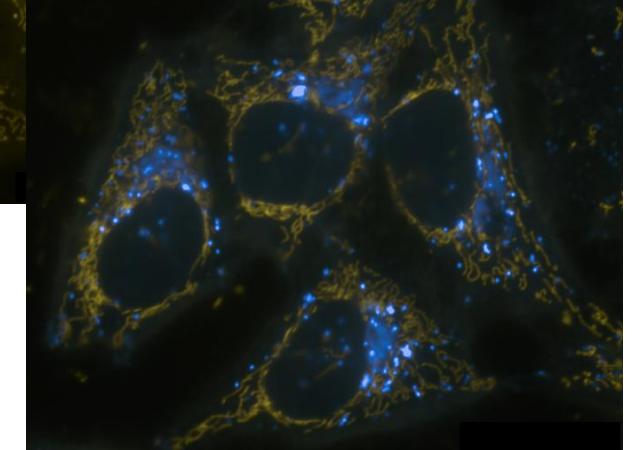
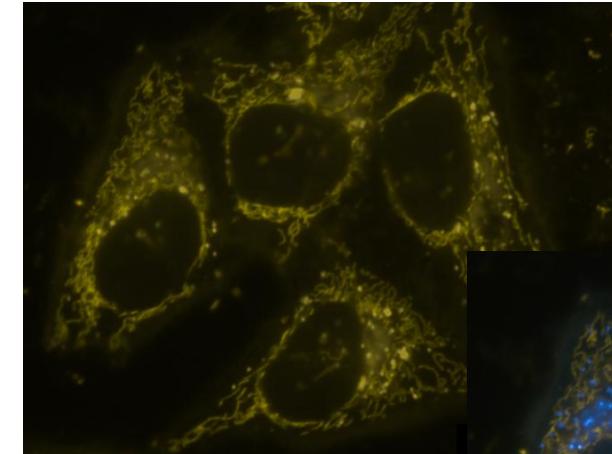
- Choose visualization of your color tables wisely!
- Think of people with red/green blindness!

Default view



Replace red with
magenta!

Red/green blind people see it like this

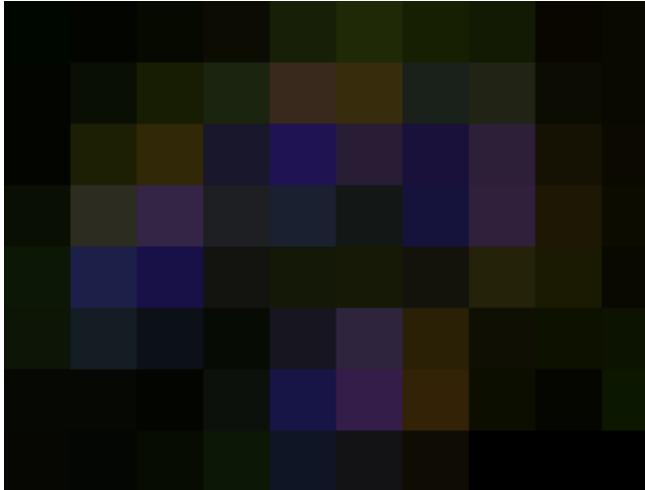


Pixel size versus resolution

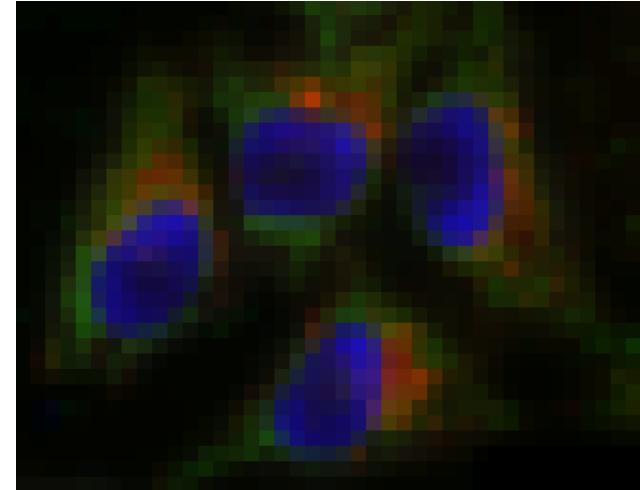
- What is the difference between pixel size and resolution?

Pixel size versus resolution

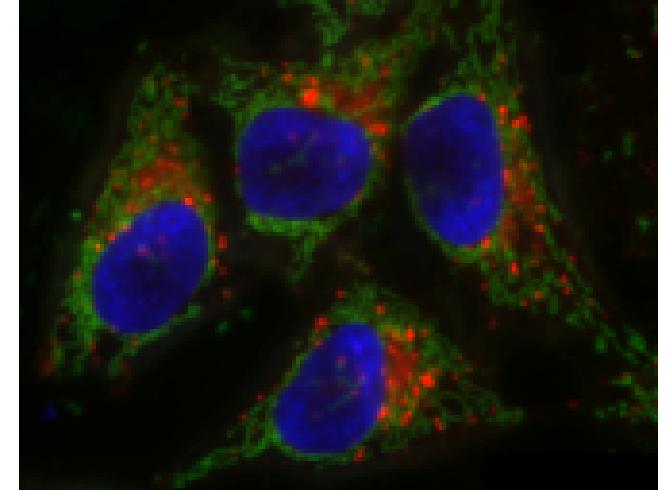
- Pixel size is a property of a digital image.
- You configure it during the imaging session at the microscope.



Pixel size: $3.3 \mu\text{m} \times 3.3 \mu\text{m}$



Pixel size: $0.8 \mu\text{m} \times 0.8 \mu\text{m}$

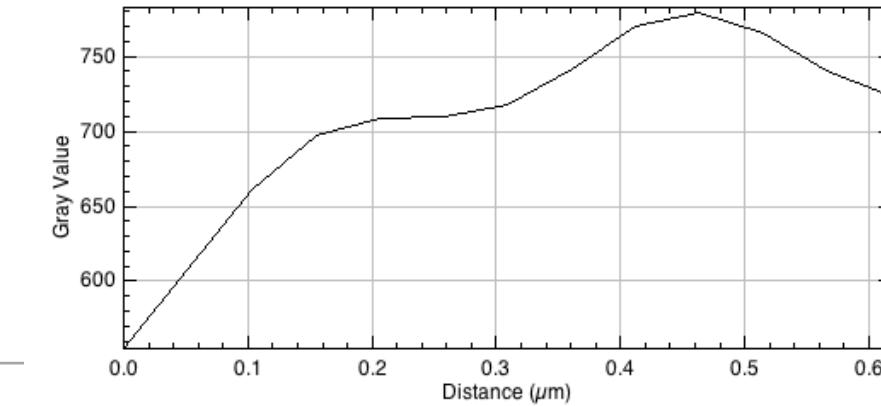
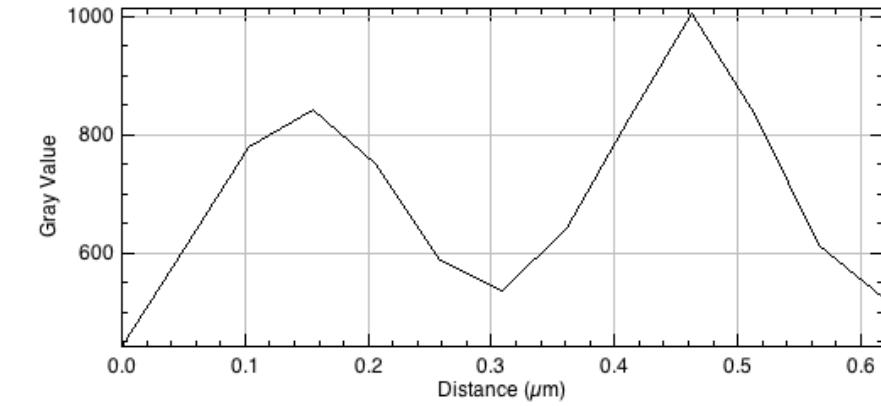
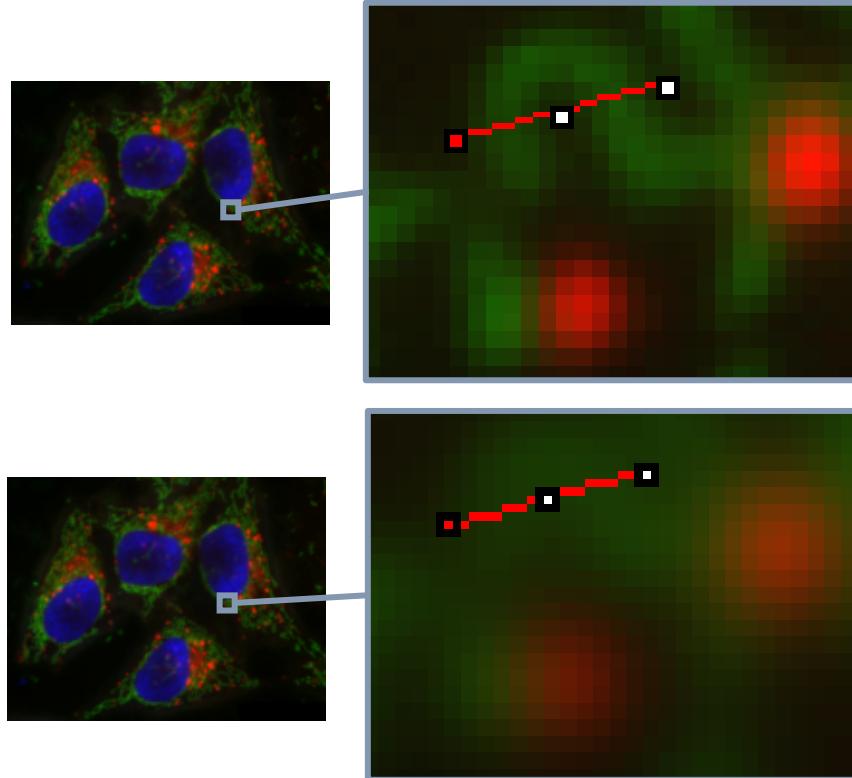


Pixel size: $0.05 \mu\text{m} \times 0.05 \mu\text{m}$

- We are not talking about resolution!

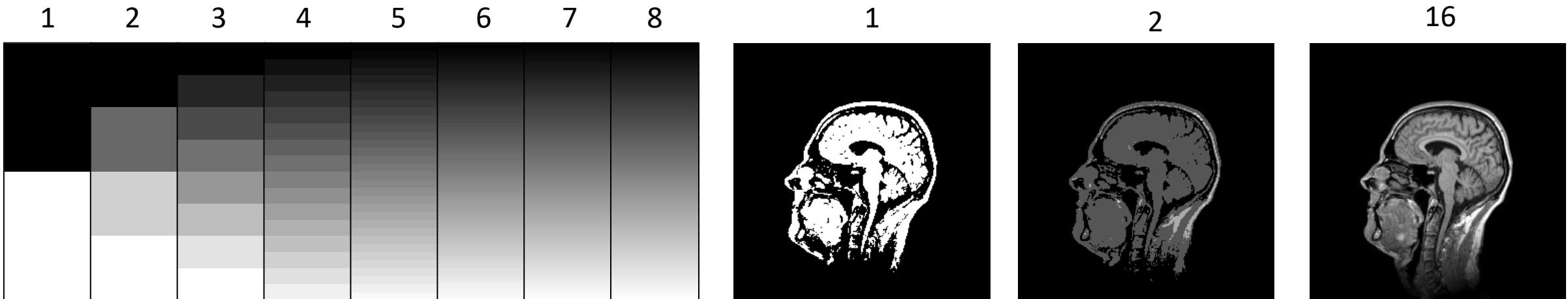
Pixel size versus resolution

- Resolution is a property of your imaging system.
- How small can objects be, to be still differentiable?



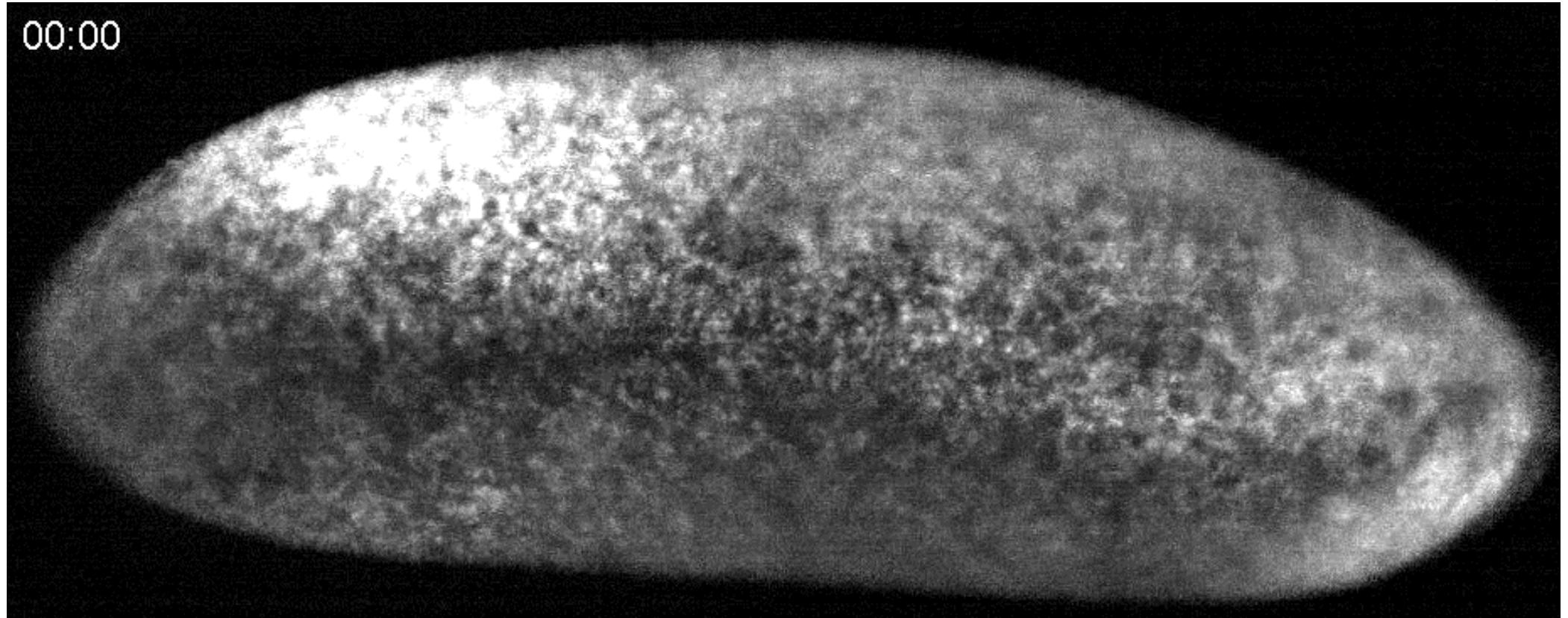
Bit-depth

- A bit is the smallest memory unit in computers, *atomic data*.
- The bit-depth n enumerates how many different intensity values are present in an image:
 - 2^n grey values
- In microscopy, images are usually stored as 8, 12 or 16-bit images.
- In computer vision, 8-bit integer and 32-bit float (range 0...1) is more common



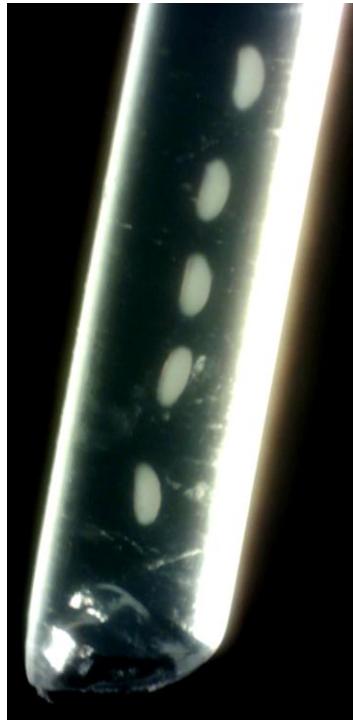
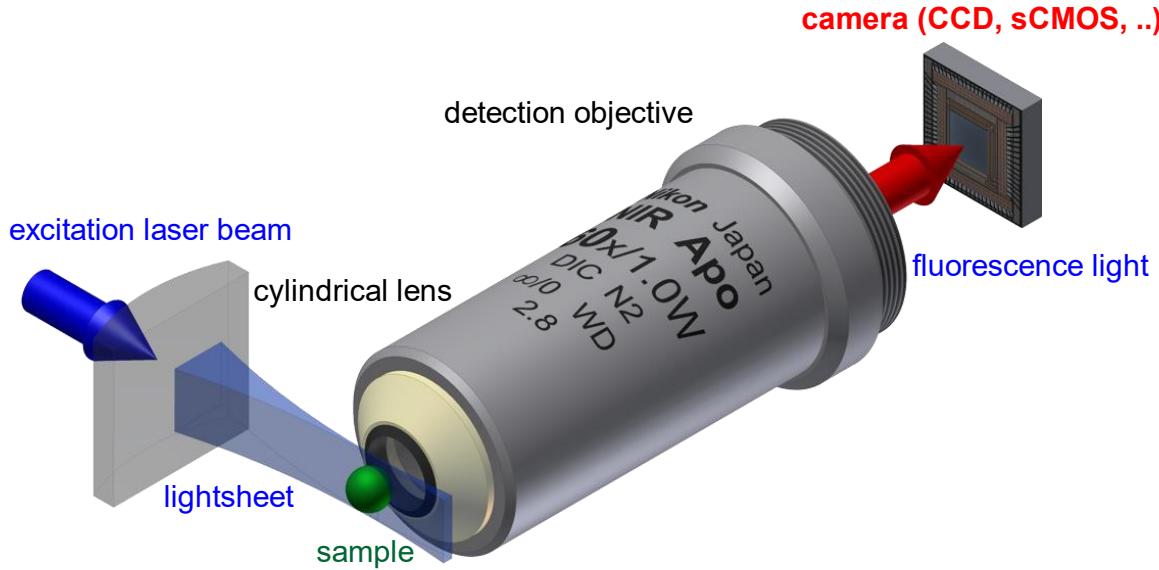
Light-sheet fluorescence microscopy

