

# Reproducible Bio-Image Analysis using Python, Napari, Jupyter and AI

Robert Haase



These slides and the related training materials can be reused under the  
terms of the [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) license unless mentioned otherwise.

<https://scads.github.io/embo-lsm-bia-2025>

# Reproducibility: Python

- Technical: I can rerun the workflow because it is well documented.
- Inter-personal: I understand a workflow and can explain all steps.

```
1 from skimage.io import imread, imsave
2 import stackview
3 import pyclesperanto as cle
4 import numpy as np
5
6 image = imread('data/lund.tif')
7
8 # Apply top hat filter to the image
9 processed_image = np.asarray(cle.top_hat_box(image, radius_x=10, radius_y=10, radius_z=0))
10 # Segment the image using voronoi otsu labeling
11 label_image = cle.voronoi_otsu_labeling(processed_image, spot_sigma=2, outline_sigma=2)
12
13 print(label_image.max())
```

Output: 285

Technical: ✓

Inter-personal: ✗

We cannot follow the workflow, because we do not see intermediate results.

# Reproducibility: Python + Jupyter

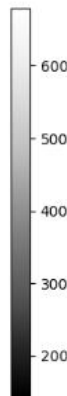
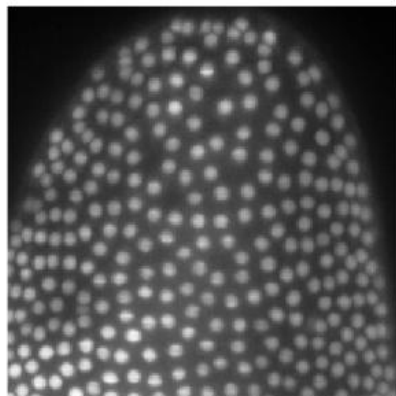
- Technical: I can rerun the workflow because it is well documented.
- Inter-personal: I understand a workflow and can explain all steps.

```
[2]: image = imread('data/lund.tif')  
  
image.shape
```

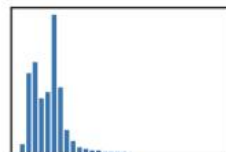
```
[2]: (100, 256, 256)
```

```
[3]: stackview.insight(image)
```

```
[3]:
```

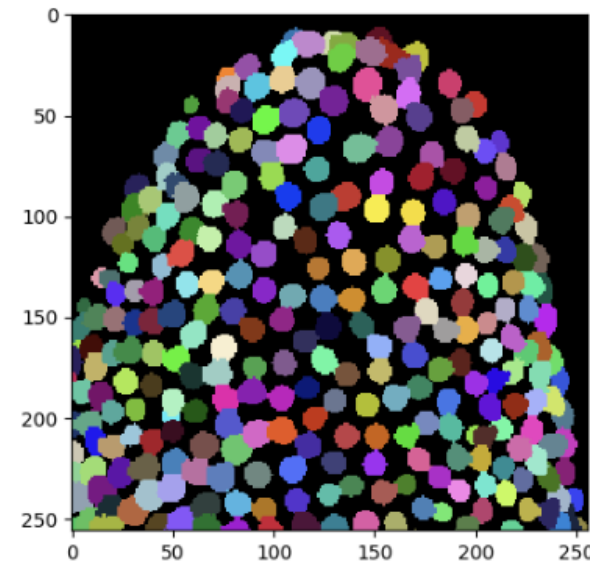


```
shape (100, 256, 256)  
dtype uint16  
size 12.5 MB  
min 125  
max 680
```



```
[12]: # Segment the image using voronoi otsu labeling  
label_image = cle.voronoi_otsu_labeling(processed_image, spot_sigma=2, outline_sigma=2)  
label_image
```

```
[12]:
```



cle.\_image

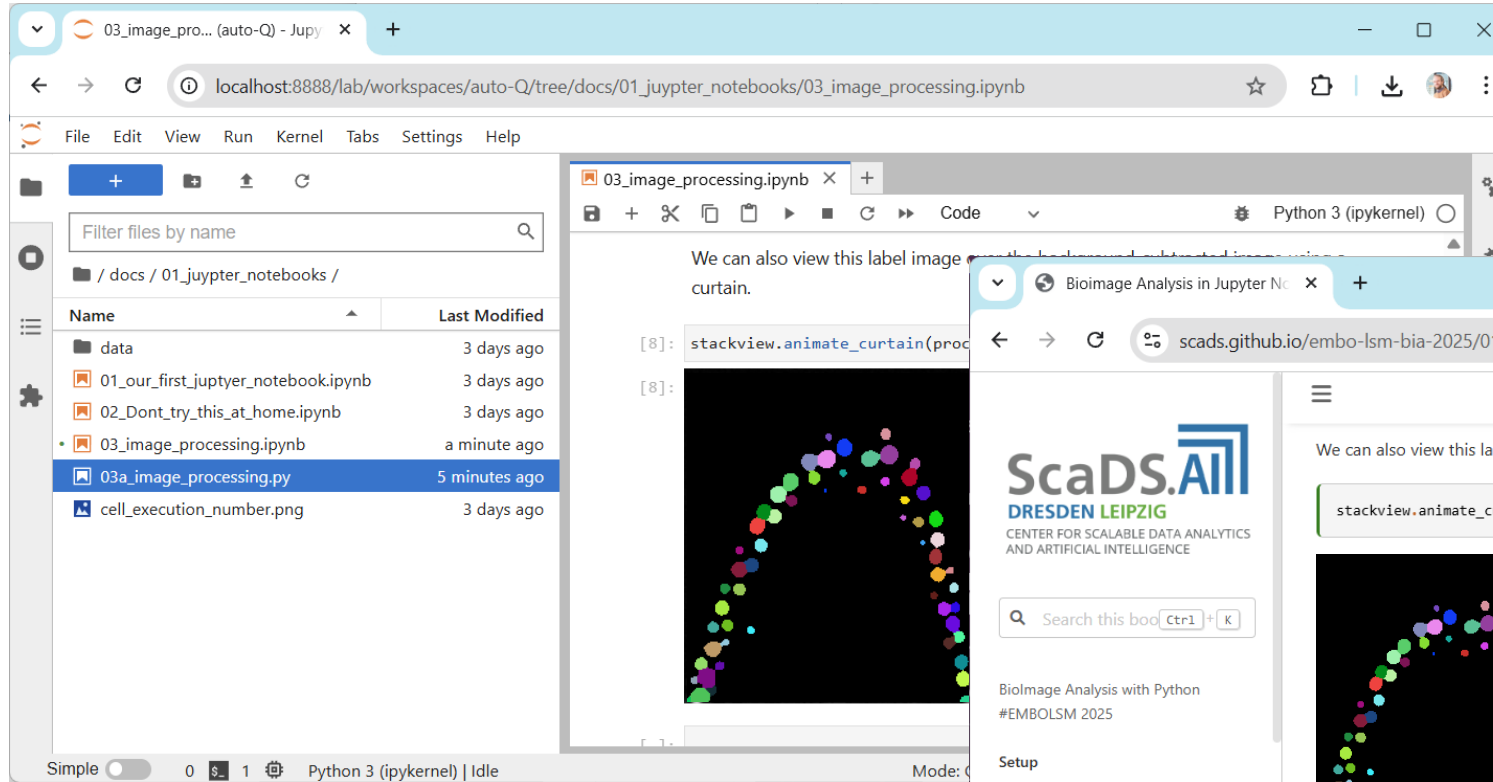
```
shape (100, 256, 256)  
dtype uint32  
size 25.0 MB  
min 0.0  
max 285.0
```

Technical:

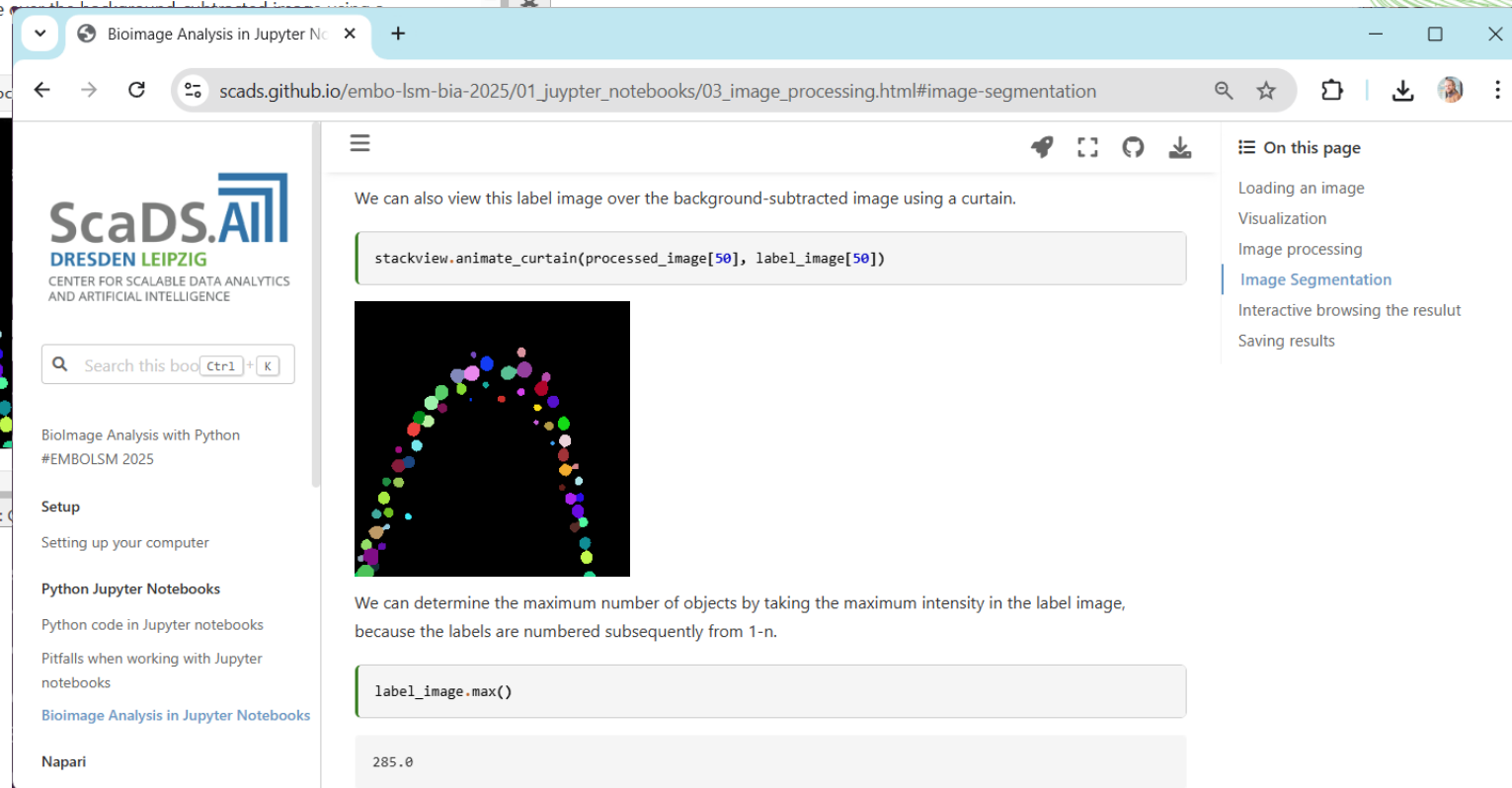
Inter-personal:

# Reproducibility : Python + Jupyter

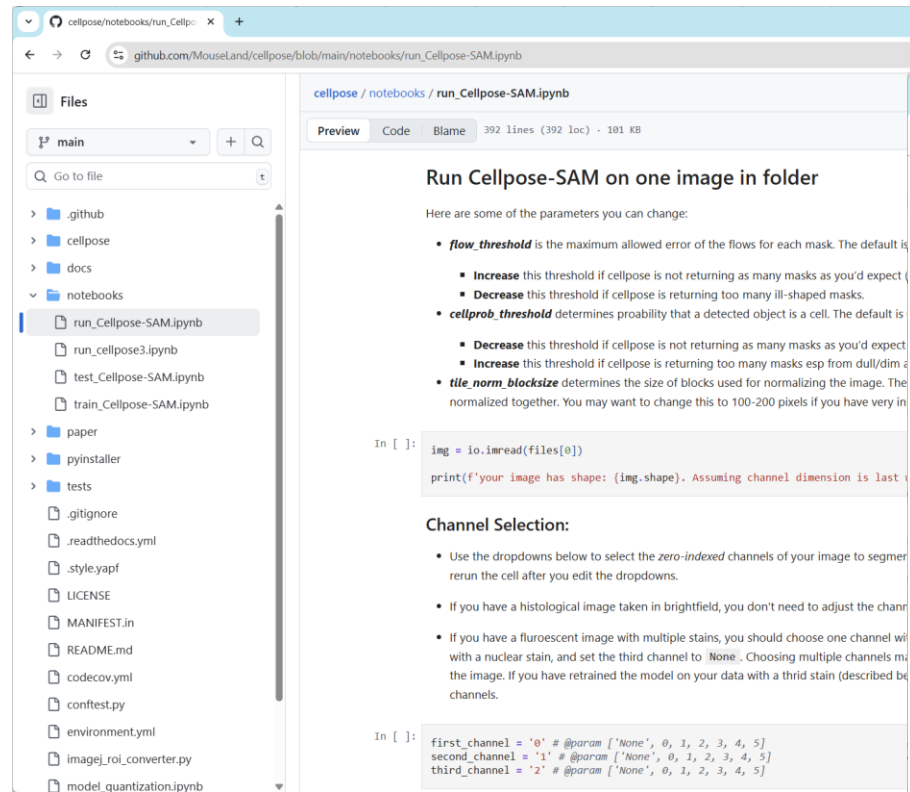
Other's view when  
you share the  
workflow online



