CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE

NATIONAL RESEARCH DATA MANAGEMENT INFRASTRUCTURE
FOR MICROSCOPY AND BIOIMAGE ANALYSIS

GLOBAL BIOIMAGE
ANALYST'S SOCIETY

Using
Large Language Models for Data Analysis

Robert Haase

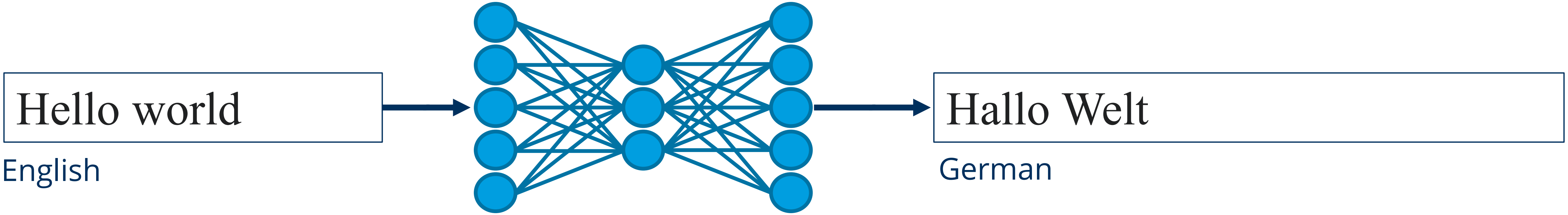
These slides can be reused under the terms of the CC-BY4.0 license.

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.

Large Language Models (LLMs)

Text-to-text, translation



Hello world

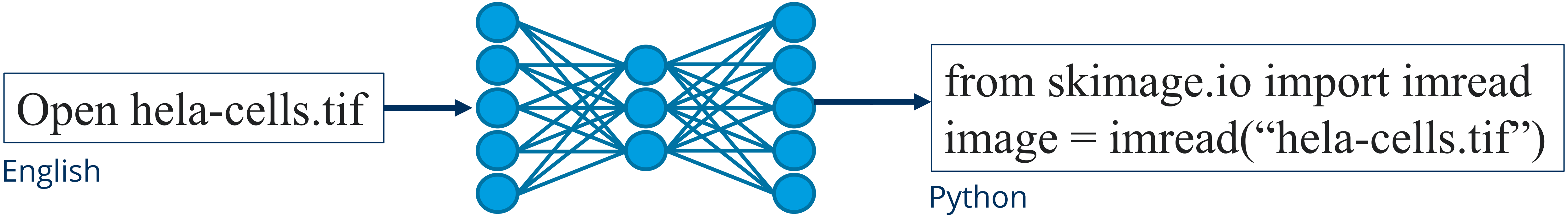
English

Hallo Welt

German

Large Language Models (LLMs)

Text-to-text, translation, code generation

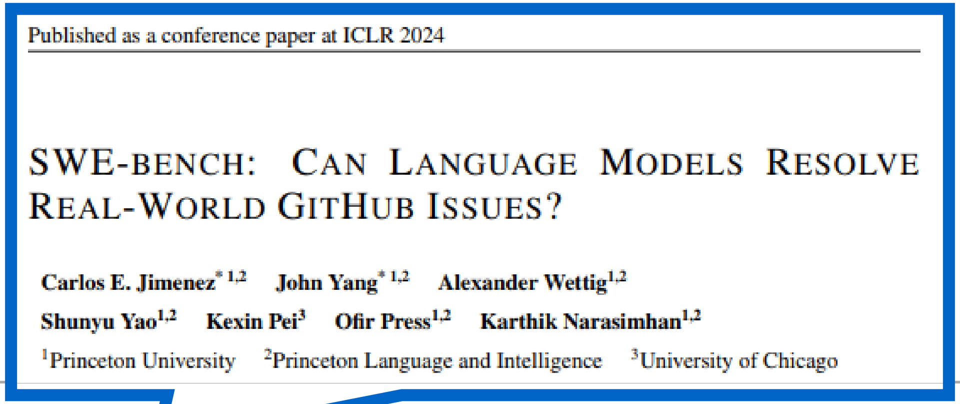


Open hela-cells.tif

English

```
from skimage.io import imread
image = imread(“hela-cells.tif”)
```

Python



Published as a conference paper at ICLR 2024

SWE-BENCH: CAN LANGUAGE MODELS RESOLVE REAL-WORLD GITHUB ISSUES?

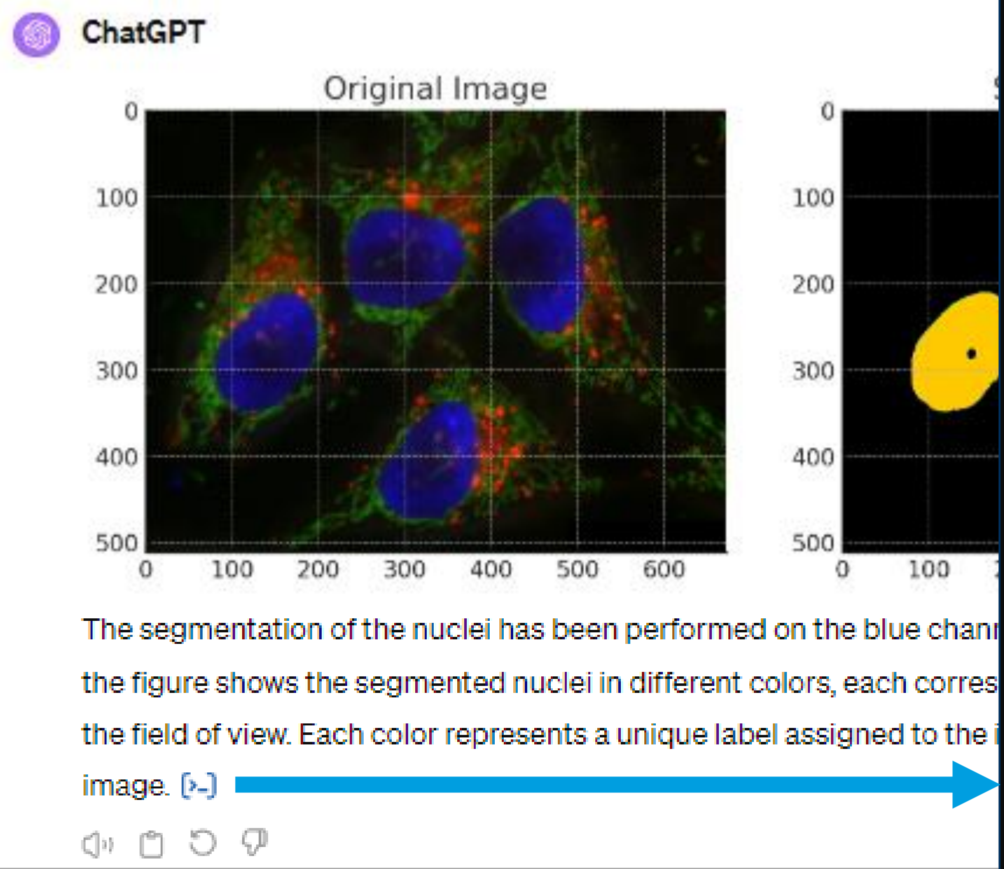
Carlos E. Jimenez*[1,2] **John Yang***[1,2] **Alexander Wettig**[1,2]
Shunyu Yao[1,2] **Kexin Pei**[3] **Ofir Press**[1,2] **Karthik Narasimhan**[1,2]
[1]Princeton University [2]Princeton Language and Intelligence [3]University of Chicago

https://arxiv.org/abs/2310.06770

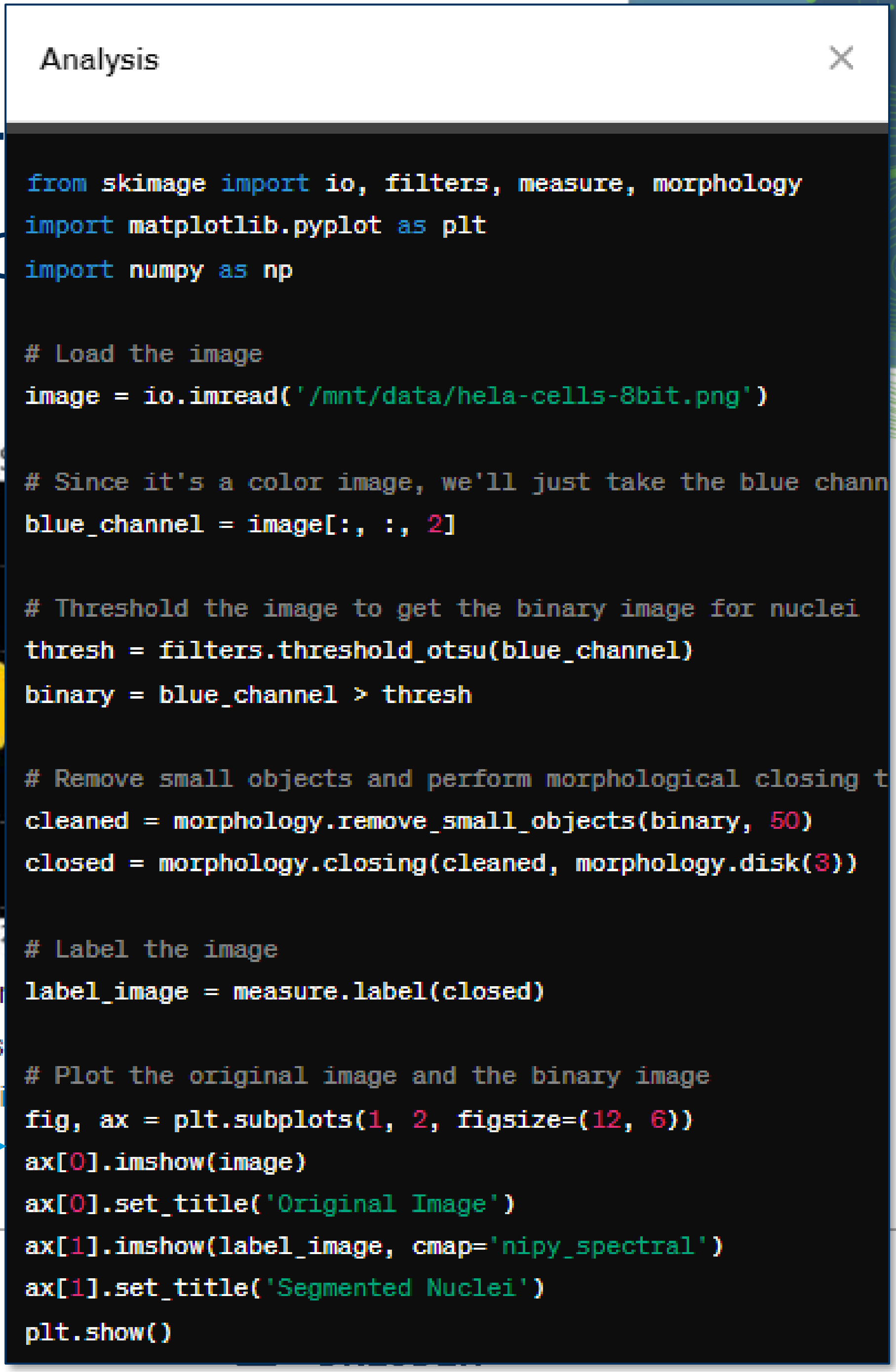
Bio-image Analysis using Large-L

Analysing images in the browser, e.g. using Chat

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



The segmentation of the nuclei has been performed on the blue chann
the figure shows the segmented nuclei in different colors, each corres
the field of view. Each color represents a unique label assigned to the i
image. [>_]

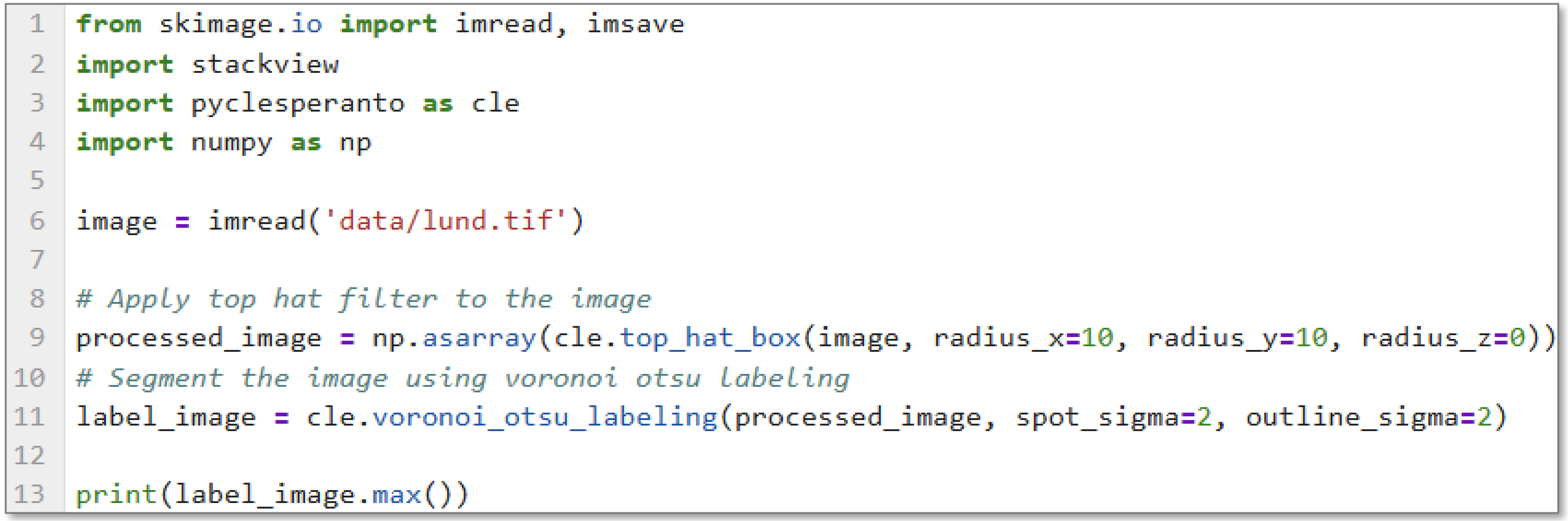


```
from skimage import io, filters, measure, morphology
import matplotlib.pyplot as plt
import numpy as np

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

# Since it's a color image, we'll just take the blue chann
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 t
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()
```

Reproducibility: Python

Technical: I can rerun the workflow because it is well documented.

Inter-personal: I understand a workflow and can explain all steps.



```python
from skimage.io import imread, imsave
import stackview
import pyclesperanto as cle
import numpy as np

image = imread('data/lund.tif')

# Apply top hat filter to the image
processed_image = np.asarray(cle.top_hat_box(image, radius_x=10, radius_y=10, radius_z=0))
# Segment the image using voronoi otsu labeling
label_image = cle.voronoi_otsu_labeling(processed_image, spot_sigma=2, outline_sigma=2)

print(label_image.max())
```

Output: 285

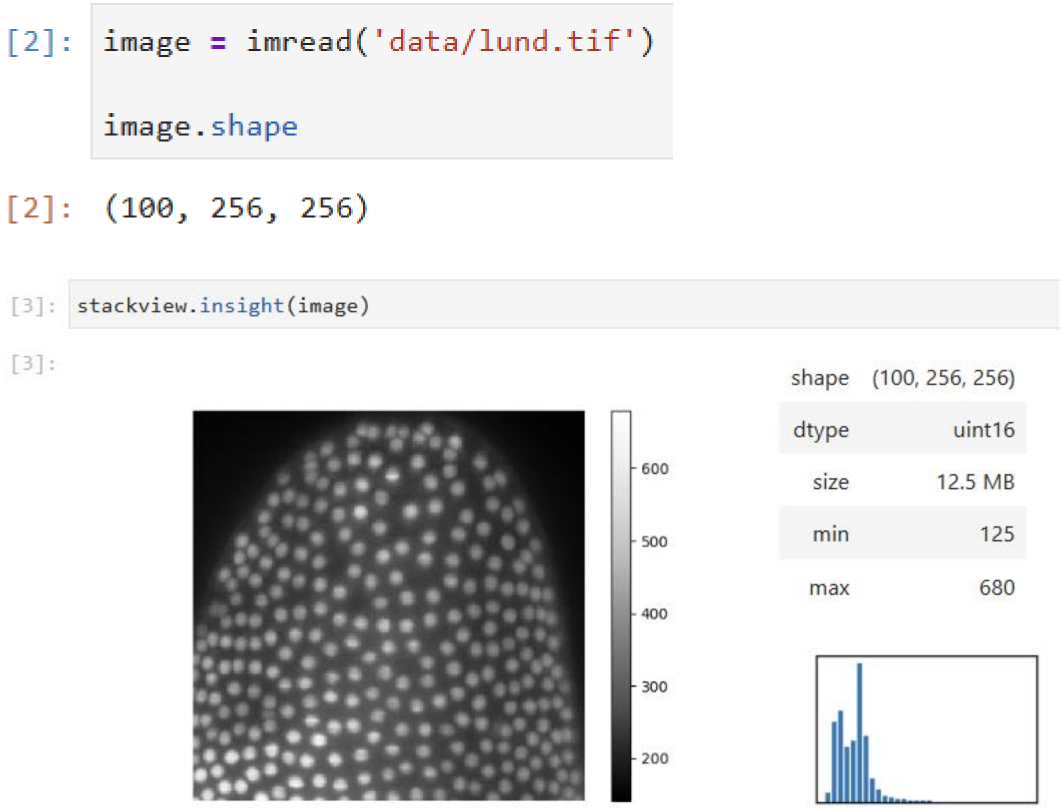
We cannot follow the workflow, because we do not see intermediate results.