- Popular tool for large-volume imaging



Light-sheet fluorescence microscopy

- Single-plane illumination microscopy (SPIM)



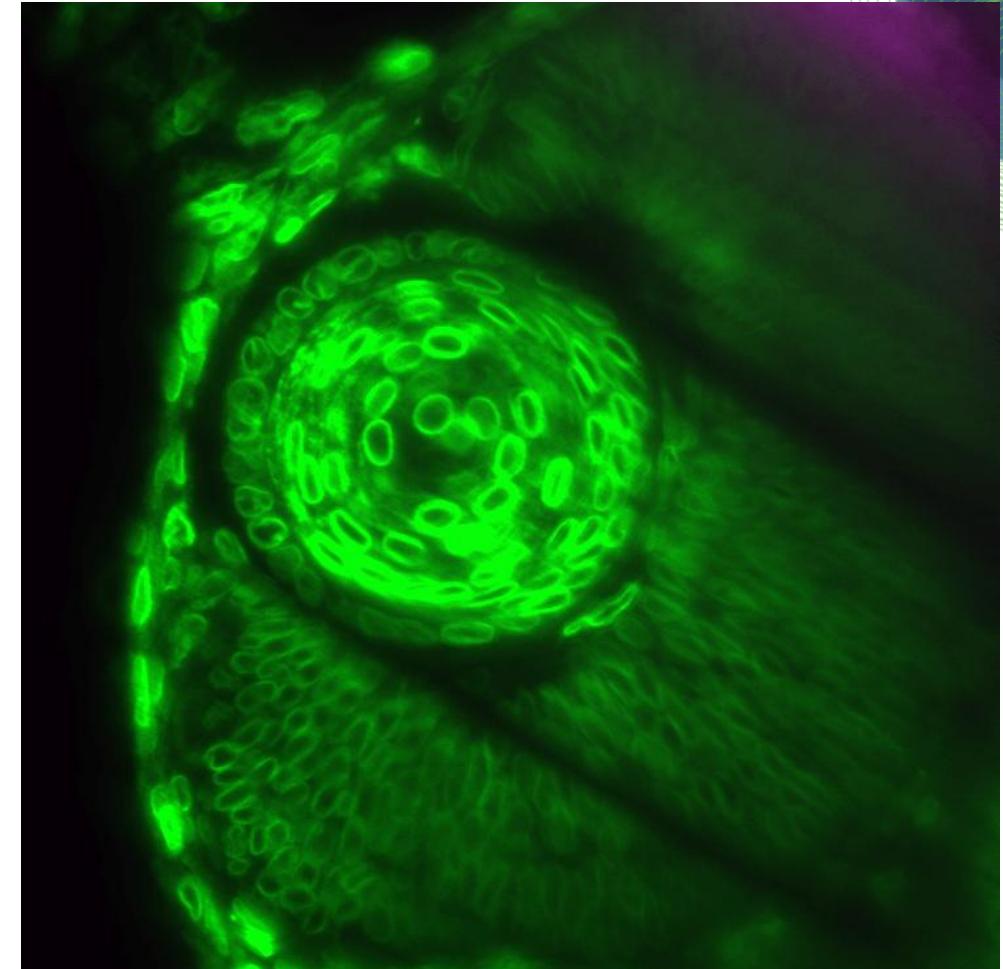
Sample mounting



Plane scanning

Light-sheet fluorescence microscopy

- The beam interacts with the tissue while passing matter
 - Refraction
 - [Reflection]
 - Scattering
- Challenge for quantitative analysis: image quality varies within the image

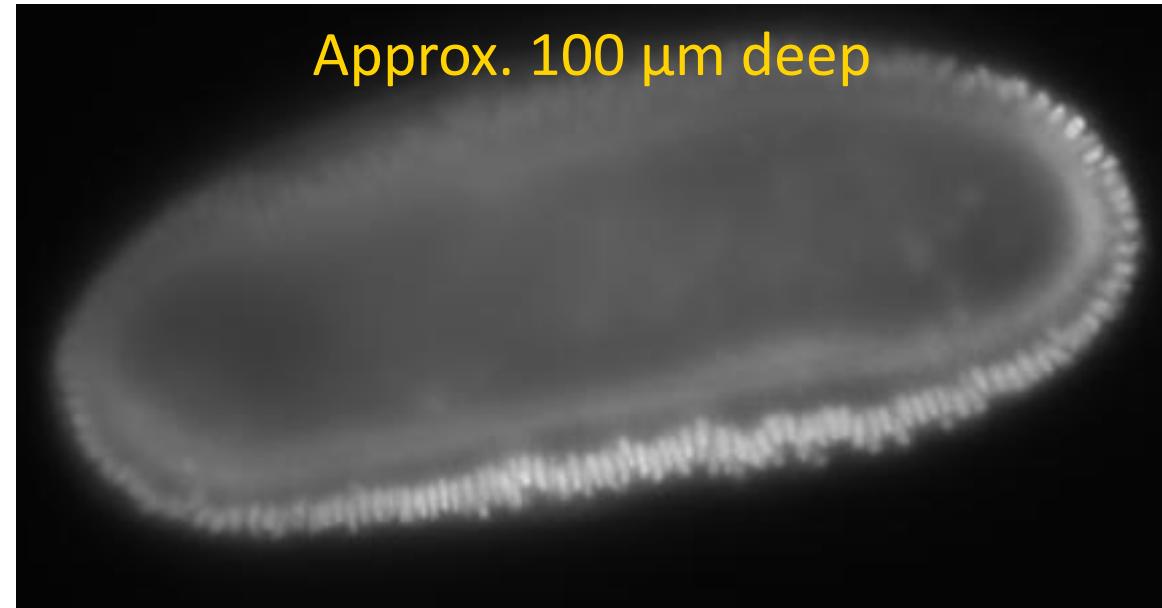
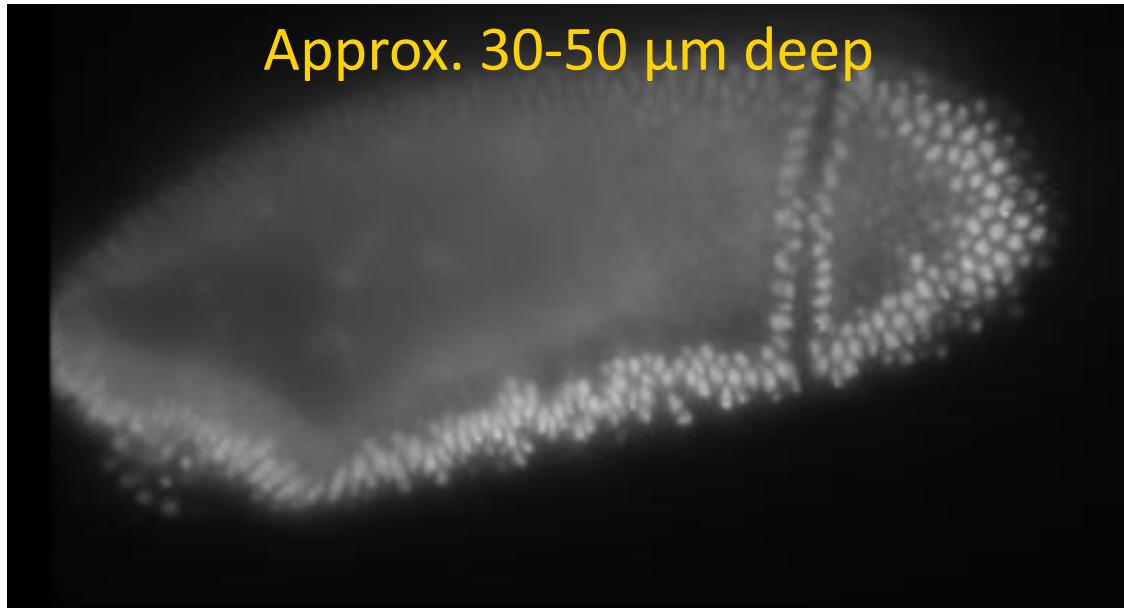
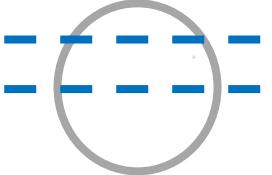


Light-sheet fluorescence microscopy

- These effects depend on the sample

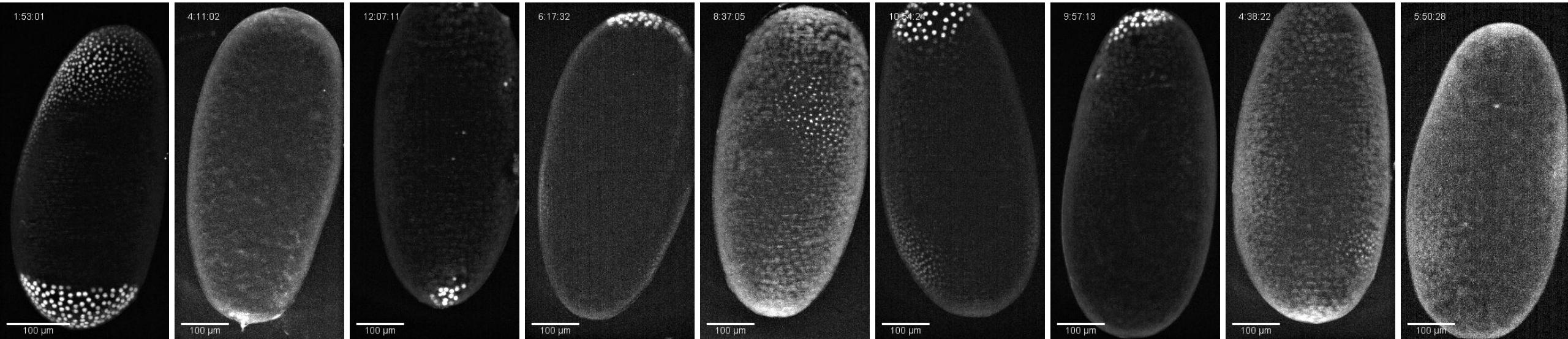


- Big challenge in practice: *Image deep inside* :



High-throughput imaging

- Quality variation between samples, in space and time

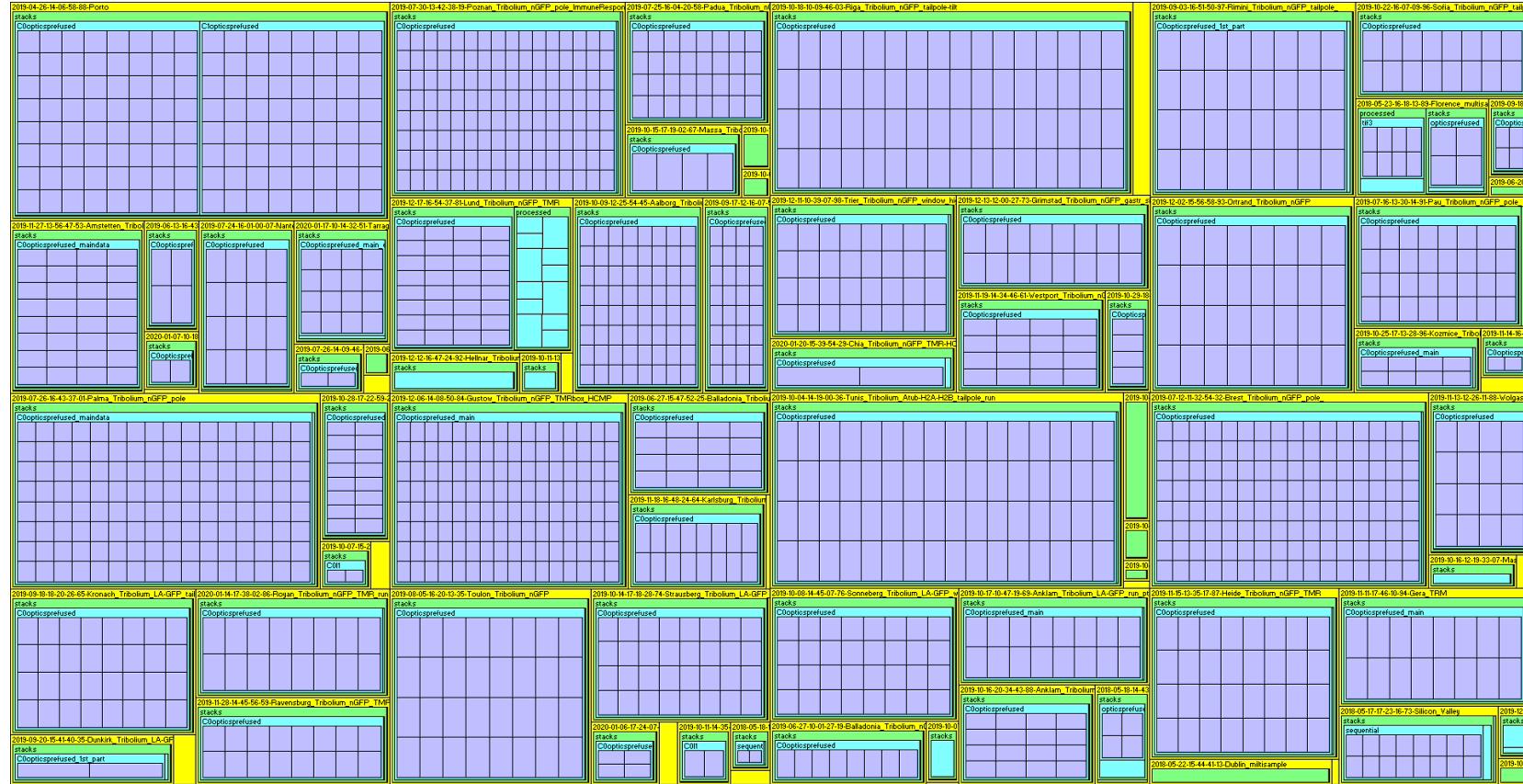


Big Data

- Big Data is a challenge in the data analysis context.
- What do you think is “big” microscopy imaging data?

High-throughput imaging

- “Big data”



Postdoc 2 years imaging
• 35 TB imaging data



Assuming
• one frame is about 200 MB
• assume processing takes 5 minutes per frame



Processing the data would take another 20 months.



DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE

Introduction to Bioimage Analysis

Robert Haase

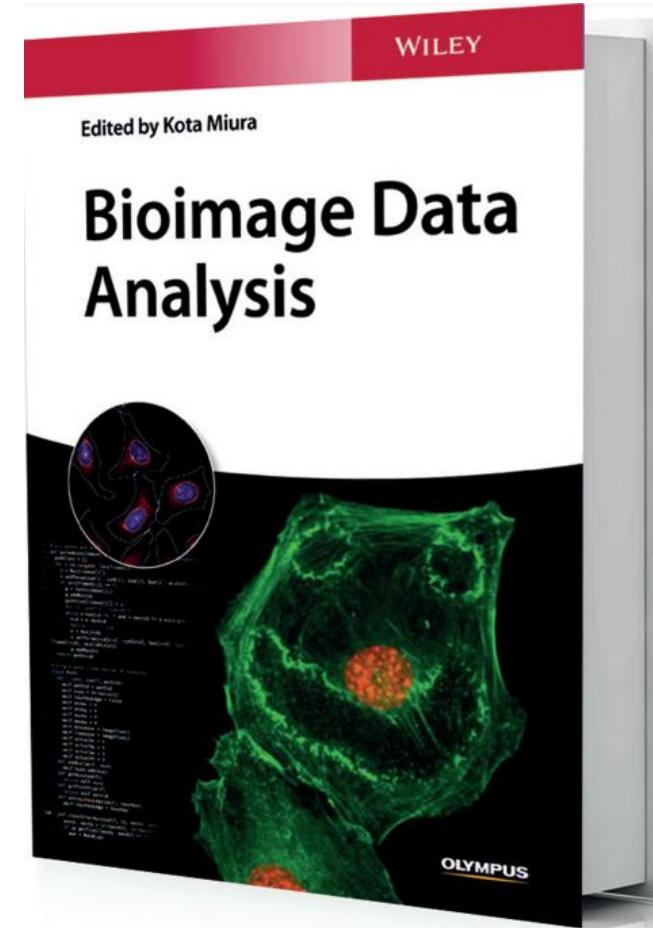


Bioimage Analysis

- Kota Miura & Sébastien Tosi 2015:

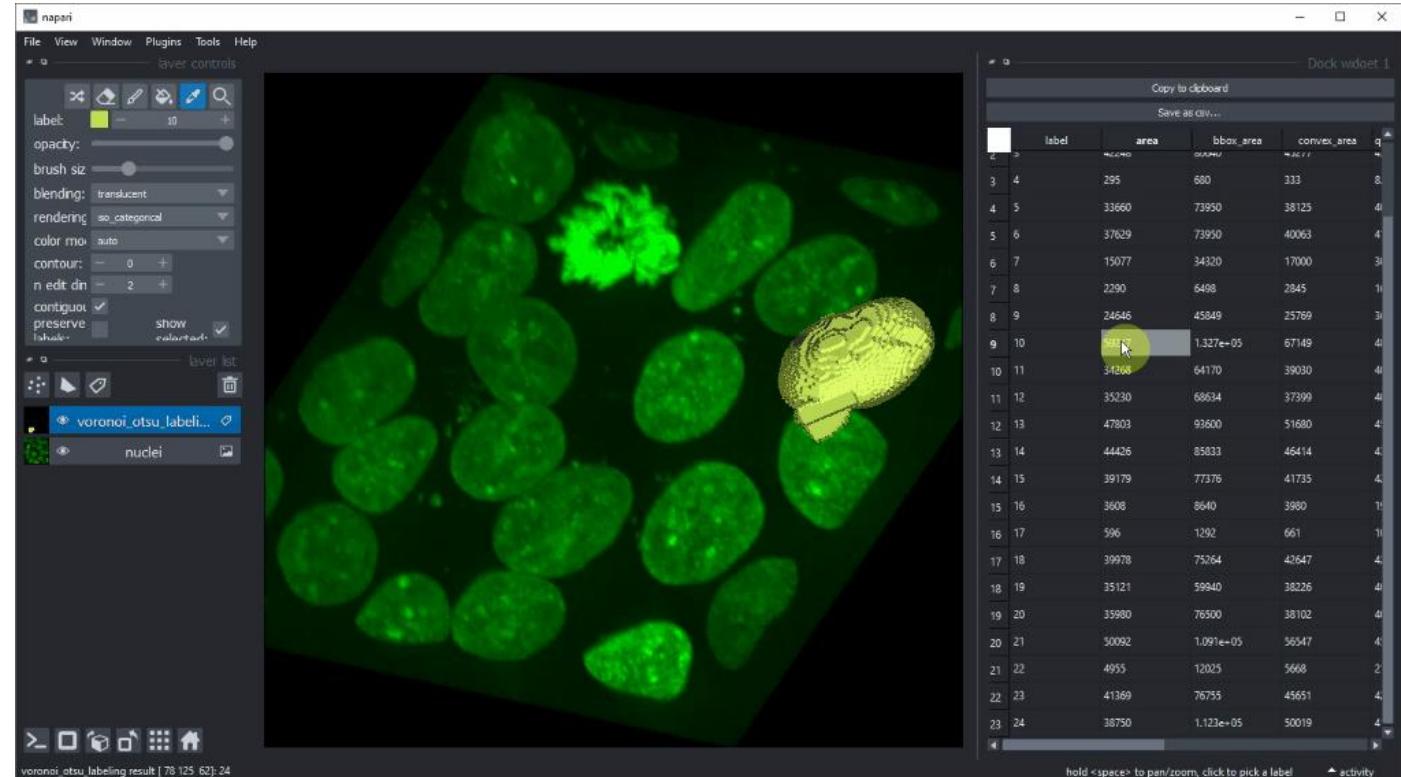
In the light of this definition, image analysis, which is also called “computer vision,” aims at mimicking the way we see the world and how we identify its visible structures. Image analysis in biology does undeniably also hold this element, but more importantly, its main goal is to *measure* biological structures and phenomena in order to study and understand biological systems in a quantitative way.

To achieve this task, we in fact do not have to be bothered with similarity to the human recognition – we have more emphasis on the objectivity of quantitative measurement, rather than how that computer-based recognition becomes in agreement with human recognition. Therefore, in biology, image analysis is a process of identifying spatial distribution of biological components in images and measuring their characteristics to study their underlying mechanisms in an unbiased way. To underline this difference in the goals of image analysis in the two fields and to distinguish them from each other, we will now refer to image analysis in biology as *bioimage analysis*.



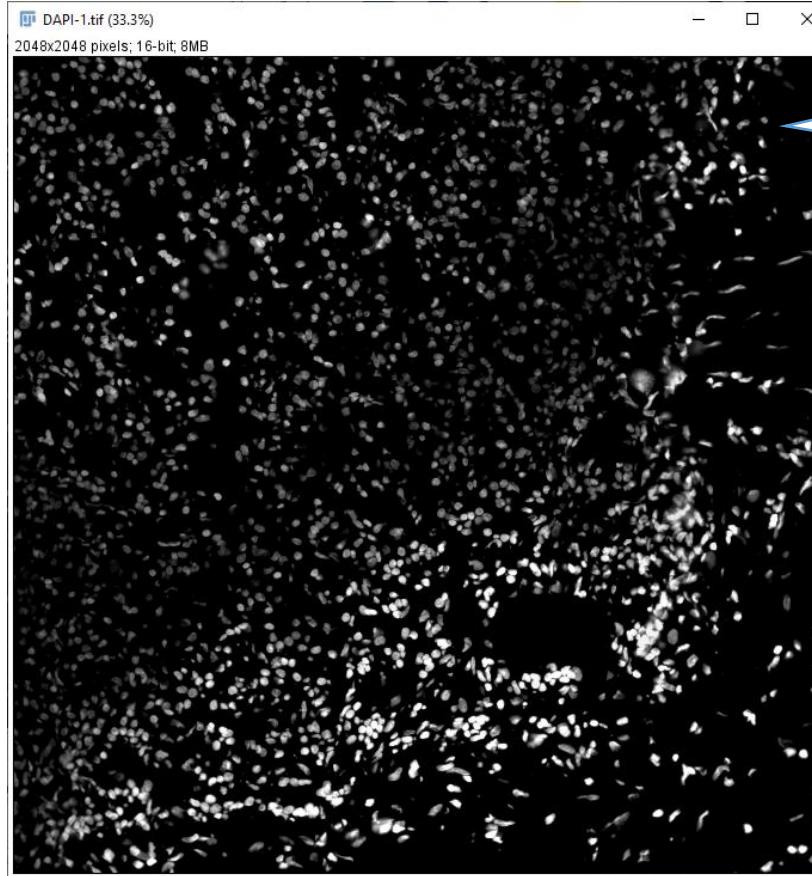
Quantitative bio-image analysis

- Deriving quantitative information from images of biological samples taken with microscopes + visualization



Objective bio-image analysis

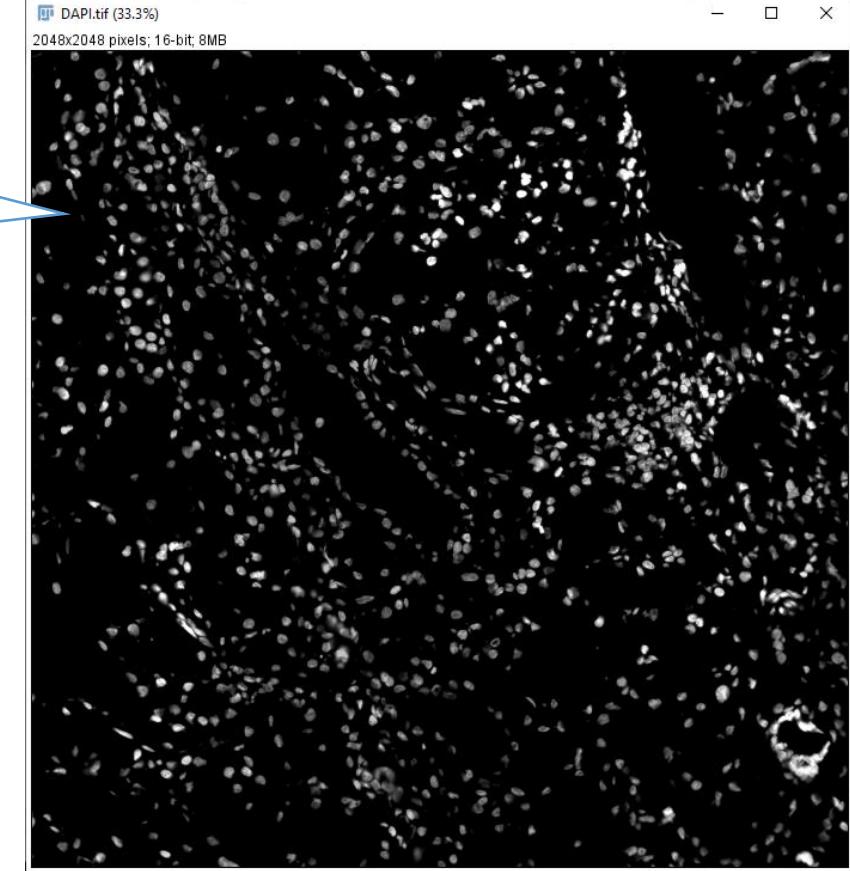
- Measurements should be objective, not influenced by human interpretation



Nuclei in this image are ...

... more dense than in this image.

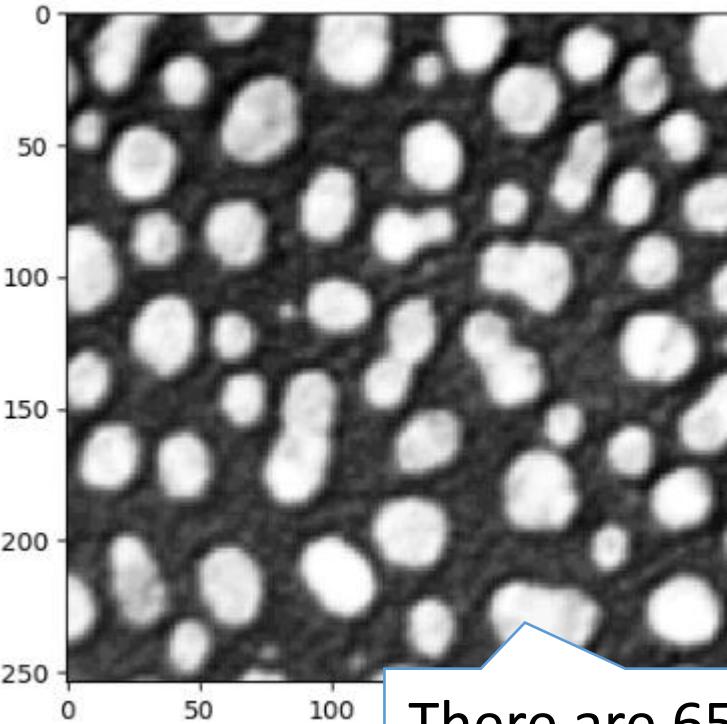
Use automation for less subjective analysis.



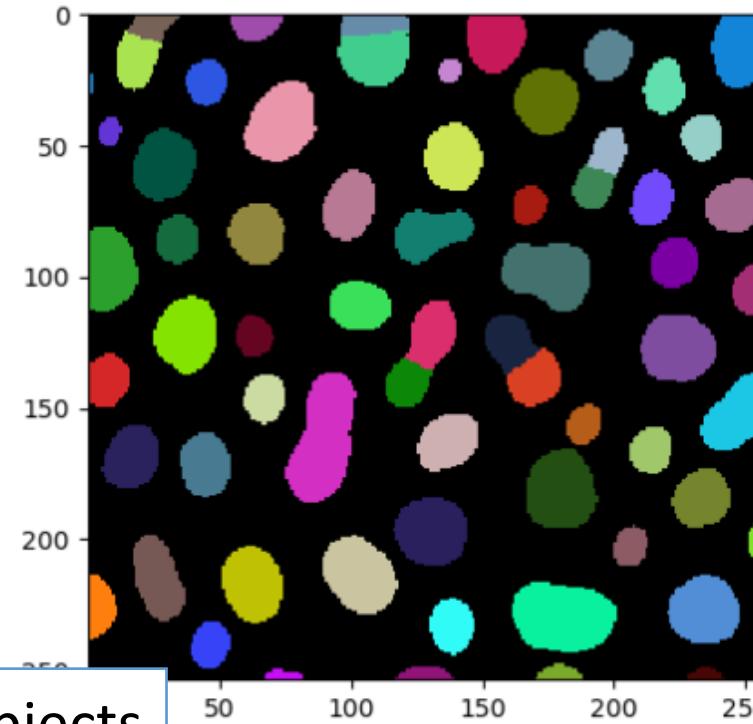
Reliable bio-image analysis

- Algorithms must be reliable (trustworthy).
- Visualization helps gaining trust in automated methods.

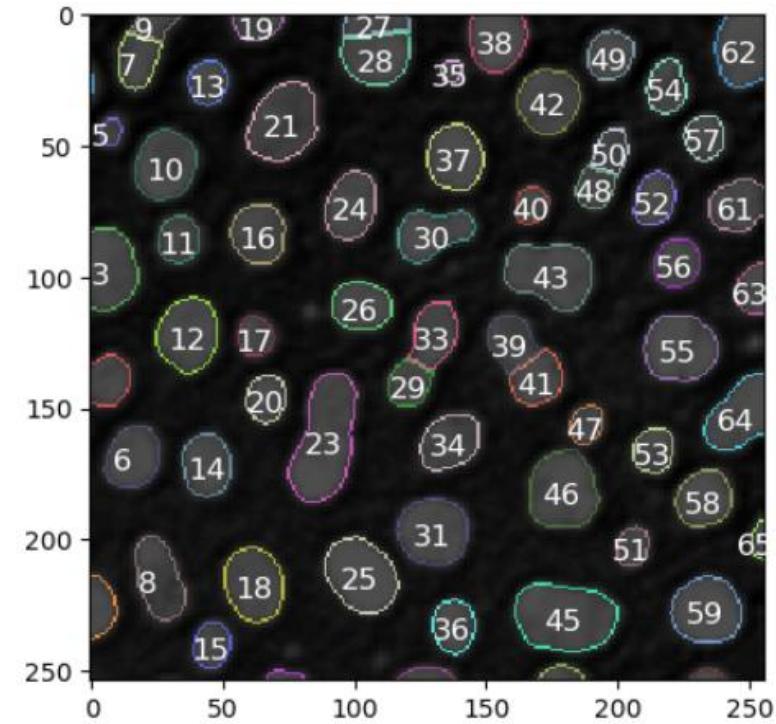
Original image



Label image



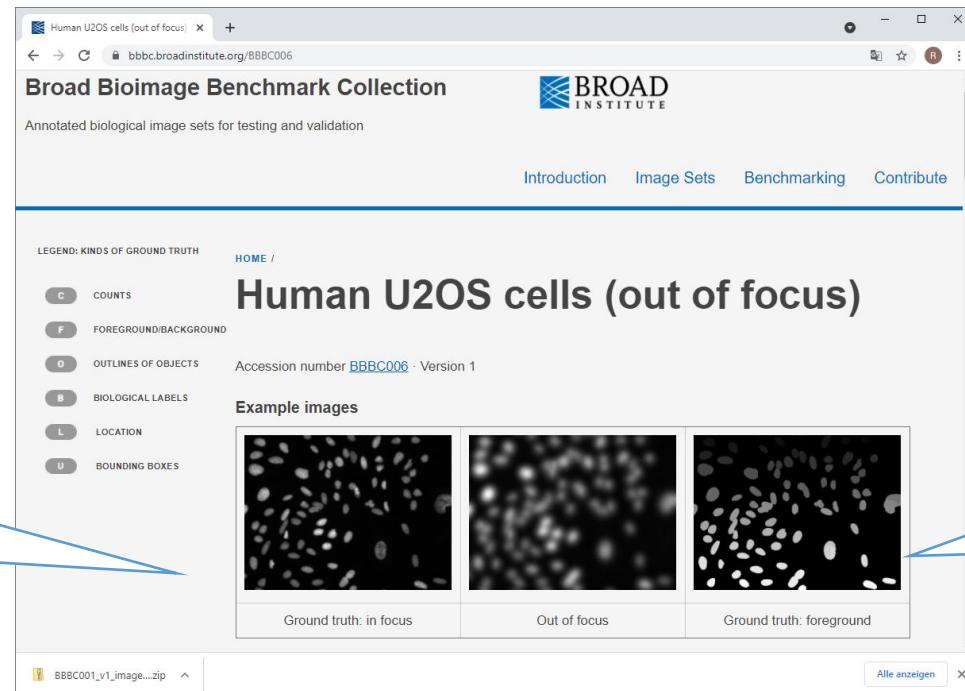
Overlay



There are 65 objects
in this image.

Reliable bio-image analysis

- Algorithms must be reliable (validated methods).
- Publicly available benchmark data sets allow to compare algorithms on common data.



Original image
data

“Ground truth”
label images

Reproducible data analysis

- Allowing others to do your experiment again.
- “The image data was analyzed with Python.”

Can you reproduce
what they did?