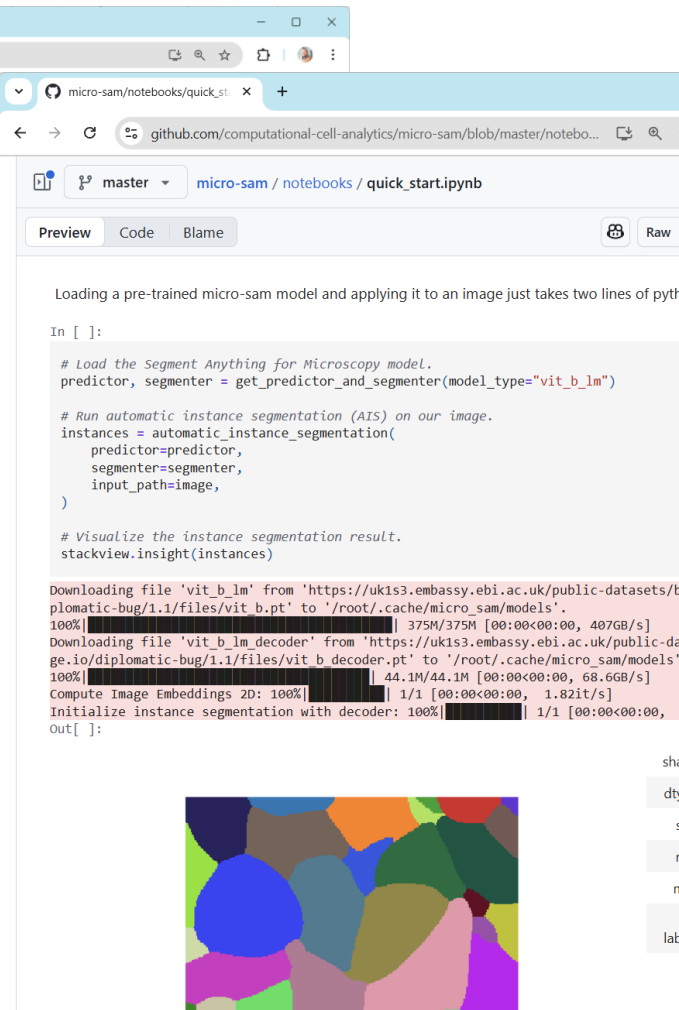
View on your laptop



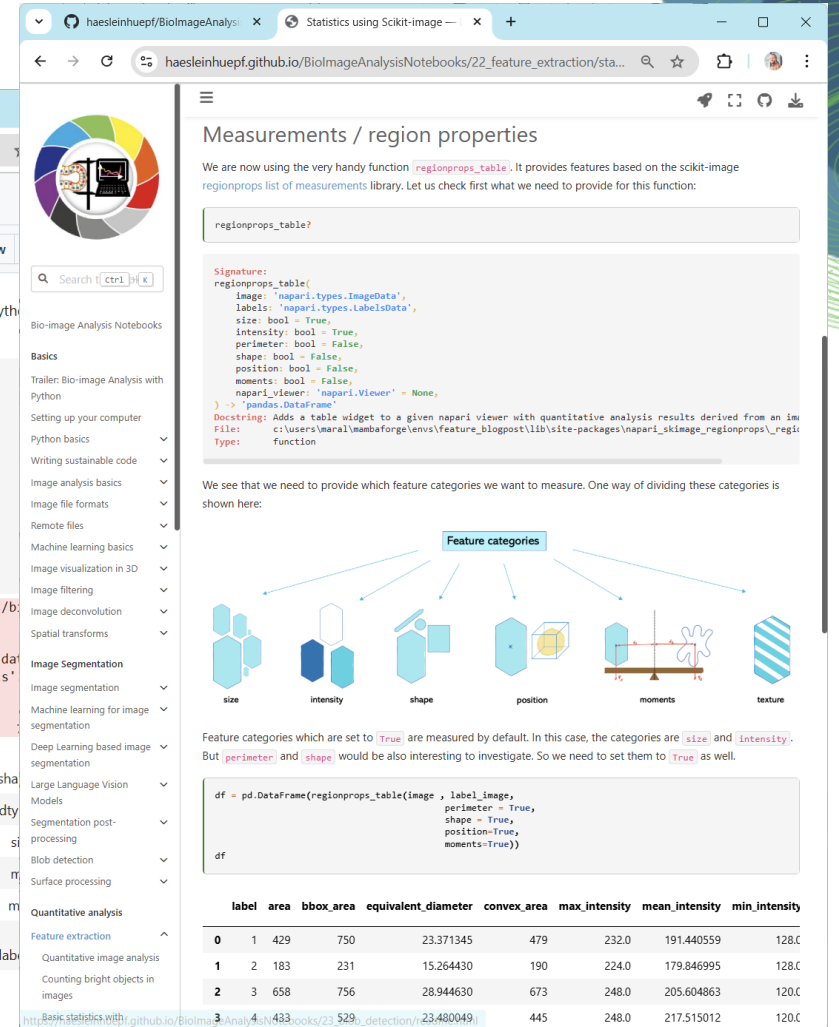
# Reproducibility : Python + Jupyter



[https://github.com/MouseLand/cellpose/blob/main/notebooks/run\\_Cellpose-SAM.ipynb](https://github.com/MouseLand/cellpose/blob/main/notebooks/run_Cellpose-SAM.ipynb)



[https://github.com/computational-cell-analytics/micro-sam/blob/master/notebooks/quick\\_start.ipynb](https://github.com/computational-cell-analytics/micro-sam/blob/master/notebooks/quick_start.ipynb)



[https://haesleinhuepf.github.io/BioImageAnalysisNotebooks/22\\_feature\\_extraction/statistics\\_with\\_scikit\\_image.html](https://haesleinhuepf.github.io/BioImageAnalysisNotebooks/22_feature_extraction/statistics_with_scikit_image.html)



# Training materials for today

The left panel shows the ScaDS website with a sidebar containing links to 'Biolmage Analysis with Python #EMBOLSM 2025', 'Setup', 'Python Jupyter Notebooks', and 'Napari'. The main content area is titled 'How to use these materials' and provides instructions on how to navigate the repository. The right panel shows the GitHub repository page for 'embo-lsm-bia-2025'. A green arrow points to the 'Code' button, and another green arrow points to the 'Download ZIP' option in the dropdown menu.

**ScaDS.DRESDEN LEIPZIG**  
CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Search

Biolmage Analysis with Python #EMBOLSM 2025

**Setup**  
Setting up your computer

**Python Jupyter Notebooks**  
Python code in Jupyter notebooks  
Pitfalls when working with Jupyter notebooks  
Bioimage Analysis in Jupyter Notebooks

**Napari**  
Interactive visualization with Napari  
The Napari Assistant  
Generating Jupyter Notebooks from the Napari Assistant  
Interactive pixel classification and object segmentation in Napari  
Interactive object classification in

**How to use these materials**  
On the top of the window, you find a Github-Button, which you can use to navigate the repository of the training materials.

Download the entire repository as ZIP and unzip the files in a place where you can find them. E.g. on your Desktop.

**GitHub repository: embo-lsm-bia-2025**  
Code Issues Pull requests Actions Projects Wiki Security Insights Settings

embo-lsm-bia-2025 Public

main

Go to file + <> Code

Local Codespaces

Clone

HTTPS SSH GitHub CLI

<https://github.com/ScaDS/embo-lsm-bia-2025.git>

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

Biolmage Analysis using Python tutorial at EMBO LSM 2025

scads.github.io/embo-lsm-bia-2...

Readme

CC-BY-4.0 license

Code of conduct

Activity

Custom properties

1 star

0 watching

The top panel shows a terminal window with the following commands and output:

```
conda activate embo25
jupyter lab
```

The bottom panel shows the Jupyter Lab interface. The file explorer on the left shows the following files and folders:

- data
- 01\_our\_first\_jupyter\_note...
- 02\_Dont\_try\_this\_at\_home...
- 03\_image\_processing.ipynb
- 03a\_image\_processing.py
- cell\_execution\_number.png

The main area displays the notebook '03\_image\_processing.ipynb' with the following content:

### Bioimage Analysis in Jupyter Notebooks

In this notebook, we will load an image stored in the file `data/lund.tif`, visualize it, process it using `pyclesperanto` for segmentation, and finally visualize the segmentation results.

We will use these libraries

- scikit-image
- numpy
- pyclesperanto
- stackview

# Reproducible Bio-Image Analysis using Python, Napari, Jupyter and AI

Robert Haase

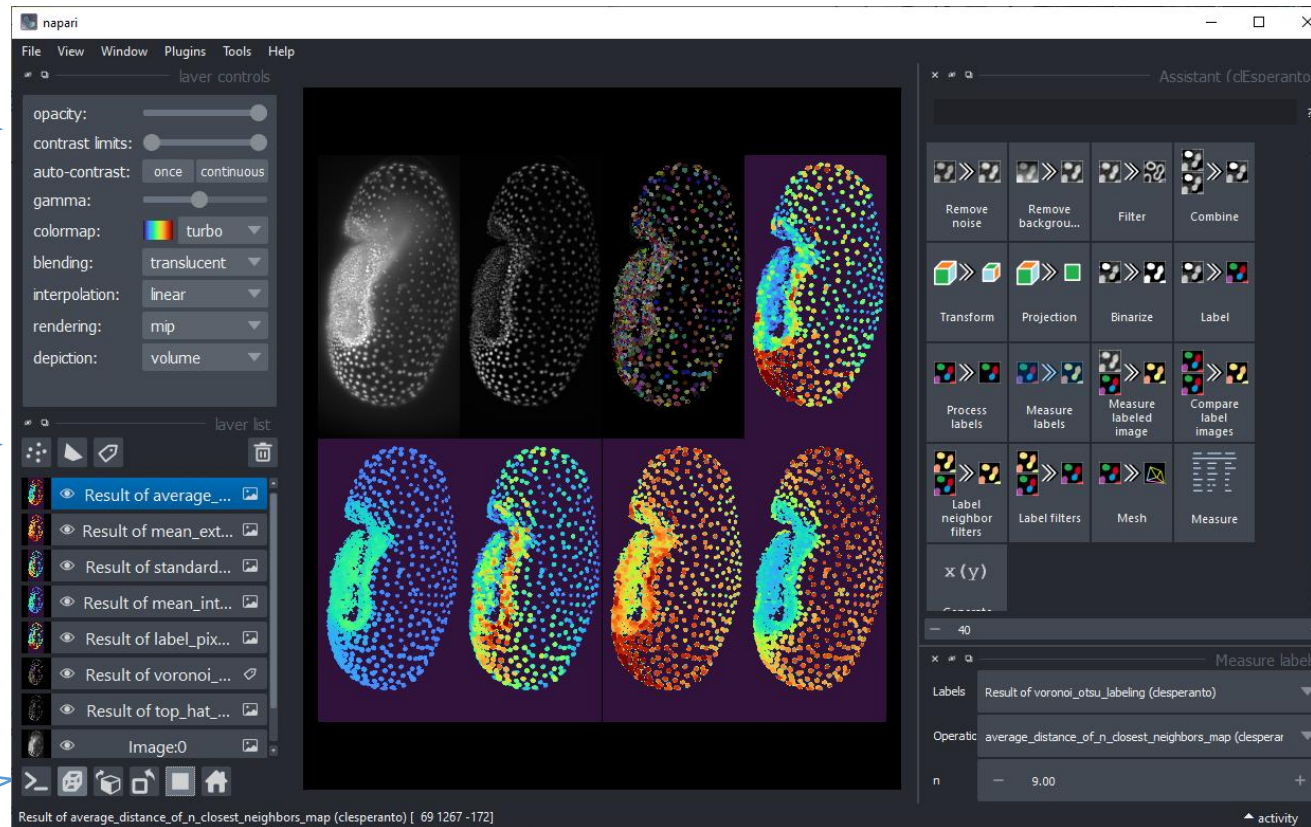
These slides and the related training materials can be reused under the  
terms of the [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) license unless mentioned otherwise.

# Napari – Graphical User Interface

View configuration /  
tools

Layers

Viewer controls



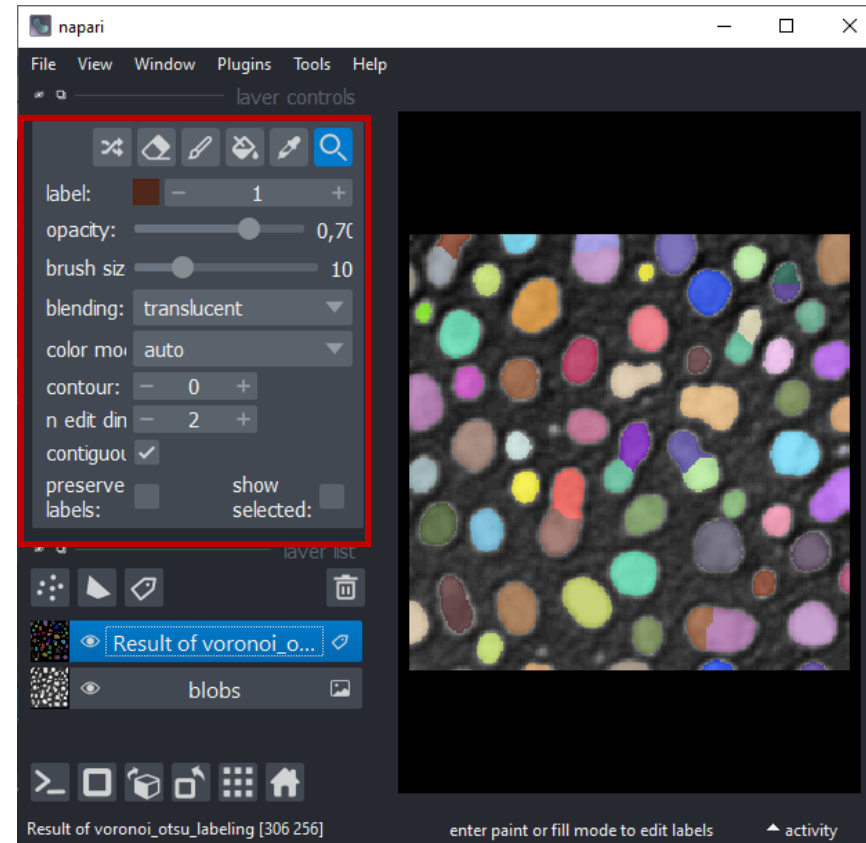
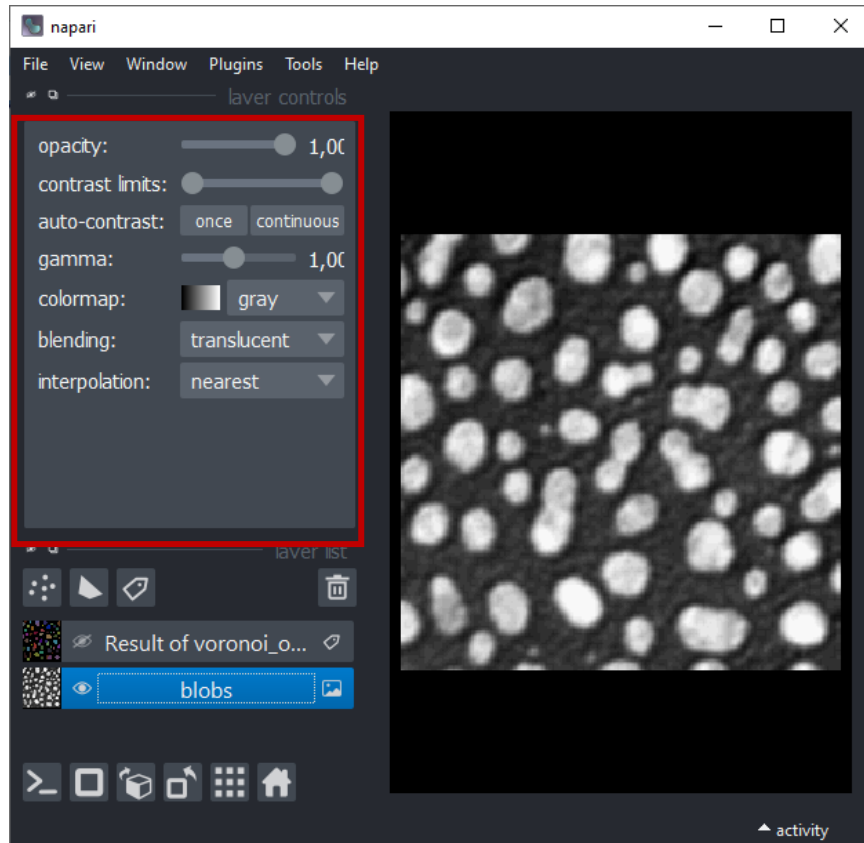
Dock widgets  
(custom plugins)