Technical: ✅
Inter-personal: ❌

Reproducibility: Python + Jupyter

Technical: I can rerun the workflow because it is well documented.

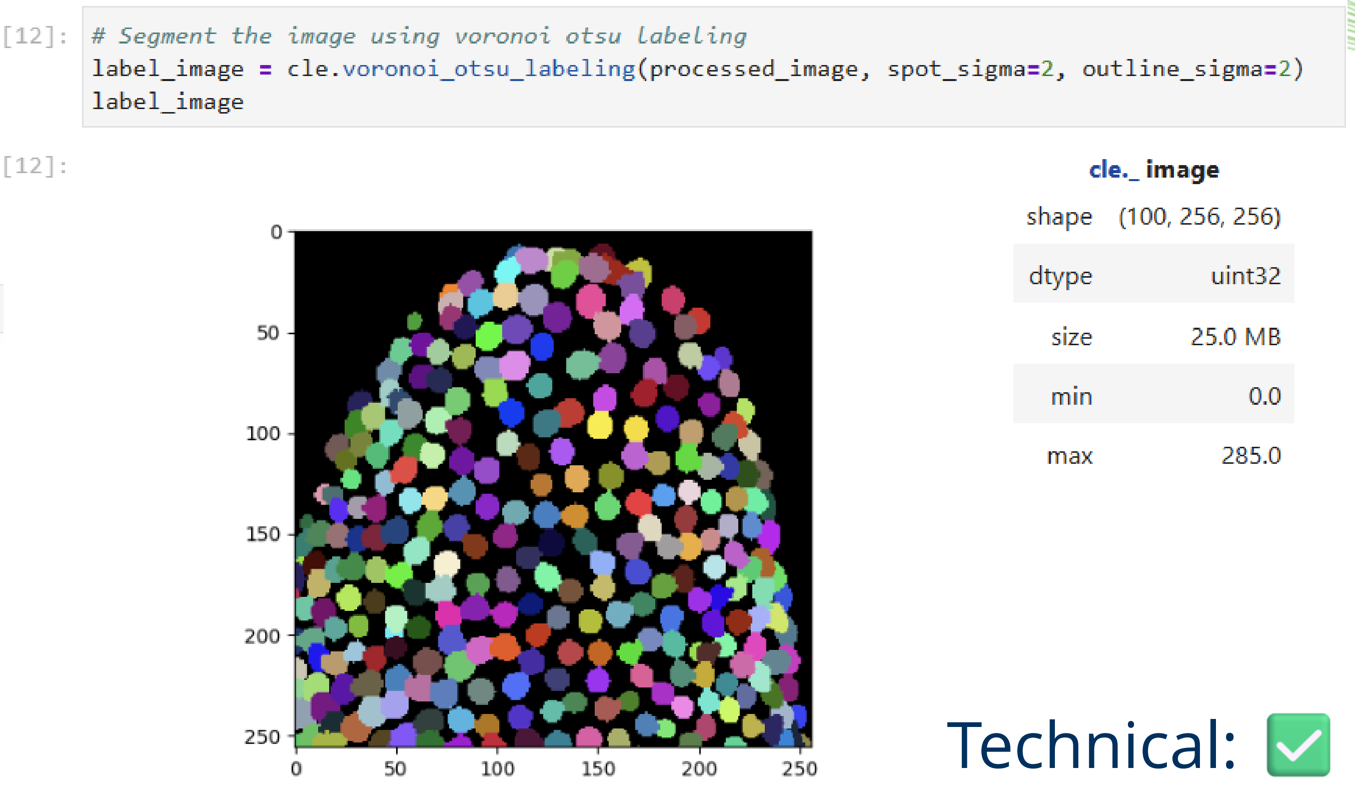
Inter-personal: I understand a workflow and can explain all steps.



```
[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

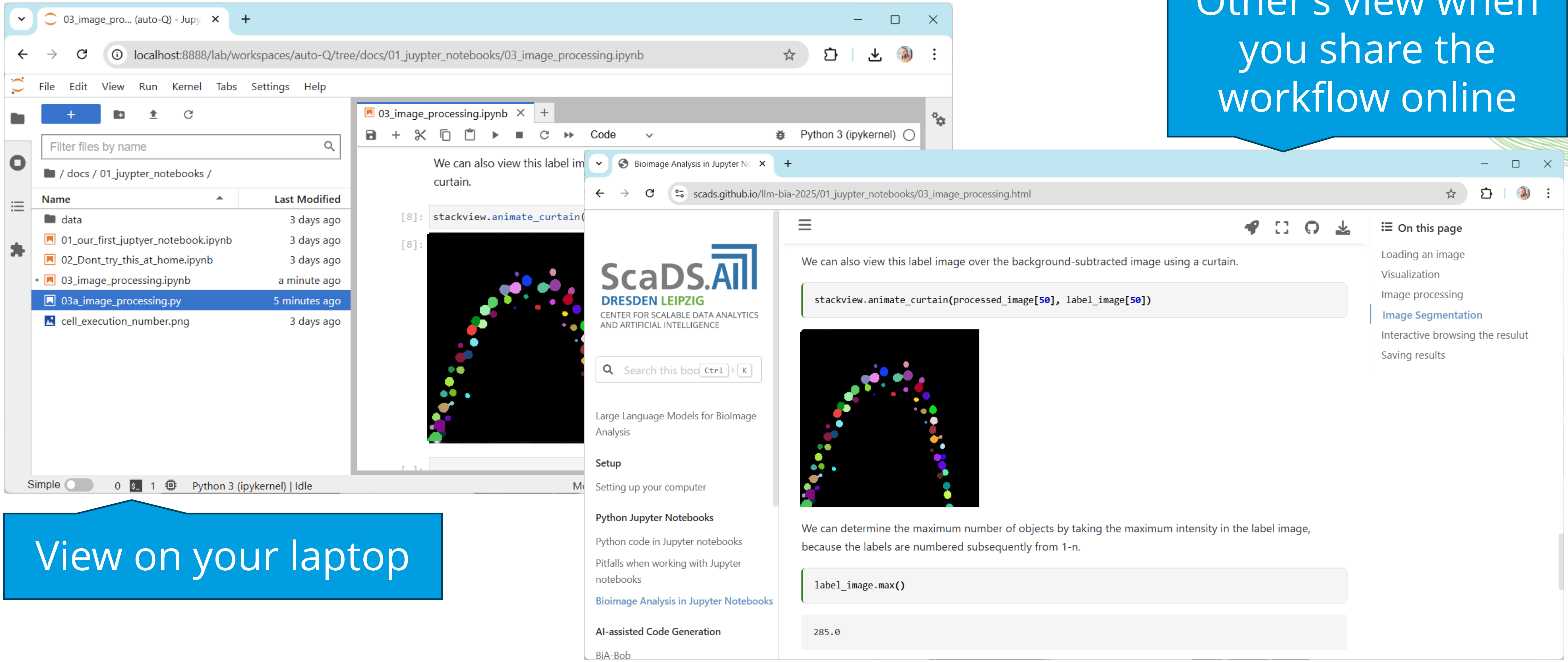
shape	(100, 256, 256)
dtype	uint32
size	25.0 MB
min	0.0
max	285.0

Technical: ✅

Inter-personal: ✅

Reproducibility : Python + Jupyter

Other's view when you share the workflow online


03_image_pro... (auto-Q) - Jupy
localhost:8888/lab/workspaces/auto-Q/tree/docs/01_juypter_notebooks/03_image_processing.ipynb
File
Edit
View
Run
Kernel
Tabs
Settings
Help
Filter files by name
/ docs / 01_juypter_notebooks /
03_image_processing.ipynb
Code
Python 3 (ipykernel)
Simple
Python 3 (ipykernel) | Idle
Bioimage Analysis in Jupyter N
scads.github.io/llm-bia-2025/01_juypter_notebooks/03_image_processing.html
Search this book Ctrl + K
On this page

Name	Last Modified
data	3 days ago
01_our_first_juypter_notebook.ipynb	3 days ago
02_Dont_try_this_at_home.ipynb	3 days ago
03_image_processing.ipynb	a minute ago
03a_image_processing.py	5 minutes ago
cell_execution_number.png	3 days ago

We can also view this label im… curtain.

```
[8]: stackview.animate_curtain(
```

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

- Large Language Models for BioImage Analysis
- Setup
  - Setting up your computer
- Python Jupyter Notebooks
  - Python code in Jupyter notebooks
  - Pitfalls when working with Jupyter notebooks
  - Bioimage Analysis in Jupyter Notebooks
- AI-assisted Code Generation
  - BiA-Bob

We can also view this label image over the background-subtracted image using a curtain.

```
stackview.animate_curtain(processed_image[50], label_image[50])
```

We can determine the maximum number of objects by taking the maximum intensity in the label image, because the labels are numbered subsequently from 1-n.

```
label_image.max()
```

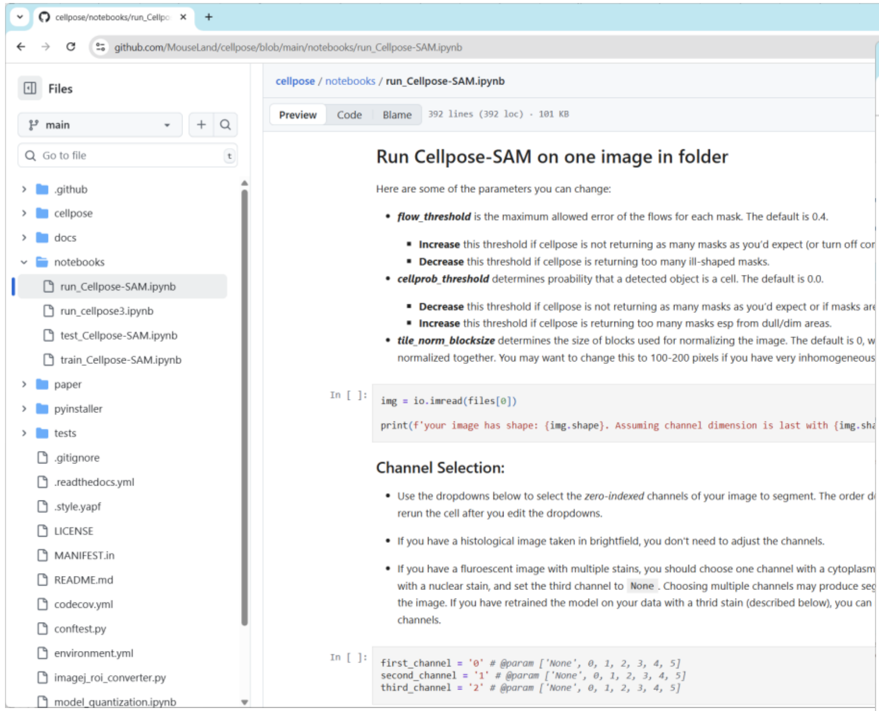
```
285.0
```

On this page
- Loading an image
- Visualization
- Image processing
- Image Segmentation
- Interactive browsing the resulut
- Saving results

View on your laptop

https://scads.github.io/llm-bia-2025/01_juypter_notebooks/03_image_processing.html

Reproducibility : Python + Jupyter



Run Cellpose-SAM on one image in folder

Here are some of the parameters you can change:

- ***flow_threshold*** is the maximum allowed error of the flows for each mask. The default is 0.4.
  - **Increase** this threshold if cellpose is not returning as many masks as you'd expect (or turn off cor
  - **Decrease** this threshold if cellpose is returning too many ill-shaped masks.
- ***cellprob_threshold*** determines proability that a detected object is a cell. The default is 0.0.
  - **Decrease** this threshold if cellpose is not returning as many masks as you'd expect or if masks are
  - **Increase** this threshold if cellpose is returning too many masks esp from dull/dim areas.
- ***tile_norm_blocksize*** determines the size of blocks used for normalizing the image. The default is 0, w normalized together. You may want to change this to 100-200 pixels if you have very inhomogeneous

In [ ]:

```
img = io.imread(files[0])

print(f'your image has shape: {img.shape}. Assuming channel dimension is last with {img.sha
```

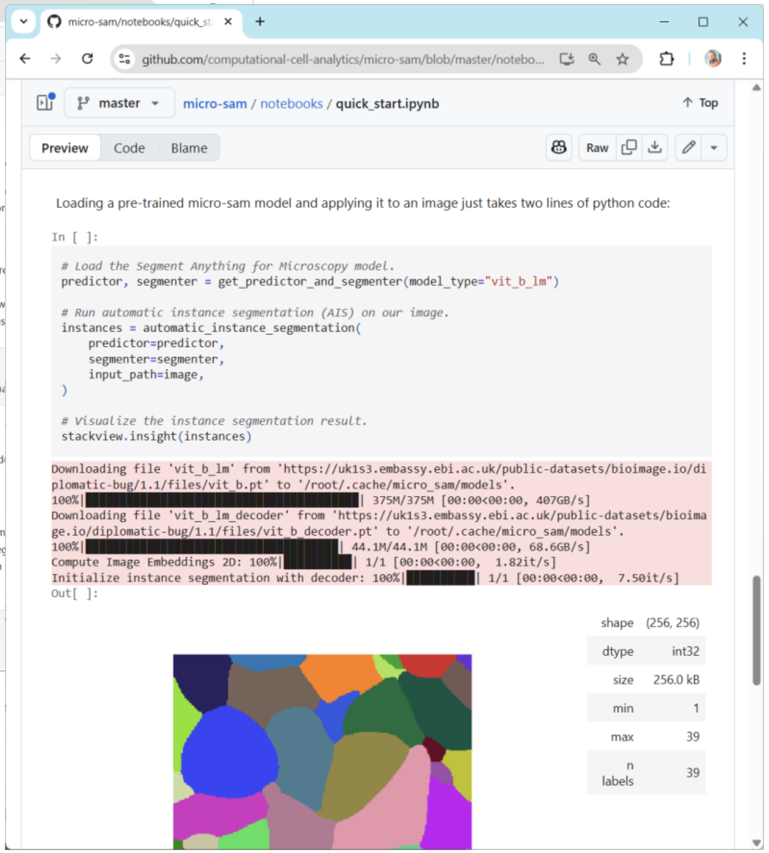
Channel Selection:

- Use the dropdowns below to select the *zero-indexed* channels of your image to segment. The order d rerun the cell after you edit the dropdowns.
- If you have a histological image taken in brightfield, you don't need to adjust the channels.
- If you have a fluroescent image with multiple stains, you should choose one channel with a cytoplasm with a nuclear stain, and set the third channel to `None`. Choosing multiple channels may produce seg the image. If you have retrained the model on your data with a thrid stain (described below), you can channels.

In [ ]:

