

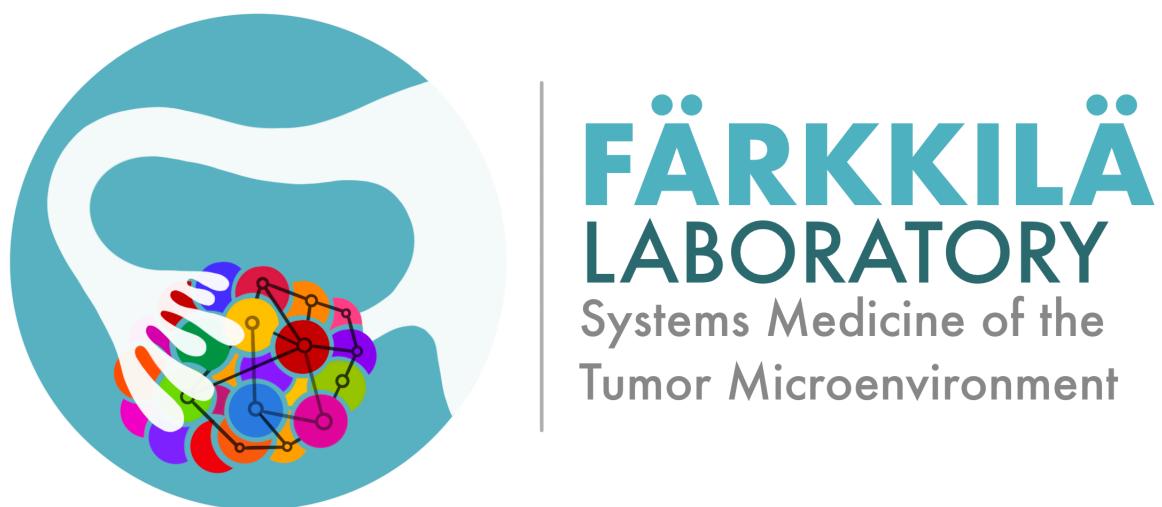


Tutorial: In-house t-CyCIF image preprocessing pipeline

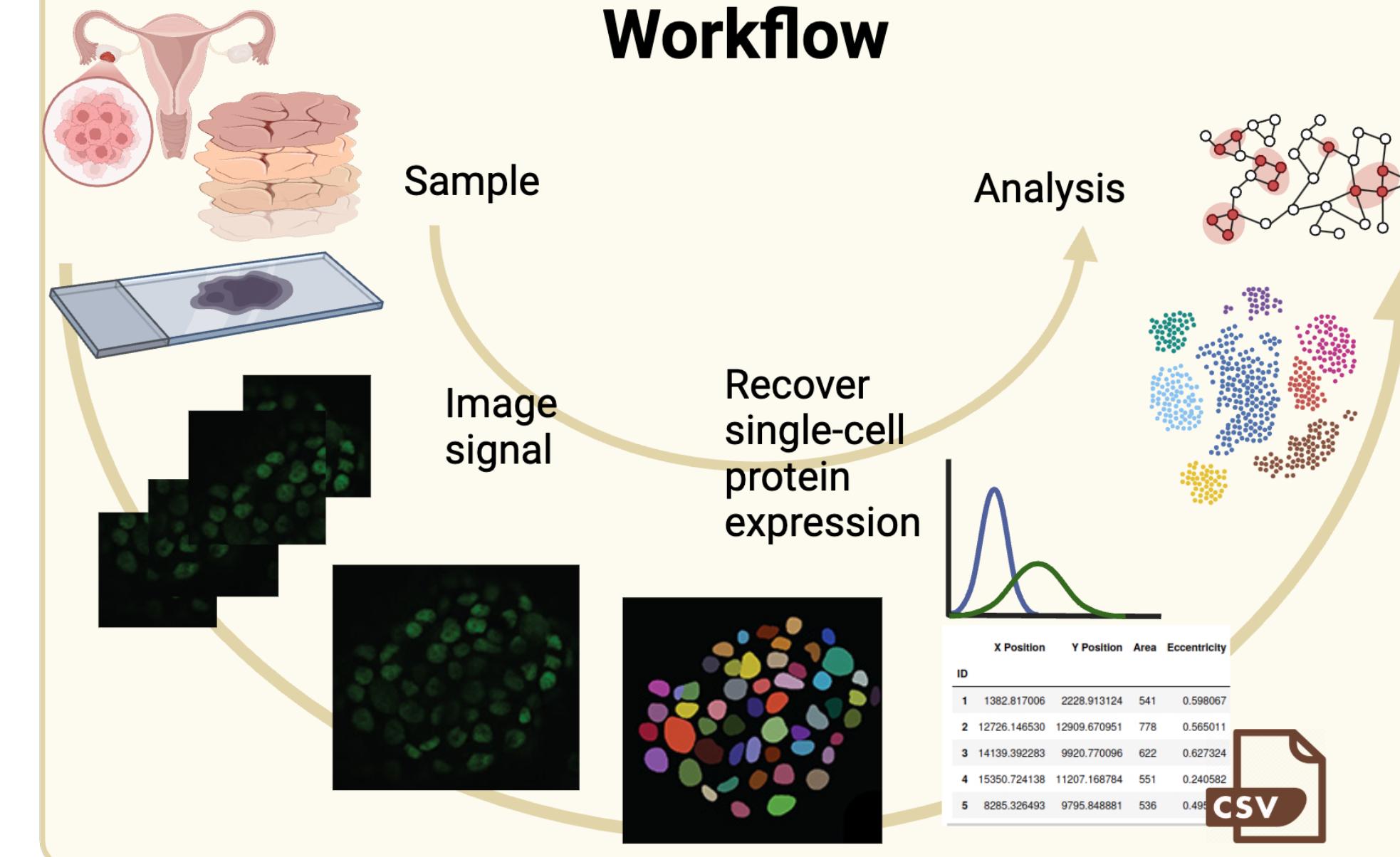
Färkkilä Lab Seminar (OncoTalks)