Function widgets  
(custom plugins)



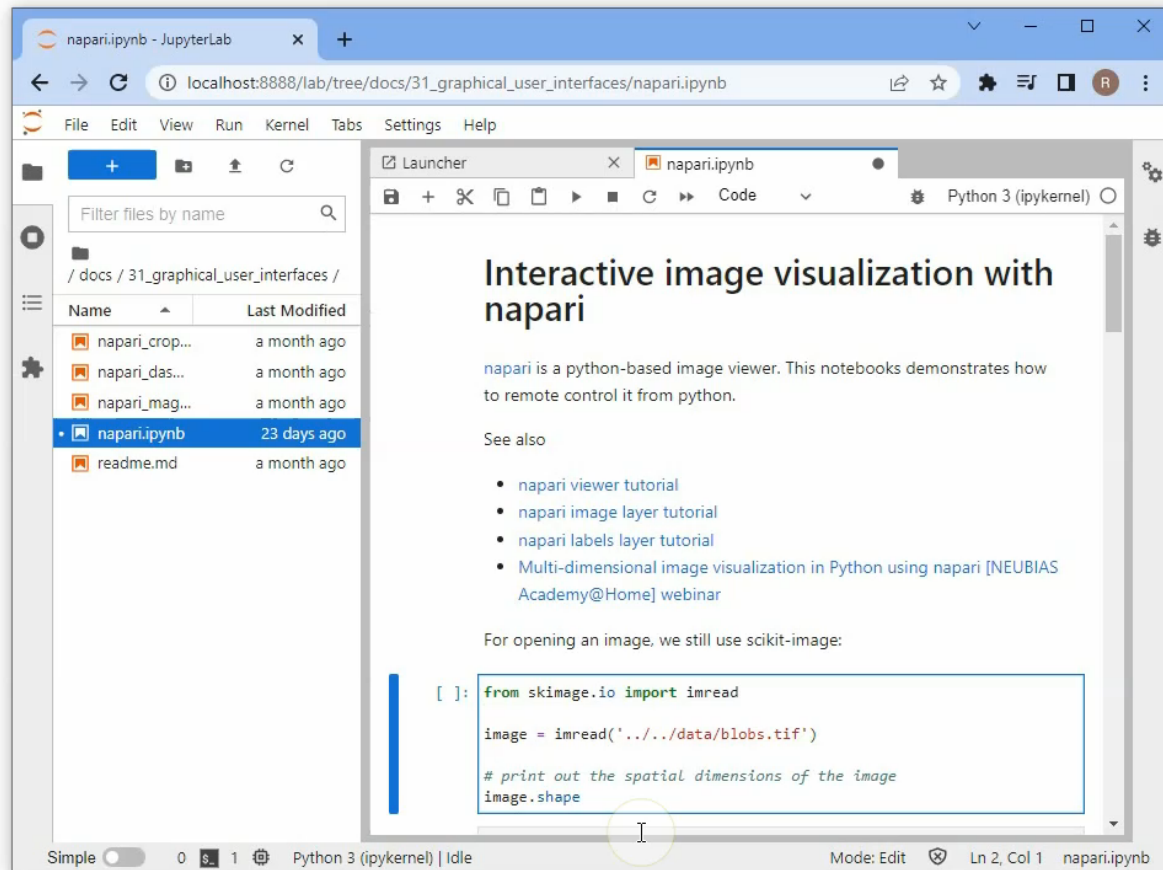
# Napari – Graphical User Interface

- Context / data type dependent tools



# Napari – Python Scripting

- Mixing interactivity and reproducibility



# Napari – Python Scripting

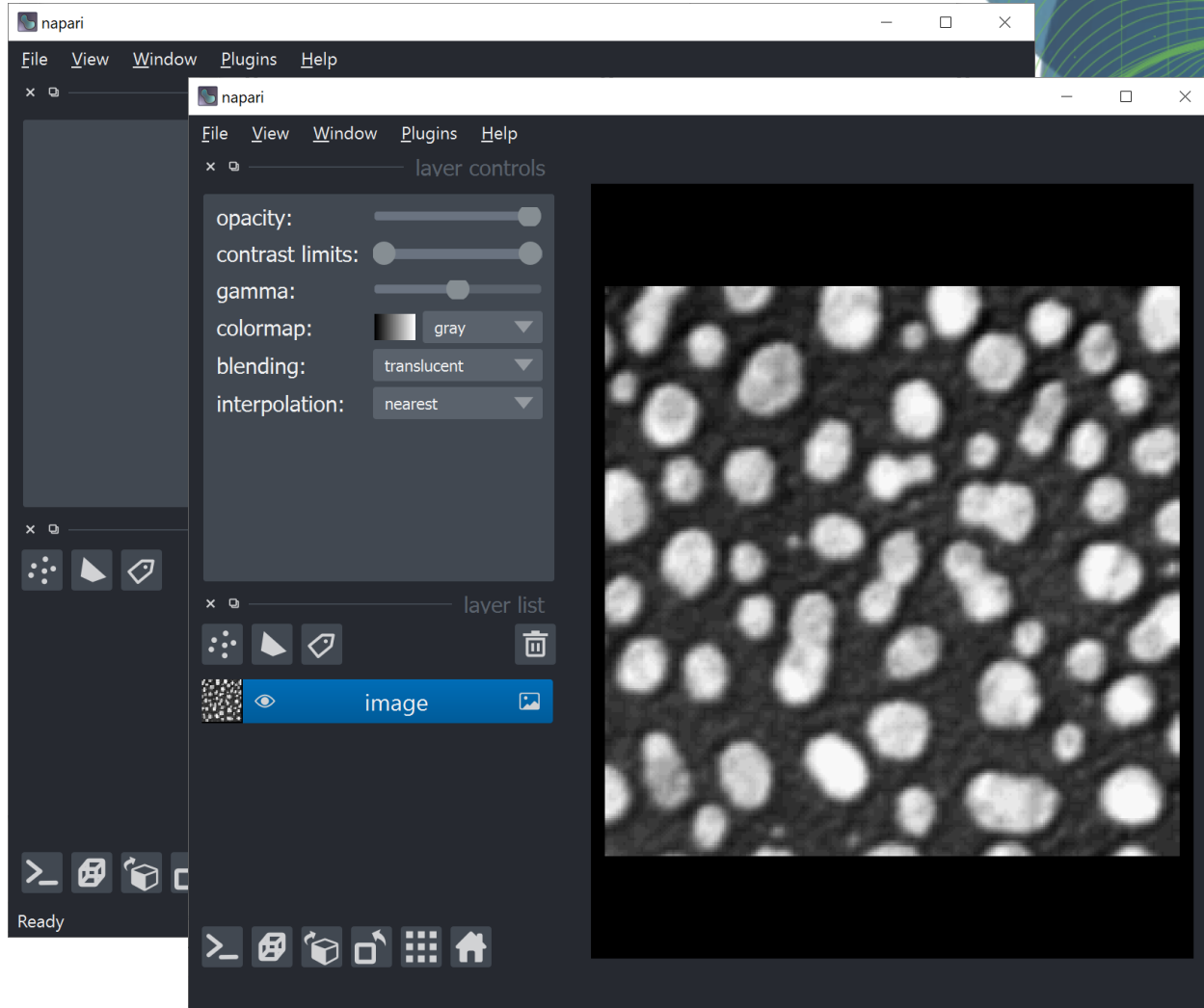
- Initialization

```
import napari
```

```
# Create an empty viewer  
viewer = napari.Viewer()
```

- Adding images

```
viewer.add_image(image)
```

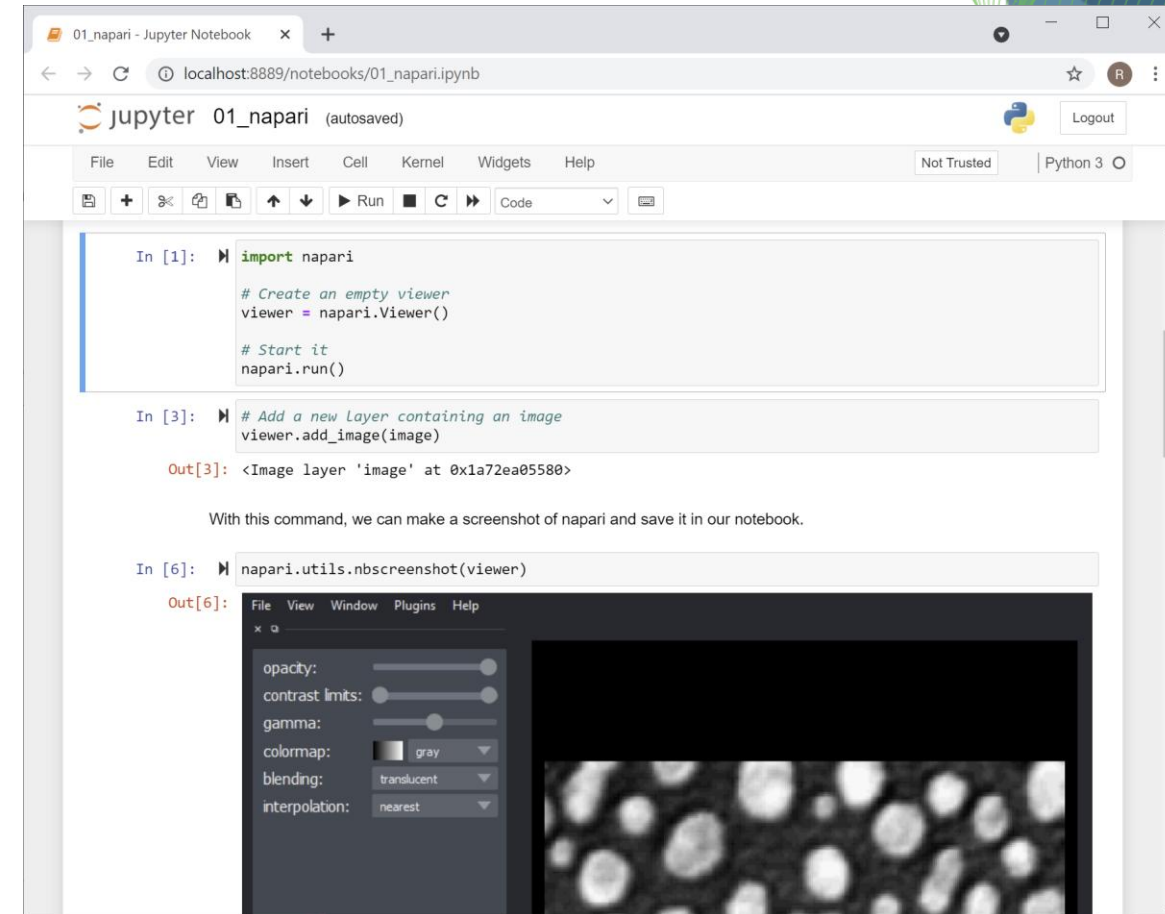


# Napari – Python Scripting

- Make screenshots from napari and put them in your jupyter notebook

```
napari.utils.nbscreenshot(viewer)
```

*Good for  
reproducibility*





# Napari – Python Scripting

- Removing layers

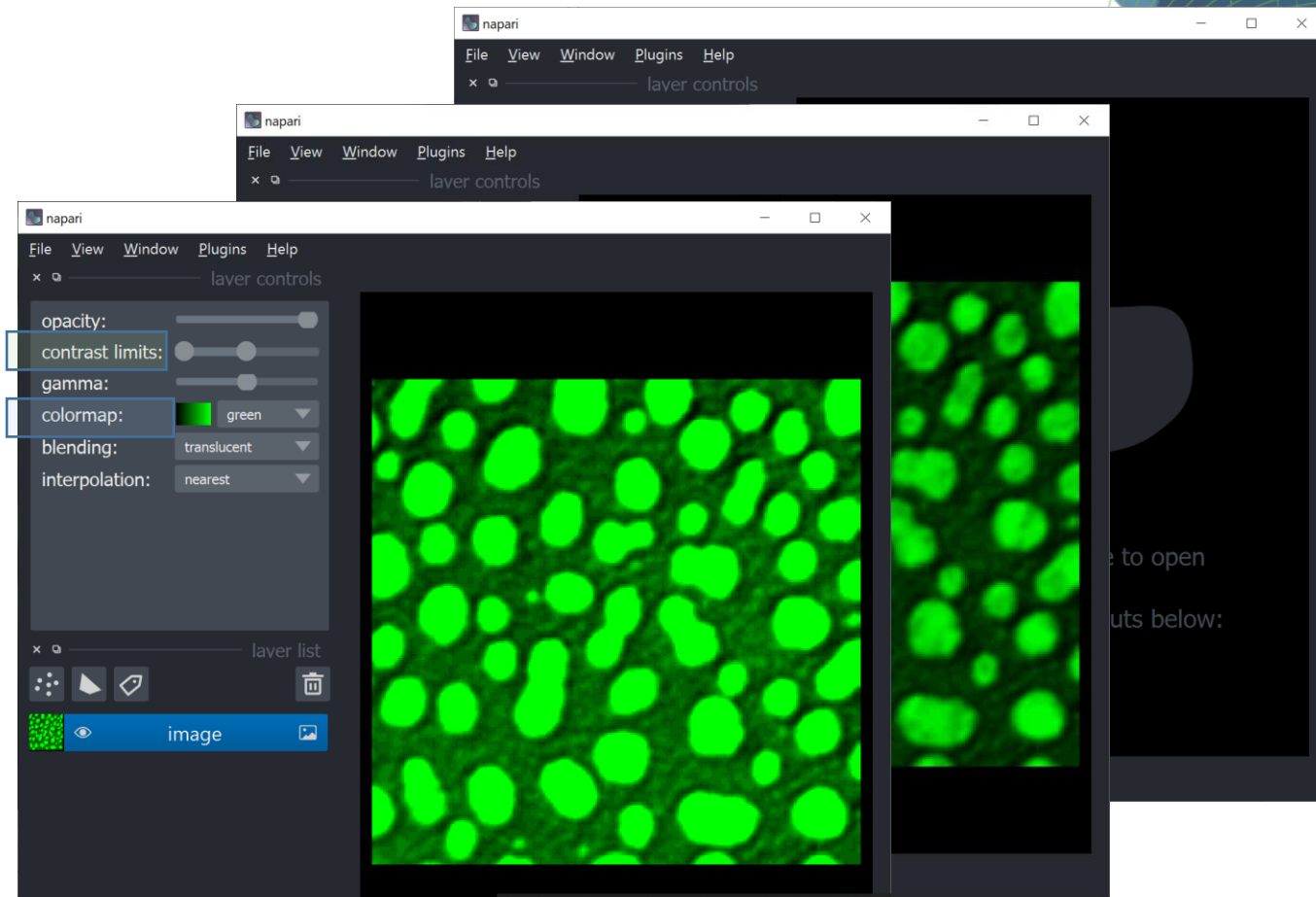
```
for l in viewer.layers:  
    viewer.layers.remove(l)
```