```
first_channel = '0' # @param ['None', 0, 1, 2, 3, 4, 5]
second_channel = '1' # @param ['None', 0, 1, 2, 3, 4, 5]
third_channel = '2' # @param ['None', 0, 1, 2, 3, 4, 5]
```

https://github.com/MouseLand/cellpose/blob/main/notebooks/run_Cellpose-SAM.ipynb


micro-sam/notebooks/quick_st
github.com/computational-cell-analytics/micro-sam/blob/master/notebo...
master
micro-sam / notebooks / quick_start.ipynb
Top
Preview
Code
Blame
Raw

Loading a pre-trained micro-sam model and applying it to an image just takes two lines of python code:

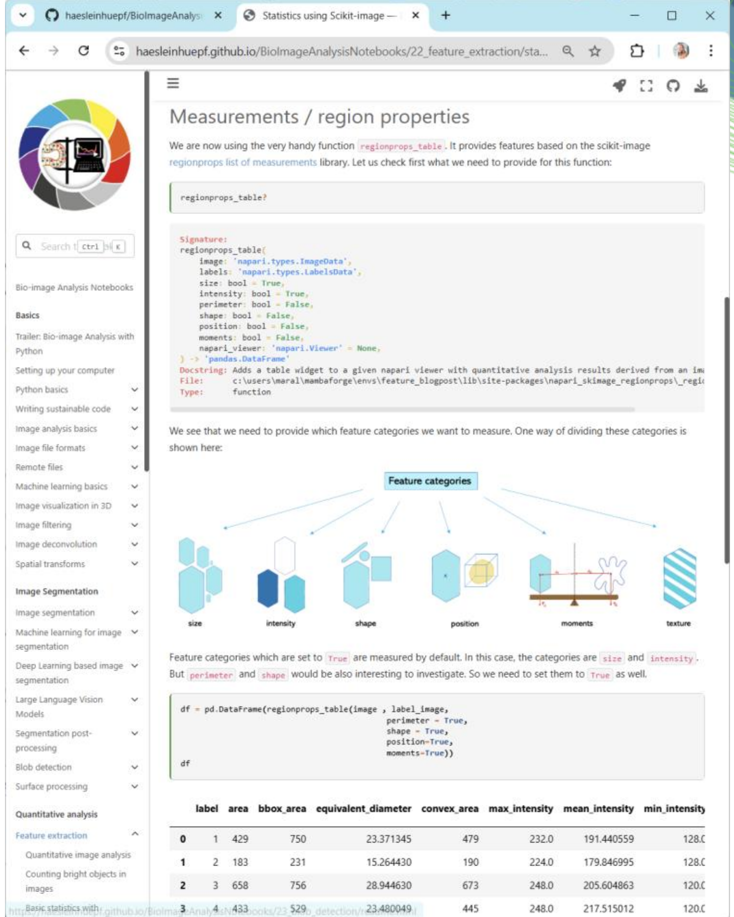
In [ ]:

```
# Load the Segment Anything for Microscopy model.
predictor, segmenter = get_predictor_and_segmenter(model_type="vit_b_lm")

# Run automatic instance segmentation (AIS) on our image.
instances = automatic_instance_segmentation(
    predictor=predictor,
    segmenter=segmenter,
    input_path=image,
)

# Visualize the instance segmentation result.
stackview.insight(instances)
```

```
Downloading file 'vit_b_lm' from 'https://uk1s3.embassy.ebi.ac.uk/public-datasets/bioimage.io/di
plomatic-bug/1.1/files/vit_b.pt' to '/root/.cache/micro_sam/models'.
100%|████████████████████████████████████████| 375M/375M [00:00<00:00, 407GB/s]
Downloading file 'vit_b_lm_decoder' from 'https://uk1s3.embassy.ebi.ac.uk/public-datasets/bioima
ge.io/diplomatic-bug/1.1/files/vit_b_decoder.pt' to '/root/.cache/micro_sam/models'.
100%|████████████████████████████████████████| 44.1M/44.1M [00:00<00:00, 68.6GB/s]
Compute Image Embeddings 2D: 100%|██████████| 1/1 [00:00<00:00,  1.82it/s]
Initialize instance segmentation with decoder: 100%|██████████| 1/1 [00:00<00:00,  7.50it/s]
Out[ ]:
```

shape	(256, 256)
dtype	int32
size	256.0 kB
min	1
max	39
n labels	39

https://github.com/computational-cell-analytics/micro-sam/blob/master/notebooks/quick_start.ipynb


haesleinhuepf/BioImageAnalys
Statistics using Scikit-image —
haesleinhuepf.github.io/BioImageAnalysisNotebooks/22_feature_extraction/sta...
Bio-image Analysis Notebooks
Basics
Trailer: Bio-image Analysis with Python
Setting up your computer
Python basics
Writing sustainable code
Image analysis basics
Image file formats
Remote files
Machine learning basics
Image visualization in 3D
Image filtering
Image deconvolution
Spatial transforms
Image Segmentation
Image segmentation
Machine learning for image segmentation
Deep Learning based image segmentation
Large Language Vision Models
Segmentation post-processing
Blob detection
Surface processing
Quantitative analysis
Feature extraction
Quantitative image analysis
Counting bright objects in images
Basic statistics with
Feature categories
size
intensity
shape
position
moments
texture

Measurements / region properties

We are now using the very handy function `regionprops_table`. It provides features based on the scikit-image regionprops list of measurements library. Let us check first what we need to provide for this function:

```
regionprops_table?
```

```
Signature:
regionprops_table(
    image: 'napari.types.ImageData',
    labels: 'napari.types.LabelsData',
    size: bool = True,
    intensity: bool = True,
    perimeter: bool = False,
    shape: bool = False,
    position: bool = False,
    moments: bool = False,
    napari_viewer: 'napari.Viewer' = None,
) -> 'pandas.DataFrame'
Docstring: Adds a table widget to a given napari viewer with quantitative analysis results derived from an ima
File:      c:\users\maral\mambaforge\envs\feature_blogpost\lib\site-packages\napari_skimage_regionprops\_regi
Type:      function
```

We see that we need to provide which feature categories we want to measure. One way of dividing these categories is shown here:

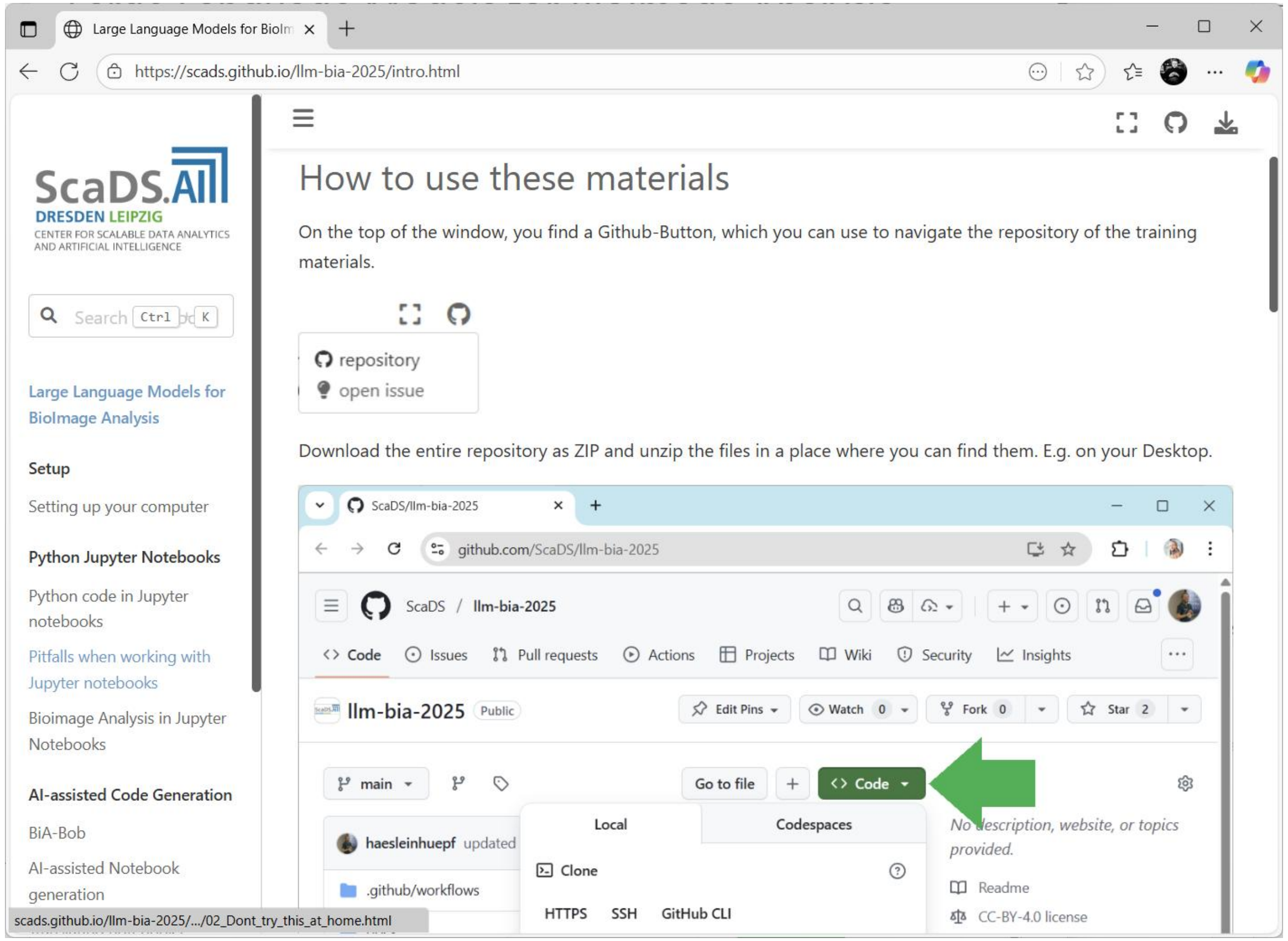
Feature categories which are set to `True` are measured by default. In this case, the categories are `size` and `intensity`. But `perimeter` and `shape` would be also interesting to investigate. So we need to set them to `True` as well.

```
df = pd.DataFrame(regionprops_table(image , label_image,
                                    perimeter = True,
                                    shape = True,
                                    position=True,
                                    moments=True))

df
```

	label	area	bbox_area	equivalent_diameter	convex_area	max_intensity	mean_intensity	min_intensity
0	1	429	750	23.371345	479	232.0	191.440559	128.0
1	2	183	231	15.264430	190	224.0	179.846995	128.0
2	3	658	756	28.944630	673	248.0	205.604863	120.0
3	4	433	529	23.480049	445	248.0	217.515012	120.0

https://haesleinhuepf.github.io/BioImageAnalysisNotebooks/22_feature_extraction/statistics_with_scikit_image.html

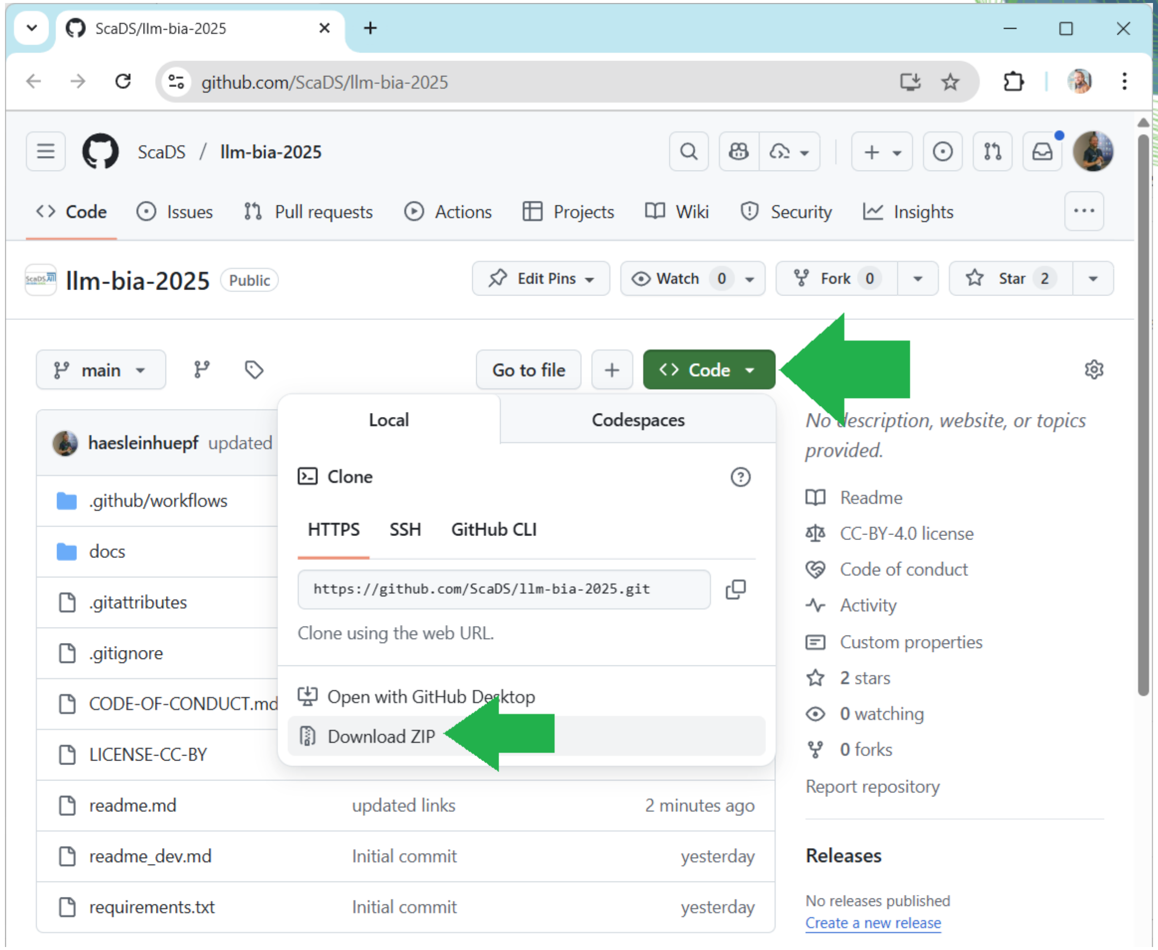
Training materials for today


Large Language Models for BioIm
https://scads.github.io/llm-bia-2025/intro.html
ScaDS.AI DRESDEN LEIPZIG
CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE
Search Ctrl K
Large Language Models for BioImage Analysis
Setup
Setting up your computer
Python Jupyter Notebooks
Python code in Jupyter notebooks
Pitfalls when working with Jupyter notebooks
Bioimage Analysis in Jupyter Notebooks
AI-assisted Code Generation
BiA-Bob
AI-assisted Notebook generation
scads.github.io/llm-bia-2025/.../02_Dont_try_this_at_home.html

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.



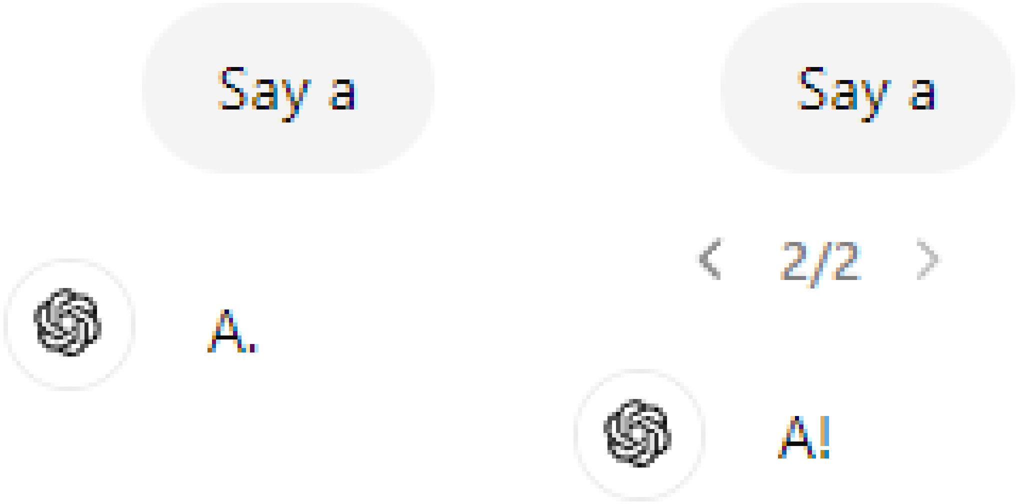
Why code generation using LLMs?

Output of LLMs is often criticized for

- Limited reproducibility
- Hard to evaluate (automatically)

Output of LLM-generated code is

- Reproducible
- Can be executed and evaluted (automatically)



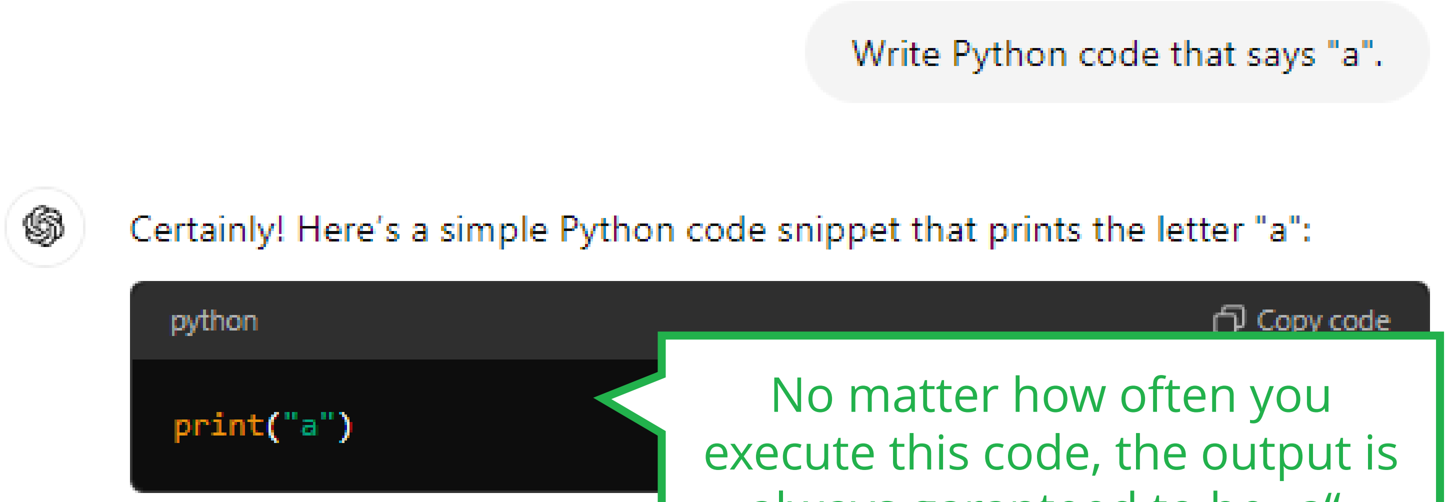
Say a

A.

Say a

2/2

A!



Write Python code that says "a".

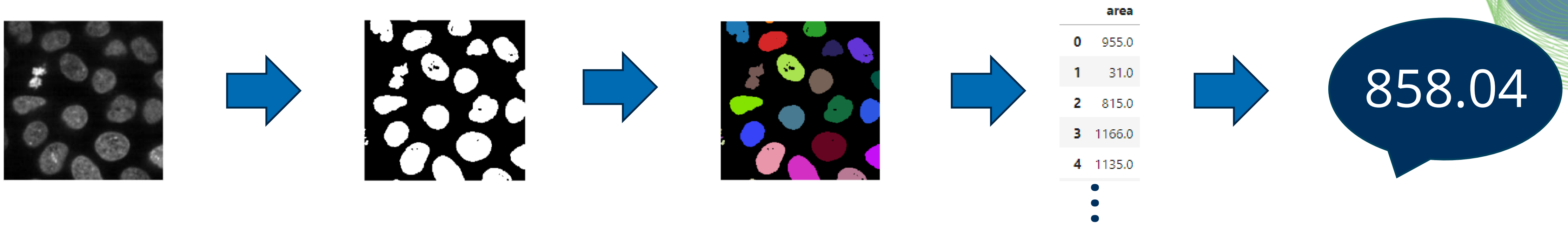
Certainly! Here's a simple Python code snippet that prints the letter "a":