Ziqi Kang, 27.02.2025

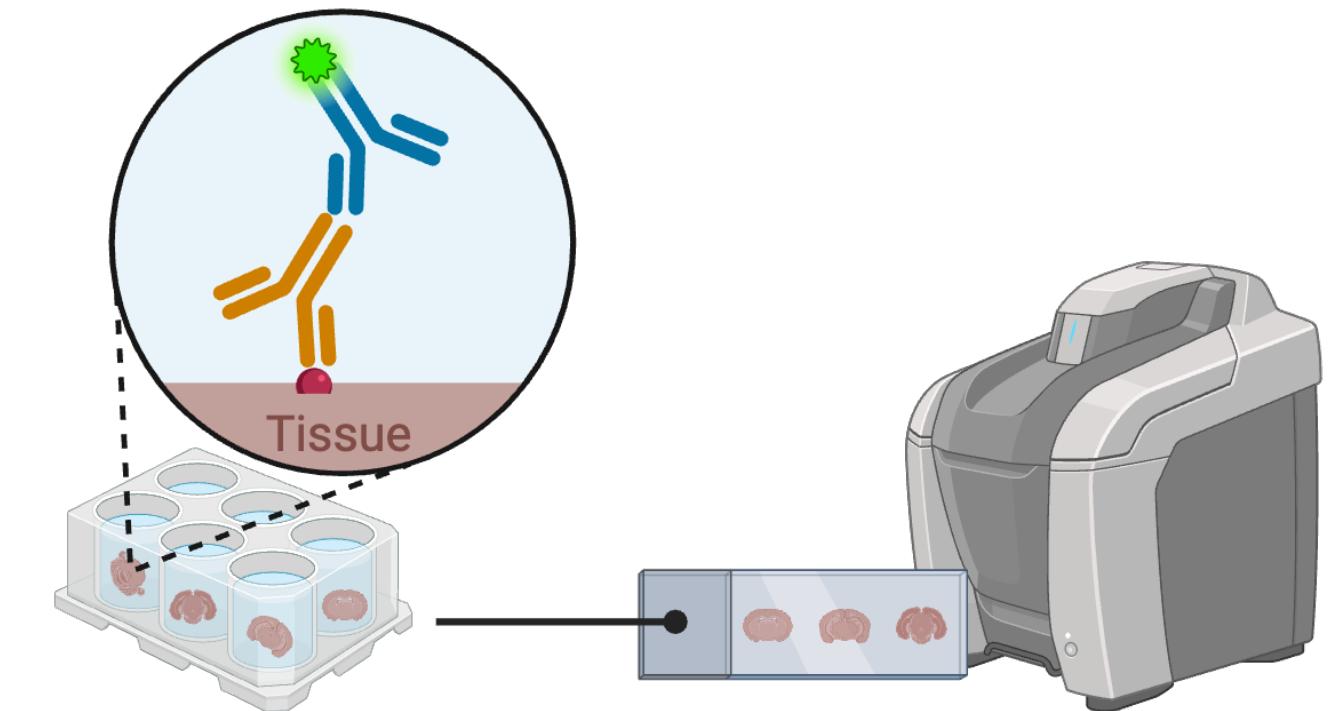
Thanks to Lina: providing the material



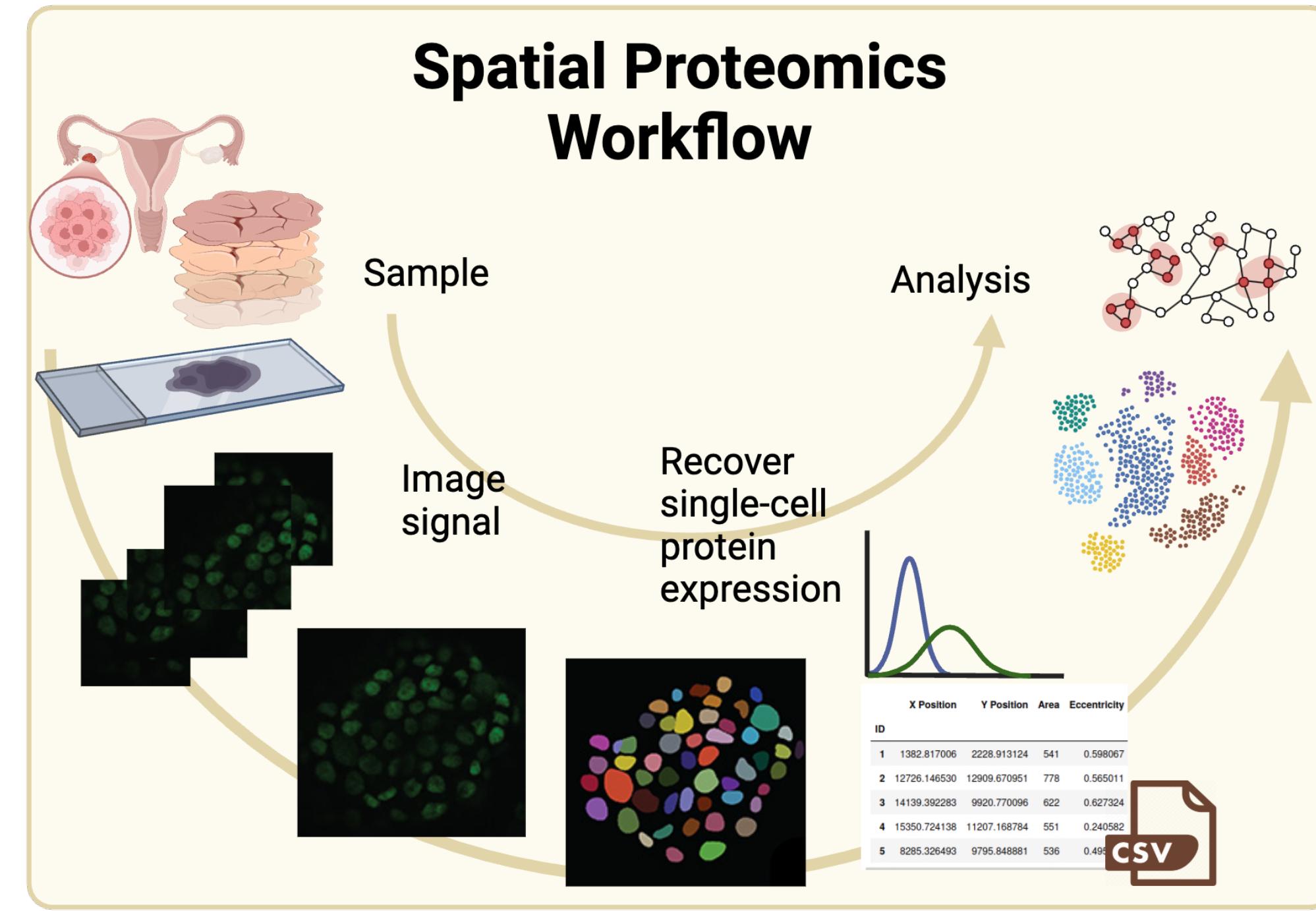
Spatial Proteomics Workflow



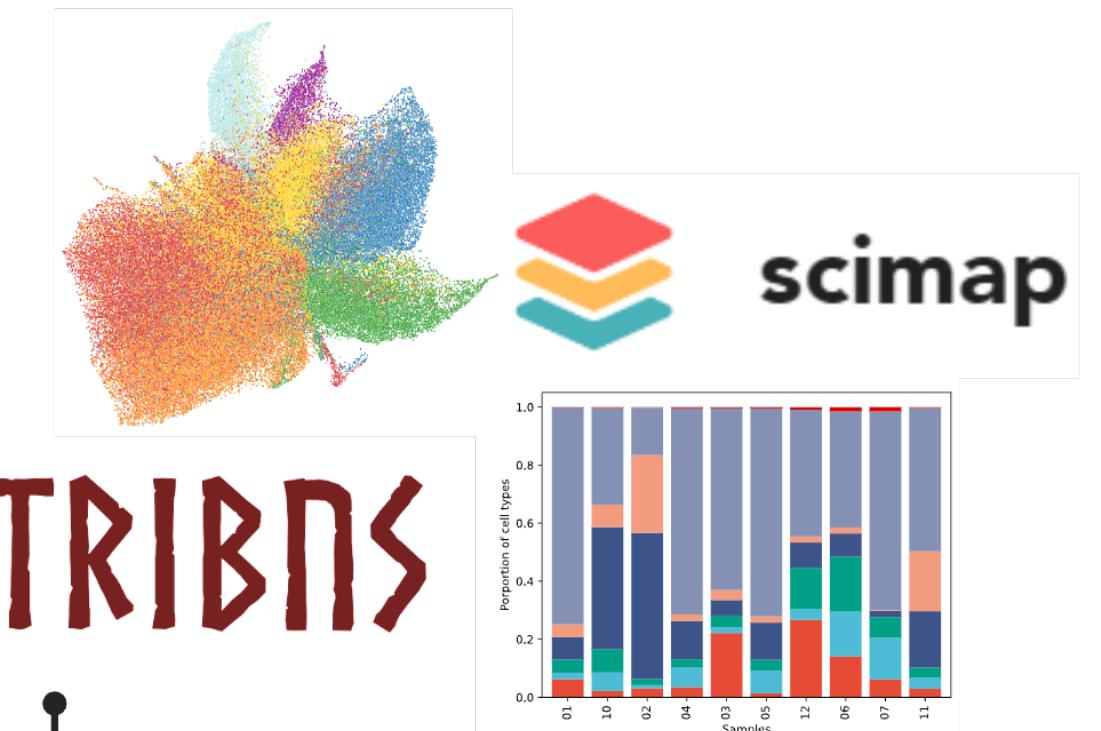
STAINING AND IMAGE ACQUISITION



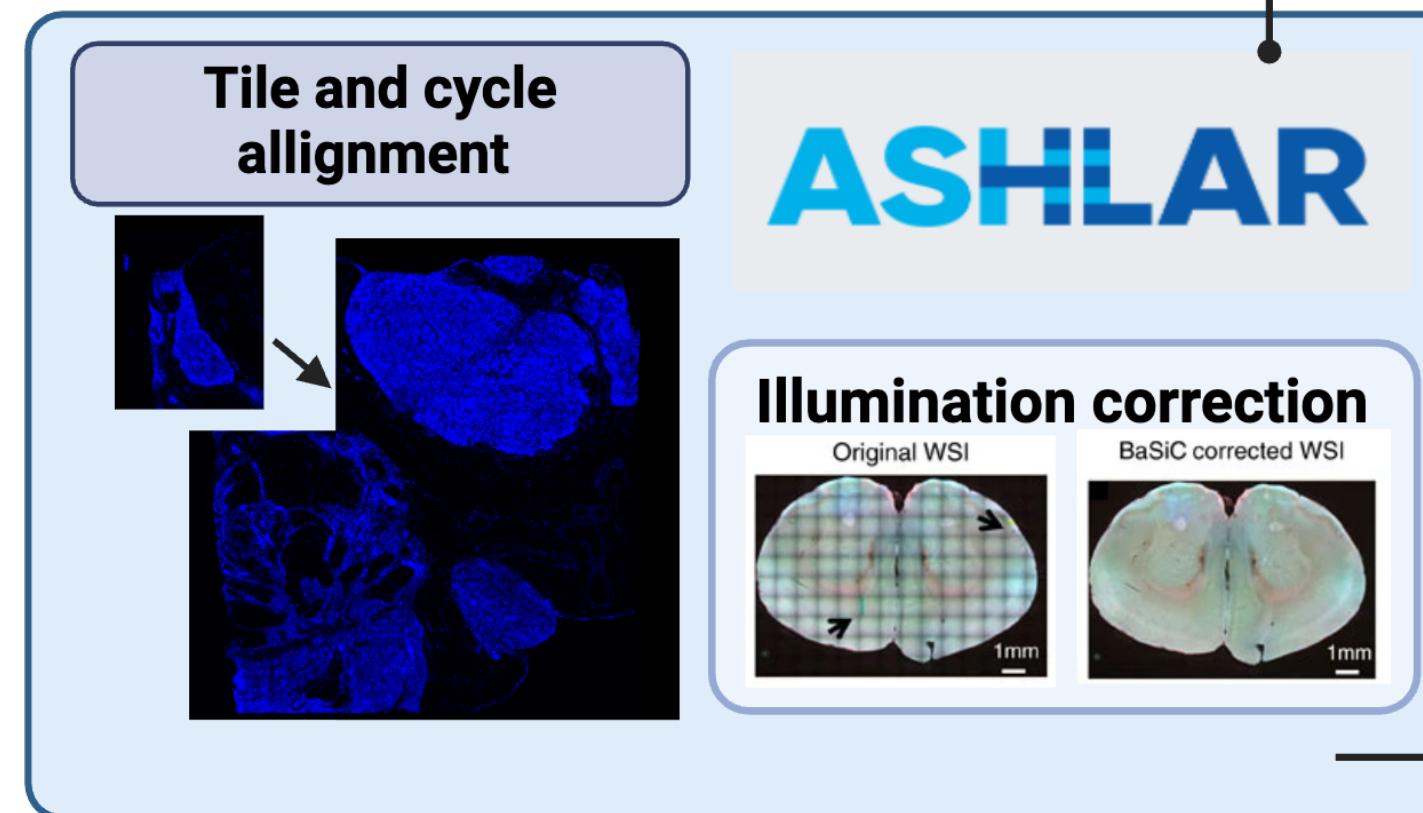
Spatial Proteomics Workflow



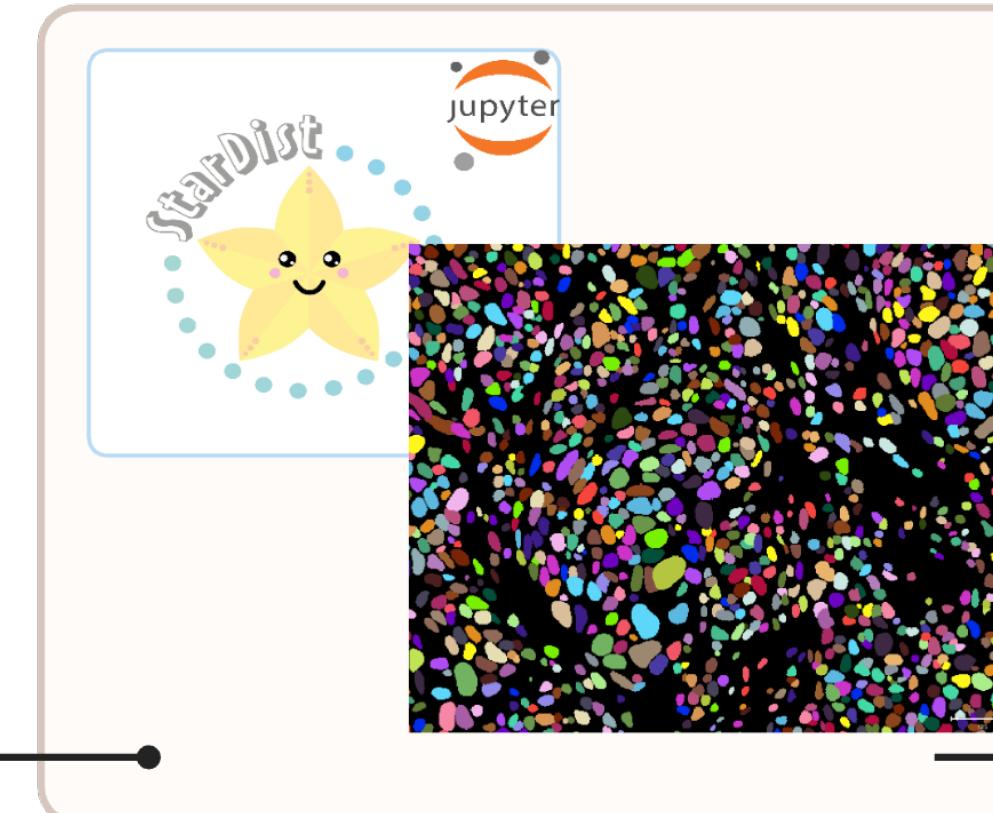
DATA ANALYSIS



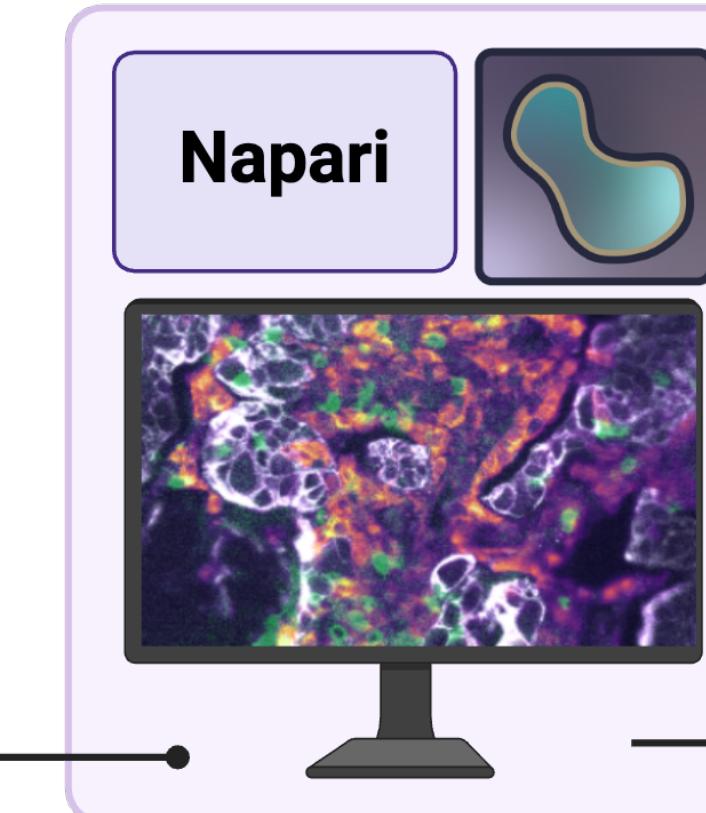
STITCHING



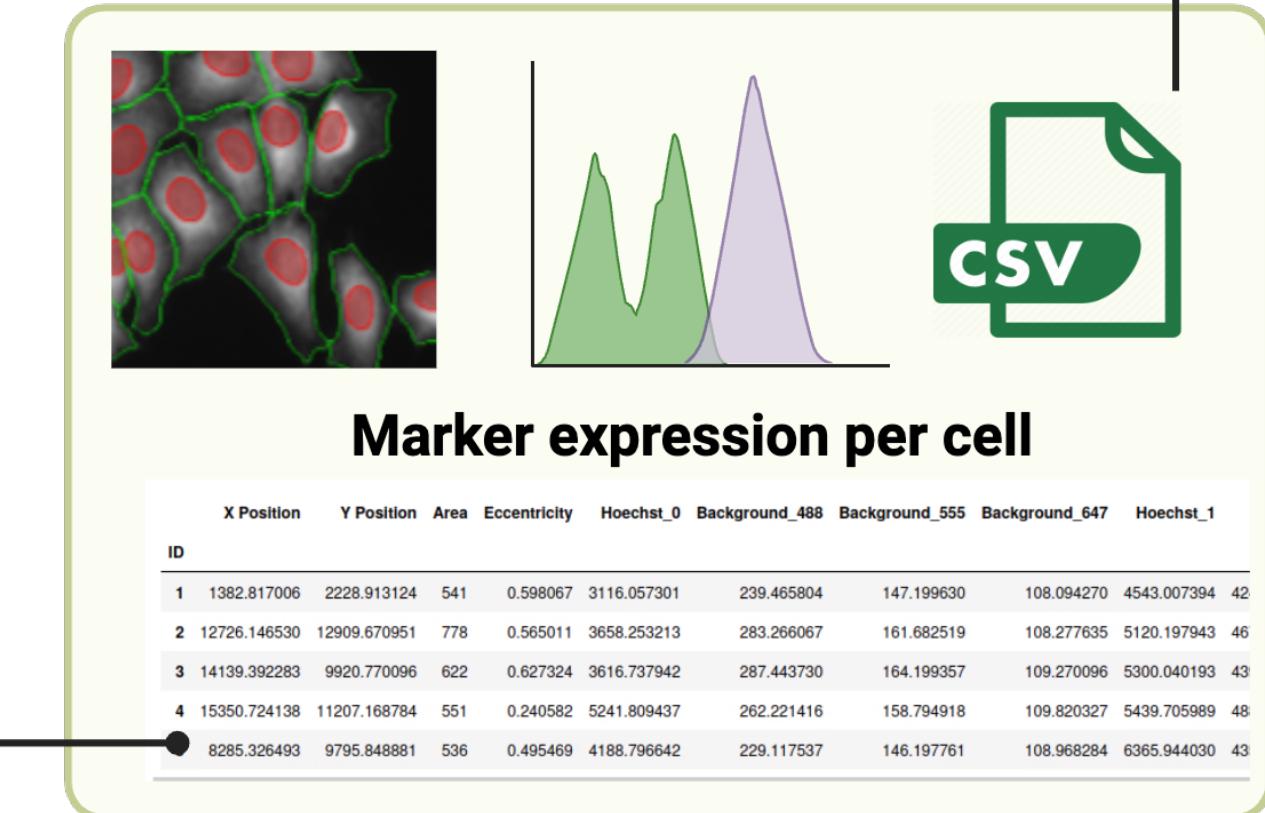
SEGMENTATION



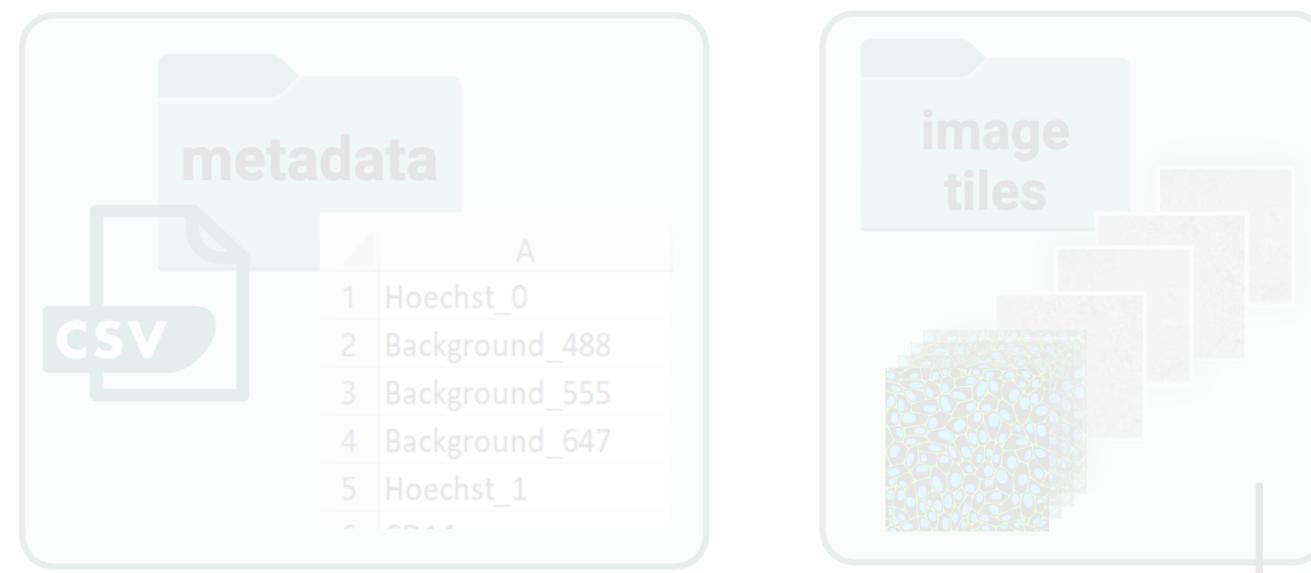
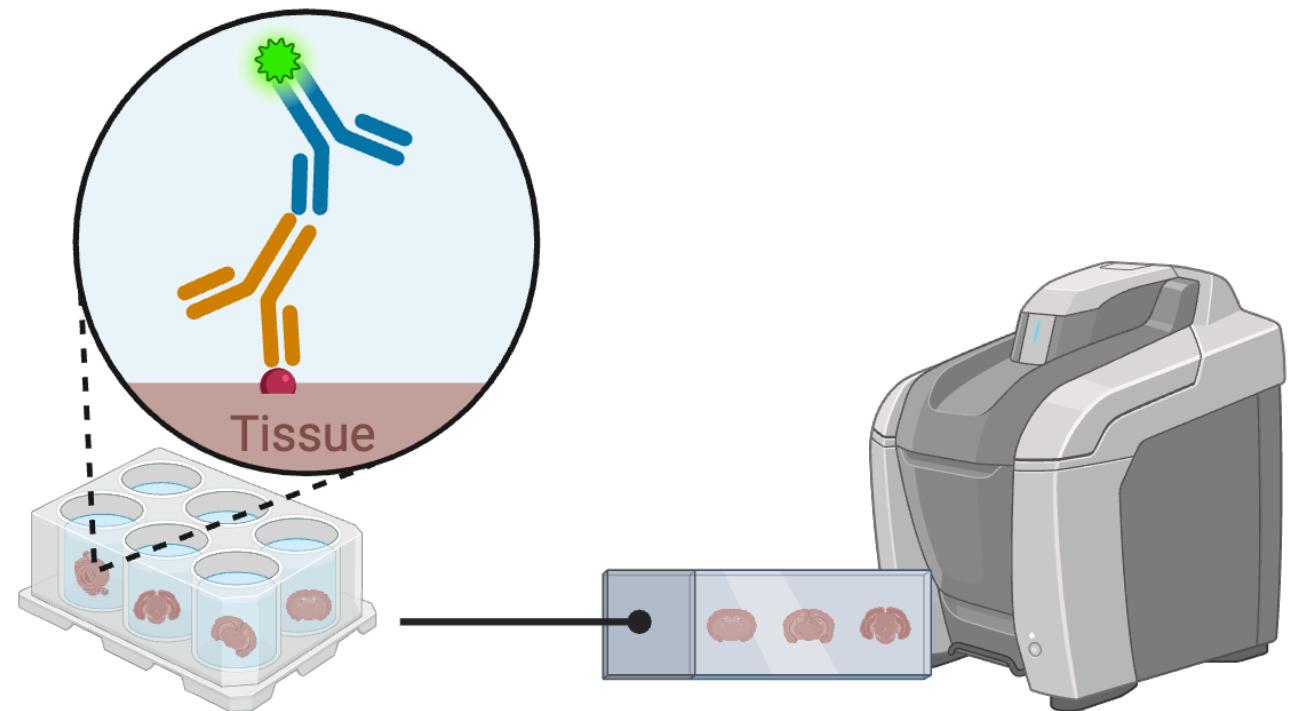
VISUALIZATION