Reproducible bio-image analysis

- Allowing others to do your experiment again.
- “The image data was analyzed with Python.”

Can you reproduce what they did?

Can you reproduce what they did?

The screenshot shows a web browser window with the URL https://haesleinhuepf.github.io/BioImageAnalysisNotebooks/01_introduction/trailer.html. The page title is "Trailer: Bio-image Analysis with Python". The left sidebar contains a logo, a search bar, and a list of chapters: "Bio-image Analysis Notebooks", "Trailer: Bio-image Analysis with Python", "Setting up your computer", "Python basics", "Advanced python programming", "Image analysis basics", "GPU accelerated image processing", "Image visualization in 3D", "Image filtering", "Image deconvolution", "Spatial transforms", "Image segmentation", "Machine learning for image segmentation", and "Comparing segmentation". The main content area starts with an introduction: "In the following chapters we will dive into image analysis, machine learning and bio-statistics using Python. This first notebook serves as a trailer of what we will be doing." It then shows a code snippet for importing Python libraries:

```
import numpy as np
from skimage.io import imread, imshow
import pyclesperanto_prototype as cle
from cellpose import models, io
from skimage import measure
import pandas as pd
import apoc
```

Below this, there is a section titled "Working with image data" with the following text: "We start with loading the image data of interest. In this example we load an image showing a zebrafish eye, courtesy of Mauricio Rocha Martins, Norden lab, MPI CBG Dresden." A code snippet for loading an image is shown:

```
# open an image file
multichannel_image = imread("../data/zfish_eye.tif")
print("Image size", multichannel_image.shape)
```

The page ends with the text "Image size (1024, 1024, 3)".

Replicable bio-image analysis

- Others run the same analysis on their data and have consistent results / same conclusions.
- Can only be achieved if data analysis protocol was documented reproducibly.

- See also: *Replication crisis*
 - In Psychology (surveys)
 - In Medicine (clinical trials)
 - In Computer Science (executable code)
 - ...

Open access, freely available online

Essay

Why Most Published Research Findings Are False

John P. A. Ioannidis

Summary

There is increasing concern that most current published research findings are false. The probability that a research claim is true may depend on study power and bias; the number of other studies on the same question; and, importantly, the ratio of true to no relationships among the relationships probed in each scientific field. In this framework, a research finding is less likely to be true when the studies conducted in a field are smaller; when effect sizes are smaller; when there is a greater number and lesser preselection of tested relationships; where there is greater flexibility in designs, definitions, outcomes, and analytical modes; when there is greater financial and other interest and prejudice; and when more teams are involved in a scientific field in chase of statistical significance. Simulations show that for most study designs and settings, it is more likely for factors that influence this problem and some corollaries thereof.

Modeling the Framework for False Positive Findings

Several methodologists have pointed out [9–11] that the high rate of nonreplication (lack of confirmation) of research discoveries is a consequence of the convenient, yet ill-founded strategy of claiming conclusive research findings solely on the basis of a single study assessed by formal statistical significance, typically for a *p*-value less than 0.05. Research is not most appropriately represented and summarized by *p*-values, but, unfortunately, there is a widespread notion that medical research articles is characteristic of the field and can vary a lot depending on whether the field targets highly likely relationships or searches for only one or a few true relationships among thousands and millions of hypotheses that may be postulated. Let us also consider, for computational simplicity, circumscribed fields where either there is only one true relationship (among many that can be hypothesized) or the power is similar to find any of the several existing true relationships. The pre-study probability of a relationship being true is $R/(R+1)$. The probability of a study finding a true relationship reflects the power $1 - \beta$ (one minus the Type II error rate). The probability of claiming a relationship when none truly exists reflects the Type I error rate, α . Assuming that c relationships are being probed in the field, the expected values of the 2×2 table are given in Table 1. After a research finding has been claimed based on

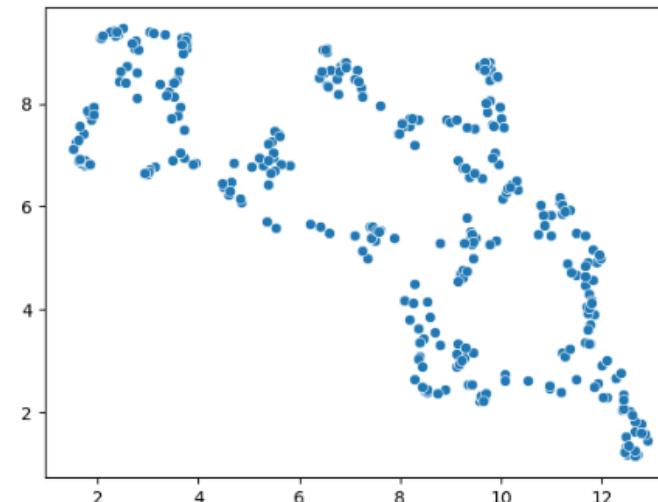
It can be proven that most claimed research findings are false.

Repeatable data analysis

- In wet-lab experiments, samples may get destroyed while executing the experiment.
- Repeatability is a property of the experiment / algorithm. You cannot improve repeatability by better documentation.

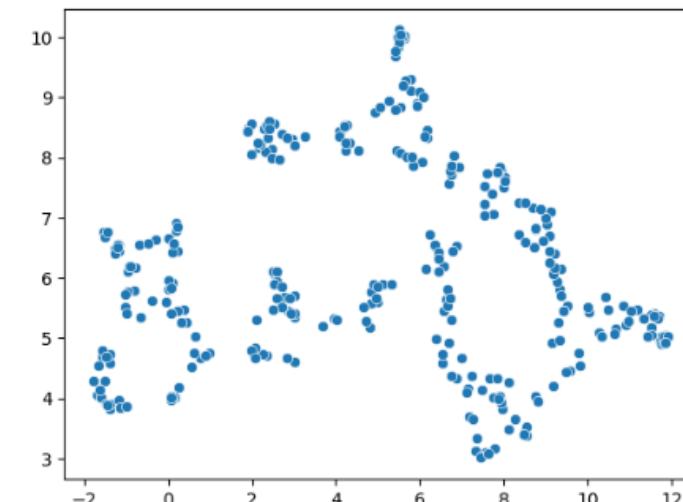
```
[11]: reducer = umap.UMAP()  
embedding2 = reducer.fit_transform(scaled_statistics)  
  
seaborn.scatterplot(x=embedding2[:, 0],  
                     y=embedding2[:, 1])
```

```
[11]: <AxesSubplot: >
```



```
[12]: reducer = umap.UMAP()  
embedding2 = reducer.fit_transform(scaled_statistics)  
  
seaborn.scatterplot(x=embedding2[:, 0],  
                     y=embedding2[:, 1])
```

```
[12]: <AxesSubplot: >
```



Bio-image Analysis: good scientific practice

Preface — Introduction to Bioimage Analysis

[bioimagebook.github.io/chapters/0-preamble/preface/preface.html](#)

When I'm confronted by an image analysis problem, my goal is never really to find the *right* way to do the analysis. That generally doesn't exist.

Instead, my goal is to find the *least wrong* way to do the analysis – and to be able to understand and explain whatever lingering limitations and biases can't be entirely overcome. It can be frustrating, I still don't feel terribly good at it, but it is – in its own strange way – kind of *enjoyable*. There's always something new to learn, and some new angle from which to look at the problem. And each new angle can help us wring more drops of knowledge out of our data.

My hope is that this book will help introduce others find the weird, frustrating pleasure of thinking more deeply about scientific images. Through this, I hope it might make a small contribution towards helping us do image analysis a bit better.

Pete Bankhead
April 2022

Front matter

Acknowledgements

License

Disclaimer

Previous [Changelog](#)

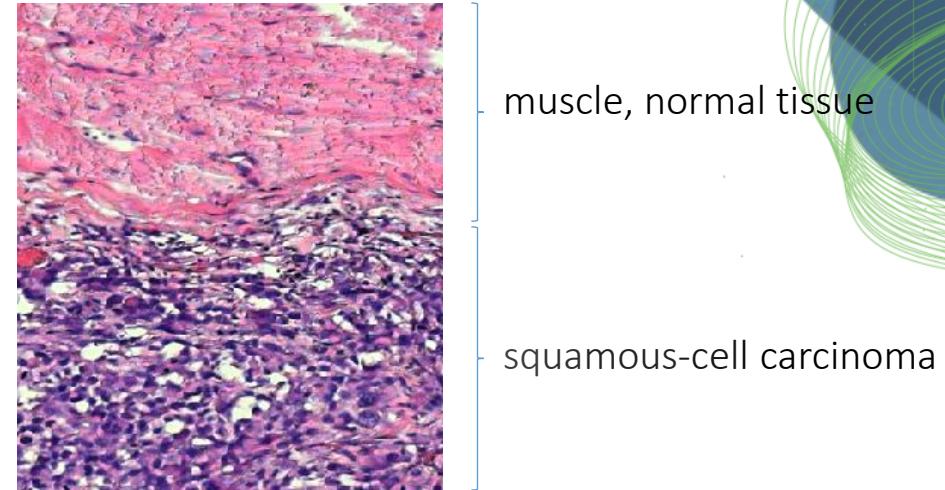
Next [How to read this book](#)

Bio-image analysis is supposed to be

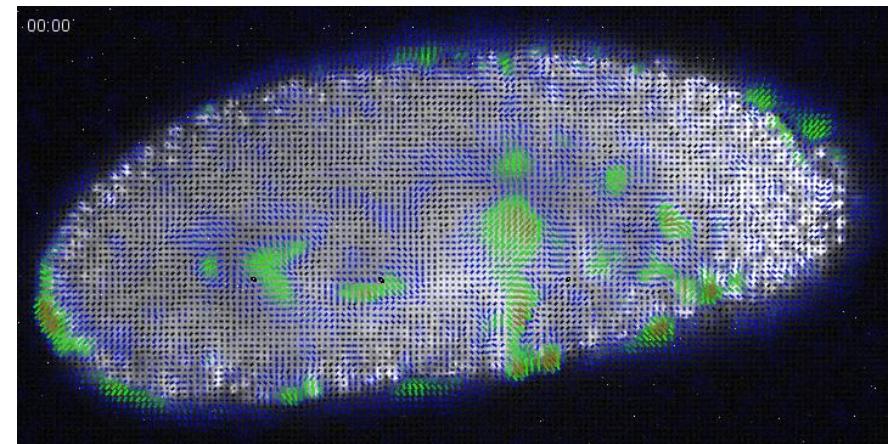
- **Quantitative**
 - We derive numbers from images which describe physical properties of the observed sample.
- **Objective**
 - The measurement does not depend on who did it. The measurement is free of interpretation.
- **Reliable (trustworthy / validated)**
 - We are confident that the measurement is describing what it is supposed to describe.
- **Reproducible**
 - Enabling others to re-do the experiment. For this, documentation is crucial!
- **Replicability**
 - Others *do* execute the same analysis, potentially on other data, and see consistent results.
- **Repeatable**
 - We can do the same experiment twice under the *same conditions* and get the same measurements.

Common topics

- Typical questions bio-image analysts deal with
 - Is signal intensity different under varying conditions?
 - How many cells are in my image?
 - How high is cell density?
 - Bio-statistics / medicine / disease staging
 - How are different tissues characterized?



- Typical questions bio-image analysts struggle with
 - What force drives the observed processes?
 - What is the lineage tree of one particular cell?
 - Are observation A and observation B related?
 - Are structures observed in different color channels colocalized?



The Image Science Community

- Ask your question online and an expert will likely reply the same day 😊

The stackoverflow of the image analysis community.

The screenshot shows a forum thread titled "Install issues with py-clesperanto-assistant". The user @WillGiang asks about installing the napari-pyclesperanto-assistant plugin. Robert Haase responds, providing a solution involving conda and pip commands. The thread continues with a follow-up from WillGiang.

Install issues with py-clesperanto-assistant

William Giang Jul 2021
Hi @haesleinhuepf ,
I'm encountering troubles with installing the napari-pyclesperanto-assistant plugin (v0.9.3) for napari 0.4.10 on Windows 10.
Here was my installation process:
First, I have Visual Studio Build Tools 2019 with "Desktop development with C++" as well as

- MSVC v142 - VS 2019 C++ x64/x86 build tools (Latest)
- Windows 10 SDK (10.0.19041.0)
- C++ CMake tools for Windows
- Testing tools core features - Build Tools
- C++ AddressSanitizer

After creating and activating a conda environment with python=3.8, I can pip install "napari[all]" without issue.
Following the [install instructions](#), I downloaded and installed a .whl file with OpenCL 1.2 and Python 3.8. Note that it was pyopenc1-2021.2.3+c112-cp38-cp38-win_amd64.whl instead of the (no longer available) 2020.3.1 version. The log for the napari-pyclesperanto plugin install suggests that it's trying to install 2021.2.1 and also suggests installing pybind11 and mako.

Install issues with py-clesperanto-assistant

Robert Haase Jul 2021
Hi @WillGiang ,
thanks for reporting! I recently hit issues with recent pyopenc1 > 2021.2.1 and thus, limited that dependency. Maybe, that was a fault. I'm considering removing this with the next minor release next week.
In the meantime, can you try installing using conda?

```
conda create --name bio11 python==3.8.5
conda activate bio11
conda install -c conda-forge pyopenc1==2021.2.1
pip install napari-pyclesperanto-assistant
pip install napari[all]
```

Let us know if this helps!
Cheers,
Robert

William Giang Jul 2021
With your instructions to specifically use python 3.8.5 and pyopenc1 v2021.2.1, the install goes through



DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE

Software Management

Robert Haase

Conda environments

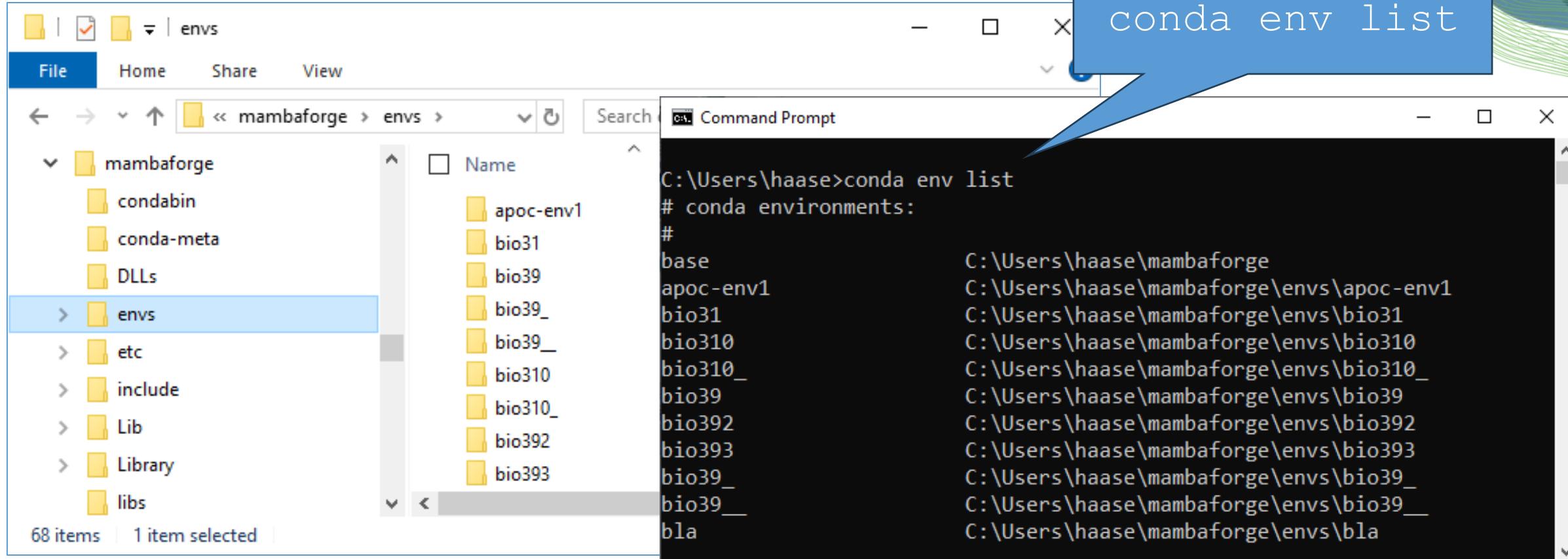
- What is the difference between
 pip install ...
 and
 conda install ...
 ?