```python
print("a")
```

No matter how often you execute this code, the output is always garanteed to be „a“.

How good are LLMs for Bio-image Analysis?

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

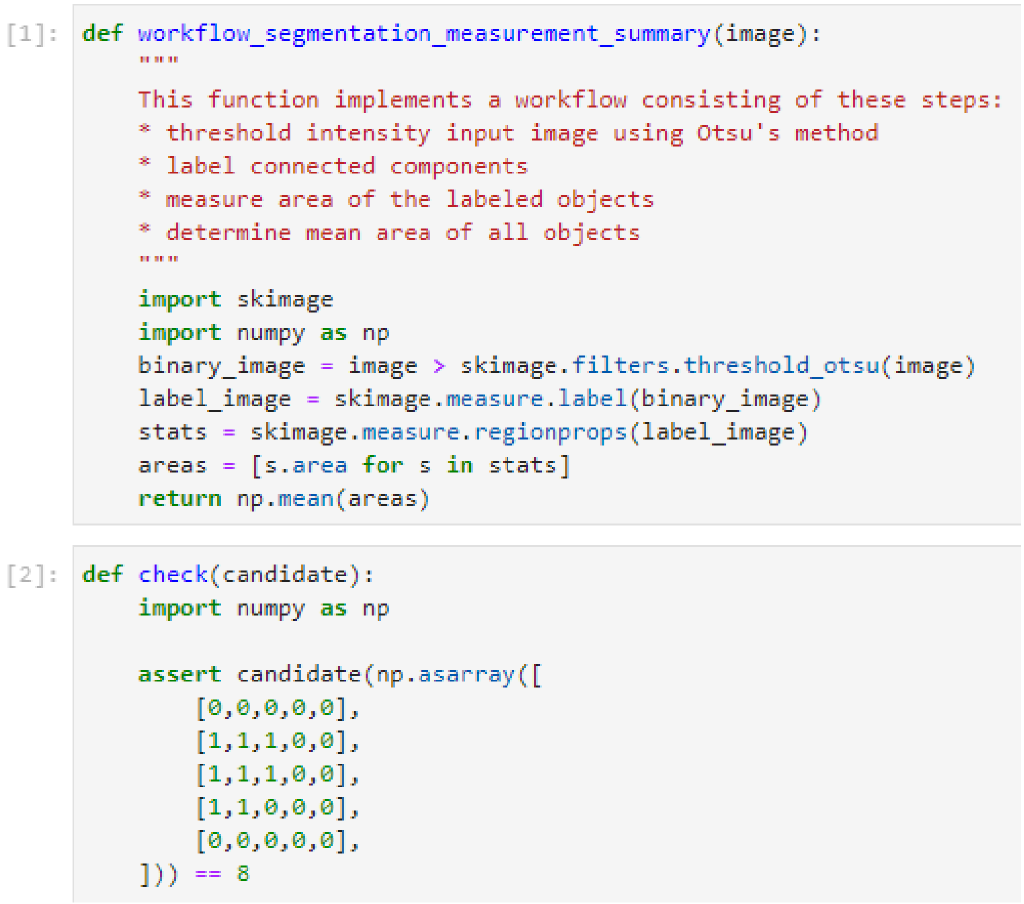


	area
0	955.0
1	31.0
2	815.0
3	1166.0
4	1135.0

https://www.biorxiv.org/content/10.1101/2024.04.19.590278v1
https://github.com/haesleinhuepf/human-eval-bia

How good are LLMs for Bio-image Analysis?

Example test-case inspired by HumaEval (Chen et al 2021)



[1]:

```python
def workflow_segmentation_measurement_summary(image):
    """
    This function implements a workflow consisting of these steps:
    * threshold intensity input image using Otsu's method
    * label connected components
    * measure area of the labeled objects
    * determine mean area of all objects
    """
    import skimage
    import numpy as np
    binary_image = image > skimage.filters.threshold_otsu(image)
    label_image = skimage.measure.label(binary_image)
    stats = skimage.measure.regionprops(label_image)
    areas = [s.area for s in stats]
    return np.mean(areas)
```

[2]:

```python
def check(candidate):
    import numpy as np

    assert candidate(np.asarray([
        [0,0,0,0,0],
        [1,1,1,0,0],
        [1,1,1,0,0],
        [1,1,0,0,0],
        [0,0,0,0,0],
    ])) == 8
```

Prompt

Reference solution

Unit test (excerpt)

Open for your (!) contributions

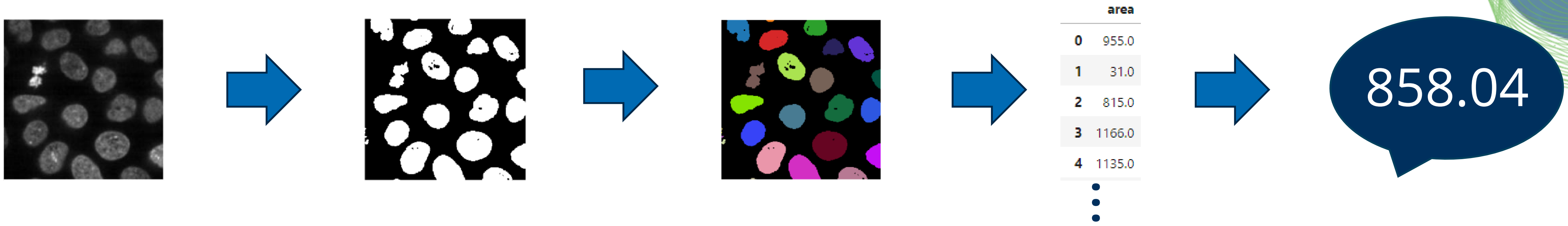
18 more currently on their way…

We formulated 57 of such test-cases (yet)

Chen at al 2021: https://arxiv.org/abs/2107.03374
https://www.biorxiv.org/content/10.1101/2024.04.19.590278v3
https://github.com/haesleinhuepf/human-eval-bia

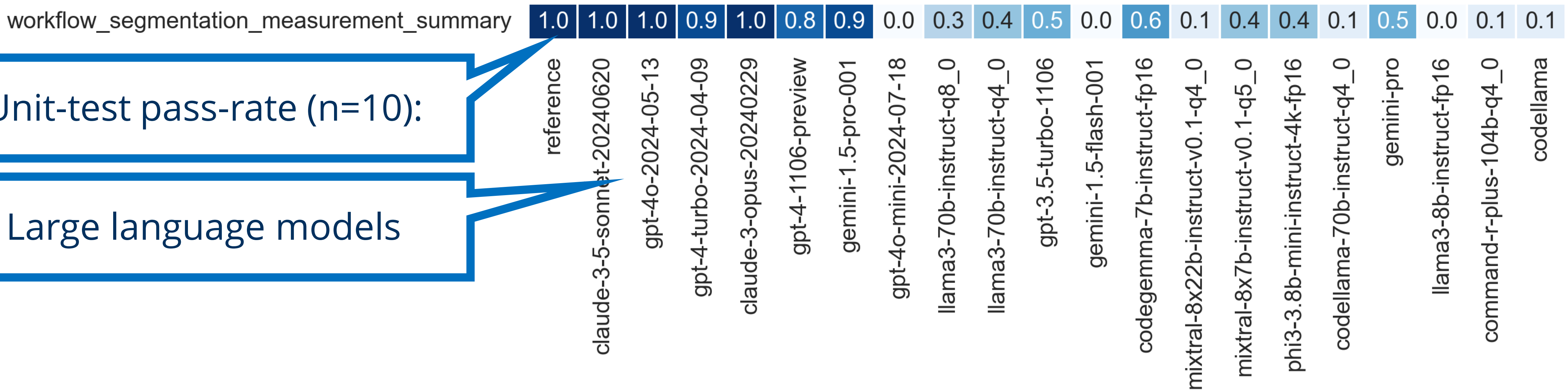
How good are LLMs for Bio-image Analysis?

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



	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
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

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

Large language models

https://www.biorxiv.org/content/10.1101/2024.04.19.590278v1
https://github.com/haesleinhuepf/human-eval-bia

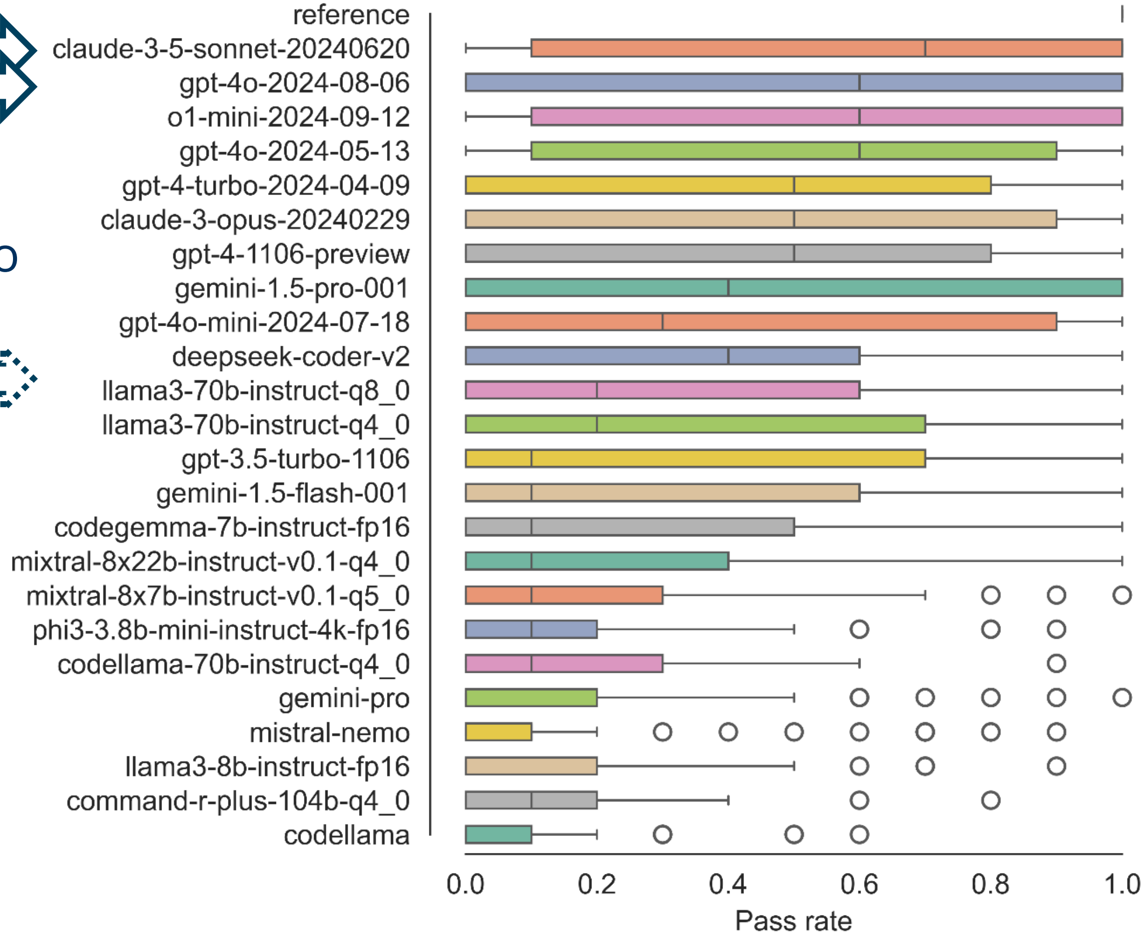
Benchmarking LLMs for Bio-image Analysis

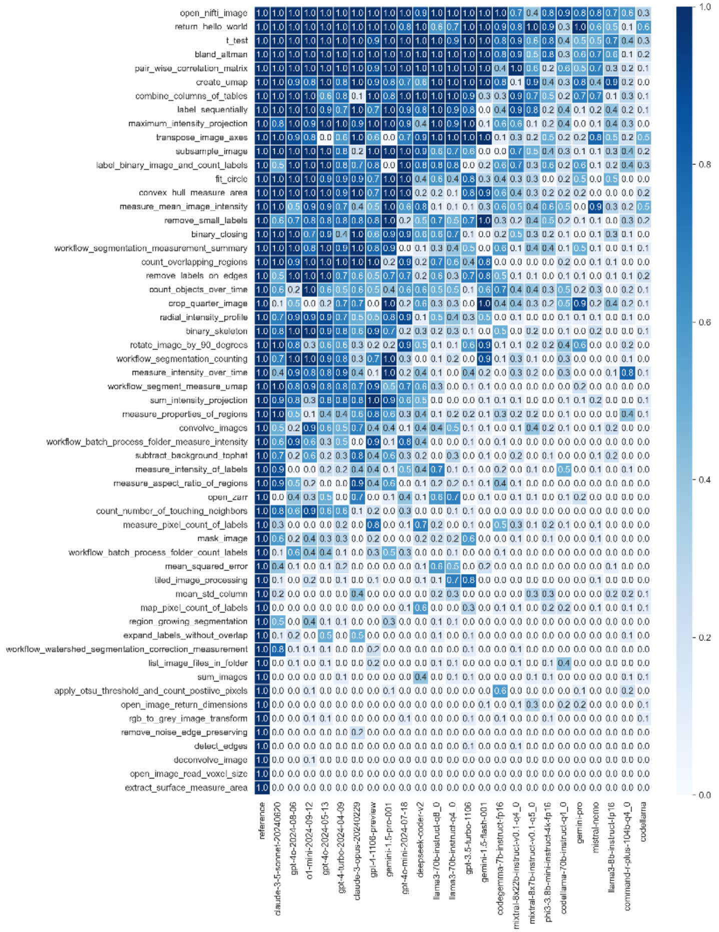
Summary: 57 use-cases (yet), 24 LLMs (yet), n=10

$

Live demo ;-)

$





https://www.biorxiv.org/content/10.1101/2024.04.19.590278v1
https://github.com/haesleinhuepf/human-eval-bia

Prompt Engineering

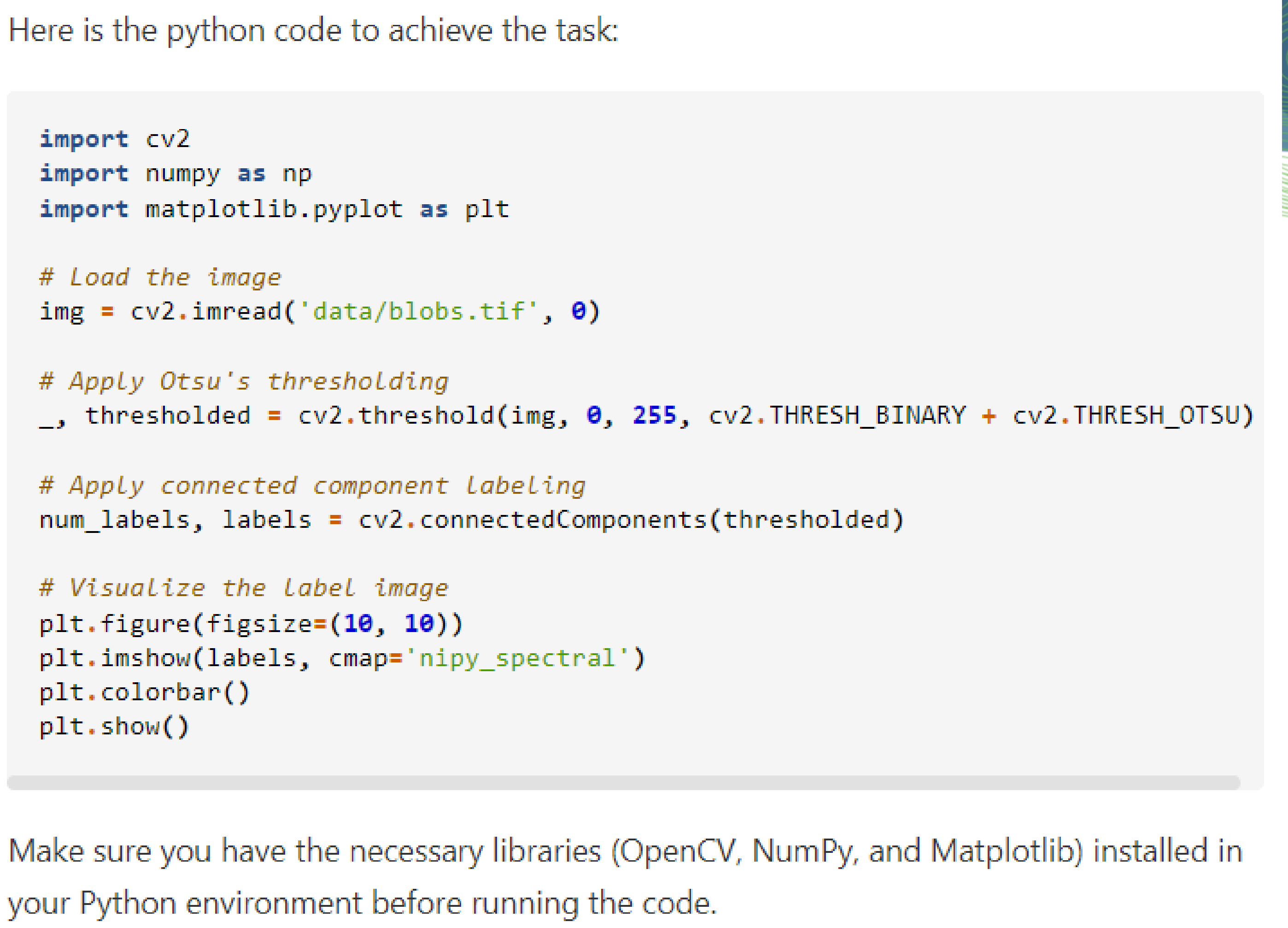
To further improve LLMs capabilities

Prompt Engineering

Modify a prompt until it works.

Write python code to open the image file 'data/blobs.tif' and threshold it using Otsu's method.

Apply connected component labeling to the result to retrieve a label image. Visualize the label image.



Here is the python code to achieve the task:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the image
img = cv2.imread('data/blobs.tif', 0)

# Apply Otsu's thresholding
_, thresholded = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Apply connected component labeling
num_labels, labels = cv2.connectedComponents(thresholded)

# Visualize the label image
plt.figure(figsize=(10, 10))
plt.imshow(labels, cmap='nipy_spectral')
plt.colorbar()
plt.show()
```