QUANTIFICATION

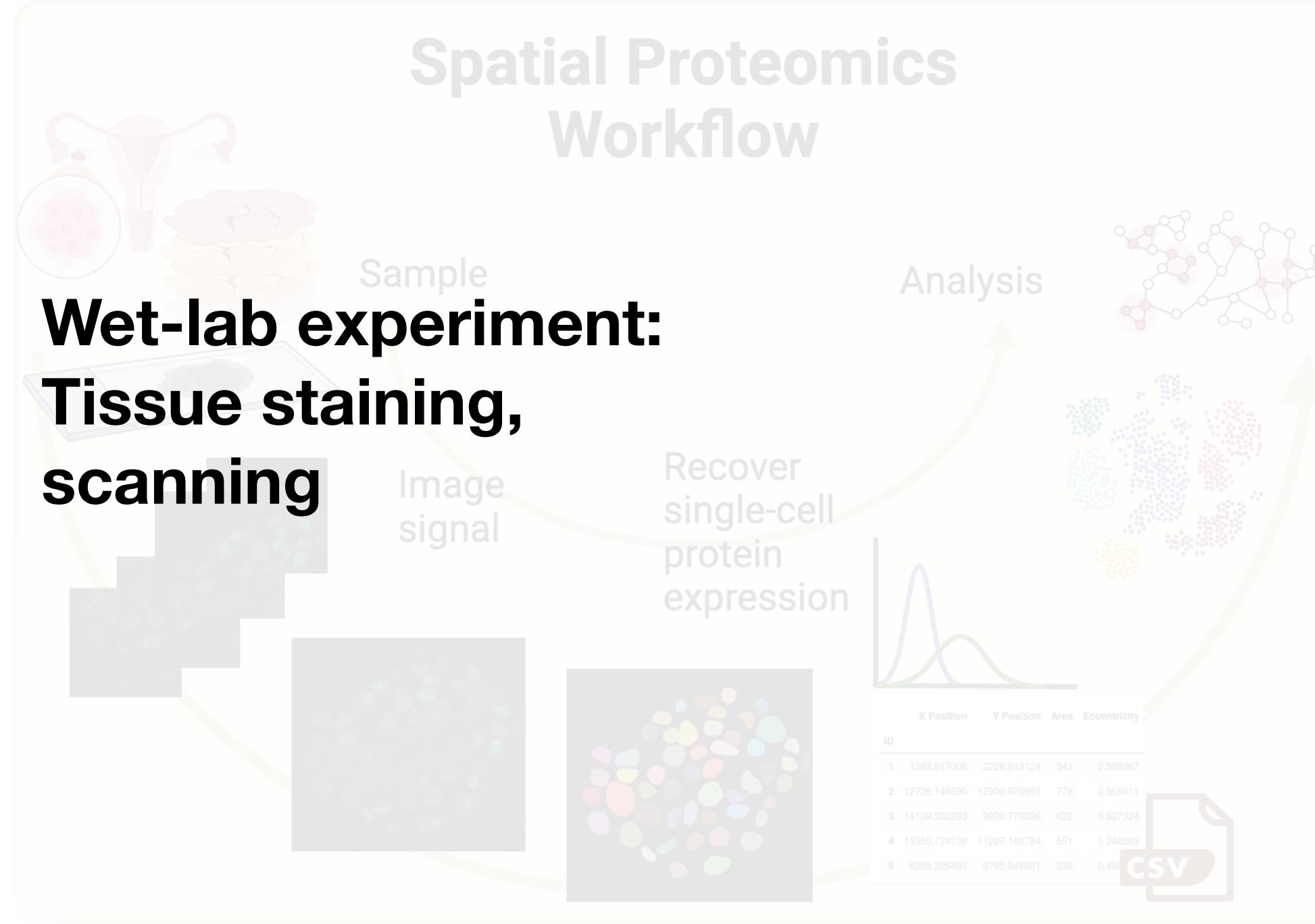


STAINING AND IMAGE ACQUISITION

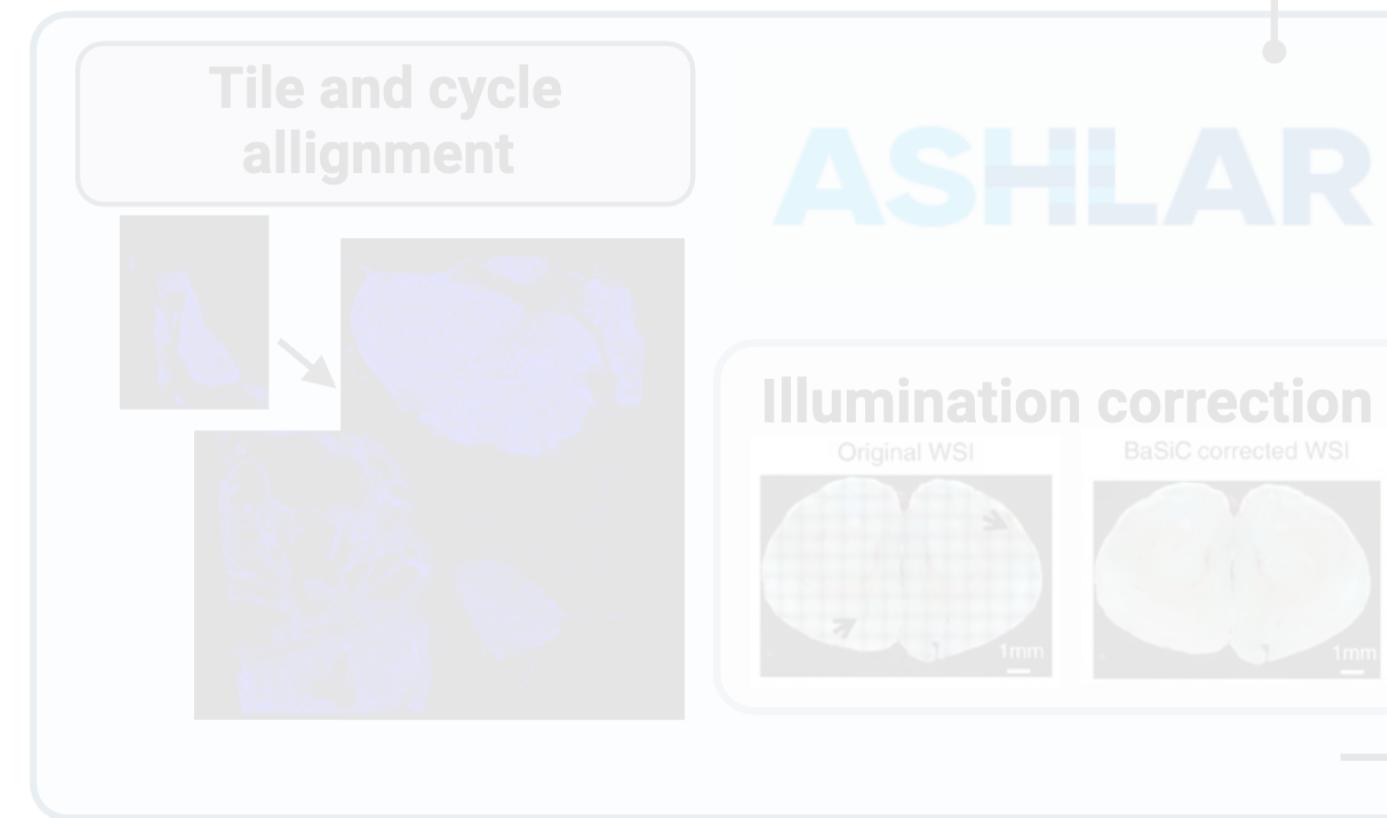


Spatial Proteomics Workflow

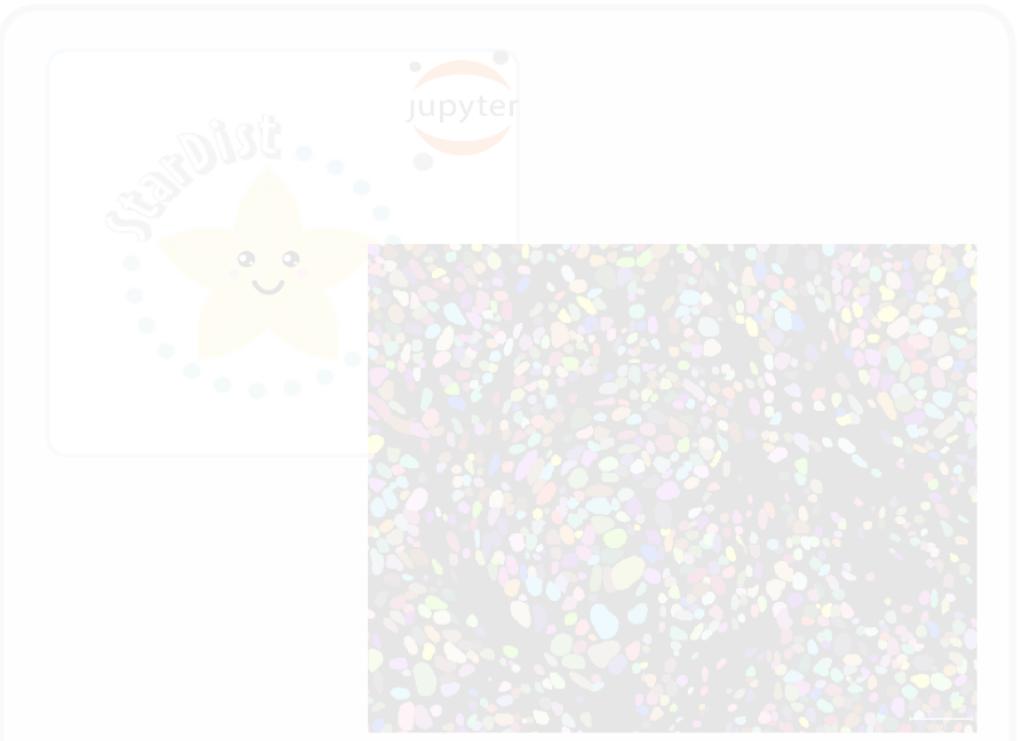
Wet-lab experiment: Tissue staining, scanning



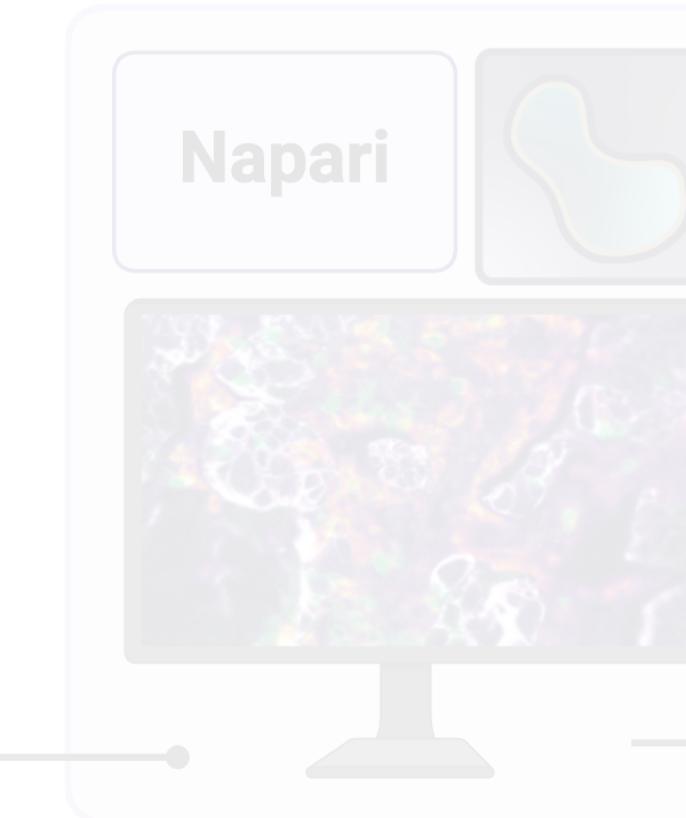
STITCHING



SEGMENTATION



VISUALIZATION



DATA ANALYSIS

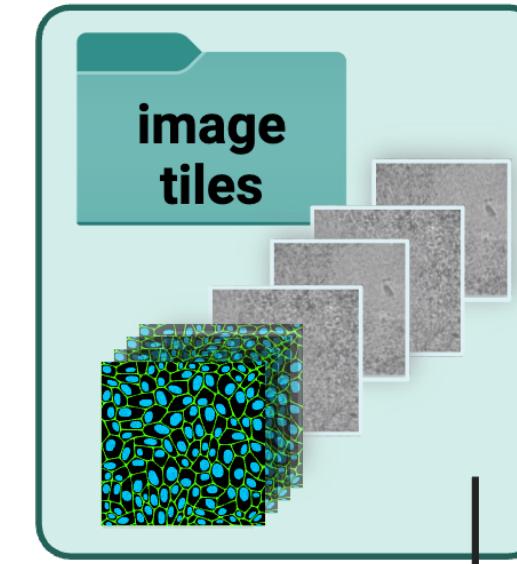
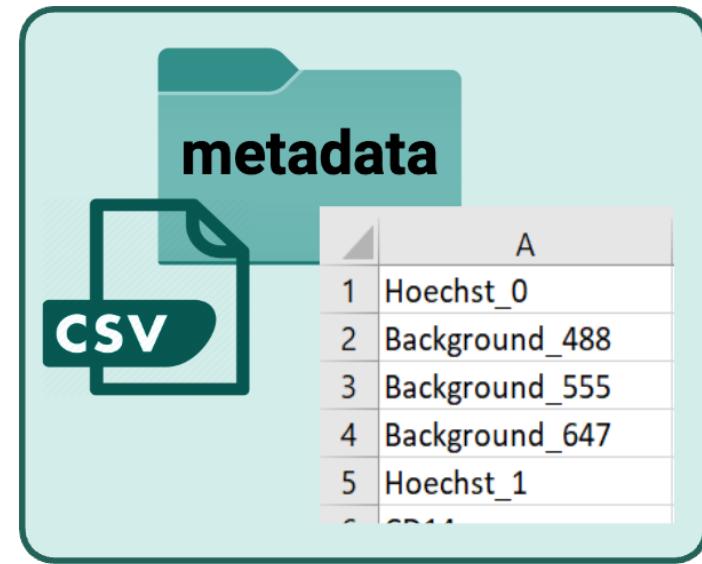
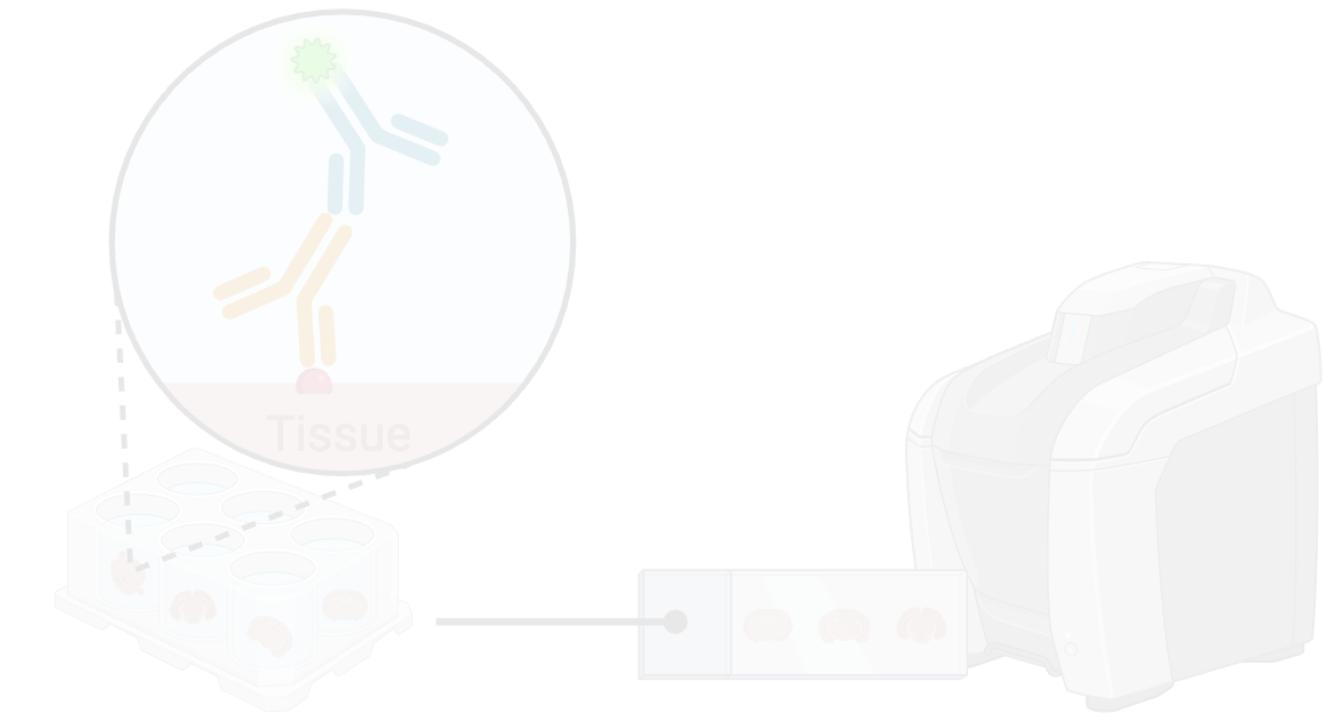