- Modify visualization while adding layers

```
viewer.add_image(image,  
                 colormap='green')
```

- Modify layers after adding

```
layer = viewer.add_image(image)  
layer.colormap = 'green'  
layer.contrast_limits = (0, 128)
```

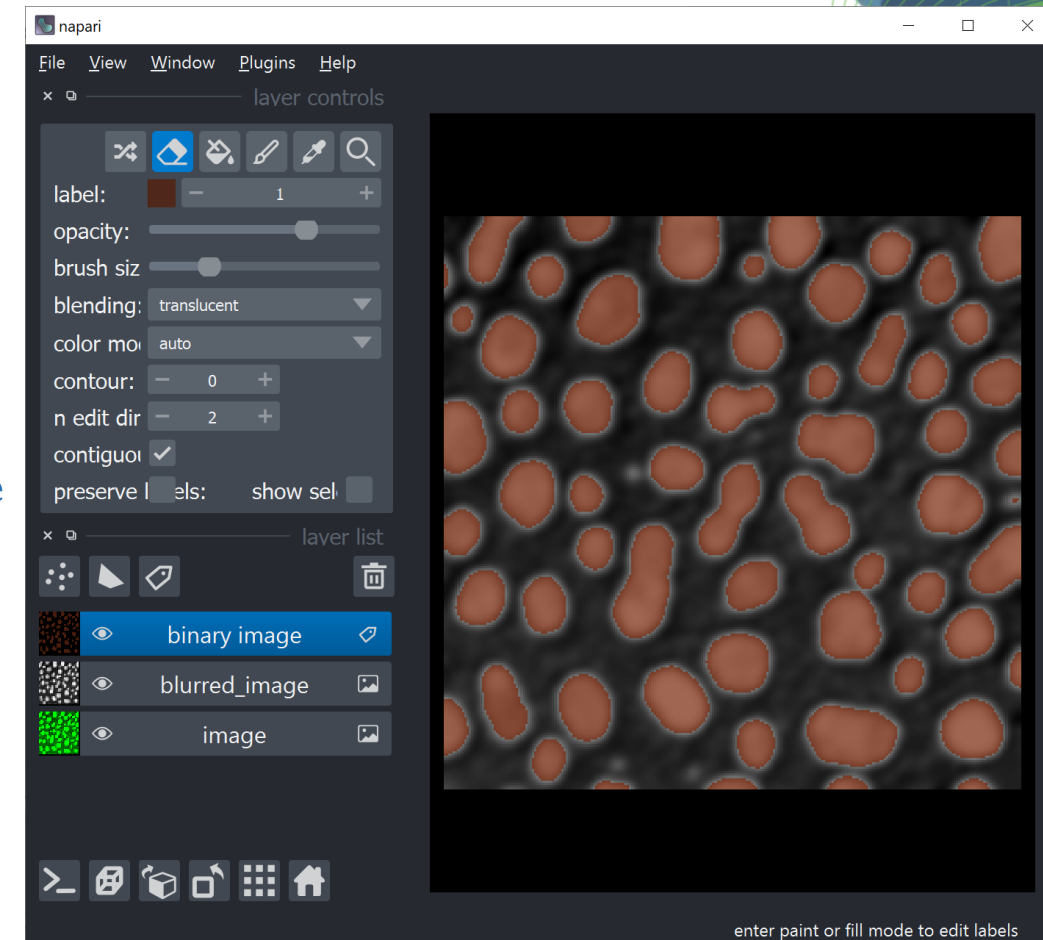


# Napari – Python Scripting

- Binary images and `label` images visualized as label layers

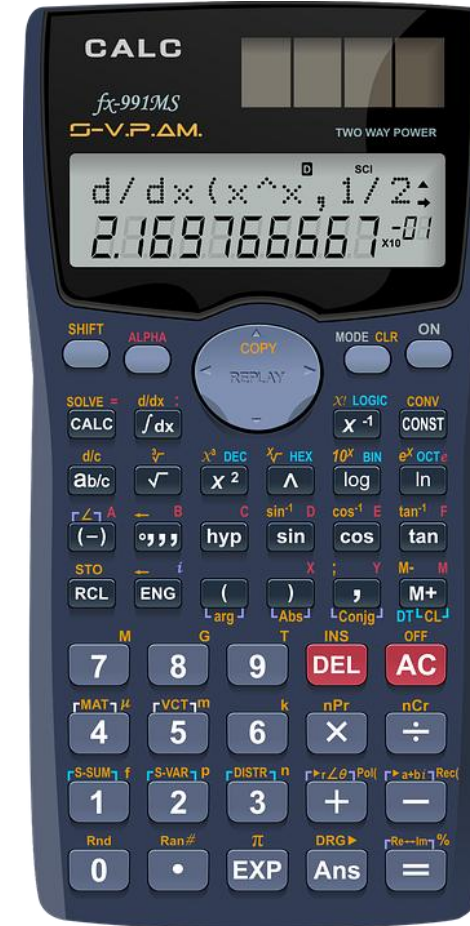
```
from skimage.filters import threshold_otsu
threshold = threshold_otsu(blurred_image)
binary_image = blurred_image > threshold

# Add a new labels layer containing an image
viewer.add_labels(binary_image)
```



# The Napari Assistant

- A pocket-calculator-like interface to build image analysis workflows



# The Napari Assistant

- Tools > Utilities > Assistant (na)

Viewer  
controls

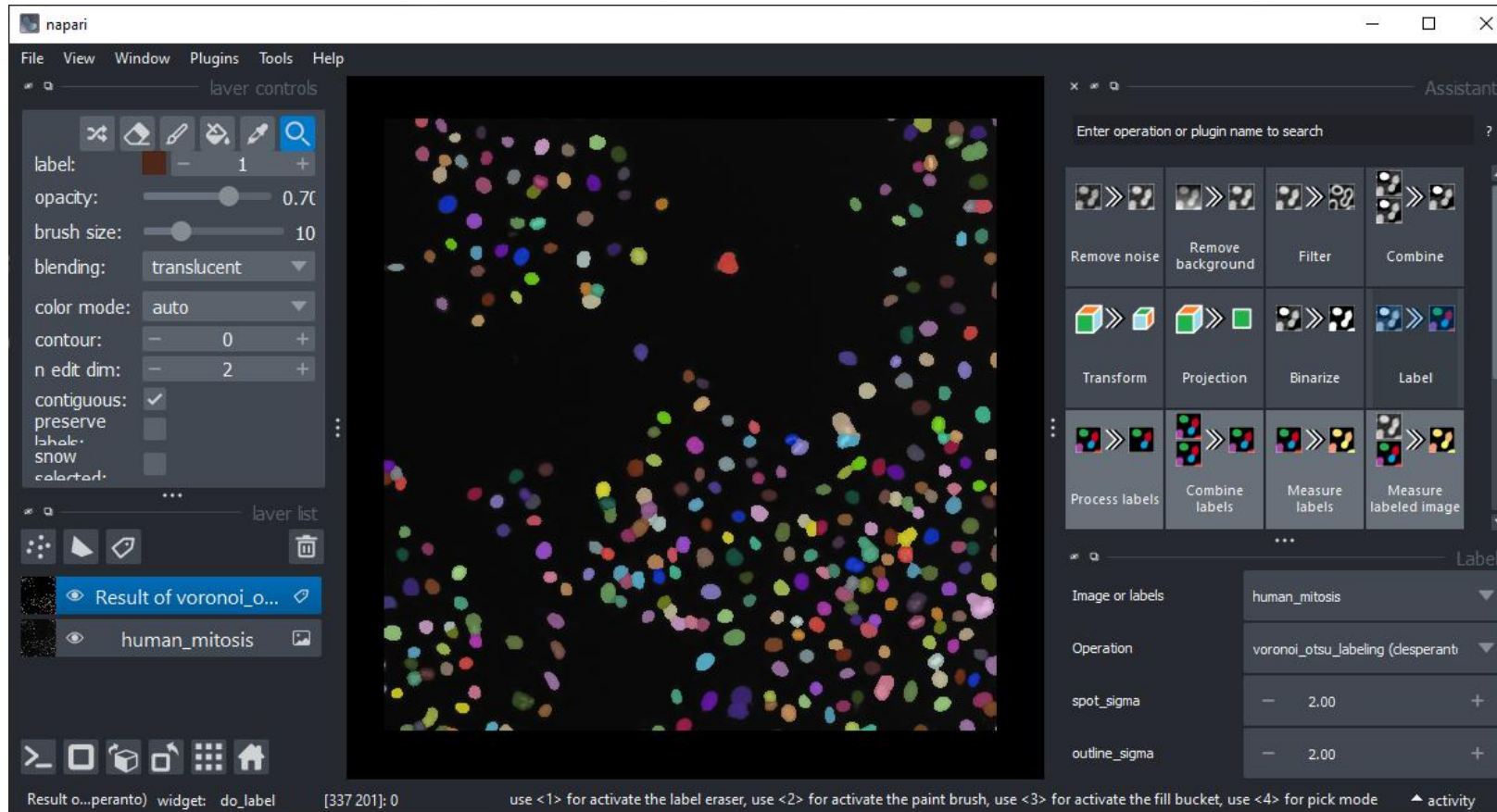


Image  
Processing

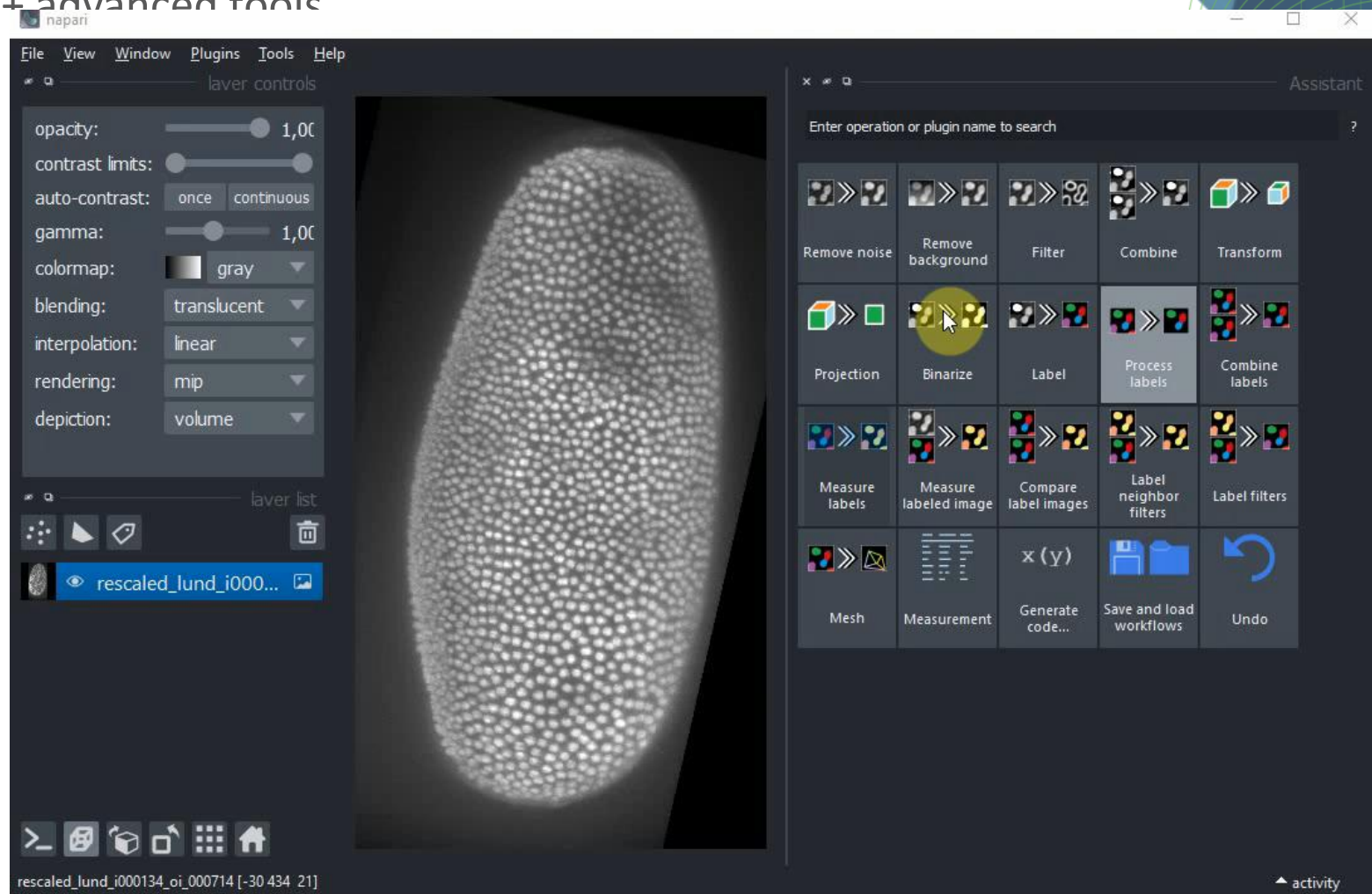


# The Napari Assistant

- Classical image processing operations + advanced tools
- Saving&loading supported
- Undo [redo]
- Hints for next steps
- ...