Make sure you have the necessary libraries (OpenCV, NumPy, and Matplotlib) installed in your Python environment before running the code.

Prompt Engineering

Modify a prompt until it works.

Write python code to open the image file 'data/blobs.tif'

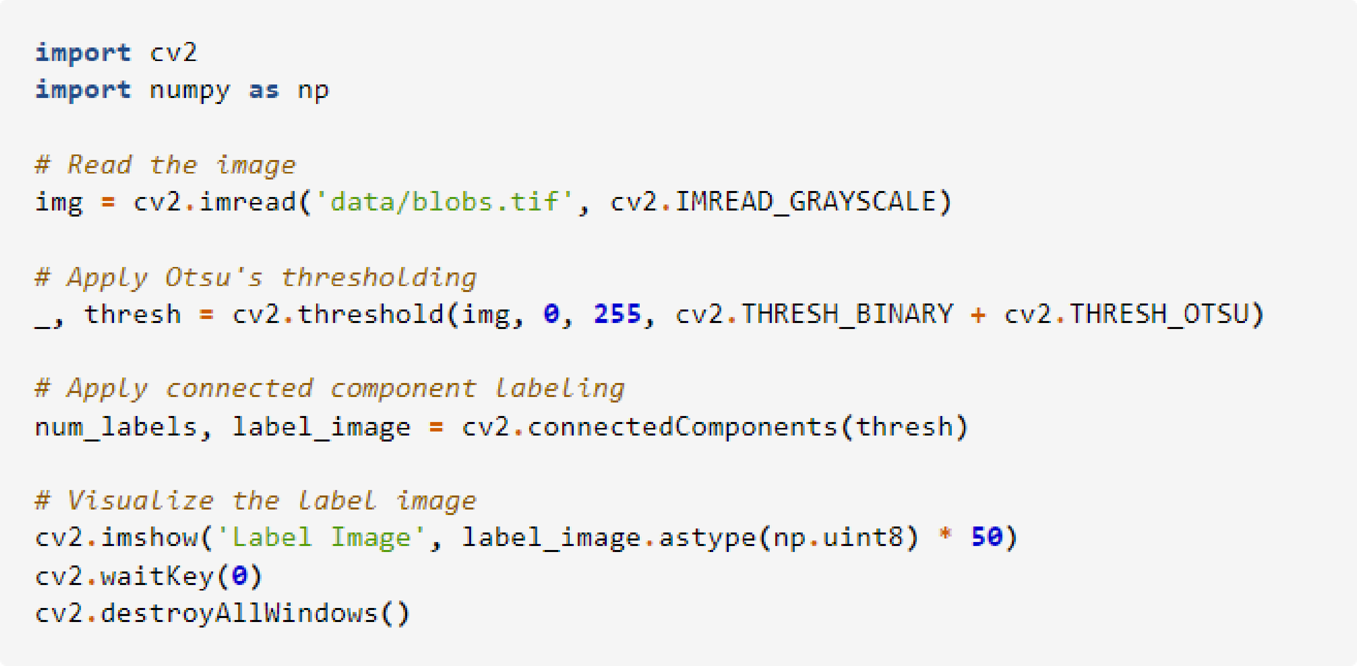
and threshold it using Otsu's method.

Apply connected component labeling to the result to

retrieve a label image. Visualize the label image.

Do not explain the solution.

Only write Python code and short comments are ok.



```python
import cv2
import numpy as np

# Read the image
img = cv2.imread('data/blobs.tif', cv2.IMREAD_GRAYSCALE)

# Apply Otsu's thresholding
_, thresh = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Apply connected component labeling
num_labels, label_image = cv2.connectedComponents(thresh)

# Visualize the label image
cv2.imshow('Label Image', label_image.astype(np.uint8) * 50)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Prompt Engineering

Modify a prompt until it works.

„Long“ Context

Write python code to open the image file 'data/blobs.tif' and threshold it using Otsu's method.

Apply connected component labeling to the result to retrieve a label image. Visualize the label image.

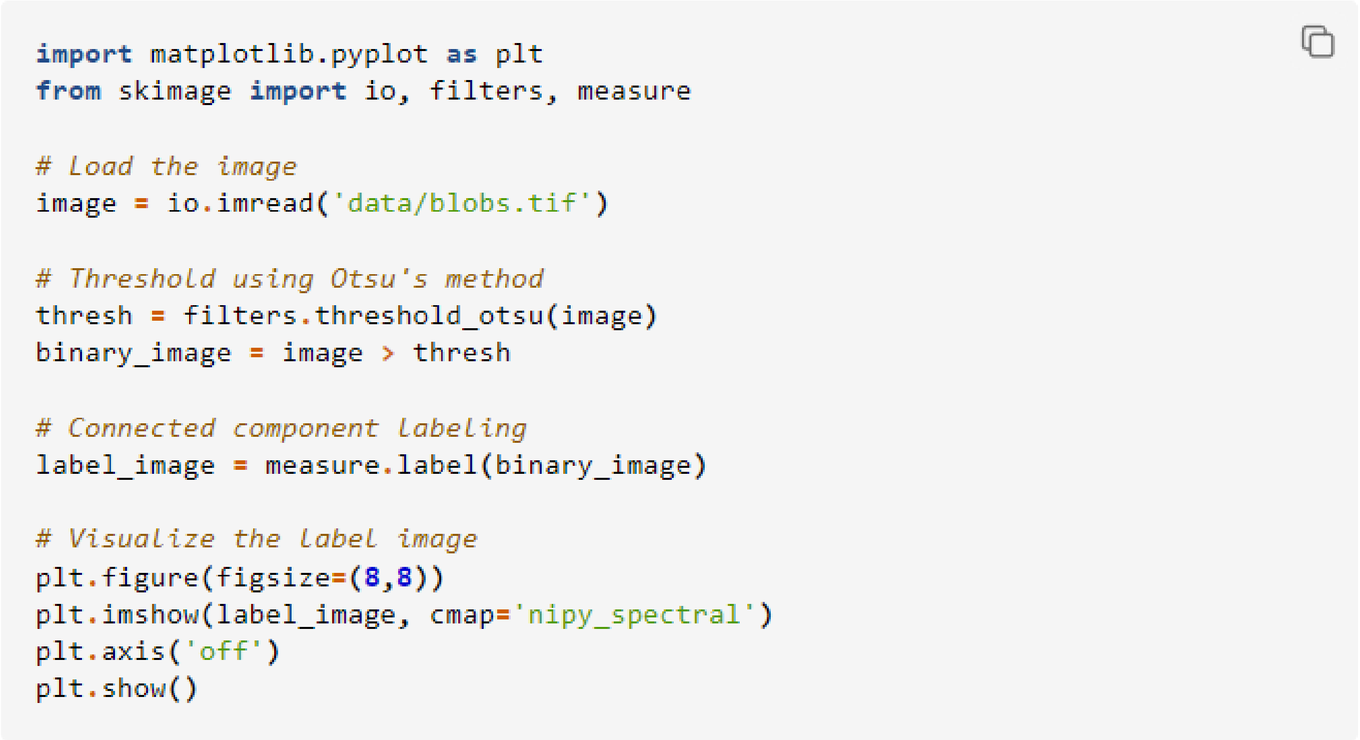
Do not explain the solution.

Only write Python code and short comments are ok.

Assume the code is executed in a Jupyter notebook.

Do not use any of these Python libraries:

* OpenCV
* PIL
* Pillow

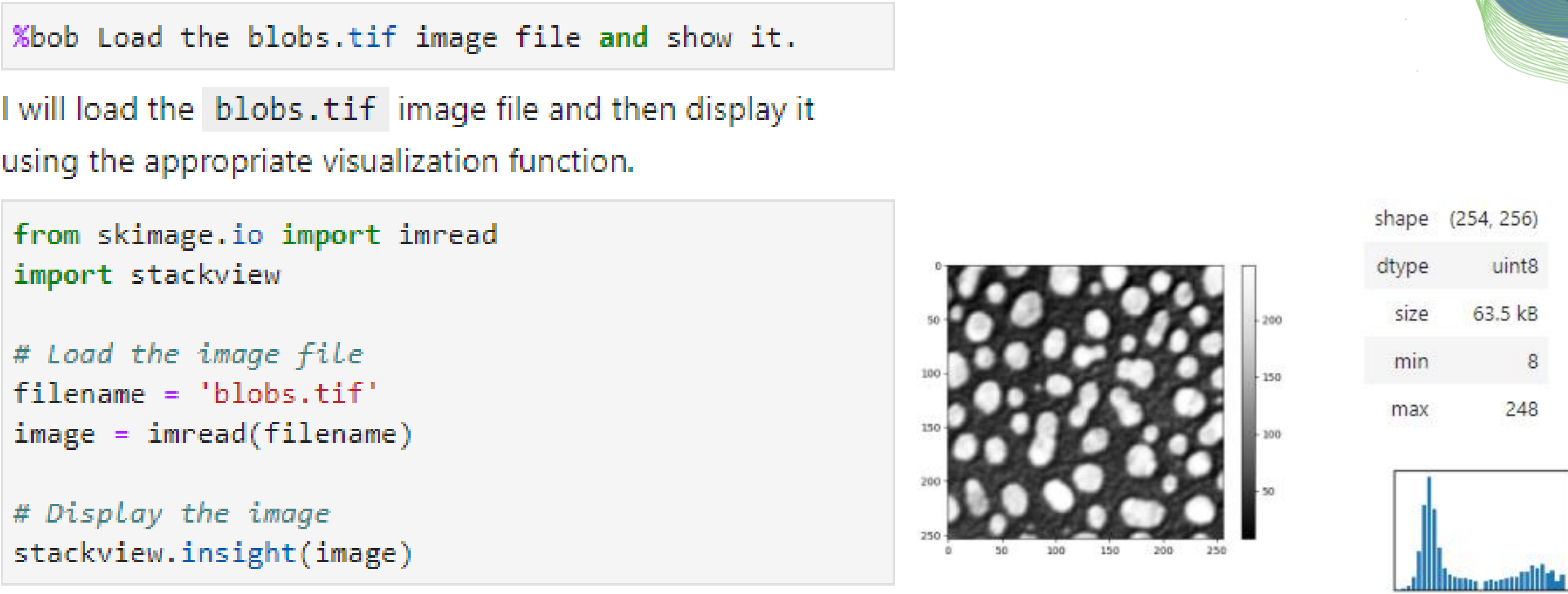