QUALITY CONTROL



QUANTIFICATION

STAINING AND IMAGE ACQUISITION



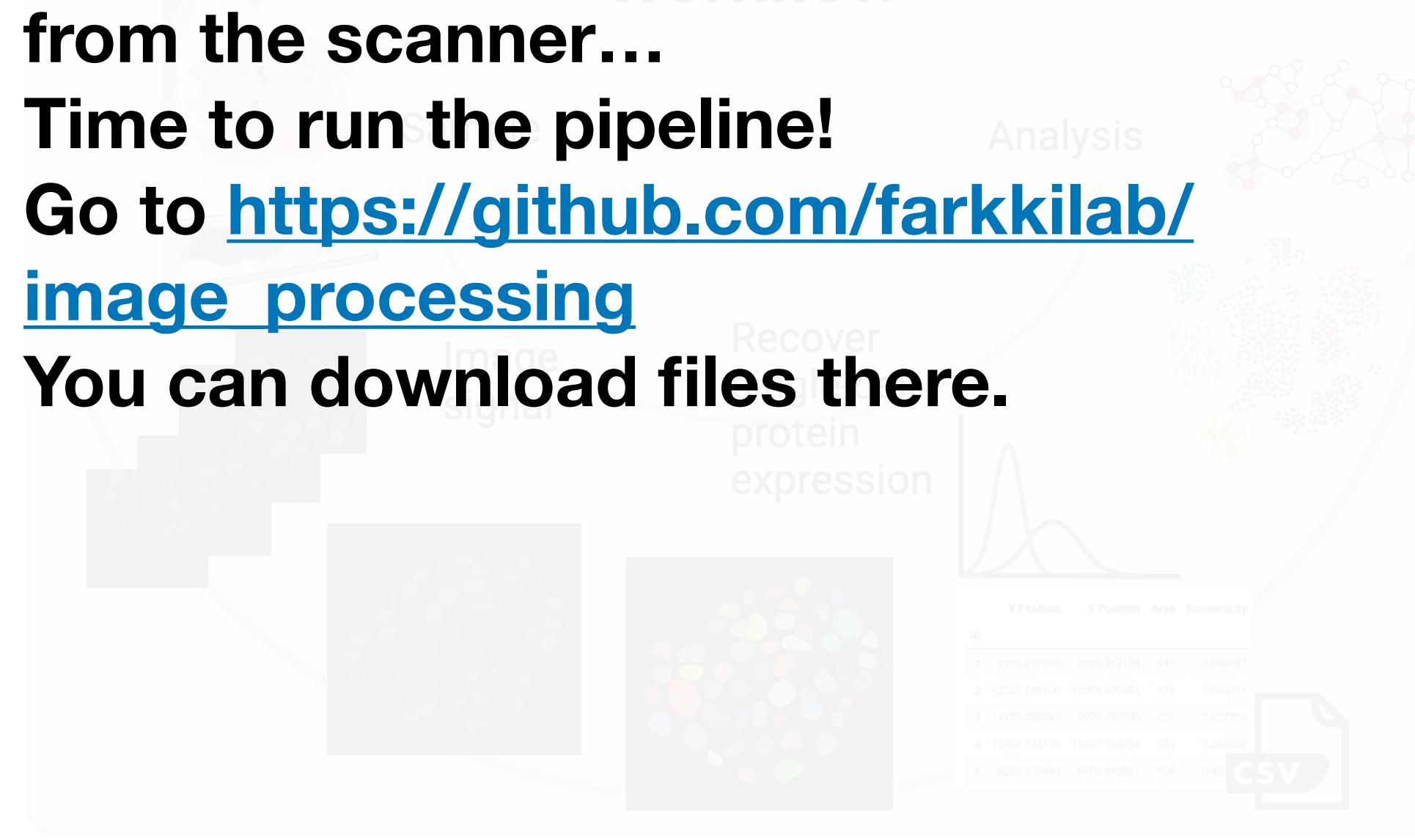
Now you have the raw images from the scanner...

Time to run the pipeline!

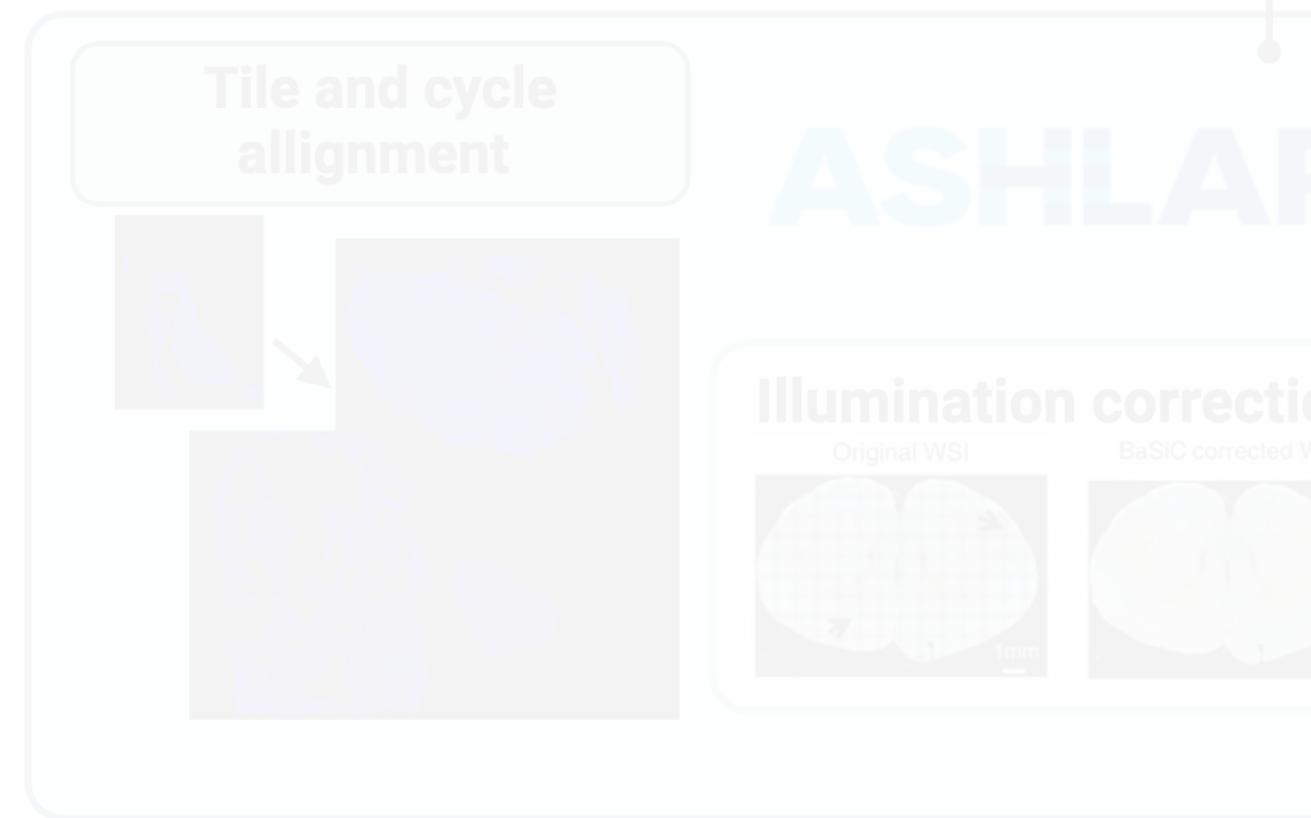
**Go to [https://github.com/farkkilab/
image processing](https://github.com/farkkilab/image_processing)**

You can download files there.

Spatial Proteomics Workflow



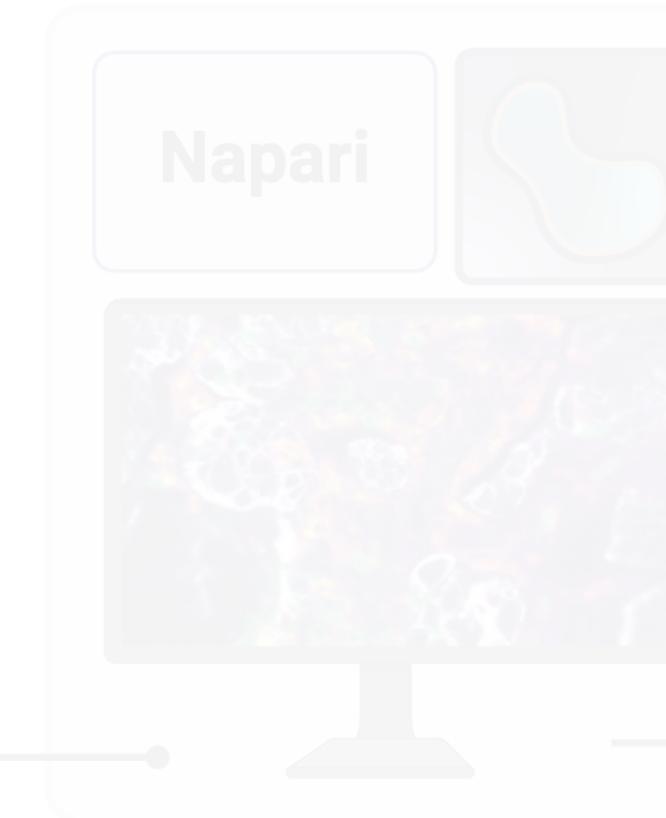
STITCHING



SEGMENTATION



VISUALIZATION



DATA ANALYSIS

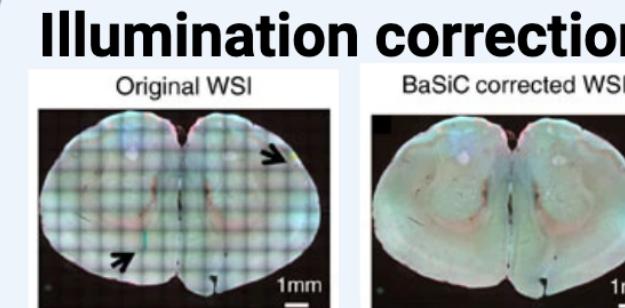


STAINING AND IMAGE ACQUISITION

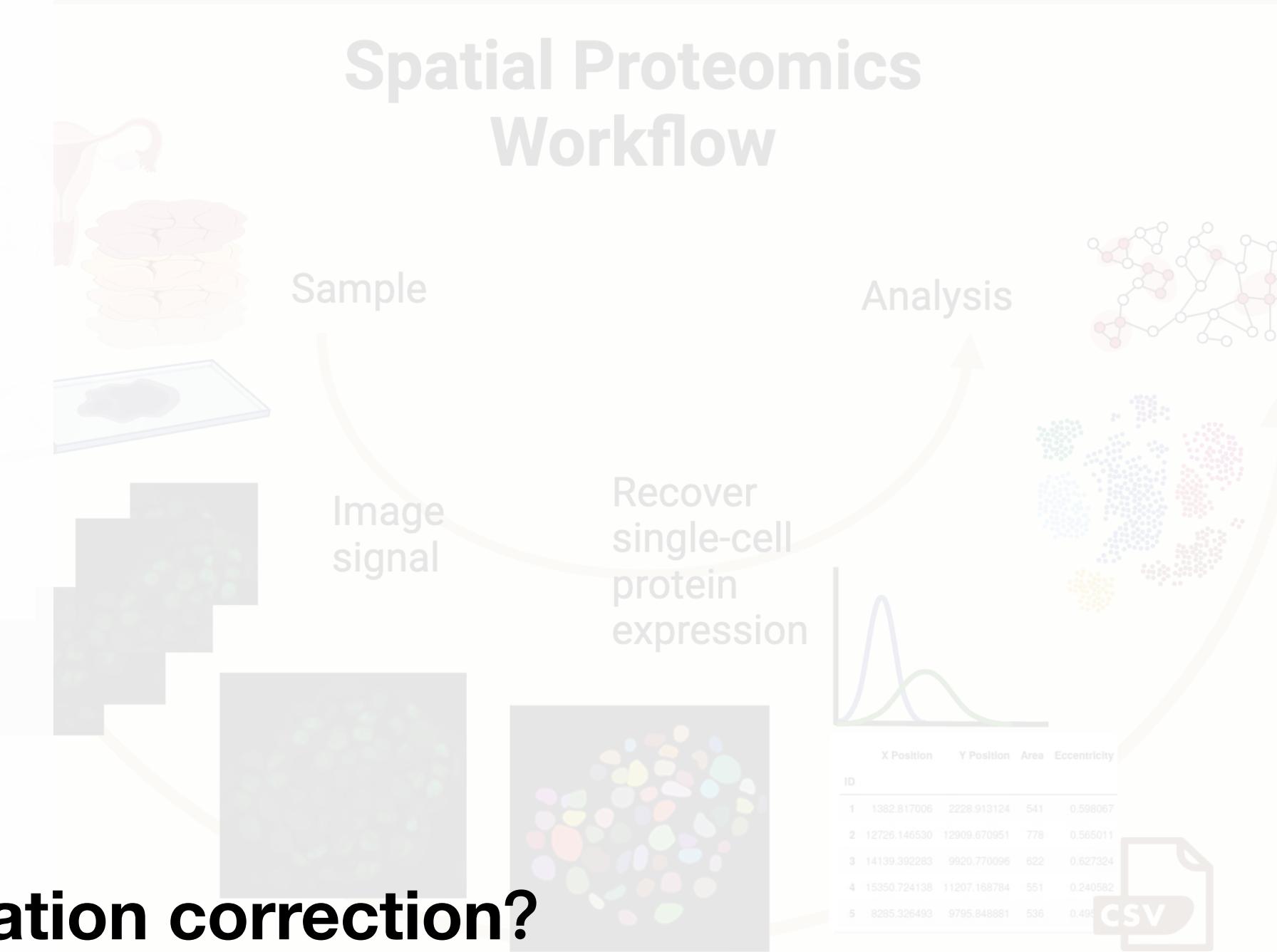


Step 0:
Do I need to perform illumination correction?