Conda environments

- Conda is a package manager
 - Allows to install Python packages
 - Allows to install other stuff (git, JDK)
- Conda is an environment manager
 - Virtual environments
 - Import/export/distribute environments
- Pip can only install Python packages (also into conda environments)

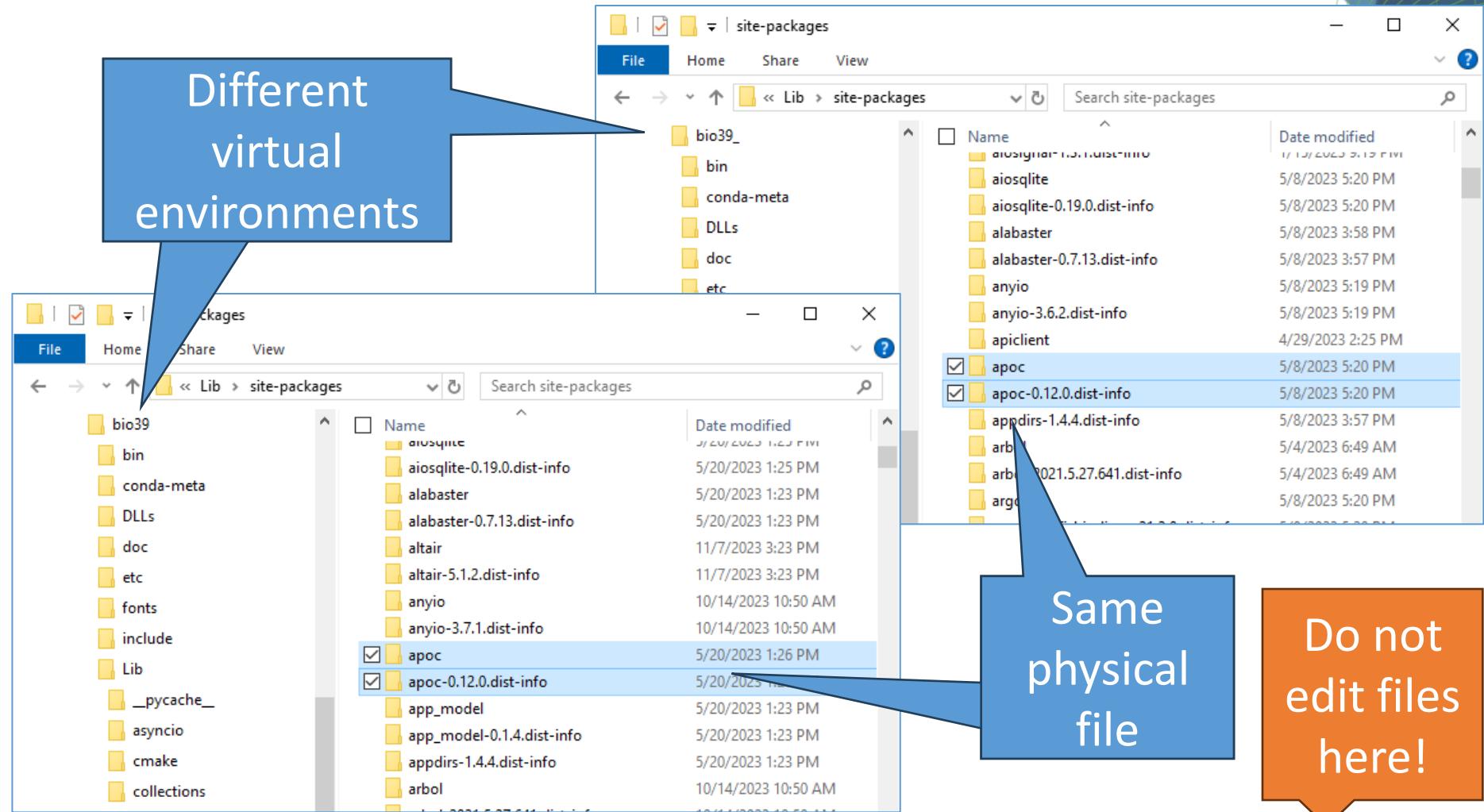
Conda environments

- A conda environment is *just a folder*



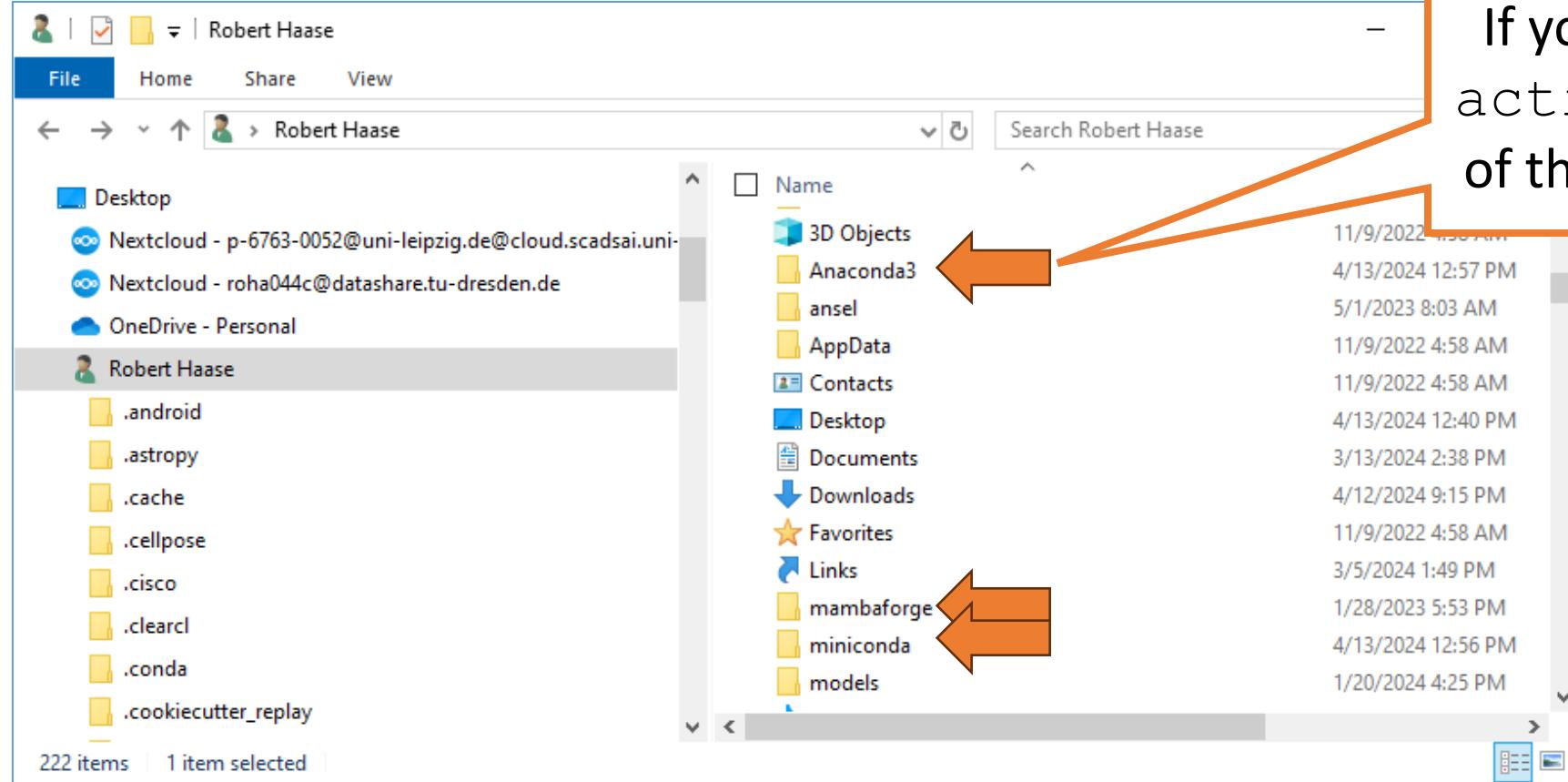
Conda environments

- Packages within the folders are just linked
 - If you change a file in env1, the same file in env2 is changed.



Multiple Python installations

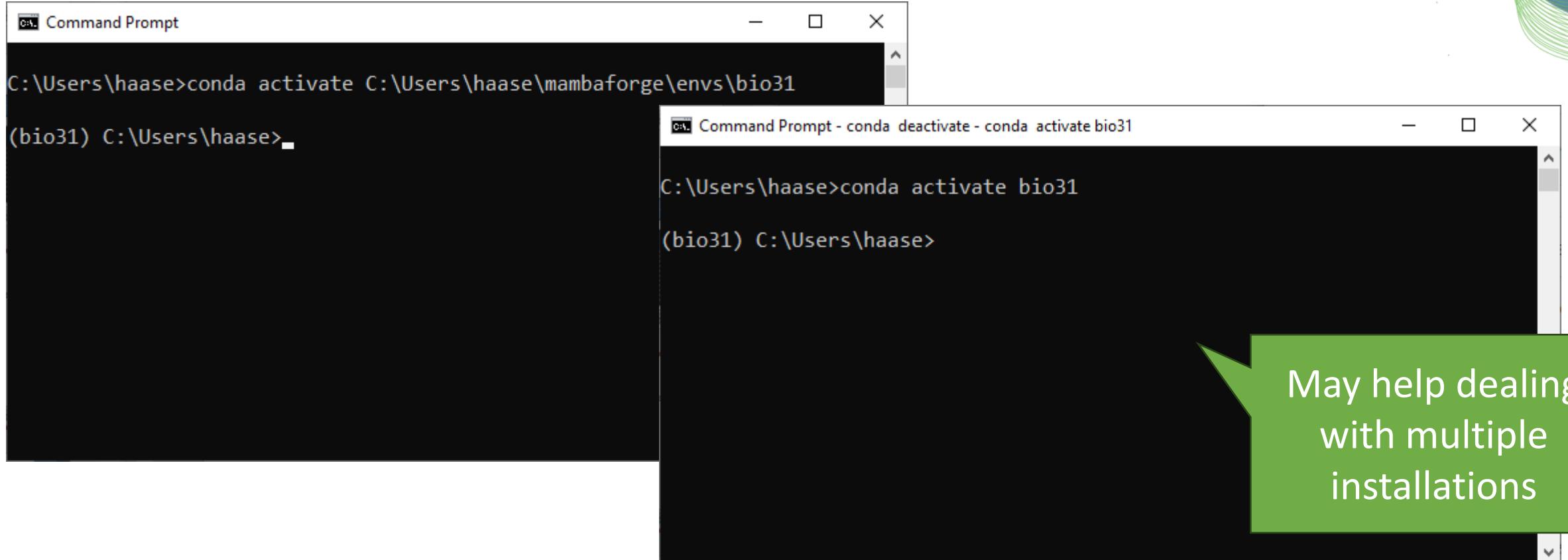
- On some computers you find this



If you enter `conda activate ...`, which of those is executed?

Conda environments

- Conda environments can be activated from anywhere



The image shows two separate Command Prompt windows side-by-side. The left window has a title bar 'Command Prompt' and contains the command: 'C:\Users\haase>conda activate C:\Users\haase\mambaforge\envs\bio31'. Below the command, the prompt shows '(bio31) C:\Users\haase>'. The right window also has a title bar 'Command Prompt - conda deactivate - conda activate bio31' and contains the command: 'C:\Users\haase>conda activate bio31'. Below the command, the prompt shows '(bio31) C:\Users\haase>'. Both windows have standard window controls (minimize, maximize, close).

May help dealing
with multiple
installations

Broken [conda] environments

- Common case of confusion in the Python ecosystem
- Happens to everyone – it's just a matter of time.

The screenshot shows a Windows desktop with three command prompt windows and a Jupyter Notebook interface.

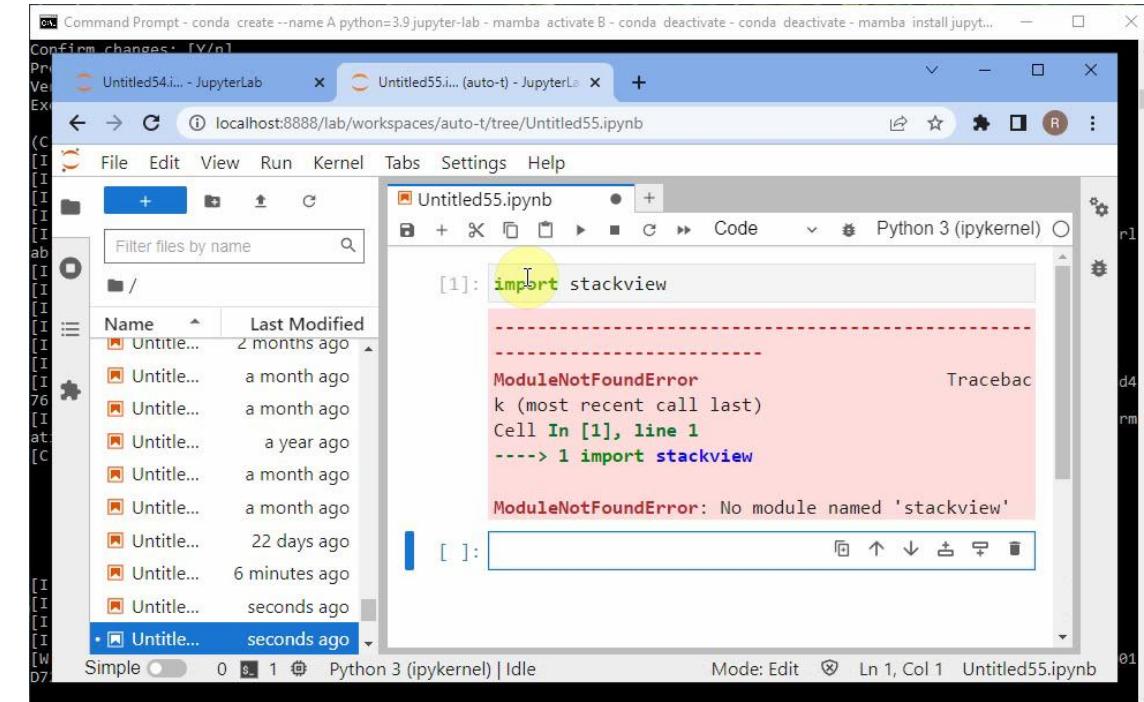
- Left Command Prompt:** Shows the output of running `mamba install stackview -c conda-forge`. It includes the MAMBA logo and version information: "mamba (0.25.0) supported by @QuantStack".
- Middle Command Prompt:** Shows the output of running `conda create --name A python=3.9 jupyter-lab - mamba activate B - conda deactivate`. It lists many packages being installed from the conda-forge channel, such as toolz, tornado, typing_extensions, tzdata, ucrt, unicodedata2, vc, vs2015_runtime, wcwidth, wheel, widgetsnbextension, xorg-libxau, xorg-libxdmcp, xz, yaml, zeromq, zfp, zipp, zlib-ng, and zstd.
- Right Command Prompt:** Shows the output of running `conda create --name A python=3.9 jupyter-lab - mamba activate B - conda deactivate - conda deactivate - mamba install jupyter-lab`. It shows a "Confirm changes: [Y/n]" prompt and transaction logs.
- Jupyter Notebook:** Shows a cell with the code `[1]: import stackview` which results in a `ModuleNotFoundError`.

Background

- The base-environment is special.
- You install packages into the base environment, if you do not run `conda activate my_env`
- You can run base-software from within other environments

Base
Installed packages:
Jupyter lab

my_env
Installed packages:
stackview



The screenshot shows a Jupyter Notebook interface with two tabs: 'Untitled54.ipynb' and 'Untitled55.ipynb'. The 'Untitled55.ipynb' tab is active, displaying a code cell with the following content:

```
[1]: import stackview
```

An error message is visible in the output area:

```
ModuleNotFoundError
k (most recent call last)
Cell In [1], line 1
----> 1 import stackview

ModuleNotFoundError: No module named 'stackview'
```

The file browser on the left shows several unnamed files in the current directory.

Broken [conda] environments

- The only cure: Uninstall and reinstall [Ana]conda.



- Alternatively:
 - Create a new environment for every project.
 - `conda activate my_env`
 - Do not install project-specific stuff in the base-environment.