```
import matplotlib.pyplot as plt
from skimage import io, filters, measure

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

# Threshold using Otsu's method
thresh = filters.threshold_otsu(image)
binary_image = image > thresh

# Connected component labeling
label_image = measure.label(binary_image)

# Visualize the label image
plt.figure(figsize=(8,8))
plt.imshow(label_image, cmap='nipy_spectral')
plt.axis('off')
plt.show()
```

https://scads.github.io/generative-ai-notebooks/50_code_generation/03_generating_code.html

Scientific code writing: bia-bob

- First steps: Keep it simple



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

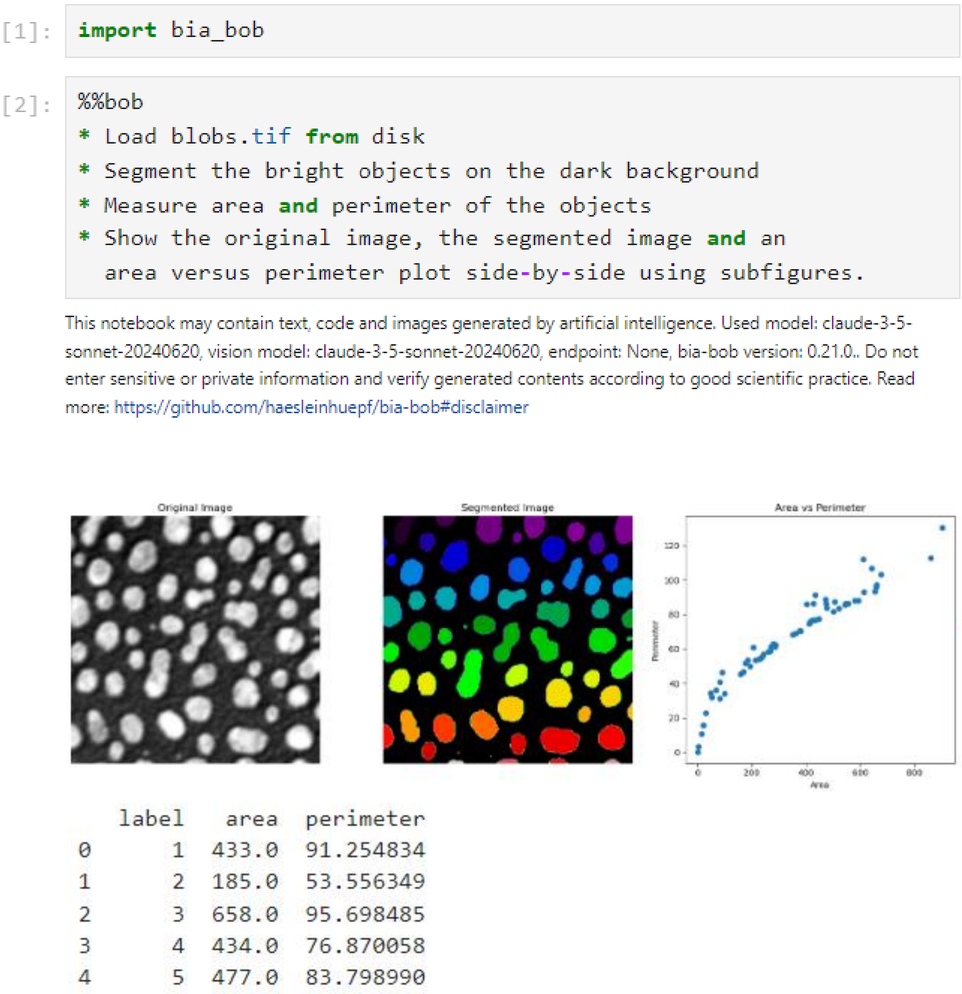
I will load the `blobs.tif` image file and then display it using the appropriate visualization function.

```
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

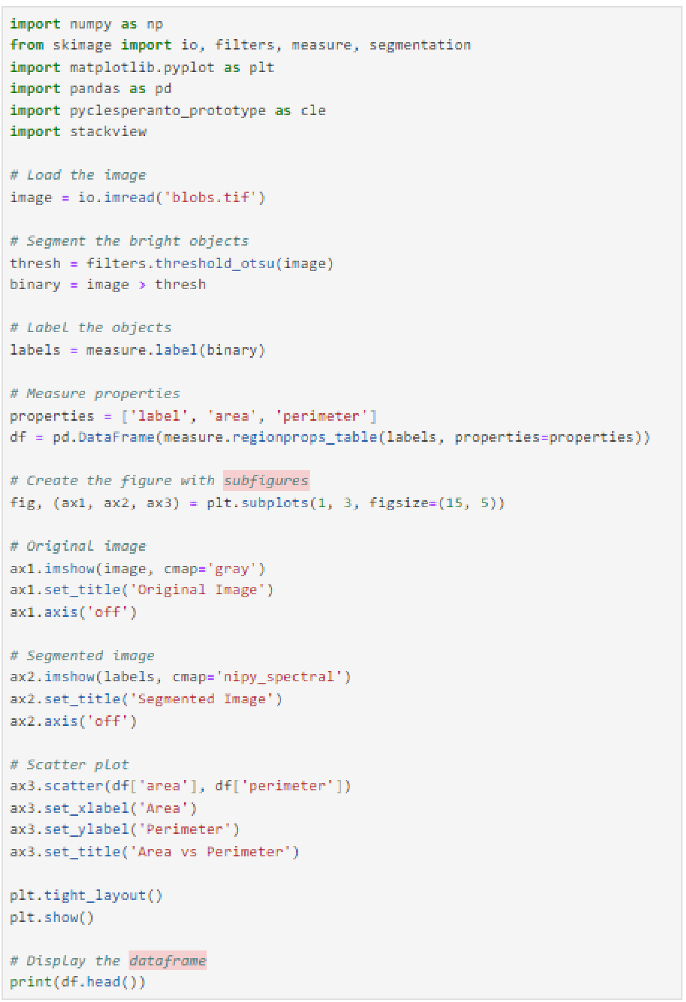


```
[1]: import bia_bob
```

```
[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

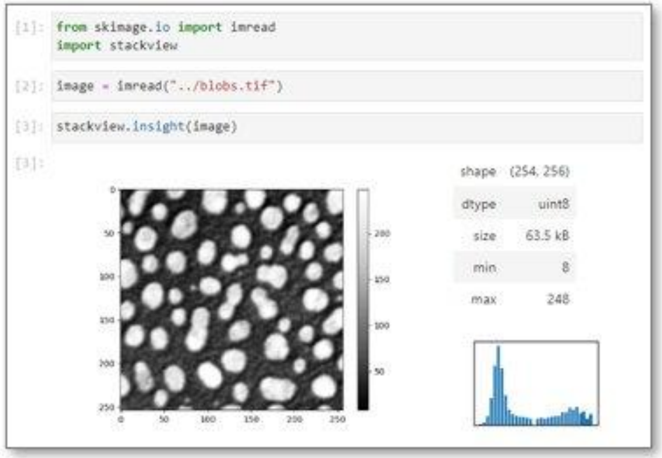
```
   label   area  perimeter
0      1  433.0  91.254834
1      2  185.0  53.556349
2      3  658.0  95.698485
3      4  434.0  76.870058
4      5  477.0  83.798990
```



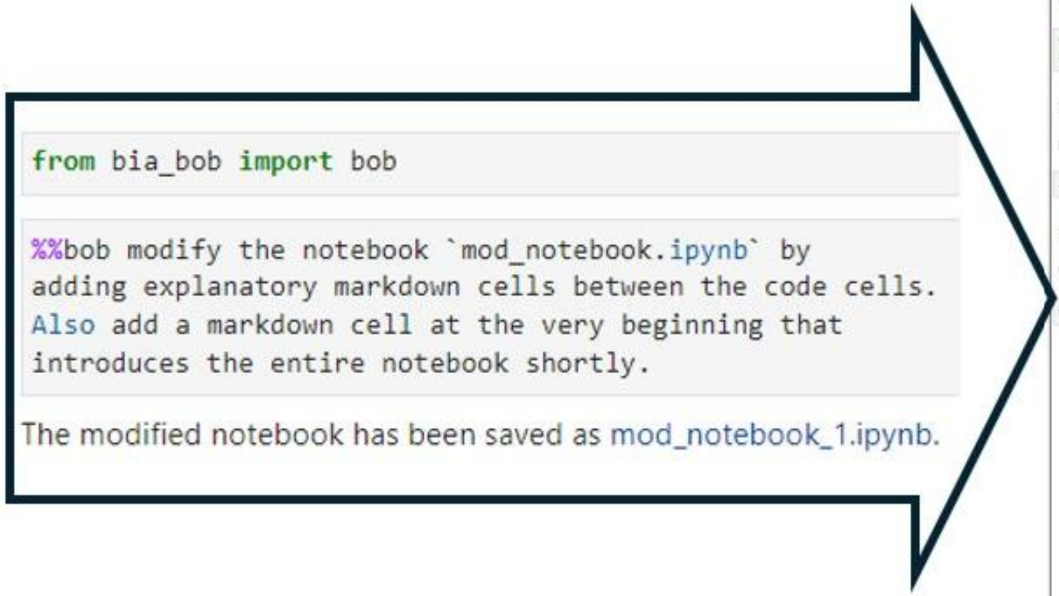
```
import numpy as np
from skimage import io, filters, measure, segmentation
import matplotlib.pyplot as plt
import pandas as pd
import pyclesperanto_prototype as cle
import stackview

# 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())
```

Optimizing documentation

E.g. to make code more readable



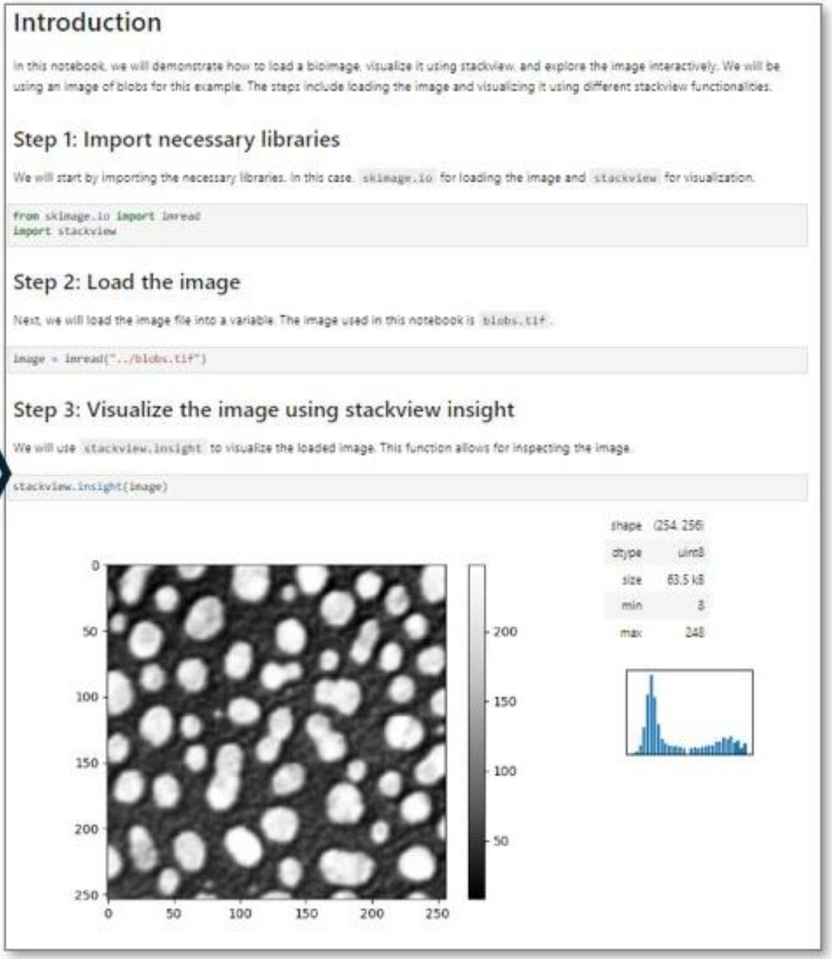
```
[1]: from skimage.io import imread
     import stackview

[2]: image = imread("../blobs.tif")

[3]: stackview.insight(image)

[3]:
shape  (254, 256)
dtype  uint8
size   63.5 kB
min    8
max    248
```



```
from bia_bob import bob
```

```
%%bob modify the notebook `mod_notebook.ipynb` by
adding explanatory markdown cells between the code cells.
Also add a markdown cell at the very beginning that
introduces the entire notebook shortly.
```

The modified notebook has been saved as mod_notebook_1.ipynb.



Introduction

In this notebook, we will demonstrate how to load a bioimage, visualize it using stackview, and explore the image interactively. We will be using an image of blobs for this example. The steps include loading the image and visualizing it using different stackview functionalities.

Step 1: Import necessary libraries

We will start by importing the necessary libraries. In this case, `skimage.io` for loading the image and `stackview` for visualization.

```
from skimage.io import imread
import stackview
```

Step 2: Load the image

Next, we will load the image file into a variable. The image used in this notebook is `blobs.tif`.

```
image = imread("../blobs.tif")
```

Step 3: Visualize the image using stackview insight

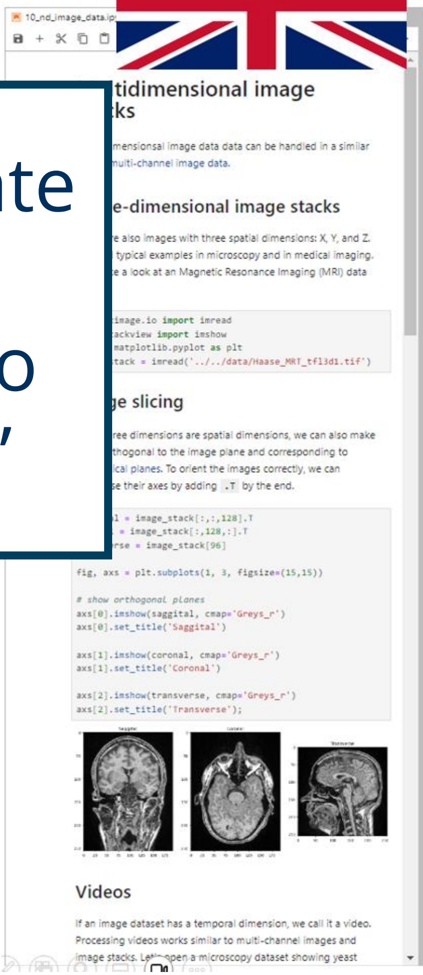
We will use `stackview.insight` to visualize the loaded image. This function allows for inspecting the image.

```
stackview.insight(image)
```

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

Translating notebooks

"Please translate notebook <xyz.ipynb> to <language>."



10_nd_image_data.ip

tidimensional image ks

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

e-dimensional image stacks

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

```
image.io import imread
ackview import imshow
matplotlib.pyplot as plt
tack = imread('../../data/Haase_MRT_tfl3d1.tif')
```

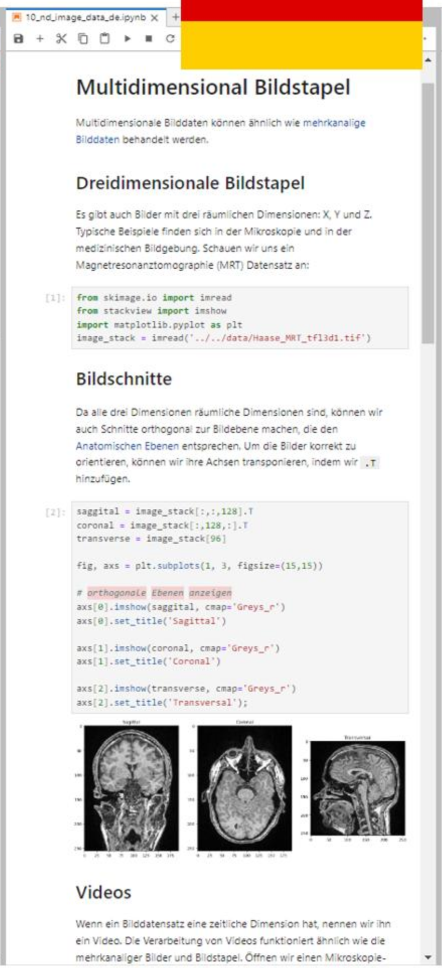
ge slicing

ree dimensions are spatial dimensions, we can also make thogonal to the image plane and corresponding to ical planes. To orient the images correctly, we can se their axes by adding `.T` by the end.