- When you do scanning, you can use Aviator to quickly check the image.
- Did you see this? →
- If so, you will need to run.



Spatial Proteomics Workflow



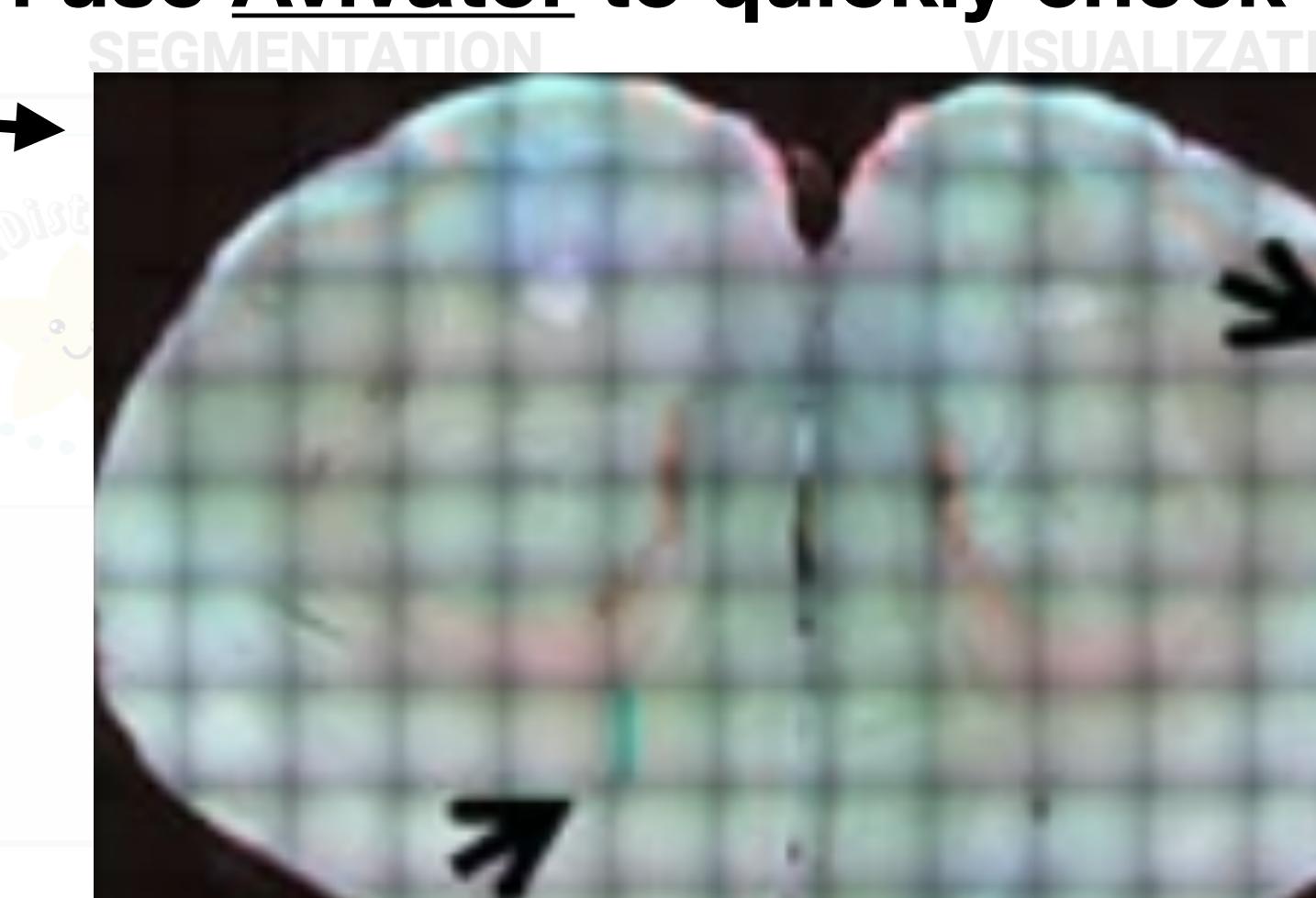
DATA ANALYSIS



TRIBNS

QUALITY CONTROL
CyLinter
IMAGE SEGMENTATION FILTER

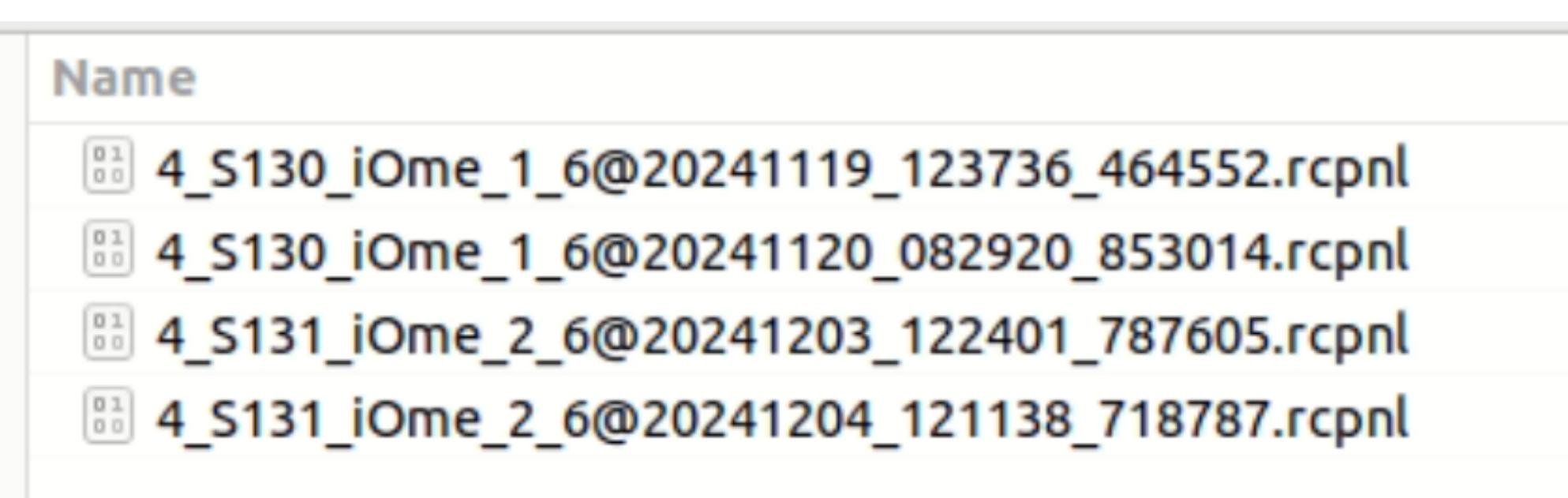
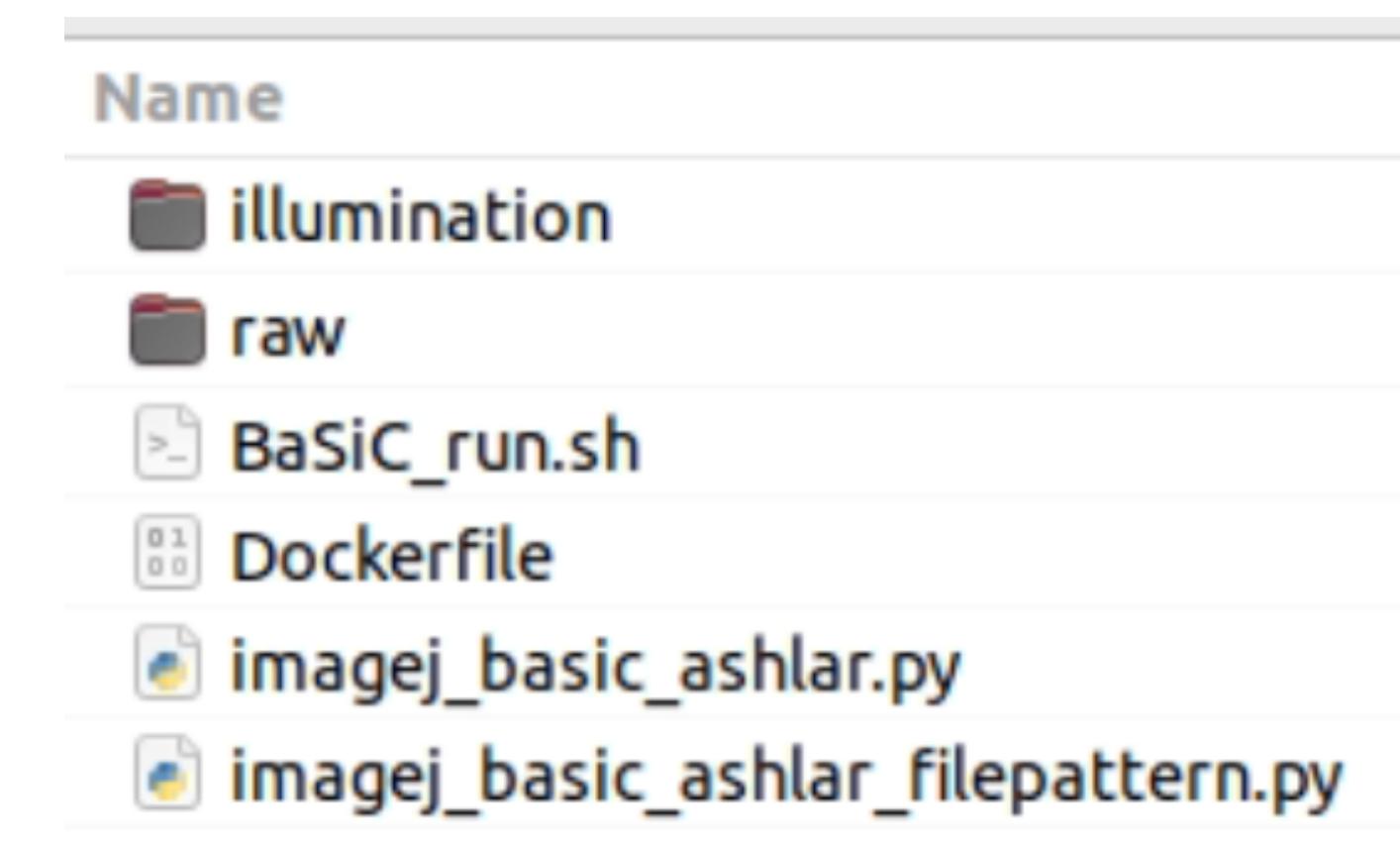
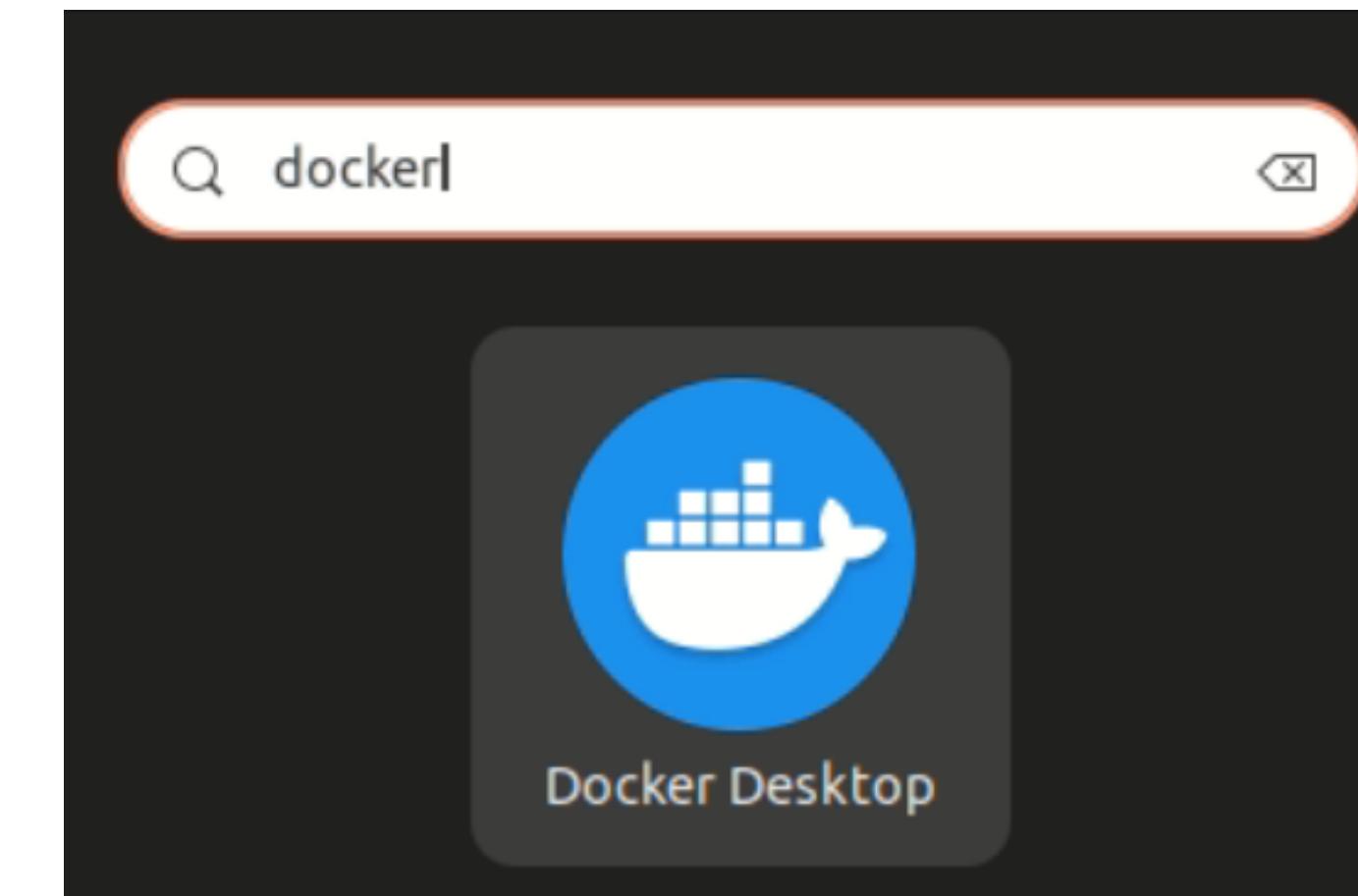
QUANTIFICATION