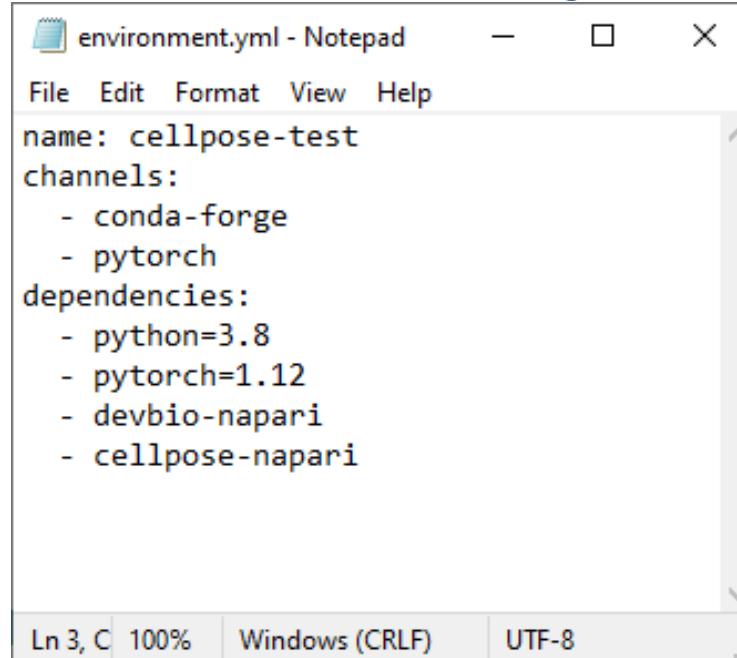
Broken [conda] environments

Do not install anything in the base environment

Always call **conda activate ...** before doing anything

Documenting dependencies

- Maintain a document with the dependencies (and versions) you need in your project!
 - The conda way
 - The pip way

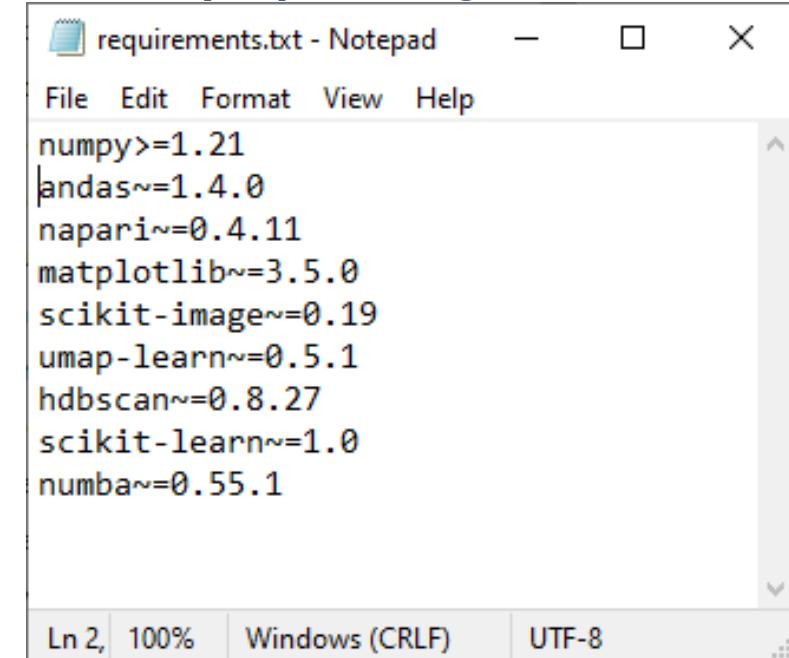


```
environment.yml - Notepad
File Edit Format View Help
name: cellpose-test
channels:
- conda-forge
- pytorch
dependencies:
- python=3.8
- pytorch=1.12
- devbio-napari
- cellpose-napari

Ln 3, C 100% Windows (CRLF) UTF-8
```

conda env create -f environment.yml

In case your
environment is screwed
up, you can rebuild it
any time.



```
requirements.txt - Notepad
File Edit Format View Help
numpy>=1.21
andas~=1.4.0
napari~=0.4.11
matplotlib~=3.5.0
scikit-image~=0.19
umap-learn~=0.5.1
hdbscan~=0.8.27
scikit-learn~=1.0
numba~=0.55.1

Ln 2, 100% Windows (CRLF) UTF-8
```

pip install -r requirements.txt

Documenting dependencies

- ... the complete way.

```
conda env export > environment.yml
```

```
environment.yml - Notepad
File Edit Format View Help
name: bio_39
channels:
- conda-forge
- defaults
dependencies:
- alabaster=0.7.12=py_0
- anyio=3.6.1=pyhd8ed1ab_1
- aom=3.5.0=h63175ca_0
- apoc-backend=0.10.0=pyhd8ed1ab_0
- appdirs=1.4.4=pyh9f0ad1d_0
- argon2-cffi=21.3.0=pyhd8ed1ab_0
- argon2-cffi-bindings=21.2.0=py39hb82d6ee_2
- asciitree=0.3.3=py_2
- asttokens=2.0.8=pyhd8ed1ab_0
- attrs=22.1.0=pyh71513ae_1
- autopep8=1.7.0=pyhd8ed1ab_0
- babel=2.10.3=pyhd8ed1ab_0
- backcall=0.2.0=pyh9f0ad1d_0
backgrounds=1.0=py_2
Ln 391, Col 46 100% Windows (CRLF) UTF
```

```
environment.yml - Notepad
File Edit Format View Help
- win_inet_pton=1.1.0=py39hcbf5309_4
- winpty=0.4.3=4
- wrapt=1.14.1=py39hb82d6ee_0
- x264=1!164.3095=h8ffe710_2
- x265=3.5=h2d74725_3
- xorg-libxau=1.0.9=hcd874cb_0
- xorg-libxdmcp=1.1.3=hcd874cb_0
- xz=5.2.6=h8d14728_0
- yaml=0.2.5=h8ffe710_2
- zarr=2.13.3=pyhd8ed1ab_0
- zeromq=4.3.4=h0e60522_1
- zfp=1.0.0=h0e60522_1
- zict=2.2.0=pyhd8ed1ab_0
- zipp=3.9.0=pyhd8ed1ab_0
- zlib-ng=2.0.6=h8ffe710_0
- zstd=1.5.2=h7755175_4
prefix: C:\Users\rober\mambaforge\envs\bio_39
Ln 391, Col 46 100% Windows (CRLF) UTF-8
```

Excellent way to document which dependencies were *actually* used...

It is *questionable* if re-creating an environment from this yml file works.

Installing dependencies

- A difference between pip and conda:

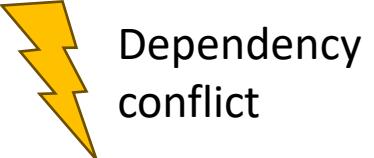
```
pip install package_a
```

```
conda install package_a
```

Depends on:
numpy<1.22.0

```
pip install package_b
```

```
conda install package_b
```



Dependency
conflict
Depends on:
numpy>=1.22.0

passes

fails

Because the environment
cannot be solved

Installing dependencies

- A difference between pip and conda:

pip install git

fails

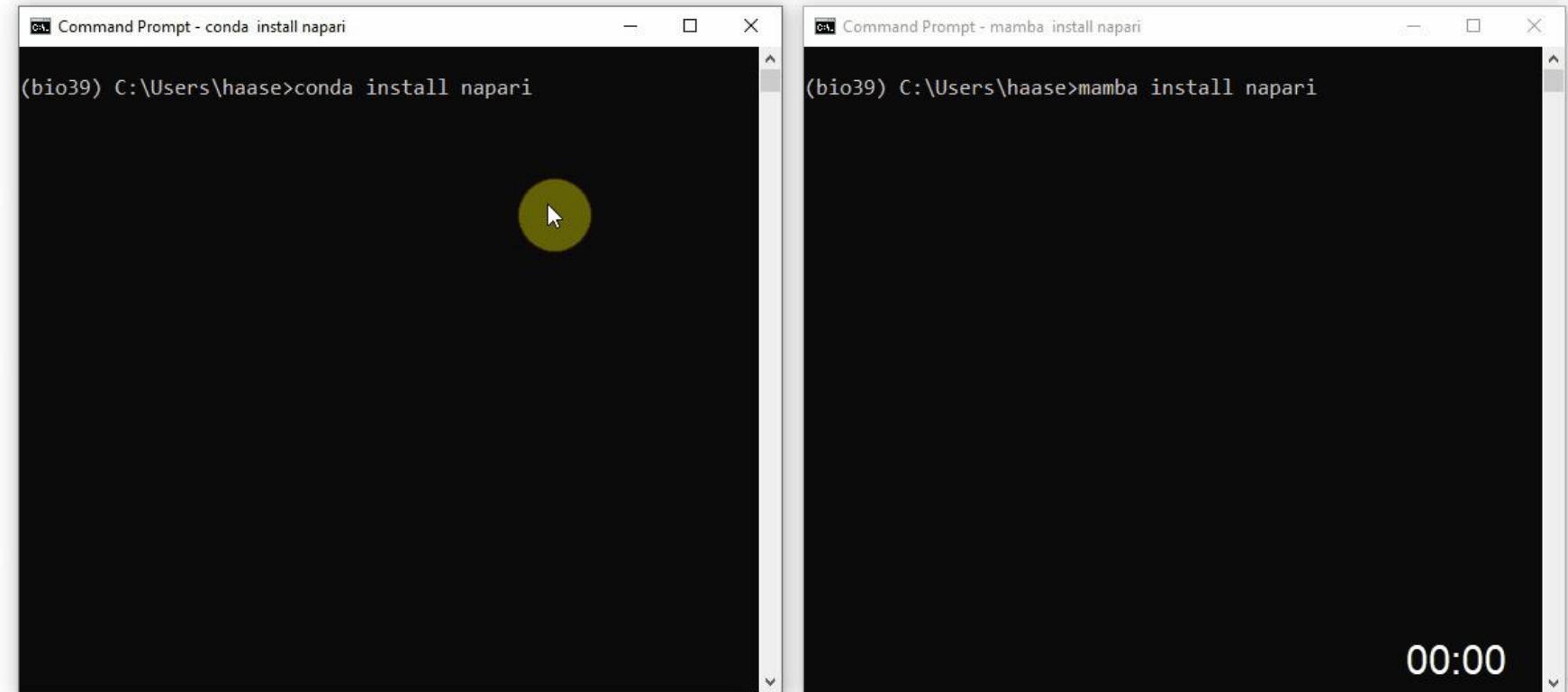
Because git is not a
python package

conda install git

passes

Conda versus mamba

- mamba is a “drop in” replacement that accepts the same commands as conda.
- mamba is much faster though.



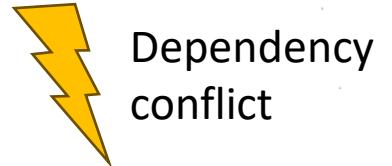
Quiz

conda install package_a

Depends on:
numpy<1.22.0

pip install package_b

Depends on:
numpy>=1.22.0



fails



passes

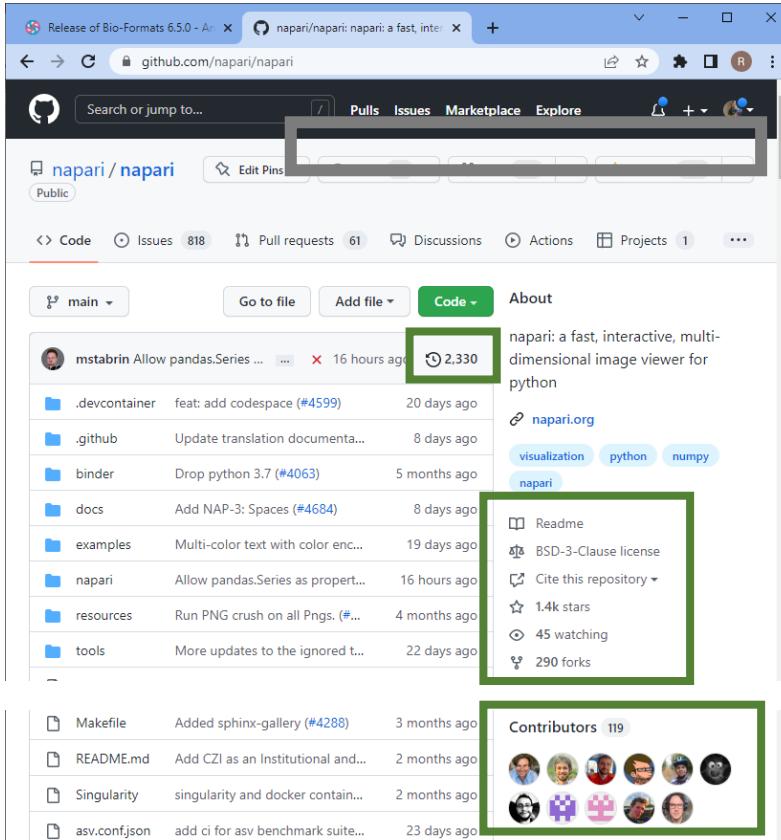


passes but...

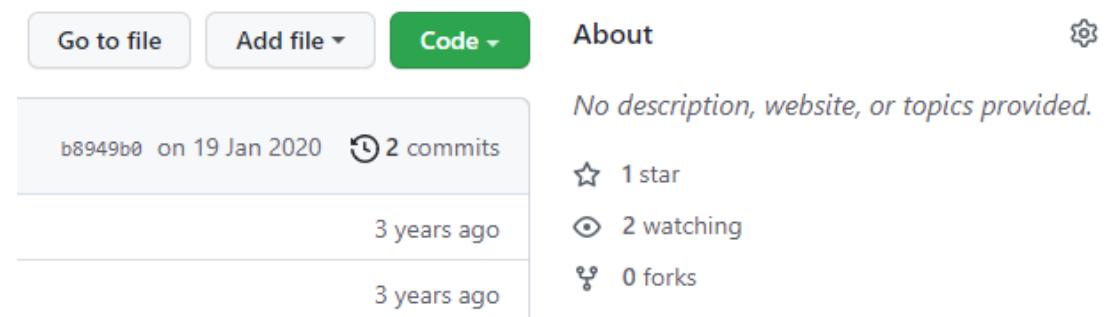


Software quality indicators

- Visit the project's github or gitlab page and review indicators.

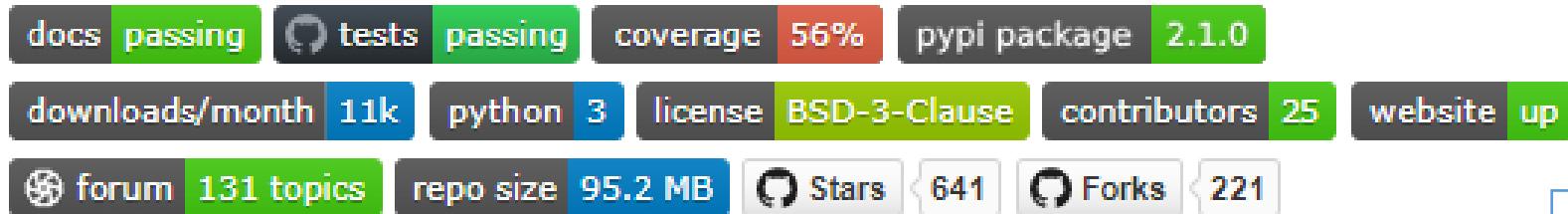


- **Stars:** People like software, similarly to tweets on Twitter
- **Watching:** People receive updates for new releases
- **Forks:** People made a copy of the code, e.g. to contribute to the project
- **Contributors:** People who contributed to the code
- **Commits:** Changes to the code



Software quality indicators

- Visit the project's github or gitlab page and review indicators.

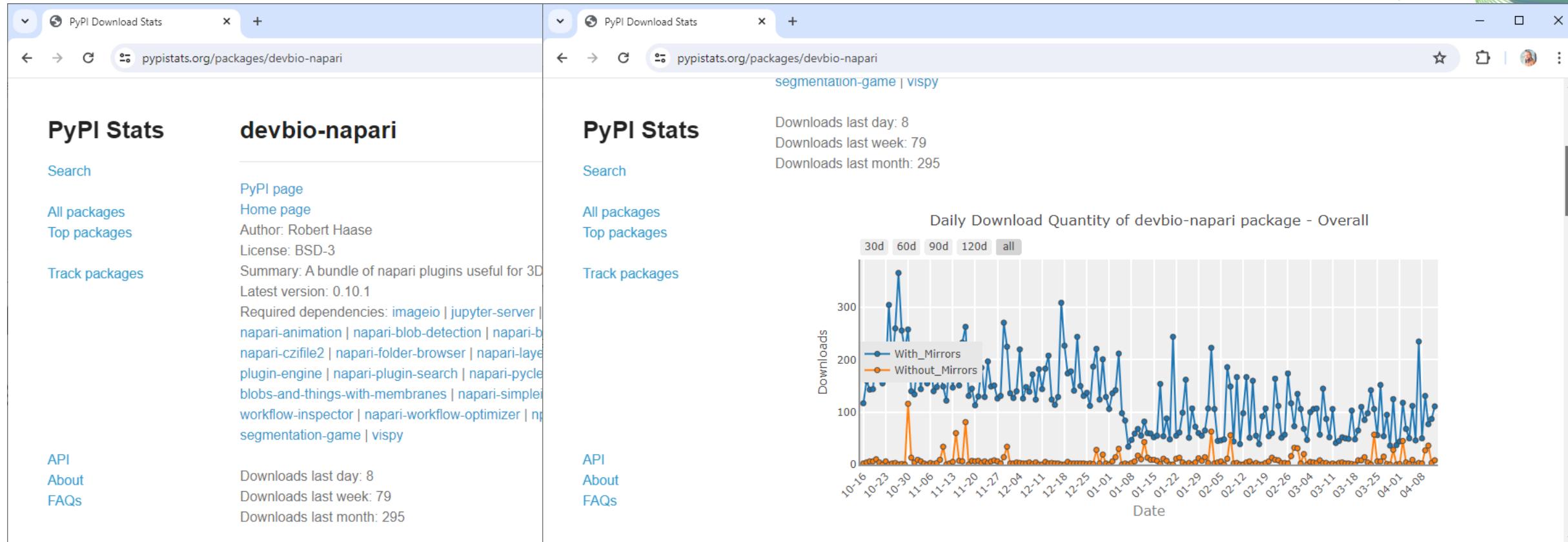


Note, github badges
cannot be *deserved*.
Developers put
them there



Software quality indicators

- Download statistics: pypi



Software quality indicators

- Scientific publications

The screenshot shows a GitHub repository page for "MouseLand/cellpose: a generalist algorithm for cellular segmentation". The page includes a README file, a BSD-3-Clause license, and several grayscale microscopy images of cell structures. A section titled "CITATION" provides citation information for Cellpose 1.0 and 2.0.