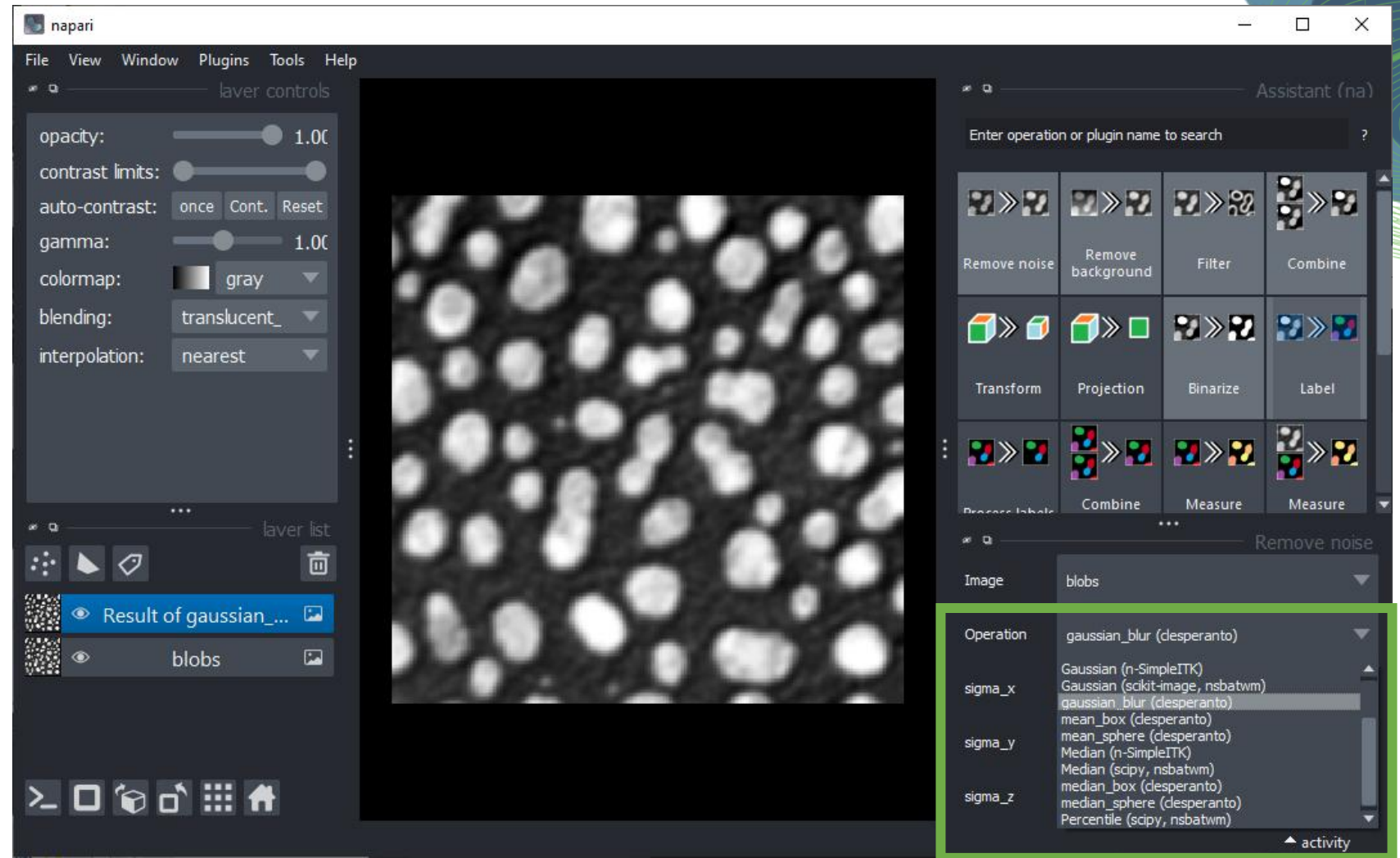


Ryan Savill ()  
@RyanSavill4



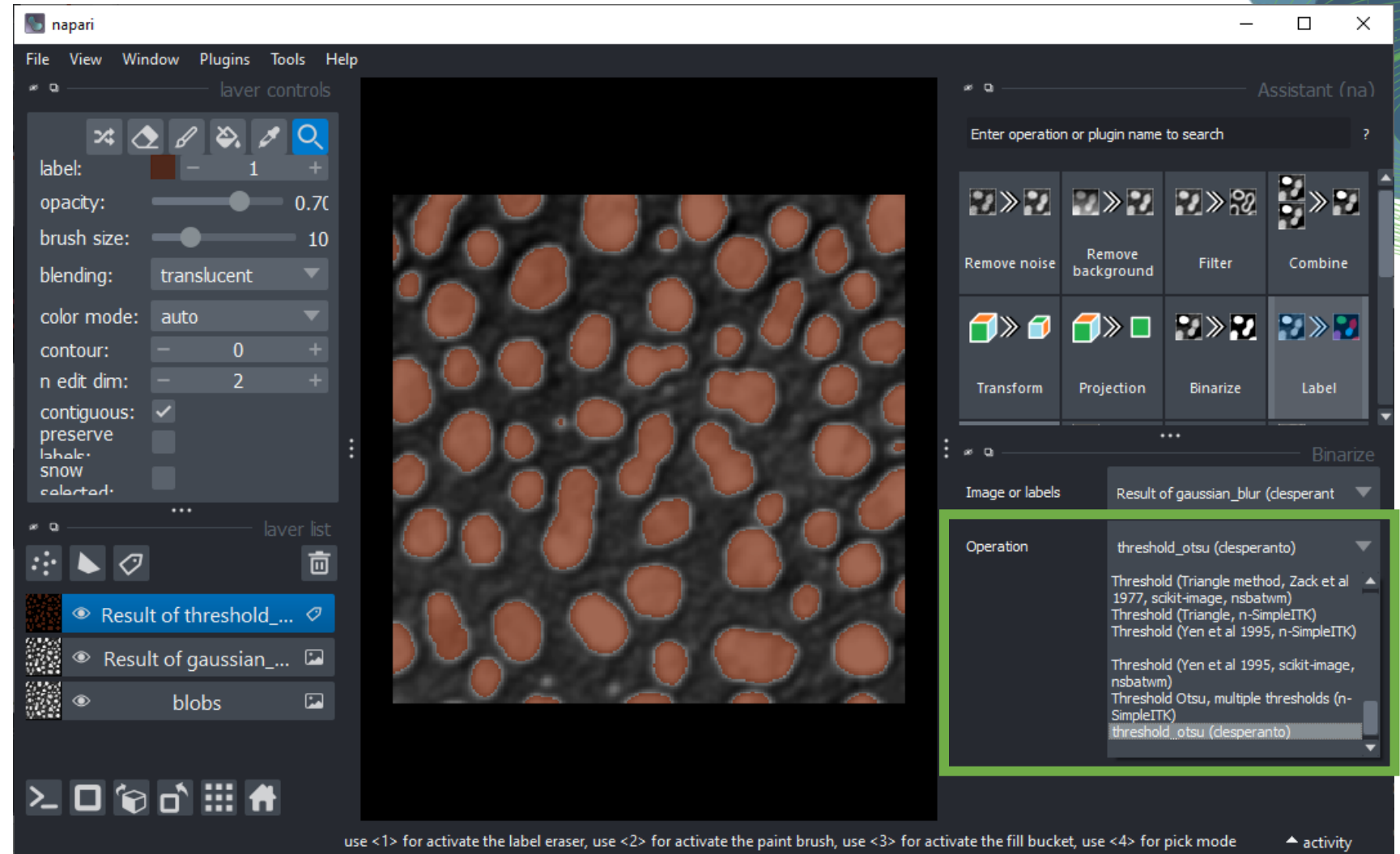
# Workflow building

- Try different algorithms, e.g. for removing noise
- Find them in the pulldown



# Workflow building

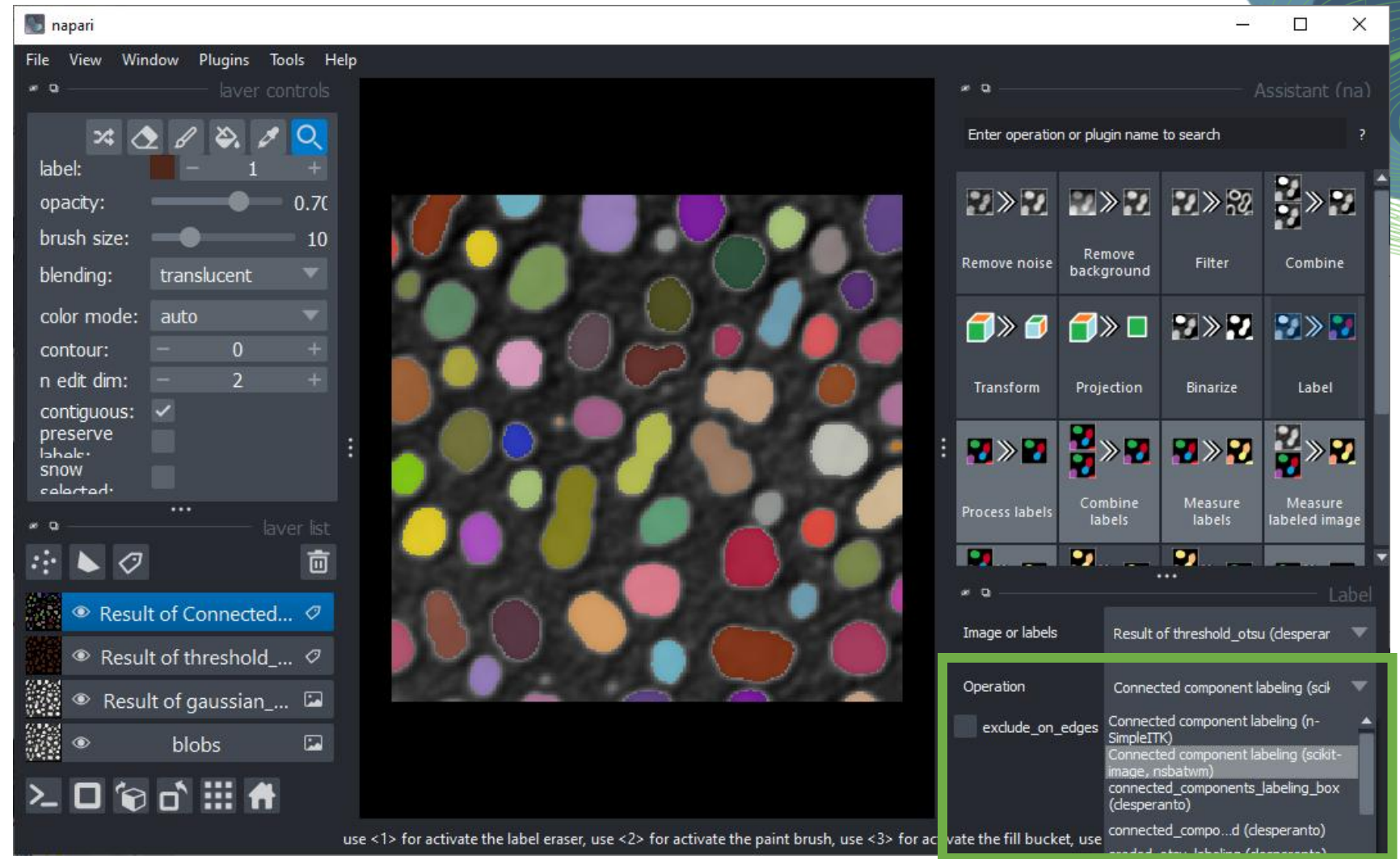
- Try different binarization algorithms





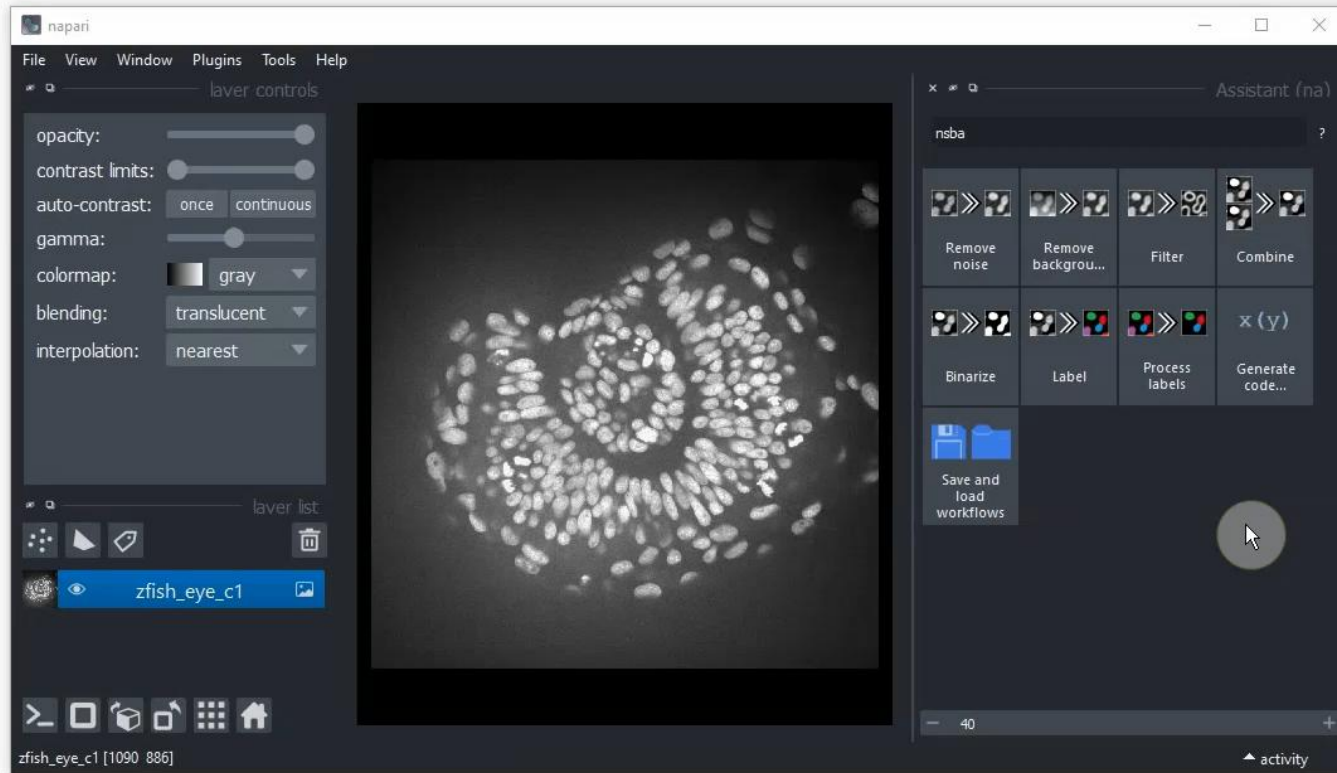
# Workflow building

- Multiple labeling algorithms available

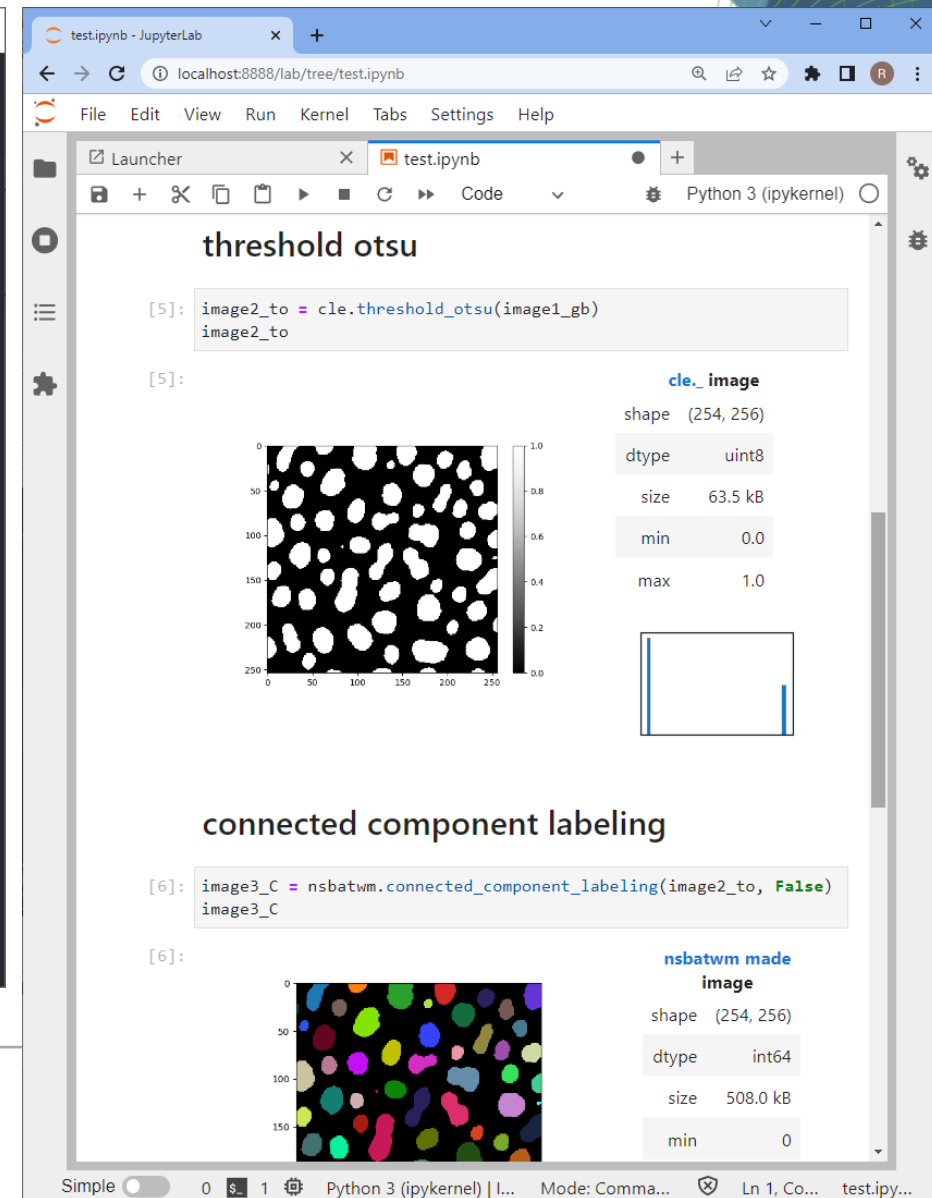
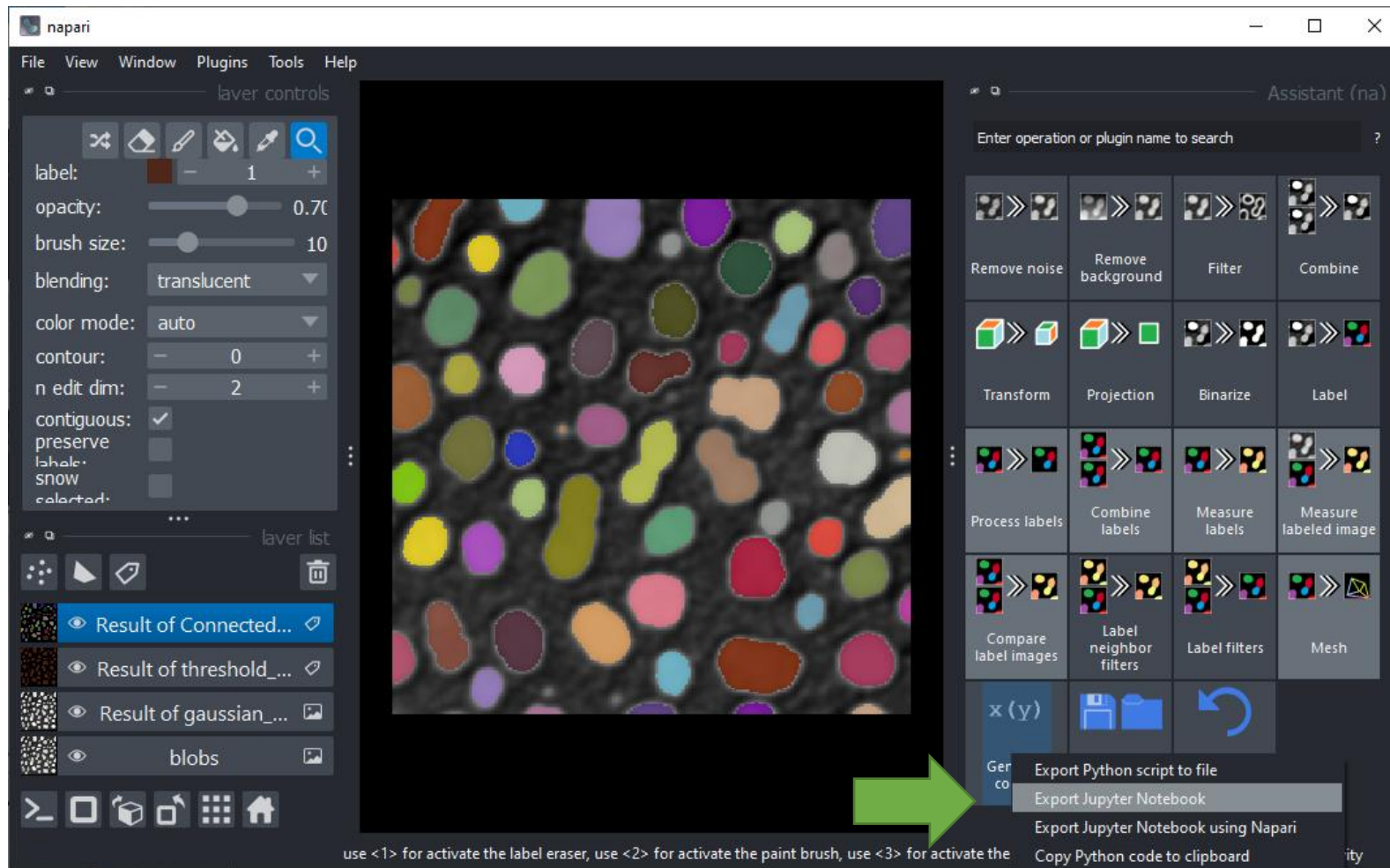




# Export code to Jupyter Notebooks



# Export code to Jupyter Notebooks



# Reproducible Bio-Image Analysis using Python, Napari, Jupyter and AI Robert Haase

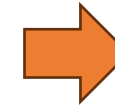
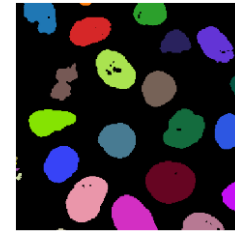
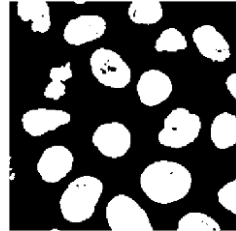
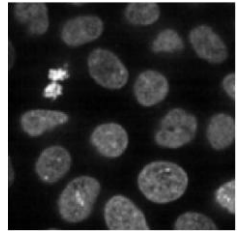
These slides and the related training materials can be reused under the  
terms of the [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) license unless mentioned otherwise.



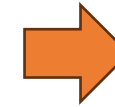


# How good are LLMs for Bio-image Analysis?

- Test case: segment the image and measure the average area of objects.



	area
0	955.0
1	31.0
2	815.0
3	1166.0
4	1135.0
...	



858.04

workflow\_segmentation\_measurement\_summary

1.0	1.0	1.0	0.9	1.0	0.8	0.9	0.0	0.3	0.4	0.5	0.0	0.6	0.1	0.4	0.4	0.1	0.5	0.0	0.1	0.1