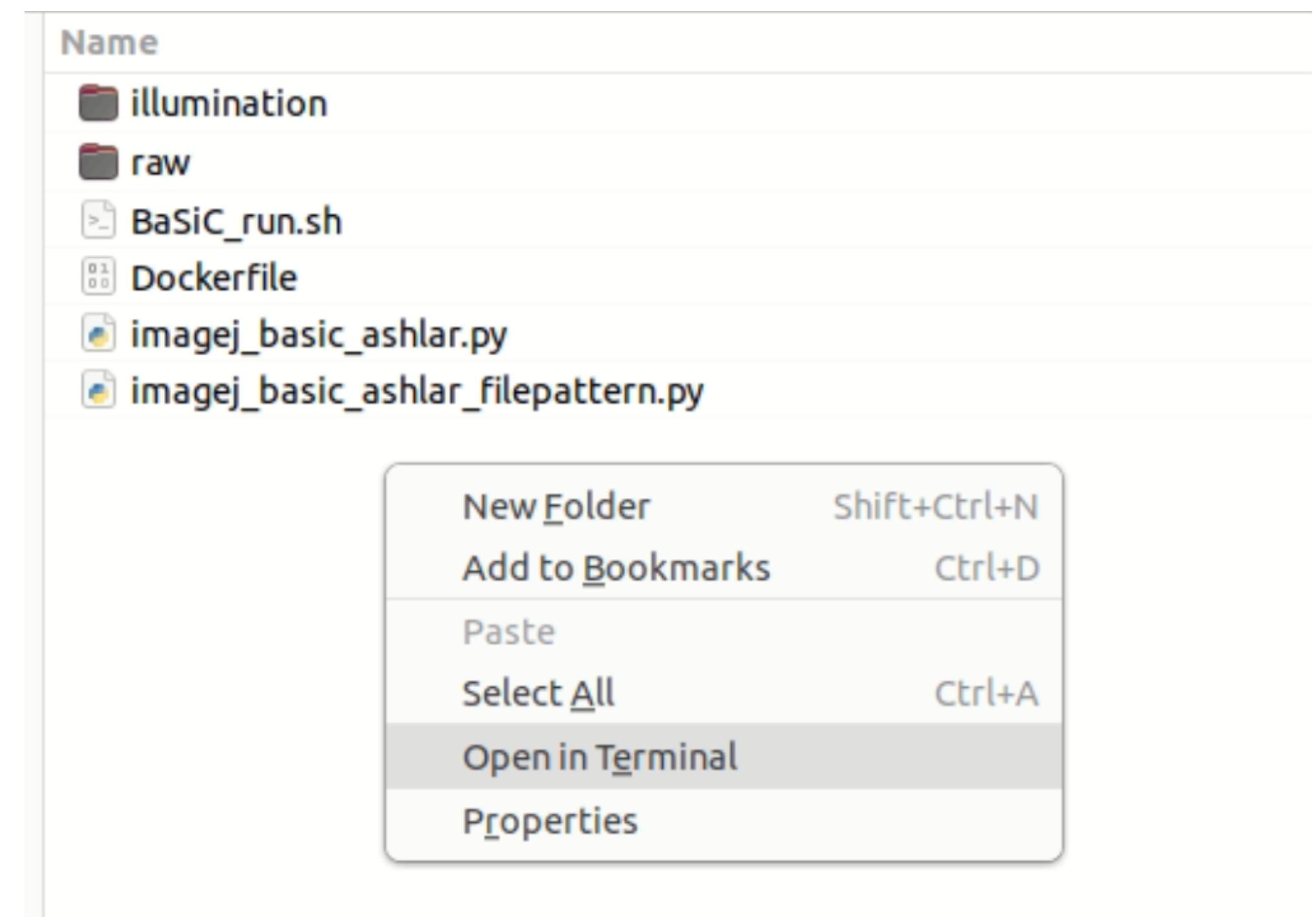
To run illumination correction (with BaSiC)

- It only works on Linux system
- Make sure you have Docker started:

- Your folder should contain those files (you can download from [GitHub](#)):
 - **Make sure your folder is in local now (under /home), not in the remote drive!**
(Docker have no access to the files in the remote drive)
 - Fold **raw** is for input.
 - Folder **illumination** is for output.
- * Notice! All rcpnls should be stored in the raw folder like this:



- Right click in your folder, choose “open in terminal”



Make sure you have full rights to the output folder. Otherwise the output files cannot be saved properly.

Use ls -l to check.

Use chmod 777 <path> to ensure you have full modification rights.

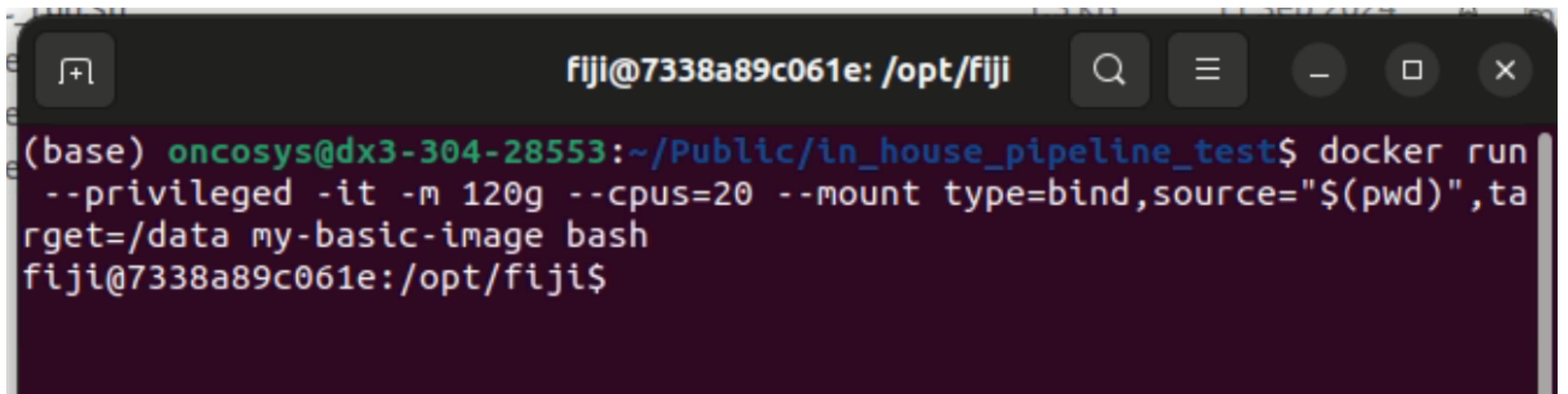
```
(base) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ ls -l
total 36
-rw-rw---- 1 oncosys oncosys 1267 Sep 11 11:35 BaSiC_run.sh
-rw-rw---- 1 oncosys oncosys 682 Sep  2 15:34 Dockerfile
drwxrwx--- 2 oncosys oncosys 4096 Feb 26 17:32 illumination
-rw-rw---- 1 oncosys oncosys 8284 Sep  2 15:24 imagej_basic_ashlar_filepattern.py
-rw-rw---- 1 oncosys oncosys 5774 Sep  2 14:48 imagej_basic_ashlar.py
drwxrwx--- 2 oncosys oncosys 4096 Feb 26 16:35 raw
(base) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ chmod 777 ./illumination/
(base) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ ls -l
total 36
-rw-rw---- 1 oncosys oncosys 1267 Sep 11 11:35 BaSiC_run.sh
-rw-rw---- 1 oncosys oncosys 682 Sep  2 15:34 Dockerfile
drwxrwxrwx 2 oncosys oncosys 4096 Feb 26 17:32 illumination
-rw-rw---- 1 oncosys oncosys 8284 Sep  2 15:24 imagej_basic_ashlar_filepattern.py
-rw-rw---- 1 oncosys oncosys 5774 Sep  2 14:48 imagej_basic_ashlar.py
drwxrwx--- 2 oncosys oncosys 4096 Feb 26 16:35 raw
(base) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ █
```

- Then follow the instructions in [GitHub](#)

```
docker run --privileged -it -m 120g --cpus=20 --  
mount type=bind,source="$(pwd)",target=/data my-  
basic-image bash # start your docker image
```

```
bash <your path> BaSic_run.sh # run bash script to  
process images
```

- Correct if you see this:



A screenshot of a terminal window titled "fiji@7338a89c061e: /opt/fiji". The window has standard macOS-style controls at the top right. The terminal output shows the command being run:

```
(base) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ docker run  
--privileged -it -m 120g --cpus=20 --mount type=bind,source="$(pwd)",ta  
rget=/data my-basic-image bash  
fiji@7338a89c061e:/opt/fiji$
```

```
fiji@7338a89c061e:/opt/fiji$ ls /data
BaSiC_run.sh  imagej_basic_ashlar.py
Dockerfile      imagej_basic_ashlar_filepattern.py
illumination  raw
```

- Next: `fiji@7338a89c061e:/opt/fiji$ bash /data/BaSiC_run.sh`

```
| RCPNLReader initializing /data/raw/4_S130_i0me_1_6@20241119_123736_46455
| 2.rcpnl
| Reading header
| Populating core metadata
| Reading extended header
| Populating original metadata
| Populating OME metadata
| Reading header
| Populating original metadata
| Populating OME metadata

Processing channel 1/4...
=====
Reading header
Populating core metadata
Reading extended header
Populating original metadata
Populating OME metadata
Reading header
Populating original metadata
Populating OME metadata
```

Done!

Illumination correction generates 1 dark field and 1 flat field file for each cycle.

But this is not the end. We need to stitch those field images with our tissue image, in order to correct the background noise.

Time to stitch!

Prepare your folders. You can now copy them to the remote drive (if no space in local):

/ Public / in_hous... ne_test / illumination : Q

Name

S130_iOme
S131_iOme

/ Public / in_... st / illu...on / S130_iOme : Q

Name

4_S130_iOme_1_6@20241119_123736_464552.rcpnl-dfp.tif
4_S130_iOme_1_6@20241119_123736_464552.rcpnl-ffp.tif
4_S130_iOme_1_6@20241120_082920_853014.rcpnl-dfp.tif
4_S130_iOme_1_6@20241120_082920_853014.rcpnl-ffp.tif

Name

4_S130_iOme_1_6@20241119_123736_464552.rcpnl-dfp.tif
4_S130_iOme_1_6@20241119_123736_464552.rcpnl-ffp.tif
4_S130_iOme_1_6@20241120_082920_853014.rcpnl-dfp.tif
4_S130_iOme_1_6@20241120_082920_853014.rcpnl-ffp.tif
4_S131_iOme_2_6@20241203_122401_787605.rcpnl-dfp.tif
4_S131_iOme_2_6@20241203_122401_787605.rcpnl-ffp.tif
4_S131_iOme_2_6@20241204_121138_718787.rcpnl-dfp.tif
4_S131_iOme_2_6@20241204_121138_718787.rcpnl-ffp.tif

Stop Criterion21 0.003768907350381717
Stop Criterion22 0.002457589799443512
Stop Criterion23 0.0016116656386085978
Stop Criterion24 0.0010507817891614545
Stop Criterion25 6.714603978811529E-4
Stop Criterion26 4.277664997004983E-4
Stop Criterion27 2.8130656517026605E-4
Stop Criterion28 1.9413107216401107E-4
Stop Criterion29 1.3820396393885546E-4
Stop Criterion30 9.658375658685047E-5
Stop Criterion31 6.409532822812613E-5
Stop Criterion32 4.03746774562557E-5
Stop Criterion33 2.4390804279297945E-5
Stop Criterion34 1.4242035163147276E-5
Stop Criterion35 8.152289319128856E-6
Stop Criterion36 4.591652231522687E-6
Stop Criterion37 2.5563996720792936E-6
Stop Criterion38 1.4103201270463032E-6
Stop Criterion39 7.743733689793542E-7

Done!
fiji@ad327eafca3b:/opt/fiji\$

/ Public / in_house_pipeline_test / raw : Q

Name

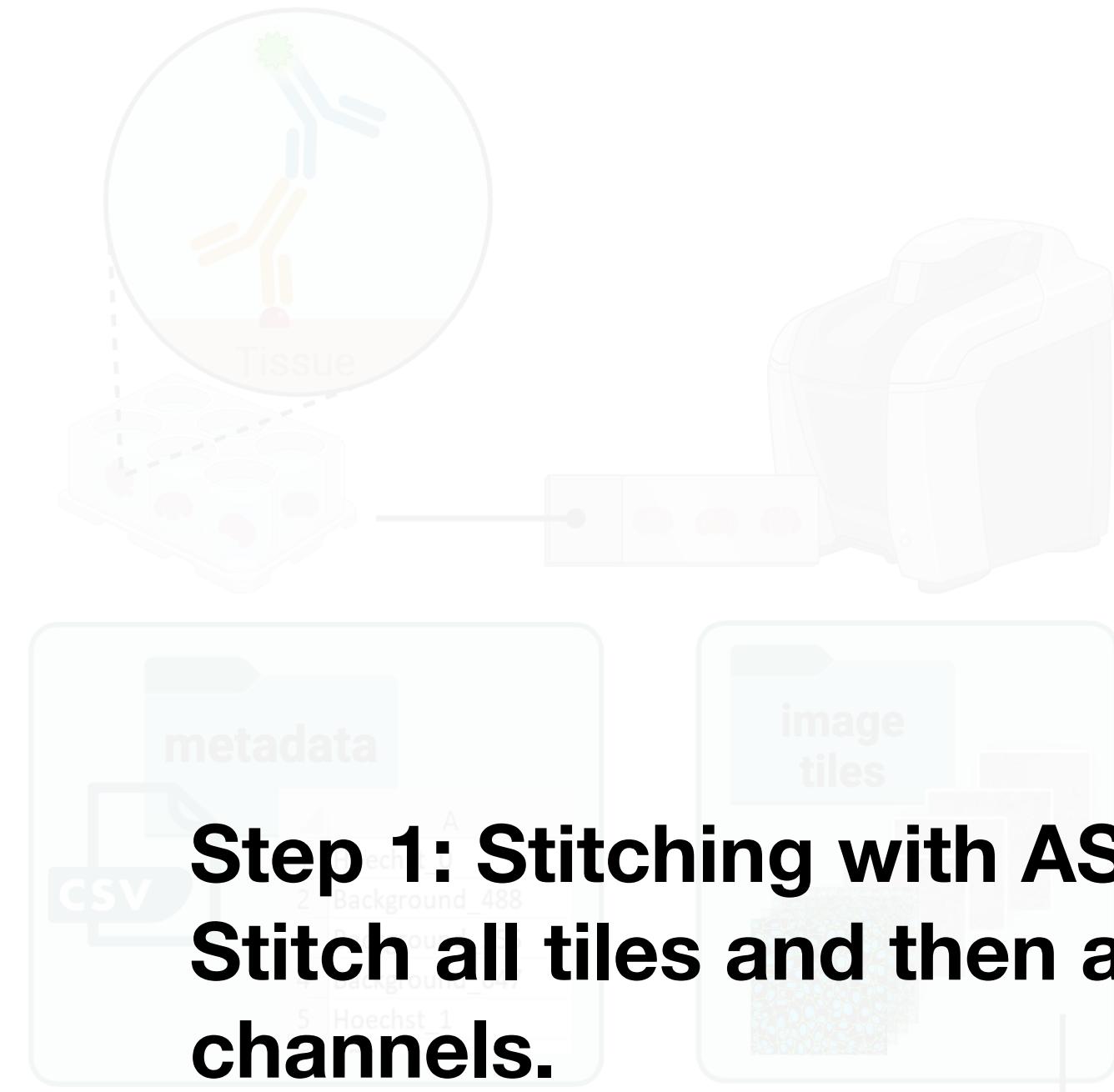
S130_iOme
S131_iOme

/ Public / in_... test / raw / S130_iOme : Q

Name

4_S130_iOme_1_6@20241119_123736_464552.rcpnl
4_S130_iOme_1_6@20241120_082920_853014.rcpnl

STAINING AND IMAGE ACQUISITION



Spatial Proteomics Workflow

Sample

Image signal

Recover single-cell protein expression

Analysis



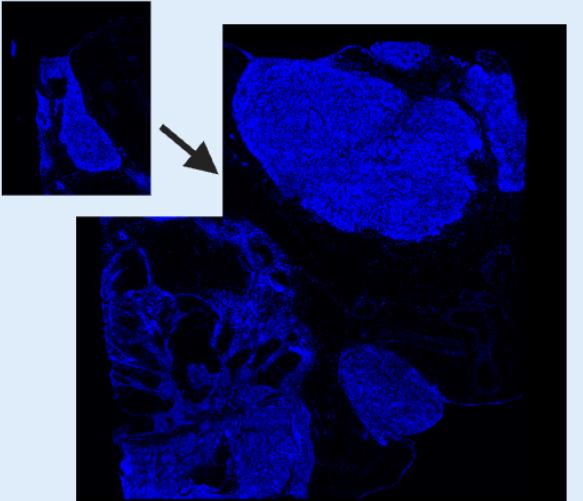
DATA ANALYSIS



STITCHING

The ASHLAR software interface is shown. It features a central panel with the ASHLAR logo and two side panels. The left panel displays 'Tile and cycle alignment' with a small image showing a blue channel and a red arrow indicating alignment. The right panel shows 'Illumination correction' comparing 'Original WSI' and 'BaSiC corrected WSI' of a tissue sample. A 3D reconstruction of the tissue is visible at the bottom.