If you use Cellpose 1, 2 or 3, please cite the Cellpose 1.0 paper:
Stringer, C., Wang, T., Michaelos, M., & Pachitariu, M. (2021). Cellpose: a generalist algorithm for cellular segmentation. *Nature methods*, 18(1), 100-106.

If you use the human-in-the-loop training, please also cite the Cellpose 2.0 paper:
Pachitariu, M. & Stringer, C. (2022). Cellpose 2.0: how to train your own model. *Nature methods*, 1-8.

The screenshot shows a Nature Methods article page for "Cellpose 2.0: how to train your own model". The page includes the journal header, navigation links, and the article title. Below the title, it lists the authors (Marius Pachitariu & Carsen Stringer), the journal issue (Nature Methods 19, 1634–1641 (2022)), and citation metrics (55k Accesses, 120 Citations, 116 Altmetric). Two arrows point to the "Accesses" and "Citations" metrics.

Article | Open access | Published: 07 November 2022

Cellpose 2.0: how to train your own model

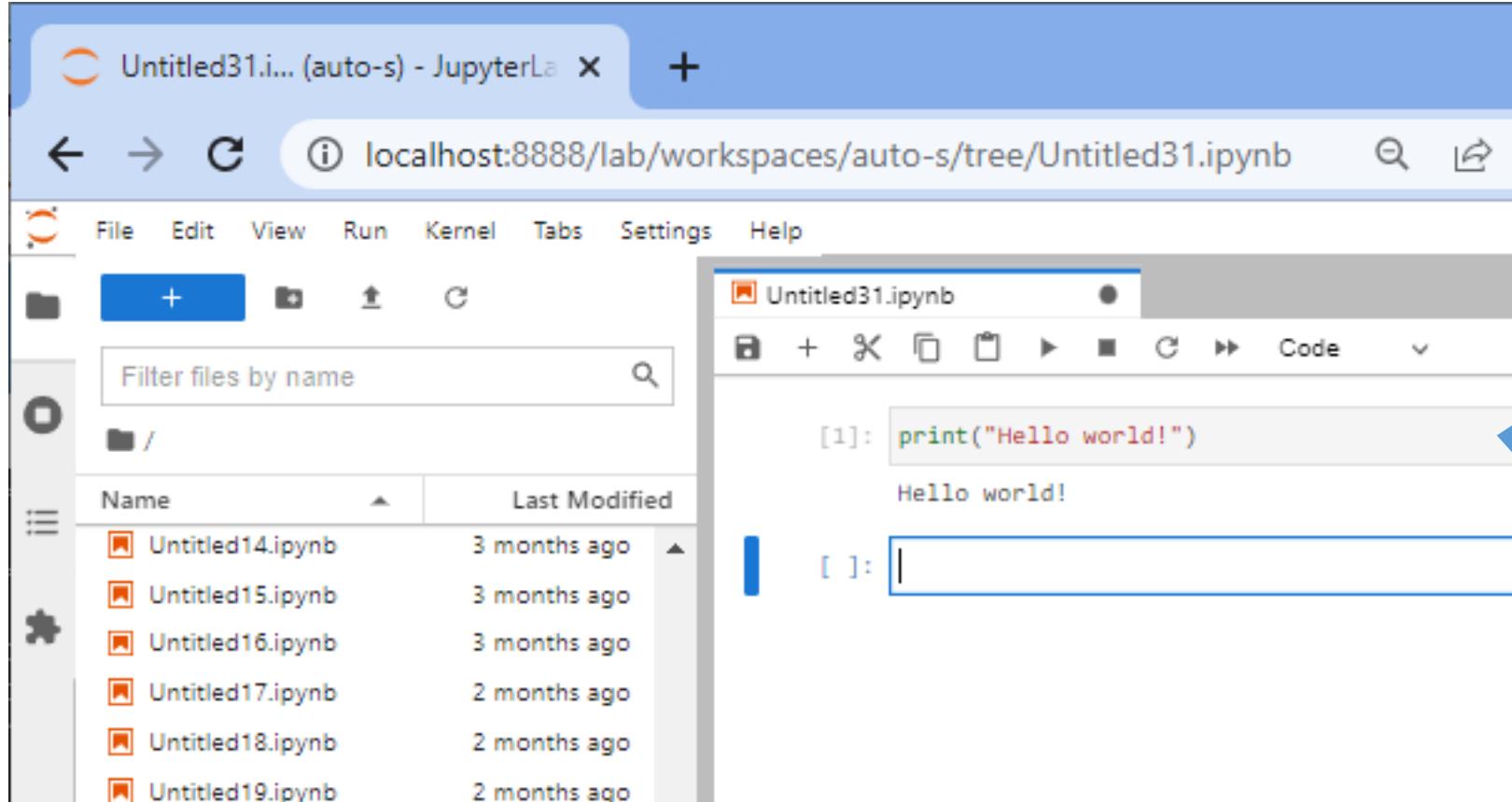
Marius Pachitariu & Carsen Stringer

Nature Methods 19, 1634–1641 (2022) | Cite this article

55k Accesses | 120 Citations | 116 Altmetric | Metrics

Jupyter lab

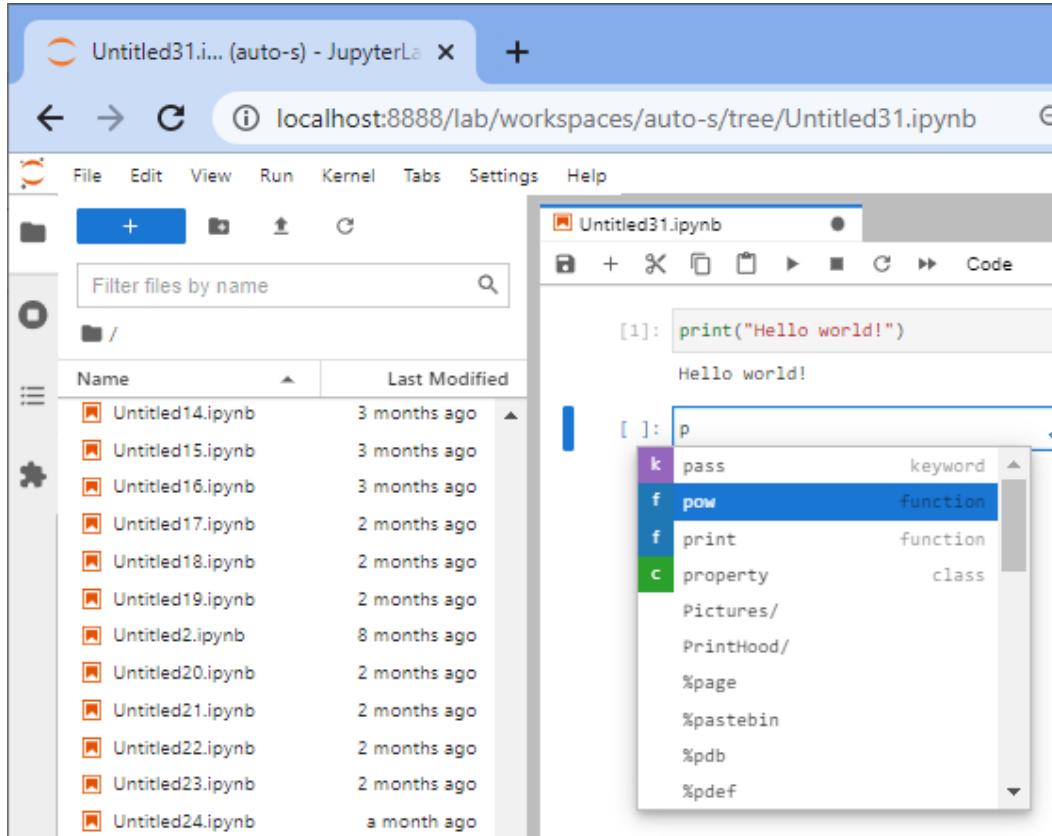
- Execute code cell-by-cell and see results instantaneously



SHIFT + ENTER
to execute a
code cell

Jupyter lab

- Context-specific help, auto-completion



TAB
to open auto-
completion

Jupyter lab

- Help / “docstrings”

The screenshot shows the Jupyter Lab interface. On the left, there's a sidebar with a file browser showing recent notebooks. In the main area, a code cell contains the command `print("Hello world!")`, which has been run and output "Hello world!". Below it, another code cell contains `print?`, which displays the `print` function's docstring:

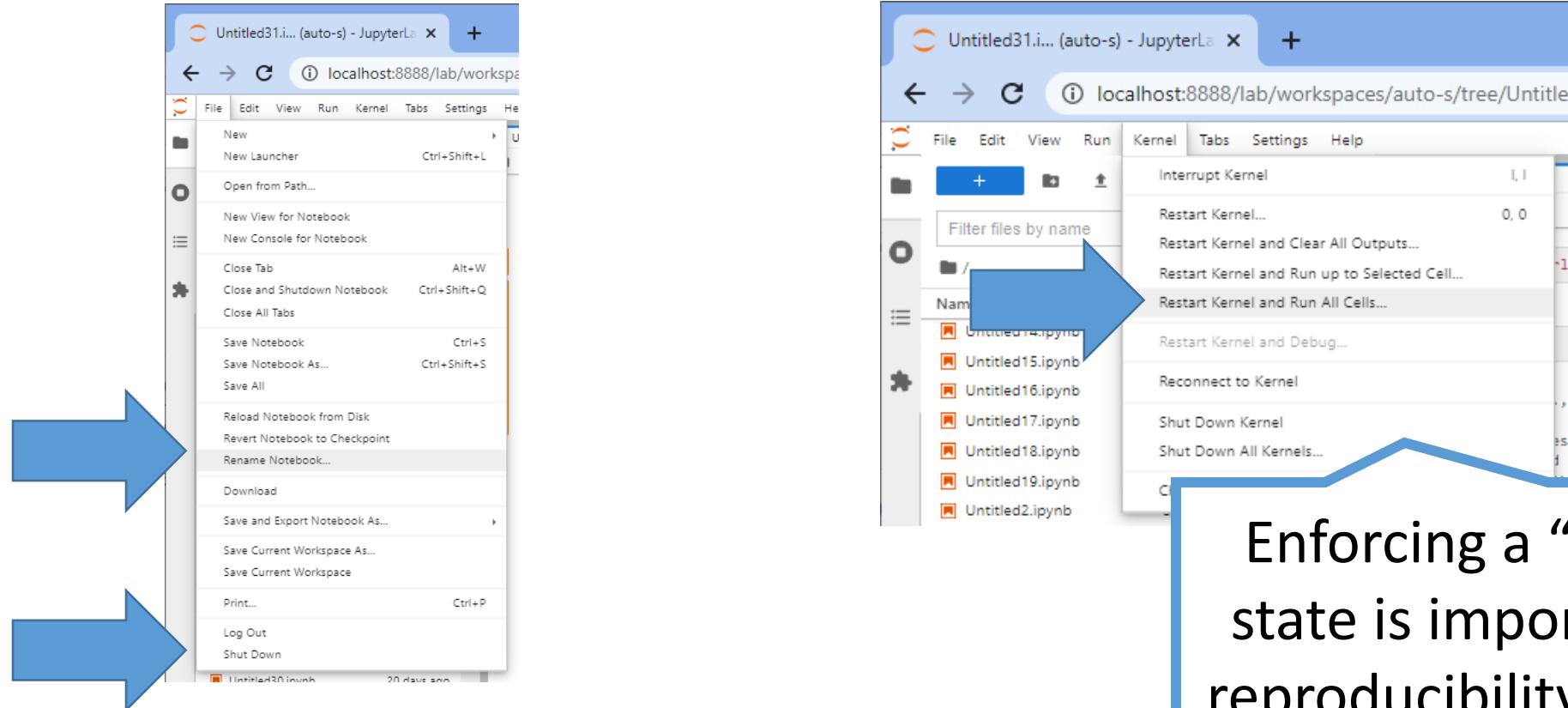
```
[1]: print("Hello world!")
Hello world!
[3]: print?
Docstring:
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file: a file-like object (stream); defaults to the current sys.stdout.
sep: string inserted between values, default a space.
end: string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
Type: builtin_function_or_method
```

?
to read what a
function does

Jupyter lab

- Saving / renaming / closing



Enforcing a “clean” execution state is important for ensuring reproducibility and repeatability

Quiz

- Enabling others to do your experiment is about ...

Repeatability

Reproducibility

Replicability

Reliability

Quiz

- Reproducibility can be achieved by

Writing documentation

Writing code

Providing example data

Recording Video tutorials

Quiz

- Resolution in imaging describes...

Size of pixels
on a screen

Size of pixels
on a camera
chip

Maximum size
of objects in
relation to the
image

Minimum size
of objects that
can be
differentiated
in an image



DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE

Exercises

Robert Haase

GEFÖRDERT VOM



Bundesministerium
für Bildung
und Forschung



Diese Maßnahme wird gefördert durch die Bundesregierung
aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf
der Grundlage des von den Abgeordneten des Sächsischen
Landtags beschlossenen Haushaltes.

Exercises

Training resources for Students at Uni Leipzig who want to dive into bio-image data science with Python. The material will develop between April and July 2024.

CC-BY-4.0 license
1 star 0 forks 2 watching 1 Branch 0 Tags Activity Public repository

main Go to file

haesleinhuepf added first exercise

01a_setting_up_local_environment added first exercise

01b_setting_up_sc_ulei_environment added first exercise

01c_testing_environment added first exercise

.gitignore added first exercise

LICENSE-CC-BY The Beginning

README.md added first exercise

Bio-image Data Science

This repository contains training resources for Students at Uni Leipzig who want to dive into bio-image data science with Python. The material will develop between April and July 2024 and shared here in this github repository.

Teaching Goal

Students learn the full workflow of common bio-image data science projects to a degree that they can execute a scientific data analysis project in this context on their own. They will be familiar with common bio-image analysis algorithms and workflows, how to choose them according to a scientific goal, and how to measure quality of derived results. Attending the lecture and executing the practicals qualifies the students to work as bio-image data scientist in the pharmaceutical industry or basic biological research.

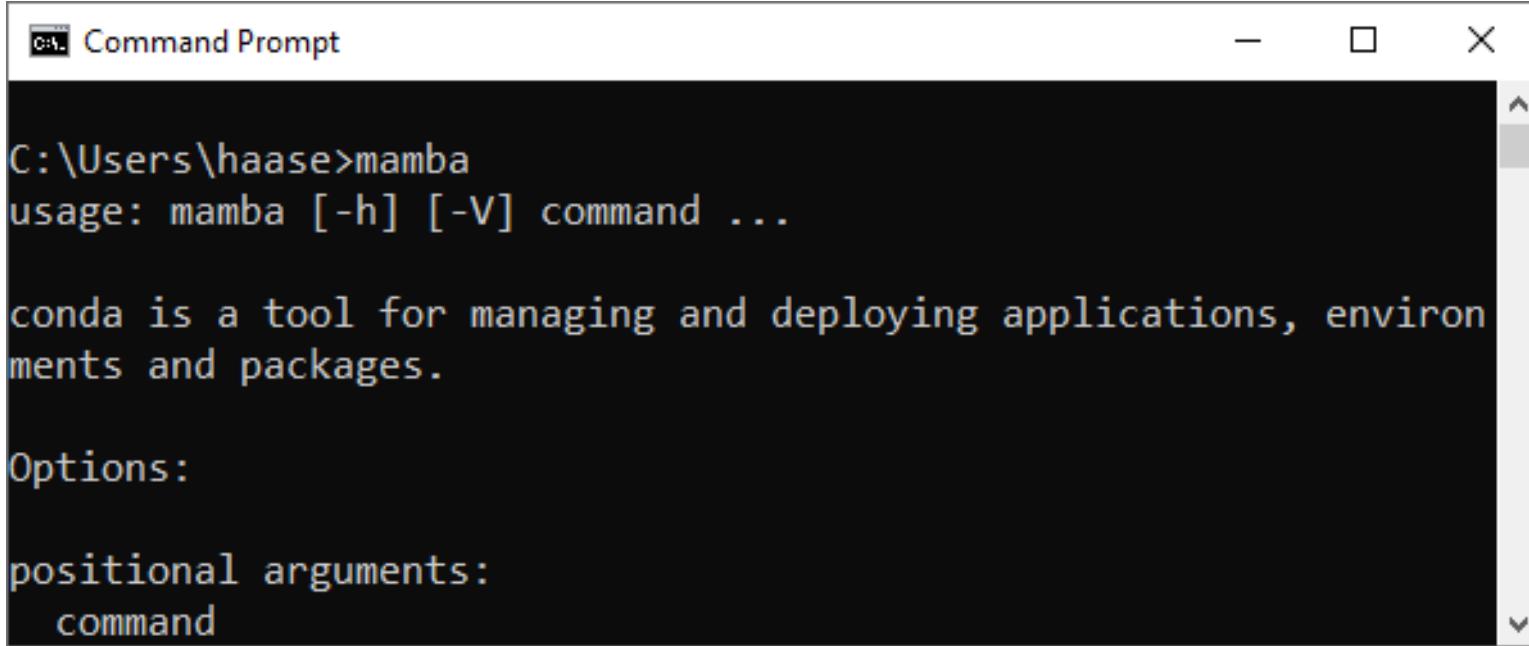
Course contents

- Introduction to Bio-image Data Science (Apr 2nd 2024)
 - Basics of microscopy
 - Introduction to Bio-image Analysis
 - Exercises:
 - Setting up a local environment
 - Setting up Jupyter Hub at Scientific Computing / Leipzig University
 - Execute the trailer notebook