reference	claude-3-5-sonnet-20240620	gpt-4o-2024-05-13	gpt-4-turbo-2024-04-09	claude-3-opus-20240229	gpt-4-1106-preview	gemini-1.5-pro-001	gpt-4o-mini-2024-07-18	llama3-70b-instruct-q8_0	llama3-70b-instruct-q4_0	gpt-3.5-turbo-1106	gemini-1.5-flash-001	codegemma-7b-instruct-fp16	mixtral-8x22b-instruct-v0.1-q4_0	mixtral-8x7b-instruct-v0.1-q5_0	phi3-3.8b-mini-instruct-4k-fp16	codellama-70b-instruct-q4_0	gemini-pro	llama3-8b-instruct-fp16	command-r-plus-104b-q4_0	codellama

Unit-test pass-rate (n=10):

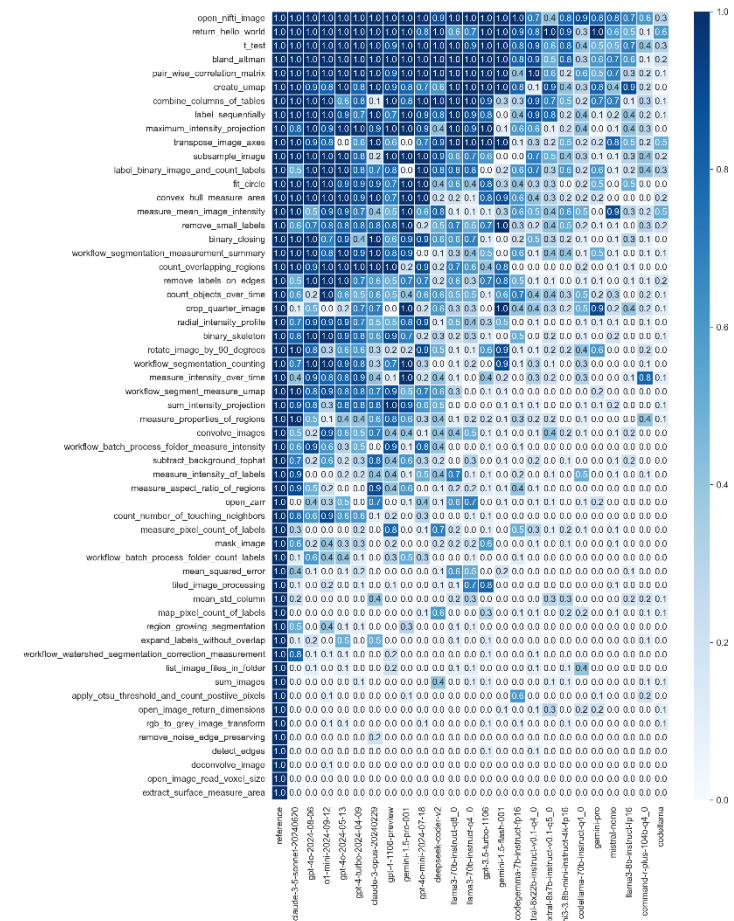
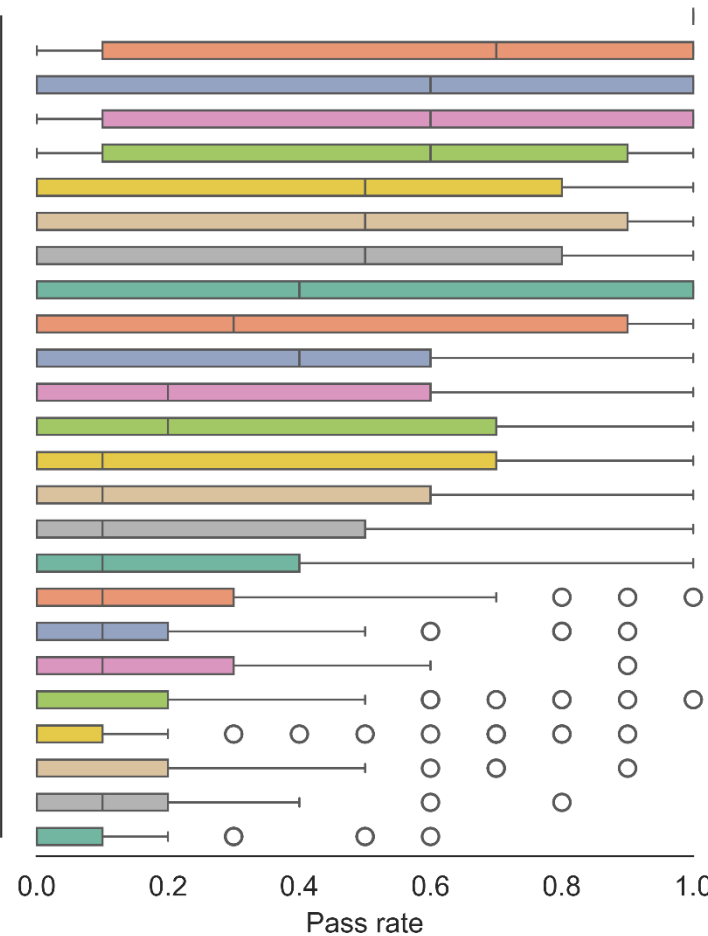
Large language models

# Benchmarking LLMs for Bio-image Analysis

- Summary: 57 use-cases (yet), 24 LLMs (yet), n=10



reference  
 claude-3-5-sonnet-20240620  
 gpt-4o-2024-08-06  
 o1-mini-2024-09-12  
 gpt-4o-2024-05-13  
 gpt-4-turbo-2024-04-09  
 claude-3-opus-20240229  
 gpt-4-1106-preview  
 gemini-1.5-pro-001  
 gpt-4o-mini-2024-07-18  
 deepseek-coder-v2  
 llama3-70b-instruct-q8\_0  
 llama3-70b-instruct-q4\_0  
 gpt-3.5-turbo-1106  
 gemini-1.5-flash-001  
 codegemma-7b-instruct-fp16  
 mixtral-8x22b-instruct-v0.1-q4\_0  
 mixtral-8x7b-instruct-v0.1-q5\_0  
 phi3-3.8b-mini-instruct-4k-fp16  
 codellama-70b-instruct-q4\_0  
 gemini-pro  
 mistral-nemo  
 llama3-8b-instruct-fp16  
 command-r-plus-104b-q4\_0  
 codellama



# Scientific code writing: bia-bob



- First steps: Keep it simple

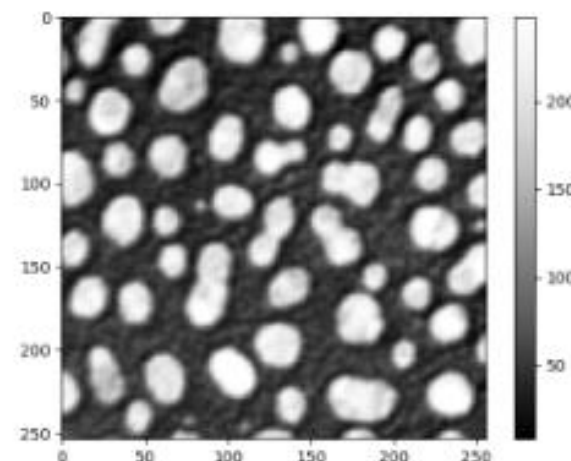
```
%bob Load the blobs.tif image file and show it.
```

I will load the `blobs.tif` image file and then display it using the appropriate visualization function.