```
l = image_stack[:,:,128].T
 = image_stack[:,128,:].T
rse = image_stack[96]

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

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

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



10_nd_image_data_de.ipynb

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_tfl3d1.tif')
```

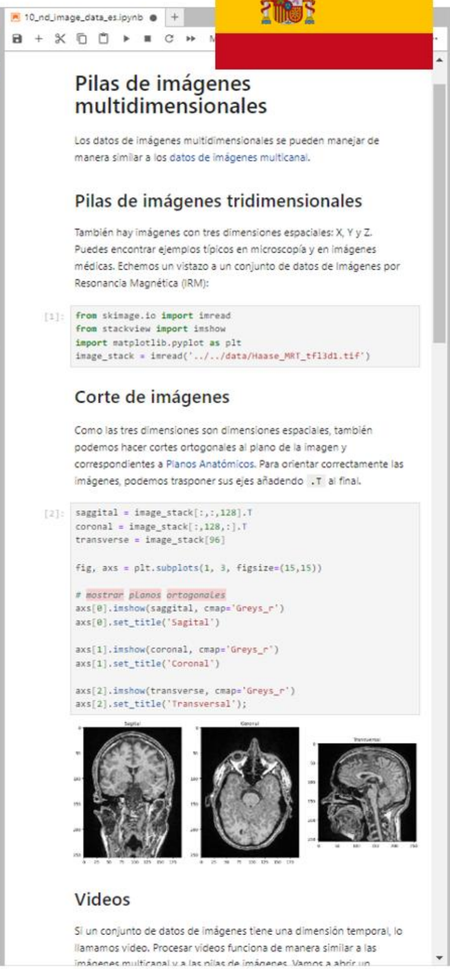
Bildschnitte

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]:
```
saggital = 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(saggital, 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 mehrkanaliger Bilder und Bildstapel. Öffnen wir einen Mikroskopie-



10_nd_image_data_es.ipynb

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_tfl3d1.tif')
```

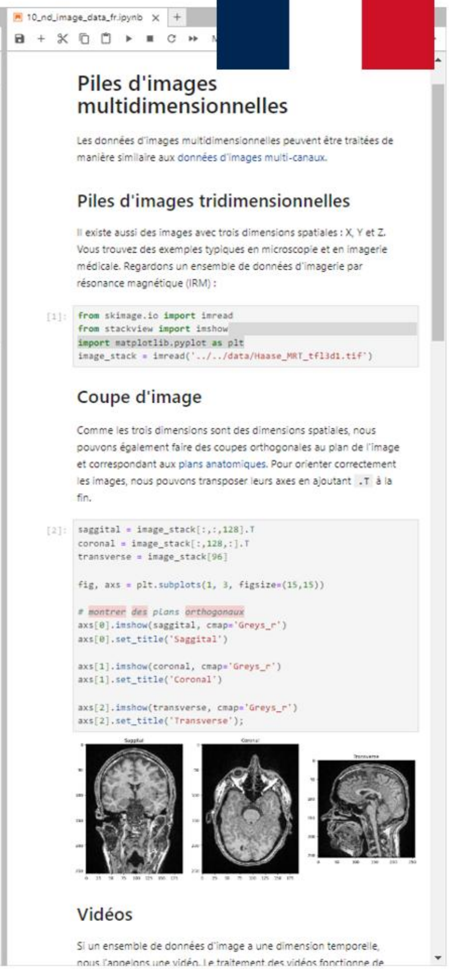
Corte de imágenes

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 trasponer sus ejes añadendo `.T` al final.

[2]:
```
saggital = 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(saggital, 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 y a las pilas de imágenes. Vamos a abrir un



10_nd_image_data_fr.ipynb

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_tfl3d1.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.

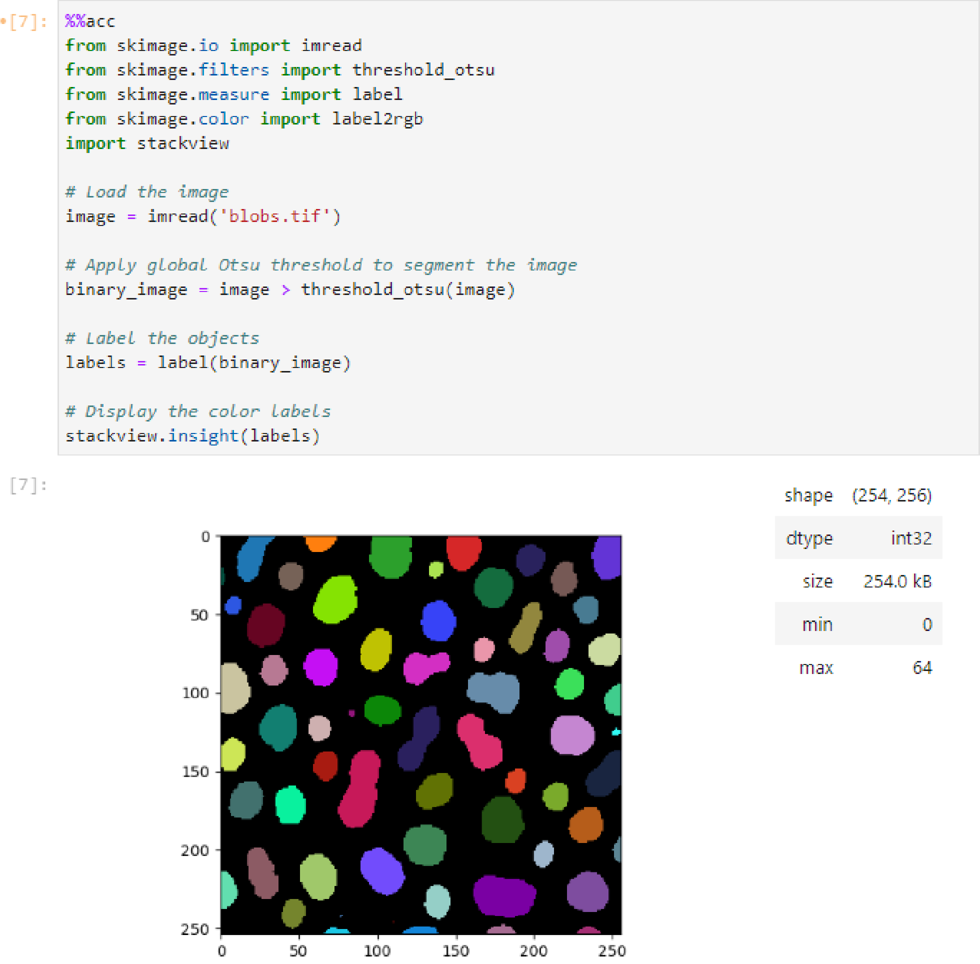
[2]:
```
saggital = image_stack[:,:,128].T
coronal = image_stack[:,128,:].T
transverse = image_stack[96]

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

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

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

GPU-Accelerating code

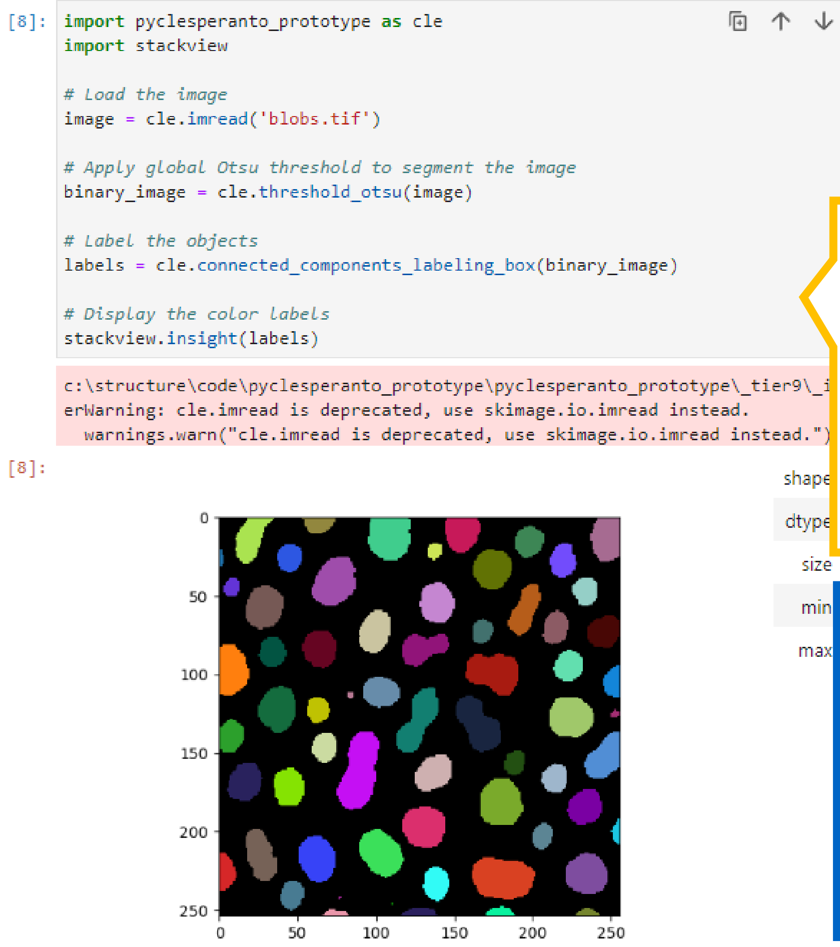
Original:



```
•[7]: %%acc
from skimage.io import imread
from skimage.filters import threshold_otsu
from skimage.measure import label
from skimage.color import label2rgb
import stackview

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

# Apply global Otsu threshold to segment the image
binary_image = image > threshold_otsu(image)

# Label the objects
labels = label(binary_image)

# Display the color labels
stackview.insight(labels)
```

```
[7]:
shape  (254, 256)
dtype       int32
size     254.0 kB
min             0
max            64
```

Replacement:



```
[8]: import pyclesperanto_prototype as cle
import stackview

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

# Apply global Otsu threshold to segment the image
binary_image = cle.threshold_otsu(image)

# Label the objects
labels = cle.connected_components_labeling_box(binary_image)

# Display the color labels
stackview.insight(labels)
```

```
c:\structure\code\pyclesperanto_prototype\pyclesperanto_prototype\_tier9\_i
erWarning: cle.imread is deprecated, use skimage.io.imread instead.
  warnings.warn("cle.imread is deprecated, use skimage.io.imread instead.")
```

```
[8]:
shape
dtype
size   254.0 kB
min
max
```

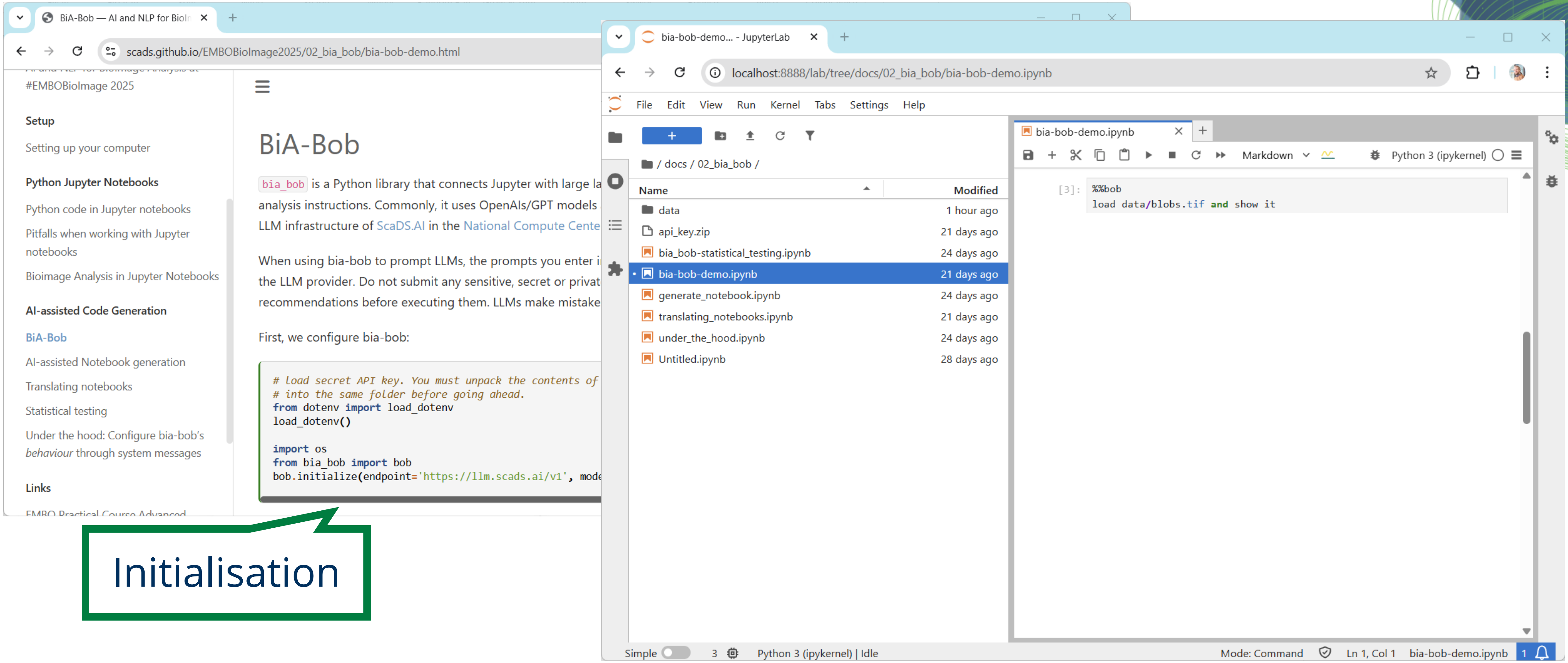
It's our responsibility that results are comparable!

https://link.springer.com/chapter/10.1007/978-3-030-76394-7_5

Exercises

https://scads.github.io/llm-bia-2025/

Bia-Bob: code generation



BiA-Bob — AI and NLP for Biol
scads.github.io/EMBOBioImage2025/02_bia_bob/bia-bob-demo.html

#EMBOBioImage 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

AI-assisted Code Generation

BiA-Bob

AI-assisted Notebook generation

Translating notebooks

Statistical testing

Under the hood: Configure bia-bob's *behaviour* through system messages

Links

BiA-Bob

`bia_bob` is a Python library that connects Jupyter with large la analysis instructions. Commonly, it uses OpenAIs/GPT models LLM infrastructure of ScaDS.AI in the National Compute Cente

When using bia-bob to prompt LLMs, the prompts you enter i the LLM provider. Do not submit any sensitive, secret or privat recommendations before executing them. LLMs make mistake

First, we configure bia-bob:

```
# load secret API key. You must unpack the contents of
# into the same folder before going ahead.
from dotenv import load_dotenv
load_dotenv()

import os
from bia_bob import bob
bob.initialize(endpoint='https://llm.scads.ai/v1', mode
```

Name	Modified
data	1 hour ago
api_key.zip	21 days ago
bia_bob-statistical_testing.ipynb	24 days ago
bia-bob-demo.ipynb	21 days ago
generate_notebook.ipynb	24 days ago
translating_notebooks.ipynb	21 days ago
under_the_hood.ipynb	24 days ago
Untitled.ipynb	28 days ago

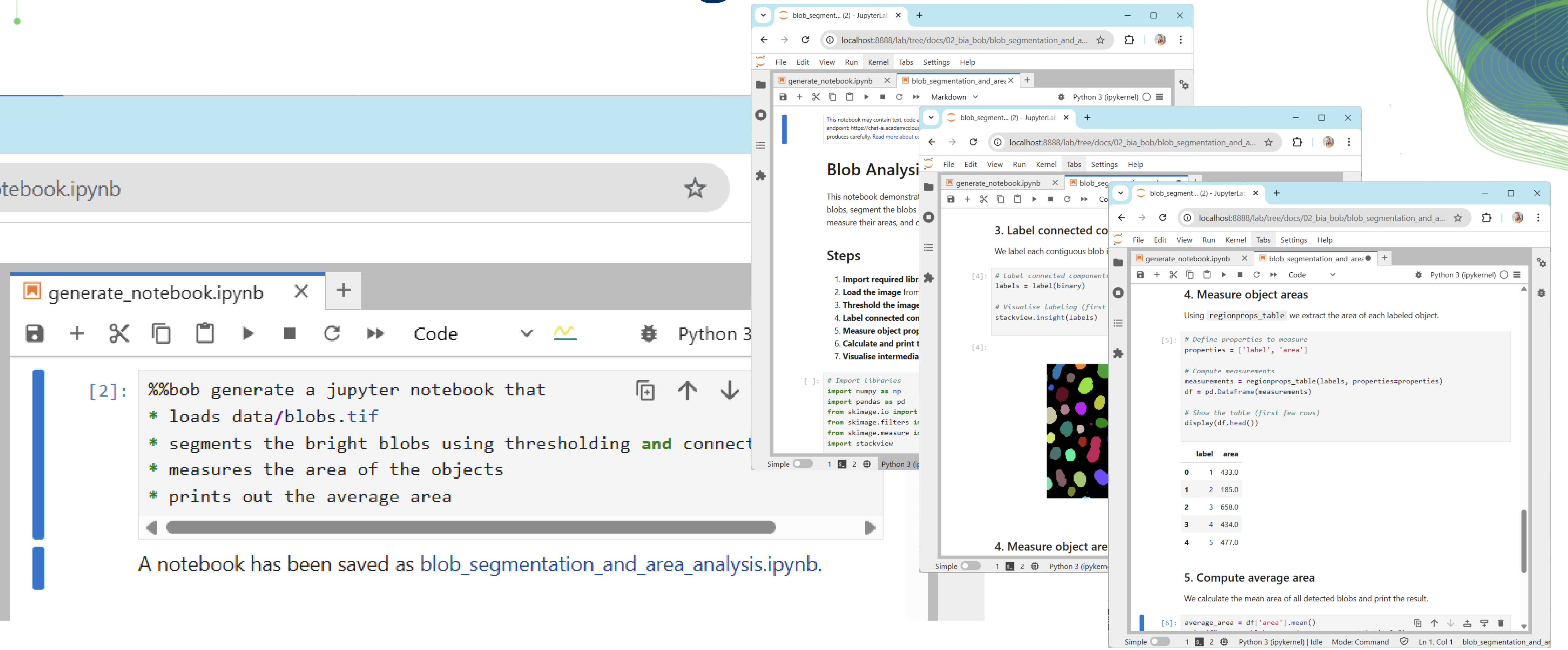
```
[3]: %%bob
     load data/blobs.tif and show it
```

Simple 3 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col 1 bia-bob-demo.ipynb 1

Initialisation

https://scads.github.io/llm-bia-2025/02_bia_bob/bia-bob-demo.html

Bia-Bob: notebook generation



```
[2]: %%bob generate a jupyter notebook that
     * loads data/blobs.tif
     * segments the bright blobs using thresholding and connect
     * measures the area of the objects
     * prints out the average area
```

A notebook has been saved as blob_segmentation_and_area_analysis.ipynb.

blob_segment... (2) - JupyterLab
localhost:8888/lab/tree/docs/02_bia_bob/blob_segmentation_and_a...
File
Edit
View
Run
Kernel
Tabs
Settings
Help
generate_notebook.ipynb
blob_segmentation_and_area
Markdown
Python 3 (ipykernel)
Simple

This notebook may contain text, code a
endpoint: https://chat-ai.academicclou
produces carefully. Read more about c

Blob Analysi

This notebook demonstrat
blobs, segment the blobs
measure their areas, and c

Steps

1. **Import required libr**
2. **Load the image** from
3. **Threshold the image**
4. **Label connected com**
5. **Measure object prop**
6. **Calculate and print t**
7. **Visualise intermedia**

```
[ ]: # Import libraries
     import numpy as np
     import pandas as pd
     from skimage.io import
     from skimage.filters in
     from skimage.measure in
     import stackview
```

3. Label connected co

We label each contiguous blob i

```
[4]: # Label connected components
     labels = label(binary)

     # Visualise labeling (first
     stackview.insight(labels)
```

4. Measure object are

4. Measure object areas

Using `regionprops_table` we extract the area of each labeled object.

```
[5]: # Define properties to measure
     properties = ['label', 'area']

     # Compute measurements
     measurements = regionprops_table(labels, properties=properties)
     df = pd.DataFrame(measurements)

     # Show the table (first few rows)
     display(df.head())
```

```
   label  area
0      1  433.0
1      2  185.0
2      3  658.0
3      4  434.0
4      5  477.0
```

5. Compute average area

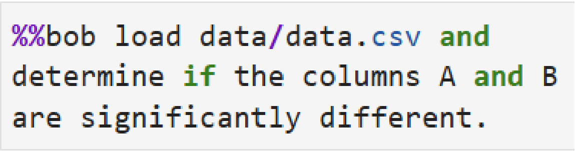
We calculate the mean area of all detected blobs and print the result.

```
[6]: average_area = df['area'].mean()
```

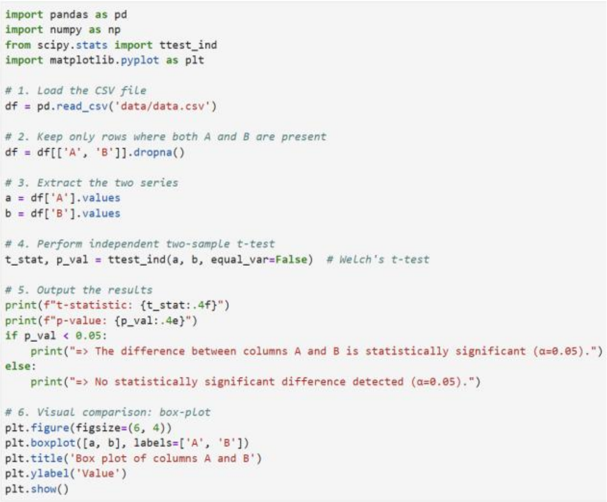
https://scads.github.io/llm-bia-2025/02_bia_bob/generate_notebook.html

Code for statistical analysis

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

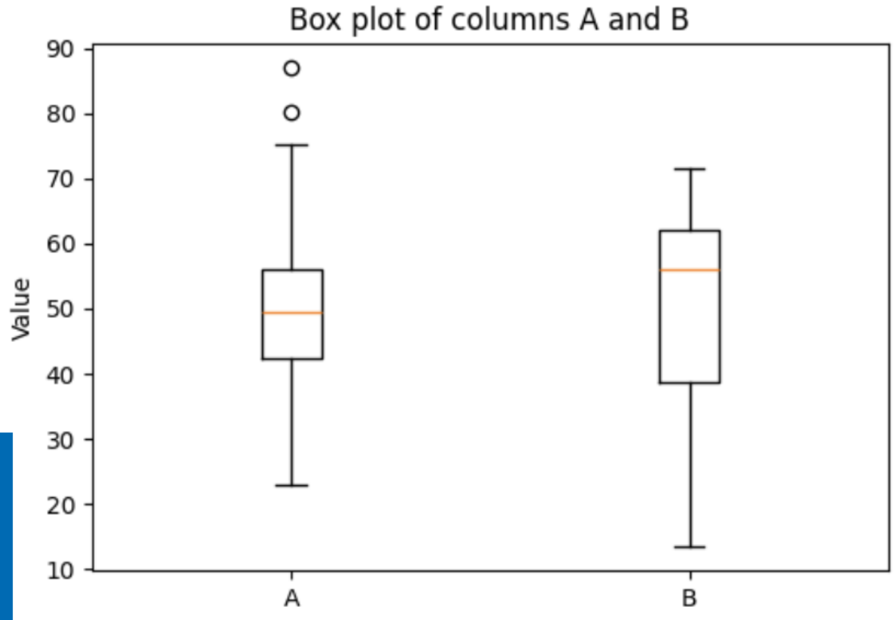


```
%%bob load data/data.csv and
determine if the columns A and B
are significantly different.
```



```python
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 (α=0.05).
```

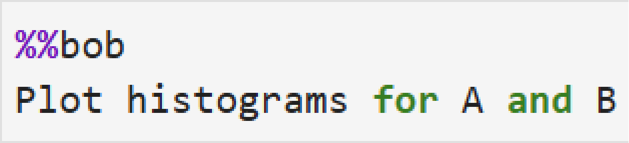


LLMs often provide “simple” solutions.

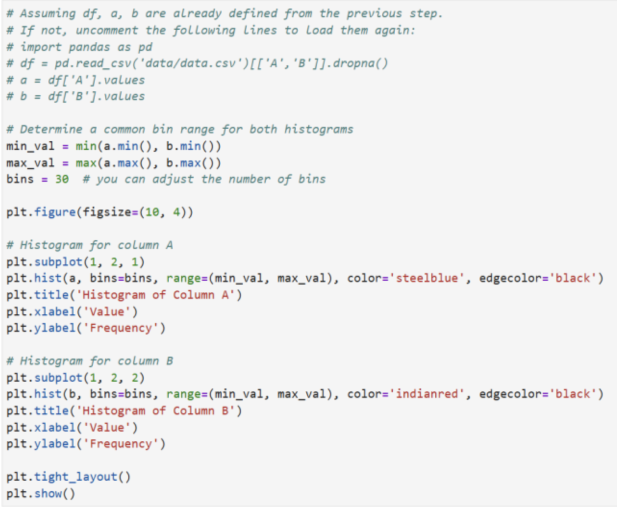
https://scads.github.io/embo-lsm-bia-2025/03_bia_bob/bia_bob-statistical_testing.html

Code for statistical analysis

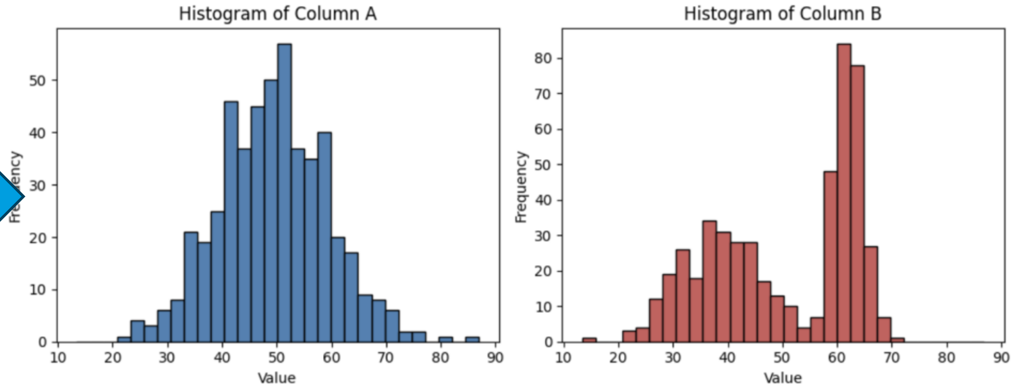
Be careful when prompting for code and provide as many details as you can!



```
%%bob
Plot histograms for A and B
```



```python
# 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()
```

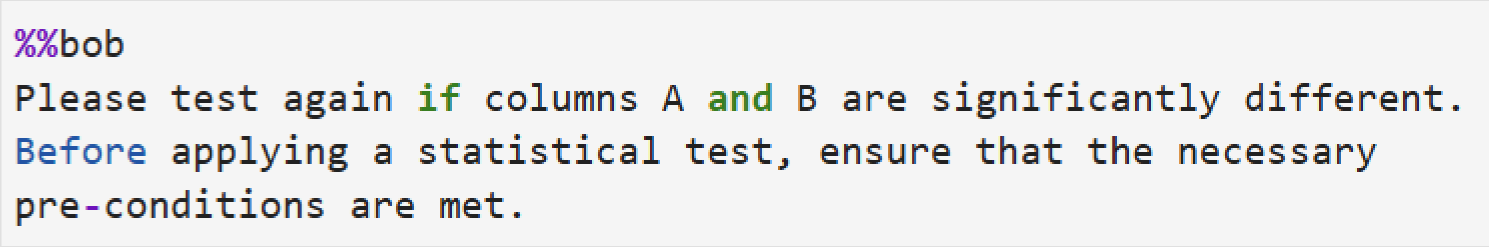


Look at your data!

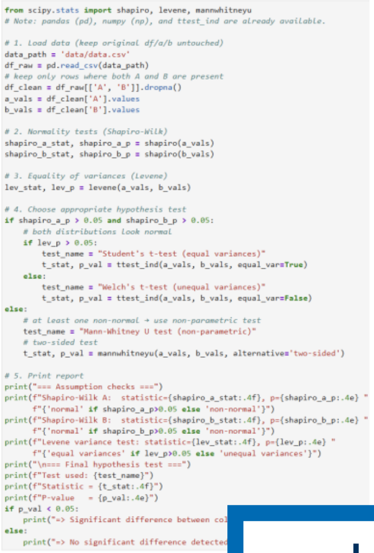
https://scads.github.io/embo-lsm-bia-2025/03_bia_bob/bia_bob-statistical_testing.html

Code for statistical analysis

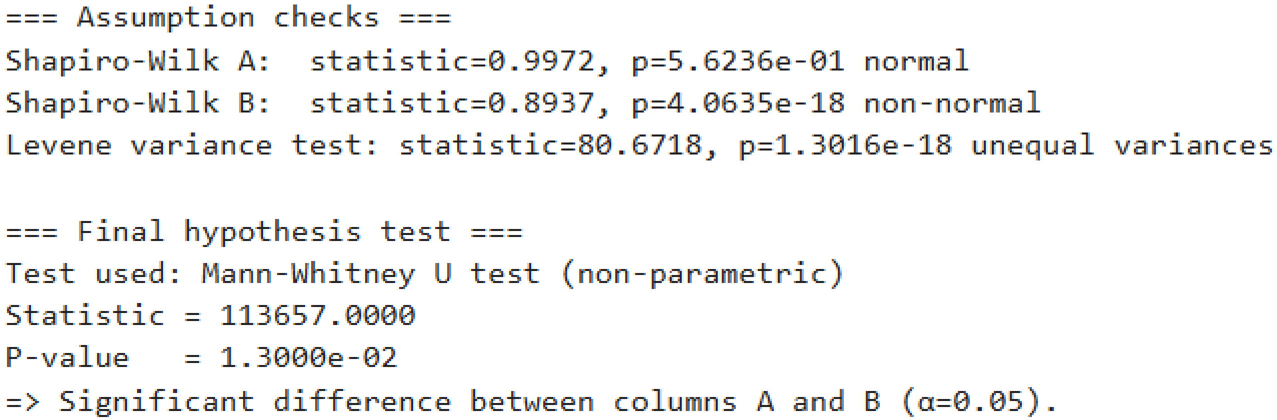
Be careful when prompting for code and provide as many details as you can!



```
%%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.
```



```python
from scipy.stats import shapiro, levene, mannwhitneyu
# Note: pandas (pd), numpy (np), and ttest_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[['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 = ttest_ind(a_vals, b_vals, equal_var=True)
    else:
        test_name = "Welch's t-test (unequal variances)"
        t_stat, p_val = ttest_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, alternative='two-sided')

# 5. Print report
print("=== Assumption checks ===")
print(f"Shapiro-Wilk A:  statistic={shapiro_a_stat:.4f}, p={shapiro_a_p:.4e} "
      f"{'normal' if shapiro_a_p>0.05 else 'non-normal'}")
print(f"Shapiro-Wilk B:  statistic={shapiro_b_stat:.4f}, p={shapiro_b_p:.4e} "
      f"{'normal' if shapiro_b_p>0.05 else 'non-normal'}")
print(f"Levene variance test: statistic={lev_stat:.4f}, p={lev_p:.4e} "
      f"{'equal variances' if lev_p>0.05 else 'unequal variances'}")
print("\n=== Final hypothesis test ===")
print(f"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 col
else:
    print("=> No significant difference detected
```



```
=== 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 (α=0.05).
```

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

https://scads.github.io/embo-lsm-bia-2025/03_bia_bob/bia_bob-statistical_testing.html

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

GloBIAS Global BioImage Analysts' Society

NFDI4 BIOIMAGE

image.sc

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.

DFG Deutsche Forschungsgemeinschaft

BIDS Team @ ScaDS.AI / UL

Lea Kabjesz, Lea Gihlein, Mara Lampert

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