Installation

- You can skip the first local installation steps if you already use conda or mamba



```
Command Prompt

C:\Users\haase>mamba
usage: mamba [-h] [-V] command ...

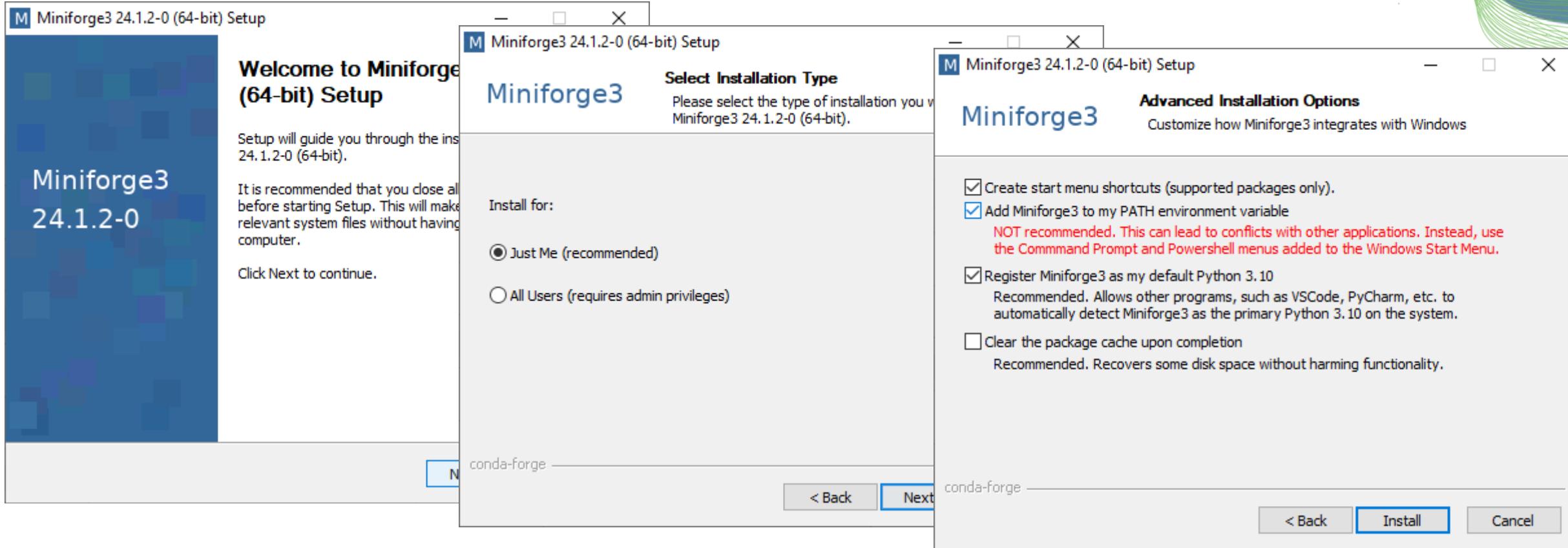
conda is a tool for managing and deploying applications, environments and packages.

Options:

positional arguments:
  command
```

Installation

- Install mini-forge



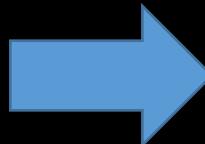
Setting up an environment

- Create a conda/mamba environment

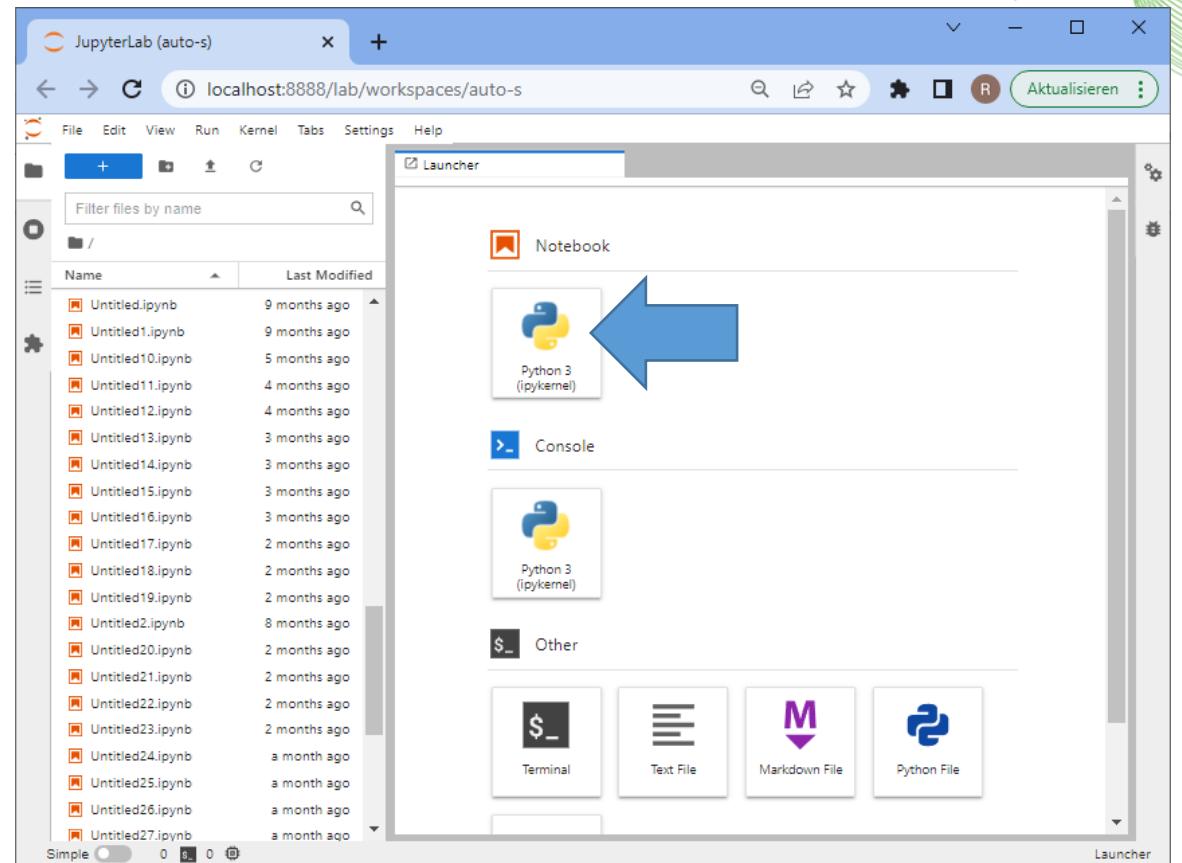
```
mamba create --name my_first_env  
python=3.9 devbio-napari pyqt -c  
conda-forge
```

Jupyter lab

- Start Jupyter lab from the folder you want to work in
- Create a new notebook

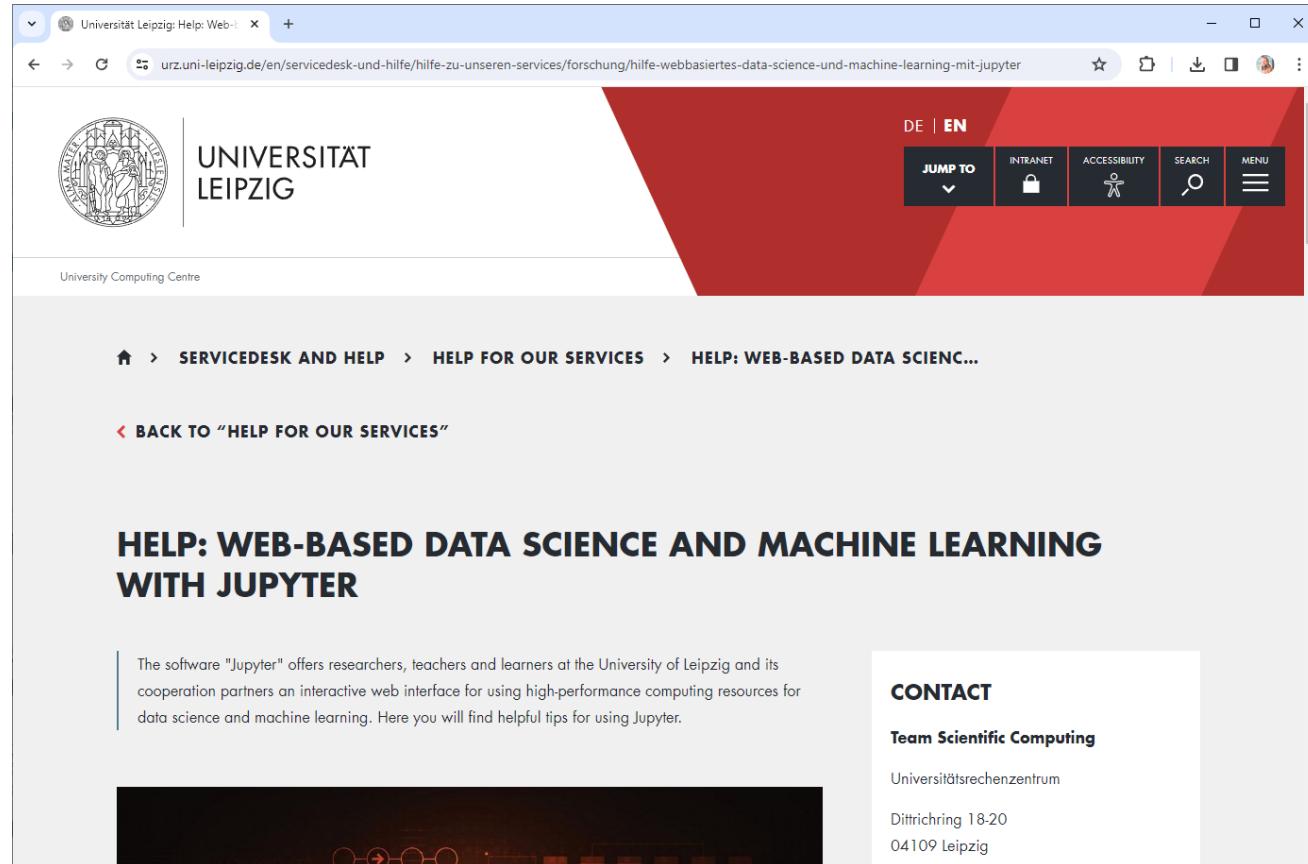


```
c:\ Command Prompt - conda deactivate - cond...
c:\Users\rober>conda activate bio_39
(bio_39) c:\Users\rober>jupyter lab
```

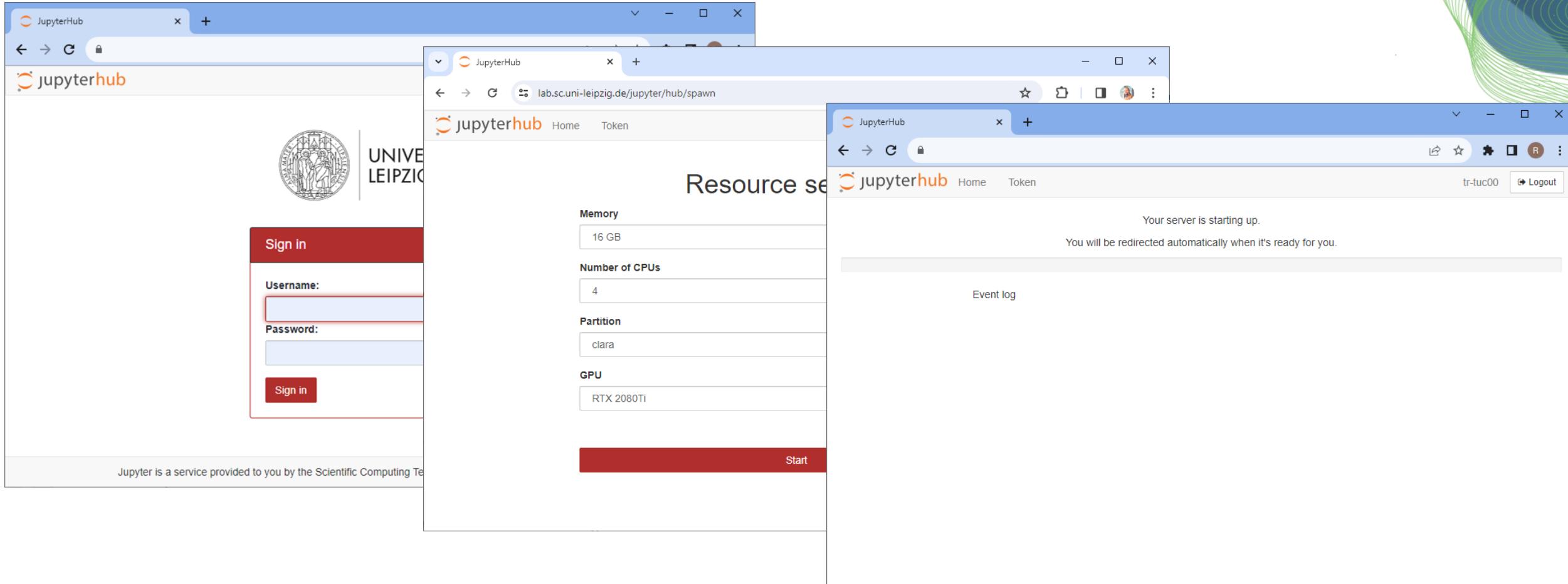


Setting up the JupyterHub @ Scientific Computing ULEI

- Alternatively: Register @ Scientific Computing / Uni Leipzig



Setting up the JupyterHub @ Scientific Computing ULEI



Setting up the JupyterHub @ Scientific Computing ULeI

- Detailed instructions

The image shows two side-by-side screenshots of a web browser window. Both windows have a blue header bar with tabs labeled "JupyterHub" and "BIDS-lecture-2024/01b_setting_". The address bar for both shows "github.com/ScaDS/BIDS-lecture-2024/tree/main/01b_setting_up_sc_ulei_environment".

Left Screenshot Content:

- Section Title:** Installation instructions for devbio-napari on clara
- Text:** These instructions are derived/modified from [this page](#).
- Text:** Request a [Scientific Computing Account at ULei](#).
- Text:** Login to VPN and [Jupyter Hub](#) and open a terminal.
- Text:** Install [ana]conda
- ```
module load Anaconda3
conda init bash
```
- Text:** At this point, we need to reopen the terminal. Afterwards, create a conda environment with one specific python version only.
- ```
conda create --name my_first_env python=3.9
conda activate my_first_env
```

Right Screenshot Content:

- Text:** pip install devbio-napari
pip install scikit-image==0.19.3
pip install ipykernel
- Text:** Make this conda environment available to Jupyter hub like this:

```
python -m ipykernel install --user --name 'my_first_env' --display-name "my_first_env"
```
- Text:** Sometimes this is necessary too:

```
# Reload page (Jupyter hub in browser)
# jupyter labextension install @jupyter-widgets/jupyterlab-manager ipycanvas
```
- Text:** To make stackview work interactively, install it outside the conda environment:

```
conda deactivate
pip install stackview --user
```

Exercise: Test the environment

- Download and test the trailer notebook

The image shows four sequential screenshots of a Jupyter Notebook titled "trailer.ipynb" demonstrating bio-image data science. The notebook is run in a Python 3 (ipykernel) environment.

- Step 1: Working with images**
The code reads an image from a URL and displays it. The output shows a grayscale image of cellular blobs with a shape of (254, 256).

```
[1]: from skimage.io import imread
from skimage.filters import threshold_otsu
from skimage.measure import label, regionprops_table
import stackview
import pandas as pd
import numpy as np
```

```
[2]: image = imread('https://samples.fiji.sc/blobs.png')[0]
stackview.insight(image)
```
- Step 2: Image segmentation**
The code applies Otsu's thresholding to the image and uses it to label connected components. The output shows a segmented image where each blob is colored differently.

```
[3]: binary = image > threshold_otsu(image)
labels = label(binary)
stackview.insight(labels)
```
- Step 3: Correlation analysis**
The code performs a correlation matrix on the measured parameters. The output is a heatmap showing the correlation coefficients between various features like area, perimeter, mean intensity, etc.

```
[7]: corr = table.corr()
def colorize(styler):
    styler.background_gradient(axis=None, cmap="seismic")
    return styler
corr.style.pipe(colorize)
```

	area	perimeter	mean_intensity	minor_axis_length	major_axis_length
area	1.000000	0.961579	0.622818	0.898116	
perimeter	0.961579	1.000000	0.707842	0.873781	
mean_intensity	0.622818	0.707842	1.000000	0.743692	
minor_axis_length	0.898116	0.873781	0.743692	1.000000	
major_axis_length	0.894802	0.962880	0.617727	0.713343	

Summary

Today, you learned

- Microscopy
 - Fluorescence microscopy
- Bio-image analysis
 - Quantitative
 - Objective
 - Reproducible
 - Repeatable
 - Reliable
- Conda Environments

Coming up next

- Exercises: Setting up your environment

In two weeks:

- Image processing for microscopy

Happy Easter!