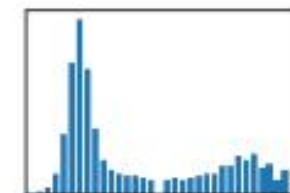
```
from skimage.io import imread
import stackview

# Load the image file
filename = 'blobs.tif'
image = imread(filename)

# Display the image
stackview.insight(image)
```



shape	(254, 256)
dtype	uint8
size	63.5 kB
min	8
max	248



# Scientific code writing: bia-bob

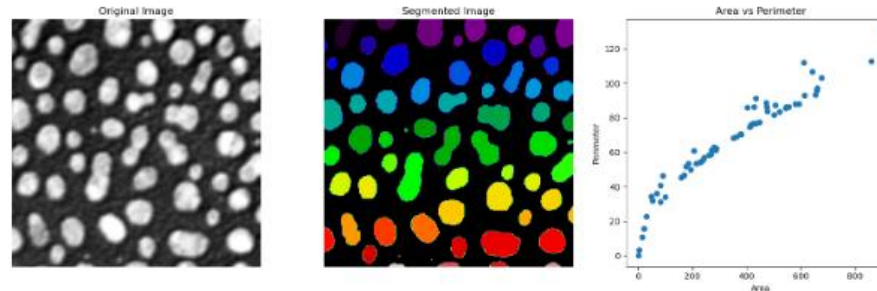


- After learning *how to prompt* for code: try more complex workflows

```
[1]: import bia_bob
```

```
[2]: %%bob
* Load blobs.tif from disk
* Segment the bright objects on the dark background
* Measure area and perimeter of the objects
* Show the original image, the segmented image and an
  area versus perimeter plot side-by-side using subfigures.
```

This notebook may contain text, code and images generated by artificial intelligence. Used model: claude-3-5-sonnet-20240620, vision model: claude-3-5-sonnet-20240620, endpoint: None, bia-bob version: 0.21.0.. Do not enter sensitive or private information and verify generated contents according to good scientific practice. Read more: <https://github.com/haesleinhuepf/bia-bob#disclaimer>



	label	area	perimeter
0	1	433.0	91.254834
1	2	185.0	53.556349
2	3	658.0	95.698485
3	4	434.0	76.870058
4	5	477.0	83.798990

```
import numpy as np
from skimage import io, filters, measure, segmentation
import matplotlib.pyplot as plt
import pandas as pd
import pycyclesperanto_prototype as cle
import stackview

# Load the image
image = io.imread('blobs.tif')

# Segment the bright objects
thresh = filters.threshold_otsu(image)
binary = image > thresh

# Label the objects
labels = measure.label(binary)

# Measure properties
properties = ['label', 'area', 'perimeter']
df = pd.DataFrame(measure.regionprops_table(labels, properties=properties))

# Create the figure with subfigures
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 5))

# Original image
ax1.imshow(image, cmap='gray')
ax1.set_title('Original Image')
ax1.axis('off')

# Segmented image
ax2.imshow(labels, cmap='nipy_spectral')
ax2.set_title('Segmented Image')
ax2.axis('off')

# Scatter plot
ax3.scatter(df['area'], df['perimeter'])
ax3.set_xlabel('Area')
ax3.set_ylabel('Perimeter')
ax3.set_title('Area vs Perimeter')

plt.tight_layout()
plt.show()

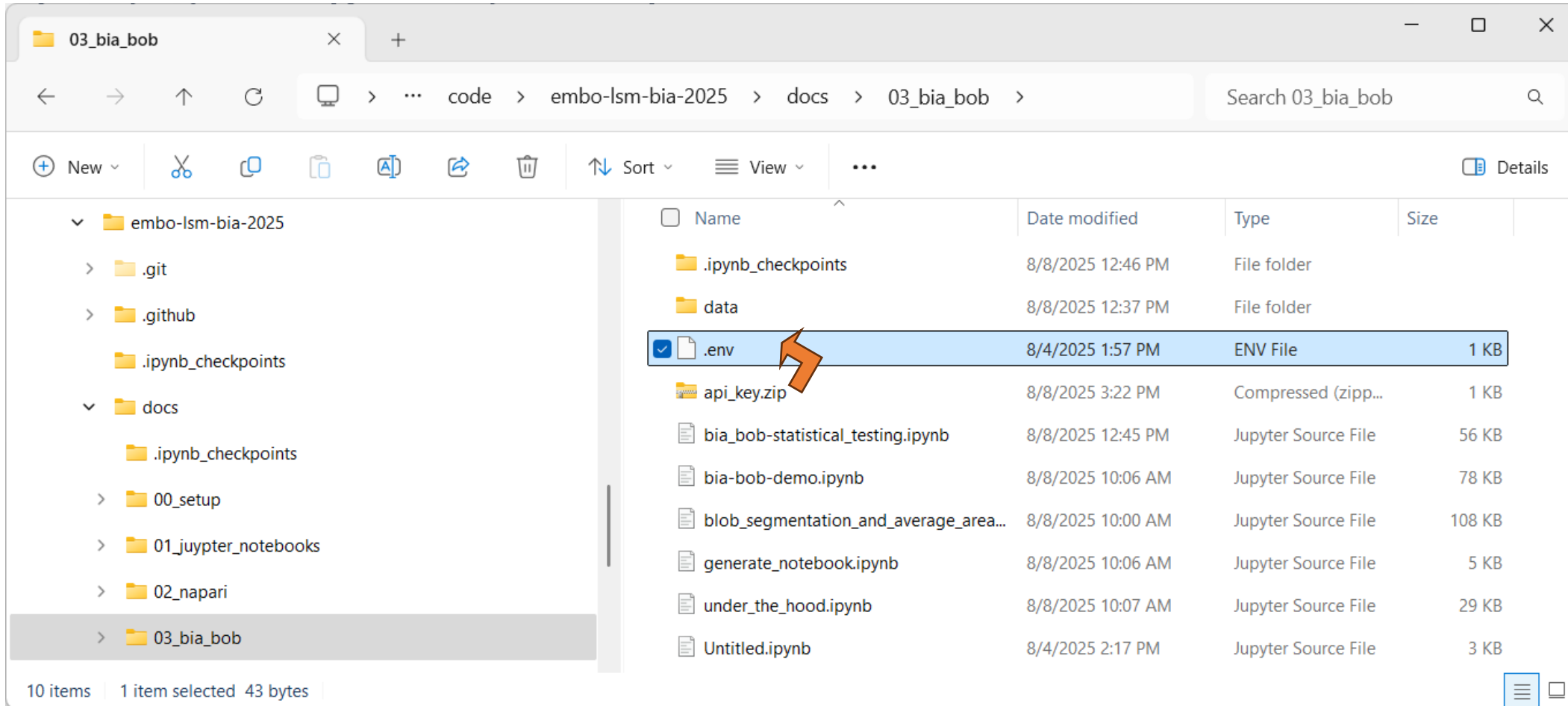
# Display the dataframe
print(df.head())
```



# Setup: bia-bob



- Unzip the “.env” file out of “api\_key.zip”.

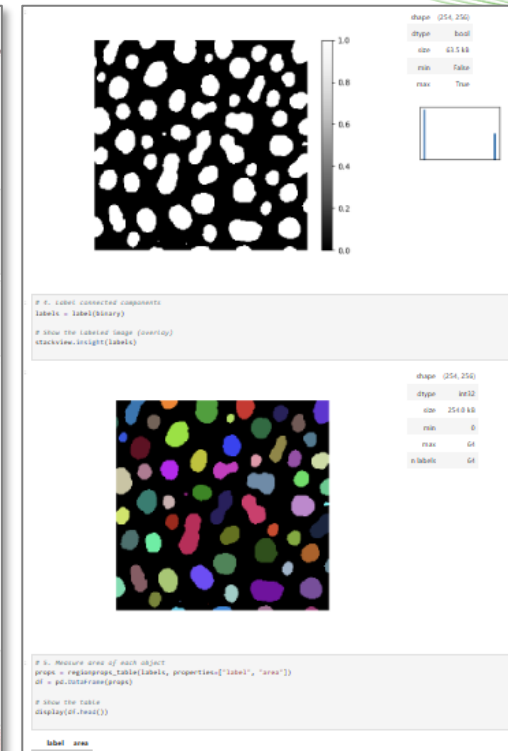
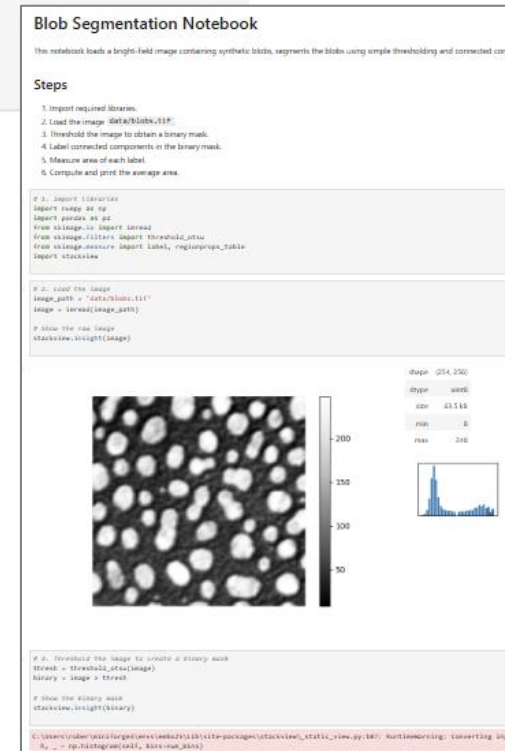


# Generating Jupyter notebooks

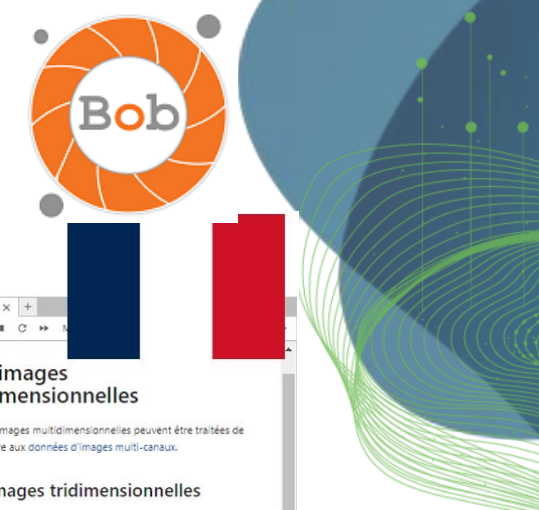
- Ask %%bob to generate a Jupyter notebook

```
%%bob generate a jupyter notebook that
* loads data/blobs.tif
* segments the bright blobs using thresholding and connected component labeling
* measures the area of the objects
* prints out the average area
```

A notebook has been saved as `blob_segmentation_and_average_area.ipynb`.



# Translating notebooks



“Please translate  
notebook  
<xyz.ipynb> to  
<language>.”



### Multidimensional image stacks

Multidimensional image data can be handled in a similar multi-channel image data.

### 3-dimensional image stacks

There are also images with three spatial dimensions: X, Y, and Z. Typical examples in microscopy and in medical imaging: a look at an Magnetic Resonance Imaging (MRI) data.

```
from skimage.io import imread
from stackview import imshow
import matplotlib.pyplot as plt
image_stack = imread('.../data/Haase_MRT_tf13d1.tif')
```

### Image slicing

Since three dimensions are spatial dimensions, we can also make orthogonal to the image plane and corresponding to anatomical planes. To orient the images correctly, we can use the axes by adding '.T' by the end.

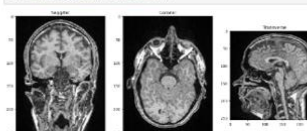
```
sagittal = image_stack[:, :, 128].T
coronal = image_stack[:, 128, :].T
transverse = image_stack[96]
```

```
fig, axs = plt.subplots(1, 3, figsize=(15,15))

# show orthogonal planes
axs[0].imshow(sagittal, cmap='Greys_r')
axs[0].set_title('Sagittal')

axs[1].imshow(coronal, cmap='Greys_r')
axs[1].set_title('Coronal')

axs[2].imshow(transverse, cmap='Greys_r')
axs[2].set_title('Transverse');
```



### Videos

If an image dataset has a temporal dimension, we call it a video. Processing videos works similar to multi-channel images and image stacks. Let's open a microscopy dataset showing yeast.



### Multidimensional Bildstapel

Multidimensionale Bilddaten können ähnlich wie mehrkanalige Bilddaten behandelt werden.

### Dreidimensionale Bildstapel

Es gibt auch Bilder mit drei räumlichen Dimensionen: X, Y und Z. Typische Beispiele finden sich in der Mikroskopie und in der medizinischen Bildgebung. Schauen wir uns ein Magnetresonanztomographie (MRT) Datensatz an:

```
[1]: from skimage.io import imread
      from stackview import imshow
      import matplotlib.pyplot as plt
      image_stack = imread('.../data/Haase_MRT_tf13d1.tif')
```

### Bildschnitte

Da alle drei Dimensionen räumliche Dimensionen sind, können wir auch Schnitte orthogonal zur Bildebene machen, die den Anatomischen Ebenen entsprechen. Um die Bilder korrekt zu orientieren, können wir ihre Achsen transponieren, indem wir '.T' hinzufügen.

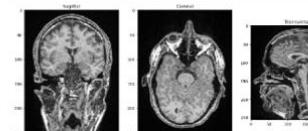
```
[2]: sagittal = image_stack[:, :, 128].T
      coronal = image_stack[:, 128, :].T
      transverse = image_stack[96]
```

```
fig, axs = plt.subplots(1, 3, figsize=(15,15))

# orthogonale Ebenen anzeigen
axs[0].imshow(sagittal, cmap='Greys_r')
axs[0].set_title('Sagittal')

axs[1].imshow(coronal, cmap='Greys_r')
axs[1].set_title('Coronal')

axs[2].imshow(transverse, cmap='Greys_r')
axs[2].set_title('Transversal');
```



### Videos

Wenn ein Bilddatensatz eine zeitliche Dimension hat, nennen wir ihn ein Video. Die Verarbeitung von Videos funktioniert ähnlich wie die mehrkanaligen Bilder und Bildstapel. Öffnen wir einen Mikroskopie-



### Pilas de imágenes multidimensionales

Los datos de imágenes multidimensionales se pueden manejar de manera similar a los datos de imágenes multicanal.

### Pilas de imágenes tridimensionales

También hay imágenes con tres dimensiones espaciales: X, Y y Z. Puedes encontrar ejemplos típicos en microscopía y en imágenes médicas. Echemos un vistazo a un conjunto de datos de imágenes por Resonancia Magnética (IRM):

```
[1]: from skimage.io import imread
      from stackview import imshow
      import matplotlib.pyplot as plt
      image_stack = imread('.../data/Haase_MRT_tf13d1.tif')
```

### Corte de imágenes

Como las tres dimensiones son dimensiones espaciales, también podemos hacer cortes ortogonales al plano de la imagen y correspondientes a Planos Anatómicos. Para orientar correctamente las imágenes, podemos transponer sus ejes añadiendo '.T' al final.

```
[2]: sagittal = image_stack[:, :, 128].T
      coronal = image_stack[:, 128, :].T
      transverse = image_stack[96]
```

```
fig, axs = plt.subplots(1, 3, figsize=(15,15))

# mostrar planos ortogonales
axs[0].imshow(sagittal, cmap='Greys_r')
axs[0].set_title('Sagital')

axs[1].imshow(coronal, cmap='Greys_r')
axs[1].set_title('Coronal')

axs[2].imshow(transverse, cmap='Greys_r')
axs[2].set_title('Transversal');
```



### Videos

Si un conjunto de datos de imágenes tiene una dimensión temporal, lo llamamos video. Procesar videos funciona de manera similar a las imágenes multicanal u a las pilas de imágenes. Vamos a abrir un



### Piles d'images multidimensionnelles

Les données d'images multidimensionnelles peuvent être traitées de manière similaire aux données d'images multi-canaux.

### Piles d'images tridimensionnelles

Il existe aussi des images avec trois dimensions spatiales: X, Y et Z. Vous trouvez des exemples typiques en microscopie et en imagerie médicale. Regardons un ensemble de données d'imagerie par résonance magnétique (IRM):

```
[1]: from skimage.io import imread
      from stackview import imshow
      import matplotlib.pyplot as plt
      image_stack = imread('.../data/Haase_MRT_tf13d1.tif')
```

### Coupe d'image

Comme les trois dimensions sont des dimensions spatiales, nous pouvons également faire des coupes orthogonales au plan de l'image et correspondant aux plans anatomiques. Pour orienter correctement les images, nous pouvons transposer leurs axes en ajoutant '.T' à la fin.


```
[2]: sagittal = image_stack[:, :, 128].T
      coronal = image_stack[:, 128, :].T
      transverse = image_stack[96]
```

```
fig, axs = plt.subplots(1, 3, figsize=(15,15))

# montrer des plans orthogonaux
axs[0].imshow(sagittal, cmap='Greys_r')
axs[0].set_title('Sagittal')

axs[1].imshow(coronal, cmap='Greys_r')
axs[1].set_title('Coronal')

axs[2].imshow(transverse, cmap='Greys_r')
axs[2].set_title('Transverse');
```



### Vidéos

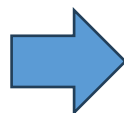
Si un ensemble de données d'image a une dimension temporelle, nous l'appelons une vidéo. Le traitement des vidéos fonctionne de

# Code for statistical analysis

- Be careful when prompting for code and **provide** as many **details** as you can!



%%bob load data/data.csv and determine if the columns A and B are significantly different.



```
import pandas as pd
import numpy as np
from scipy.stats import ttest_ind
import matplotlib.pyplot as plt

# 1. Load the CSV file
df = pd.read_csv('data/data.csv')

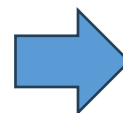
# 2. Keep only rows where both A and B are present
df = df[['A', 'B']].dropna()

# 3. Extract the two series
a = df['A'].values
b = df['B'].values

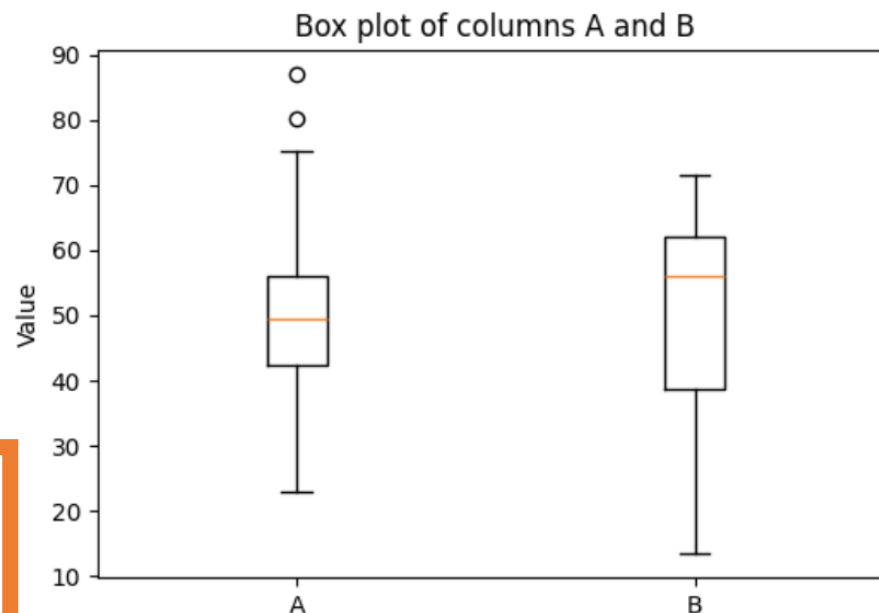
# 4. Perform independent two-sample t-test
t_stat, p_val = ttest_ind(a, b, equal_var=False) # Welch's t-test

# 5. Output the results
print(f"t-statistic: {t_stat:.4f}")
print(f"p-value: {p_val:.4e}")
if p_val < 0.05:
    print("=> The difference between columns A and B is statistically significant (α=0.05).")
else:
    print("=> No statistically significant difference detected (α=0.05).")

# 6. Visual comparison: box-plot
plt.figure(figsize=(6, 4))
plt.boxplot([a, b], labels=['A', 'B'])
plt.title('Box plot of columns A and B')
plt.ylabel('Value')
plt.show()
```



t-statistic: -1.3287  
p-value: 1.8428e-01  
=> No statistically significant difference detected ( $\alpha=0.05$ ).



LLMs often  
provide “simple”  
solutions.

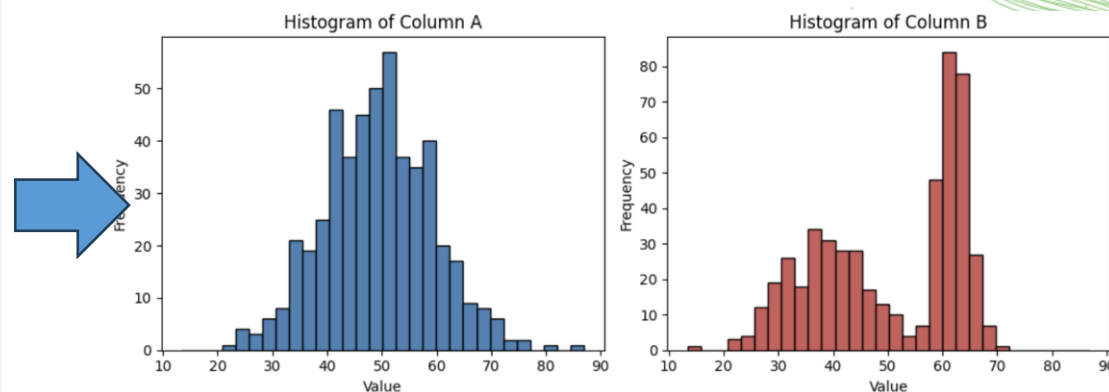


# Code for statistical analysis

- Be careful when prompting for code and **provide** as many **details** as you can!

`%bob`  
Plot histograms for A and B

```
# Assuming df, a, b are already defined from the previous step.  
# If not, uncomment the following lines to load them again:  
# import pandas as pd  
# df = pd.read_csv('data/data.csv')[['A', 'B']].dropna()  
# a = df['A'].values  
# b = df['B'].values  
  
# Determine a common bin range for both histograms  
min_val = min(a.min(), b.min())  
max_val = max(a.max(), b.max())  
bins = 30 # you can adjust the number of bins  
  
plt.figure(figsize=(10, 4))  
  
# Histogram for column A  
plt.subplot(1, 2, 1)  
plt.hist(a, bins=bins, range=(min_val, max_val), color='steelblue', edgecolor='black')  
plt.title('Histogram of Column A')  
plt.xlabel('Value')  
plt.ylabel('Frequency')  
  
# Histogram for column B  
plt.subplot(1, 2, 2)  
plt.hist(b, bins=bins, range=(min_val, max_val), color='indianred', edgecolor='black')  
plt.title('Histogram of Column B')  
plt.xlabel('Value')  
plt.ylabel('Frequency')  
  
plt.tight_layout()  
plt.show()
```



Look at your  
data!

# Code for statistical analysis

- Be careful when prompting for code and **provide** as many **details** as you can!

```
%%bob
Please test again if columns A and B are significantly different.
Before applying a statistical test, ensure that the necessary
pre-conditions are met.
```

```
from scipy.stats import shapiro, levene, mannwhitneyu
# Note: pandas (pd), numpy (np), and test_ind are already available.

# 1. Load data (keep original df/a/b untouched)
data_path = "data/data.csv"
df_raw = pd.read_csv(data_path)
# Keep only rows where both A and B are present
df_clean = df_raw[df_raw[['A', 'B']].dropna()]
a_vals = df_clean['A'].values
b_vals = df_clean['B'].values

# 2. Normality tests (Shapiro-Wilk)
shapiro_a_stat, shapiro_a_p = shapiro(a_vals)
shapiro_b_stat, shapiro_b_p = shapiro(b_vals)

# 3. Equality of variances (Levene)
lev_stat, lev_p = levene(a_vals, b_vals)

# 4. Choose appropriate hypothesis test
if shapiro_a_p > 0.05 and shapiro_b_p > 0.05:
    # both distributions look normal
    if lev_p > 0.05:
        test_name = "Student's t-test (equal variances)"
        t_stat, p_val = test_ind(a_vals, b_vals, equal_var=True)
    else:
        test_name = "Welch's t-test (unequal variances)"
        t_stat, p_val = test_ind(a_vals, b_vals, equal_var=False)
else:
    # at least one non-normal - use non-parametric test
    test_name = "Mann-Whitney U test (non-parametric)"
    # two-sided test
    t_stat, p_val = mannwhitneyu(a_vals, b_vals, alternatives='two-sided')

# 5. Print report
print("=== Assumption checks ===")
print(f"Shapiro-Wilk A: statistic={shapiro_a_stat:.4f}, p={shapiro_a_p:.4e}")
print(f"Shapiro-Wilk B: statistic={shapiro_b_stat:.4f}, p={shapiro_b_p:.4e}")
print(f"Levene variance test: statistic={lev_stat:.4f}, p={lev_p:.4e}")
print(f"Final hypothesis test used: {test_name}")
print(f"Statistic = {t_stat:.4f}")
print(f"P-value = {p_val:.4e}")
if p_val < 0.05:
    print("> Significant difference between columns A and B (alpha=0.05).")
else:
    print("> No significant difference detected between columns A and B (alpha=0.05).")
```

=== Assumption checks ===

Shapiro-Wilk A: statistic=0.9972, p=5.6236e-01 normal

Shapiro-Wilk B: statistic=0.8937, p=4.0635e-18 non-normal

Levene variance test: statistic=80.6718, p=1.3016e-18 unequal variances

=== Final hypothesis test ===

Test used: Mann-Whitney U test (non-parametric)

Statistic = 113657.0000

P-value = 1.3000e-02

=> Significant difference between columns A and B ( $\alpha=0.05$ ).

LLMs cannot be *brave*  
*scientists* yet, but you can!

# Acknowledgements

## Collaborators & contributors

Christian Tischer (EMBL Heidelberg), Jean-Karim Hériché (EMBL Heidelberg), Nico Scherf (MPI CBS Leipzig), Loïc A. Royer (CZ Biohub), Elena Nicolay (UFZ Leipzig), Kevin Yamauchi (ETH Zurich / Basel), Seth Hinz, Teun A.P.M. Huijben, Jordão Bragantini (CZ Biohub San Francisco), Ilan Theodoro (CZ Biohub and the State University of Campinas, Brazil), Ian C. (University of British Columbia), Pradeep Rajasekhar (WEHI Australia)

## Communities & platforms



NFDI 4  
BIOIMAGE



image.sc

## BIDS Team @ ScaDS.AI / UL



Lea Kabjesz, Lea Gihlein, Mara Lampert



<https://scads.github.io/embo-lsm-bia-2025>

## Funding



Bundesministerium  
für Forschung, Technologie  
und Raumfahrt



SACHSEN 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.



Deutsche  
Forschungsgemeinschaft



Robert Haase  
@haesleinhuepf  
Repro BIA Python  
EMBO LSM 2025

RH acknowledges the financial support by the Federal Ministry of Education and Research of Germany and by Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the programme Center of Excellence for AI-research „Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig“, project identification number: ScaDS.AI



TECHNISCHE  
UNIVERSITÄT  
DRESDEN



UNIVERSITÄT  
LEIPZIG