Tile and cycle alignment



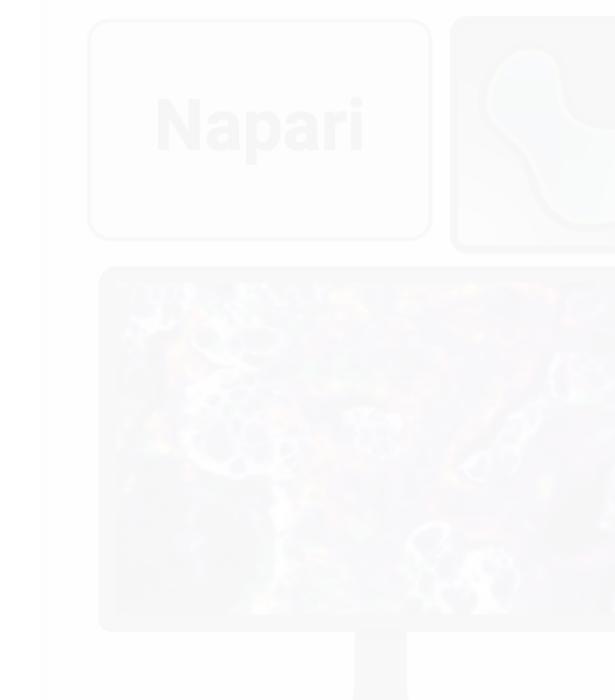
ASHLAR

Illumination correction
Original WSI BaSiC corrected WSI
1mm 1mm

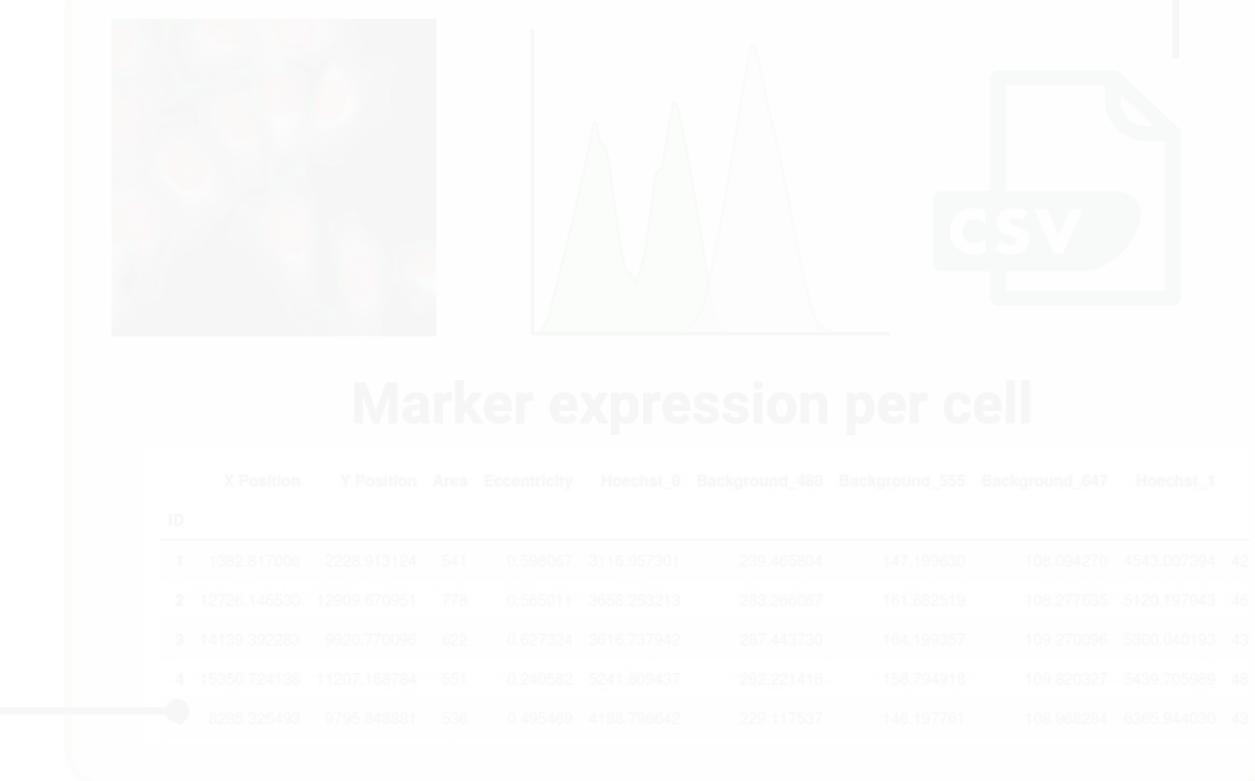
SEGMENTATION



VISUALIZATION



QUANTIFICATION



Activate the conda environment for ashlar.

```
(base) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ conda activate ashlar  
(ashlar) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$
```

Download ashlar_workflow.py from GitHub.

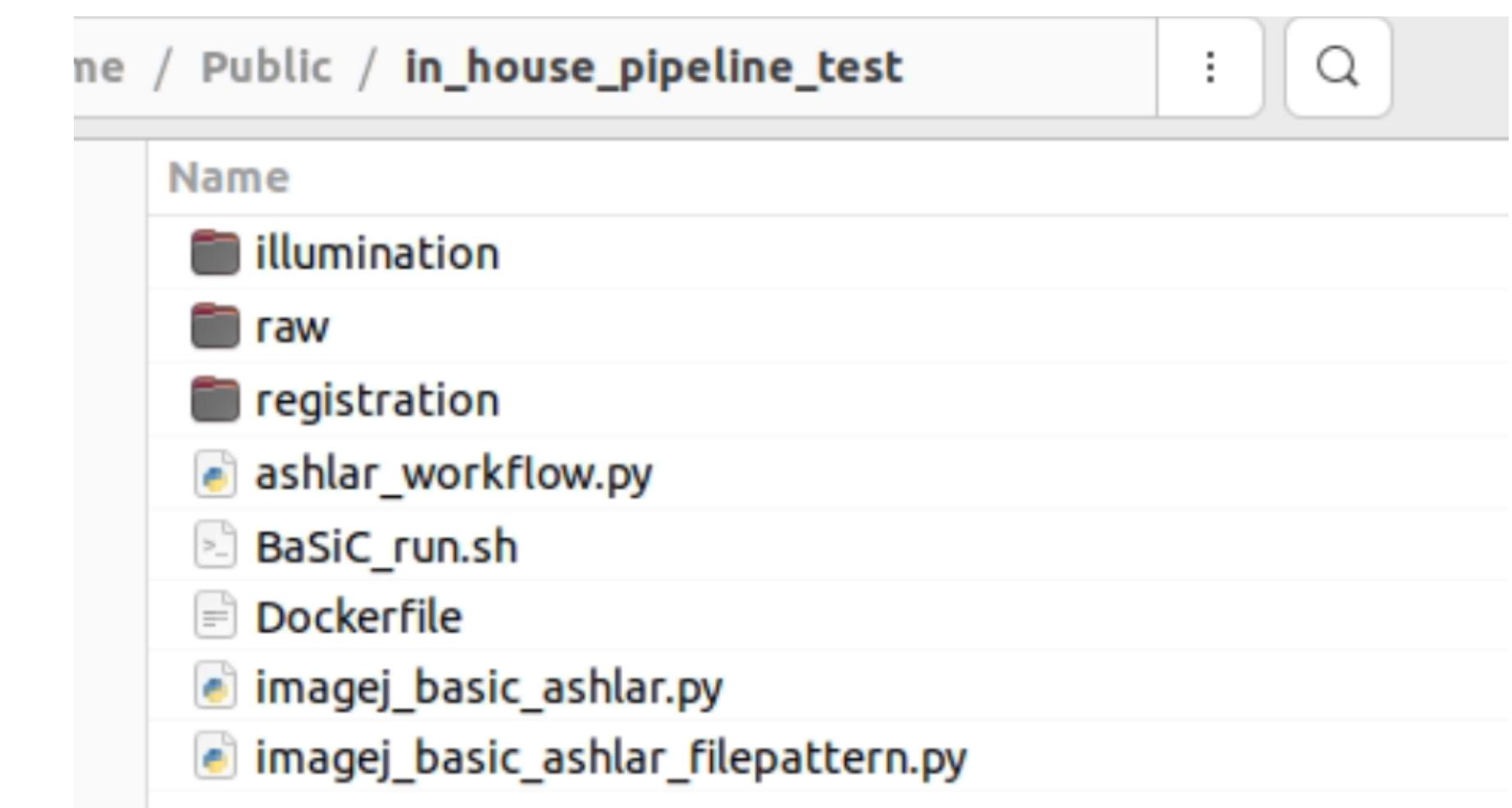
Open it and change: my_path (input), output_path, illumination (Y or N), if Y then illumination_folder.

```
!6 my_path = "./raw"  
!7 output_path = "./registration"  
!8 subfolders = [ f.path for f in os.scandir(my_path) if f.is_dir() ]  
!9 file_type = 'rcpnl' # 'nd2' 'rcpnl'  
!  
!1 illumination = 'Y' # 'N'  
!2 illumination_folder = "./illumination/"
```

Create a folder to store the output.

Here I named it “registration”.

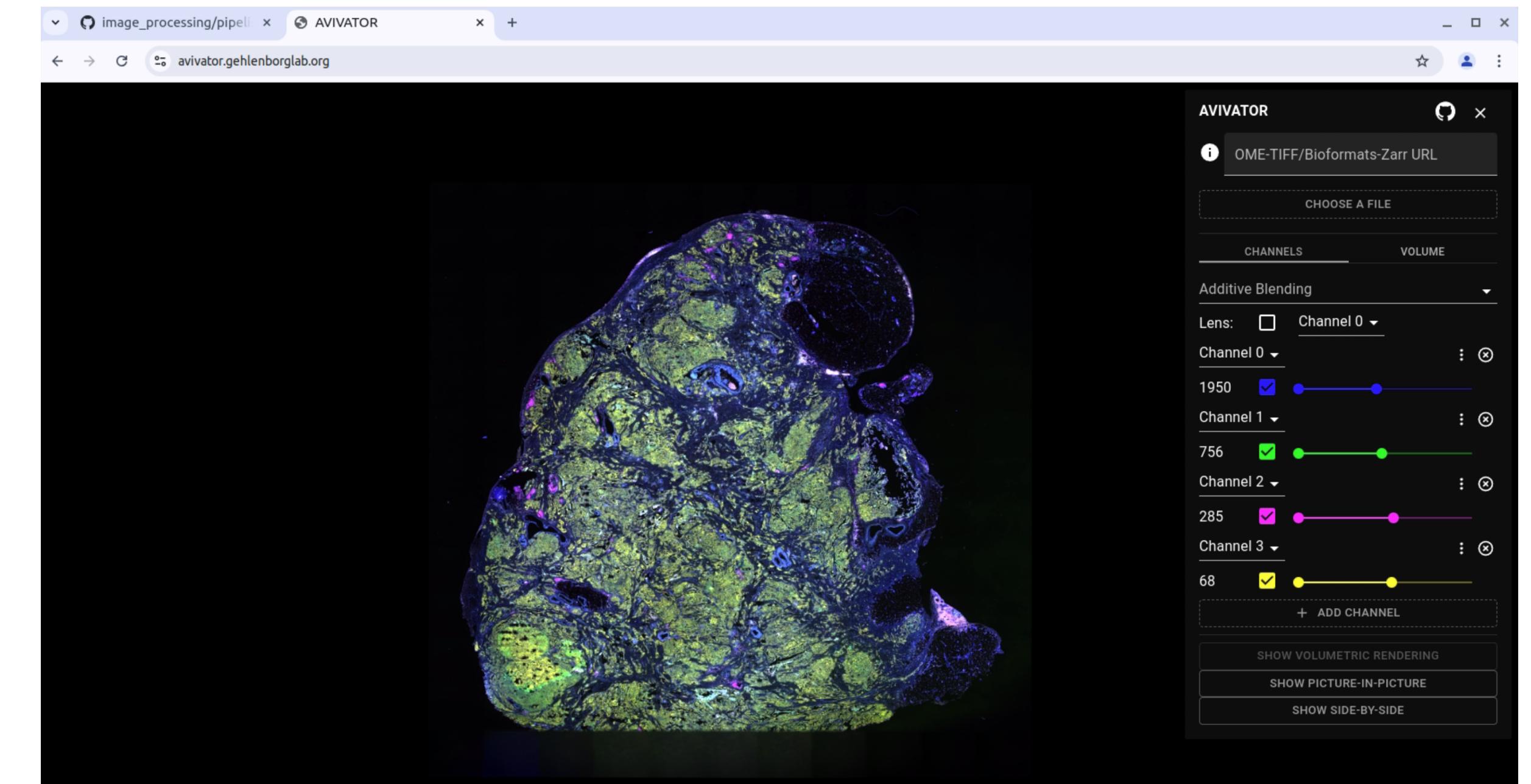
Your folder should be like this before run ASHLAR:



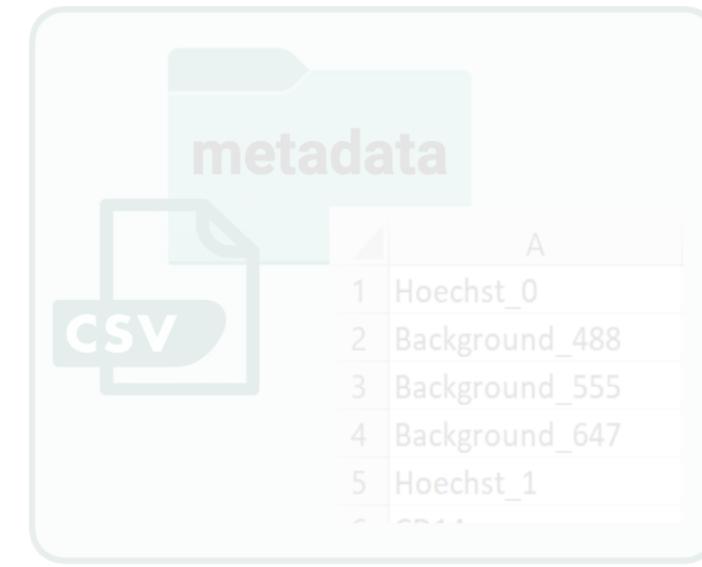
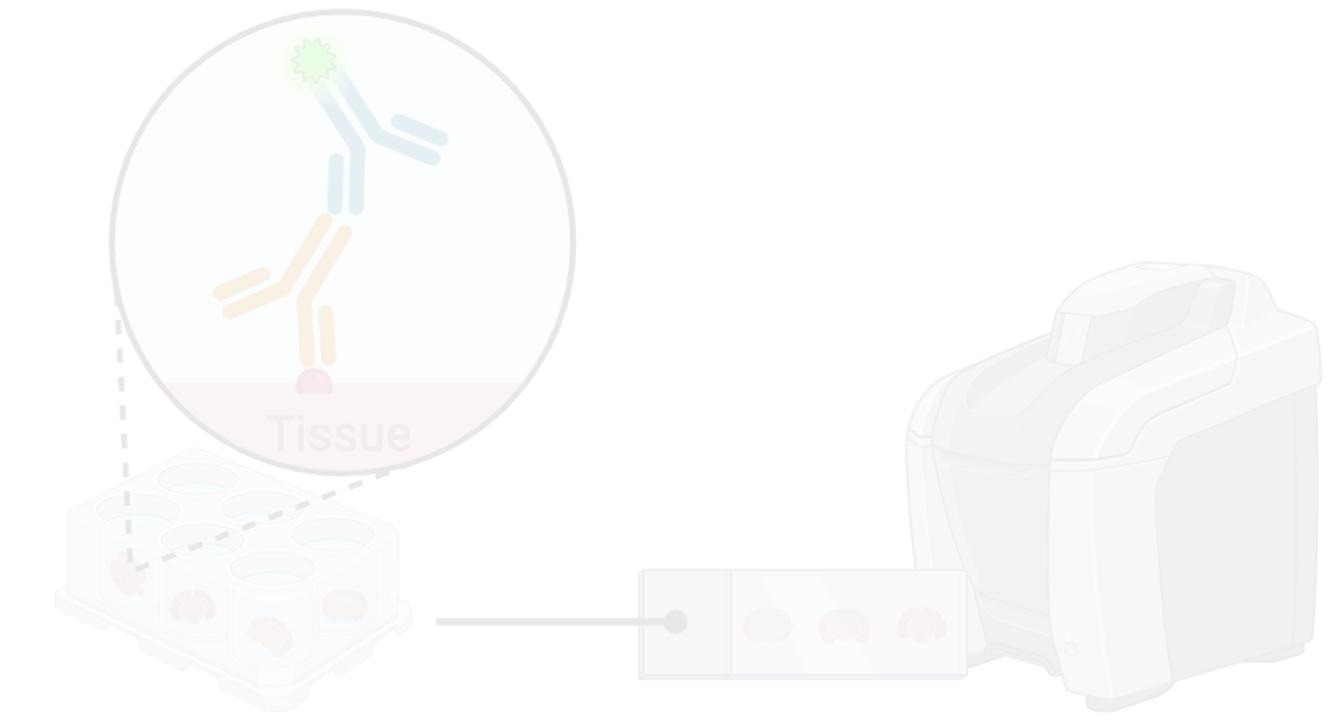
Run:
You can change the
parameter -c to start
parallelised
computation.

```
(ashlar) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ python ashlar_wo
rkflow.py -c 1
Subfolders are: ['./raw/S131_iOme', './raw/S130_iOme']
Files to stitch: ./raw/S131_iOme/4_S131_iOme_2_6@20241203_122401_787605.rcpnl ./
raw/S131_iOme/4_S131_iOme_2_6@20241204_121138_718787.rcpnl
Flat field files: ./illumination/S131_iOme/4_S131_iOme_2_6@20241203_122401_78760
5.rcpnl-ffp.tif ./illumination/S131_iOme/4_S131_iOme_2_6@20241204_121138_718787.
rcpnl-ffp.tif
Dark field files: ./illumination/S131_iOme/4_S131_iOme_2_6@20241203_122401_78760
5.rcpnl-dfp.tif ./illumination/S131_iOme/4_S131_iOme_2_6@20241204_121138_718787.
rcpnl-dfp.tif
ashlar ./raw/S131_iOme/4_S131_iOme_2_6@20241203_122401_787605.rcpnl ./raw/S131_i
Ome/4_S131_iOme_2_6@20241204_121138_718787.rcpnl -o"./registration/S131_iOme.ome
.tif" --pyramid --filter-sigma 1 -m 30 --ffp ./illumination/S131_iOme/4_S131_iOme_
2_6@20241203_122401_787605.rcpnl-ffp.tif ./illumination/S131_iOme/4_S131_iOme_
2_6@20241204_121138_718787.rcpnl-ffp.tif --dfp ./illumination/S131_iOme/4_S131_i
Ome_2_6@20241203_122401_787605.rcpnl-dfp.tif ./illumination/S131_iOme/4_S131_iOme_
2_6@20241204_121138_718787.rcpnl-dfp.tif
Stitching and registering input images
Cycle 0:
    reading ./raw/S131_iOme/4_S131_iOme_2_6@20241203_122401_787605.rcpnl
WARNING: Stage coordinates' measurement unit is undefined; assuming μm.
assembling thumbnail 143/143
```

Done! Use Aviator to quickly
check the image quality:



STAINING AND IMAGE ACQUISITION



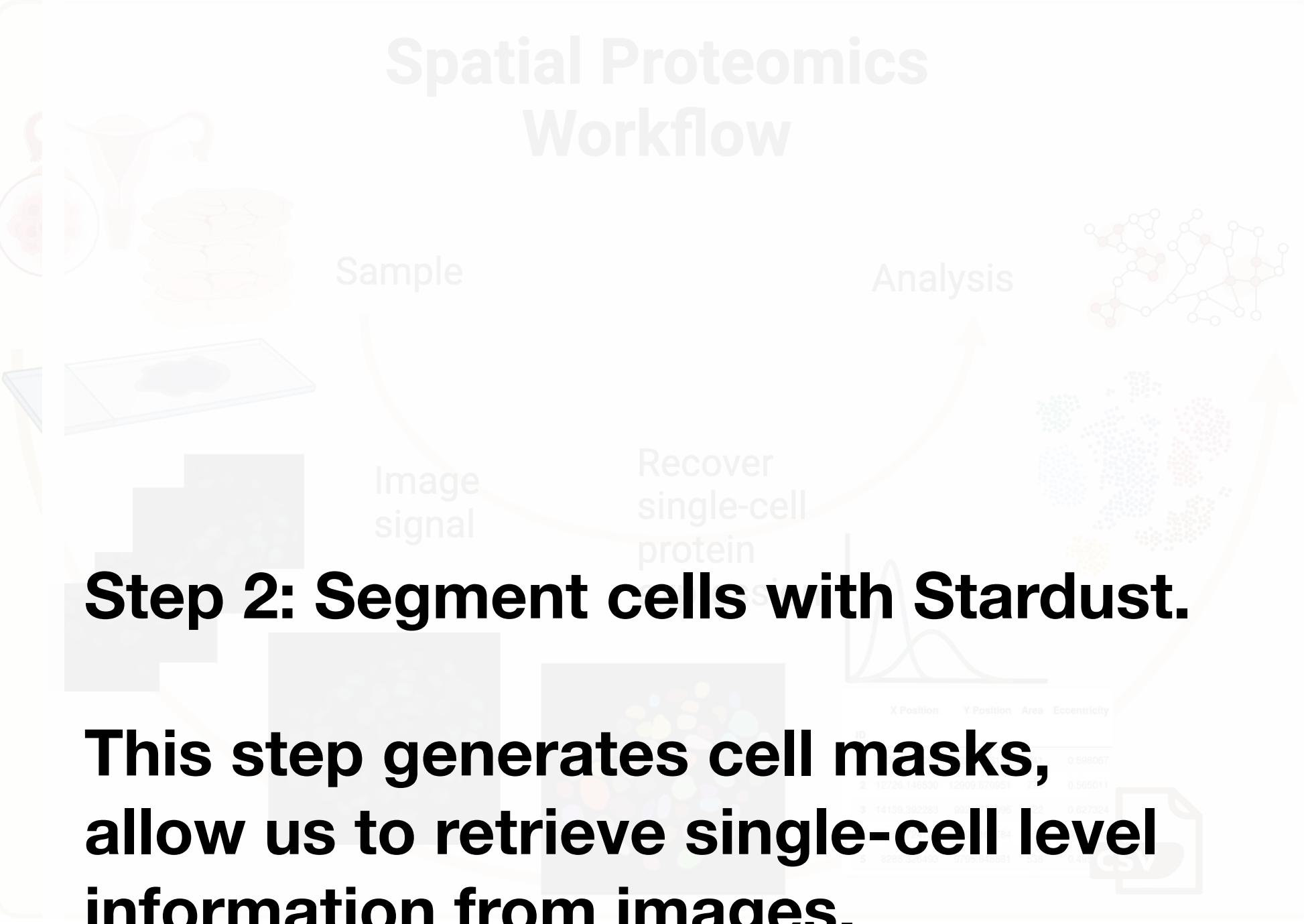
STITCHING

ASHLAR

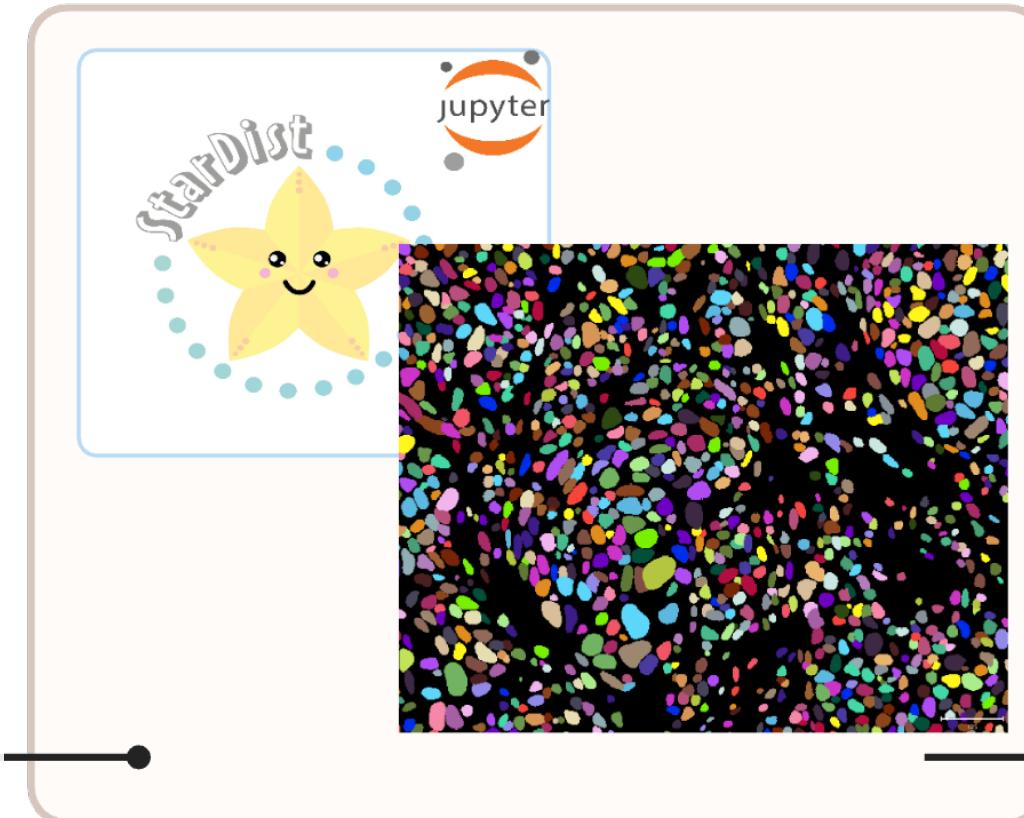
Tile and cycle alignment



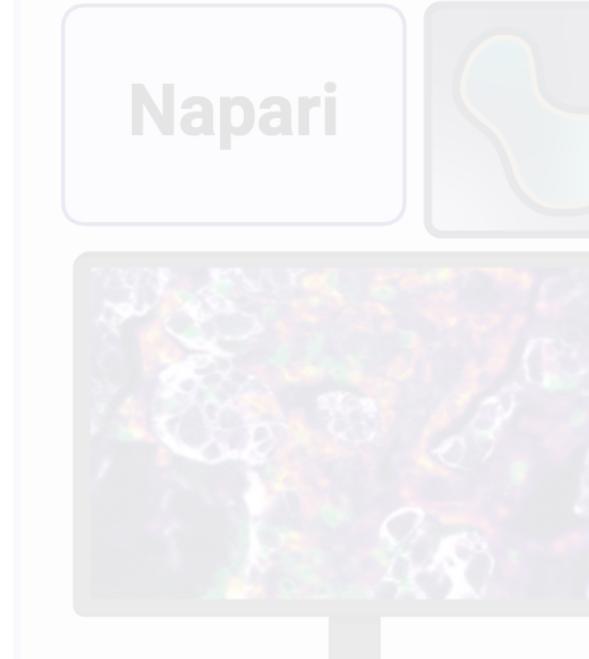
Illumination correction



SEGMENTATION



VISUALIZATION



DATA ANALYSIS



TRIBNS



QUANTIFICATION



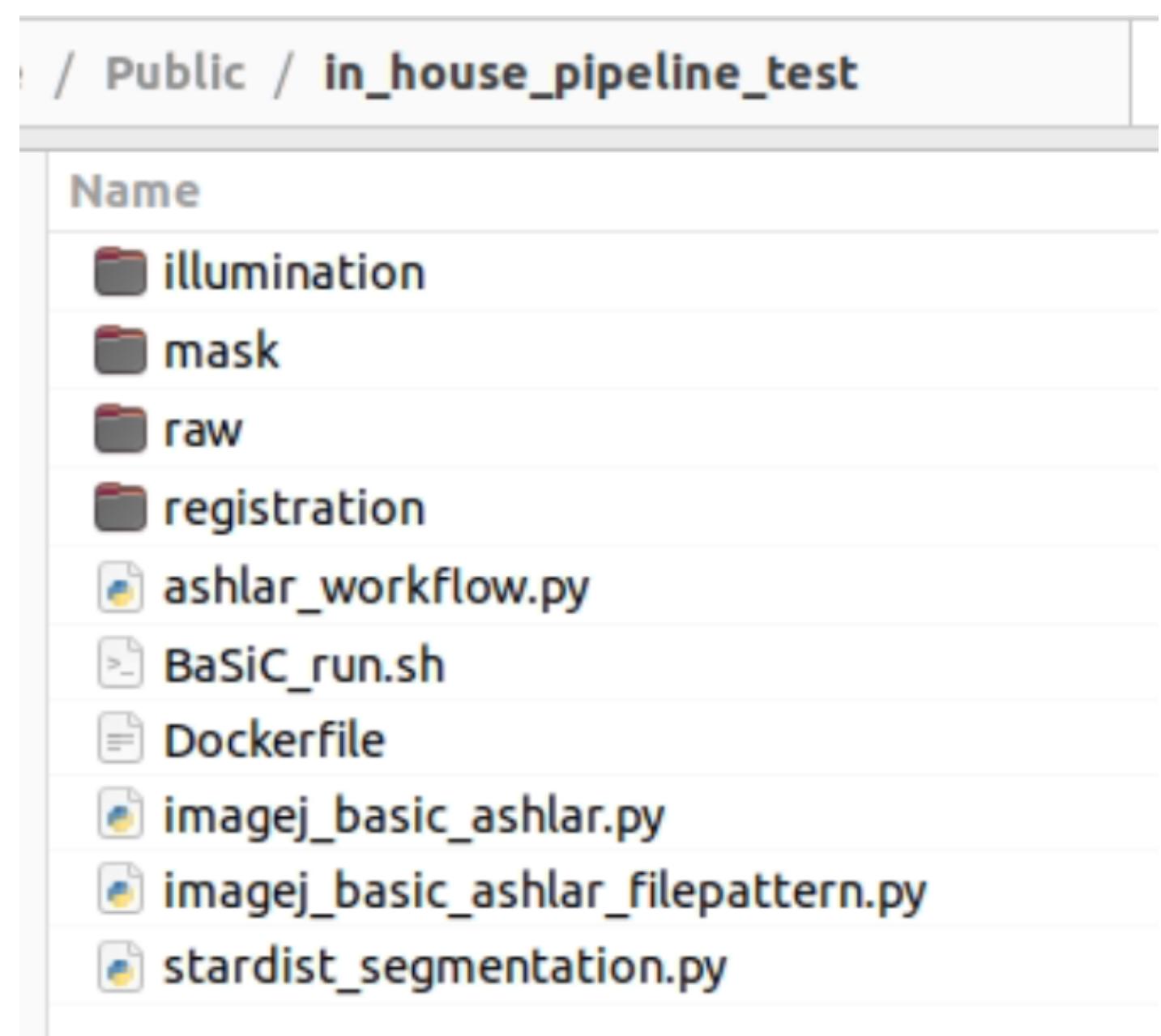
Download stardist_segmentation.py

Create new folder named mask.

Modify the input and output path in the
stardist_segmentation.py

```
30 # input path must only have the images to segment
31 # output path should be an empty folder
32 INPUT_PATH = "./registration"
33 OUTPUT_PATH = "./mask/" # remember to add / in the end
```

Activate python environment, run the script.



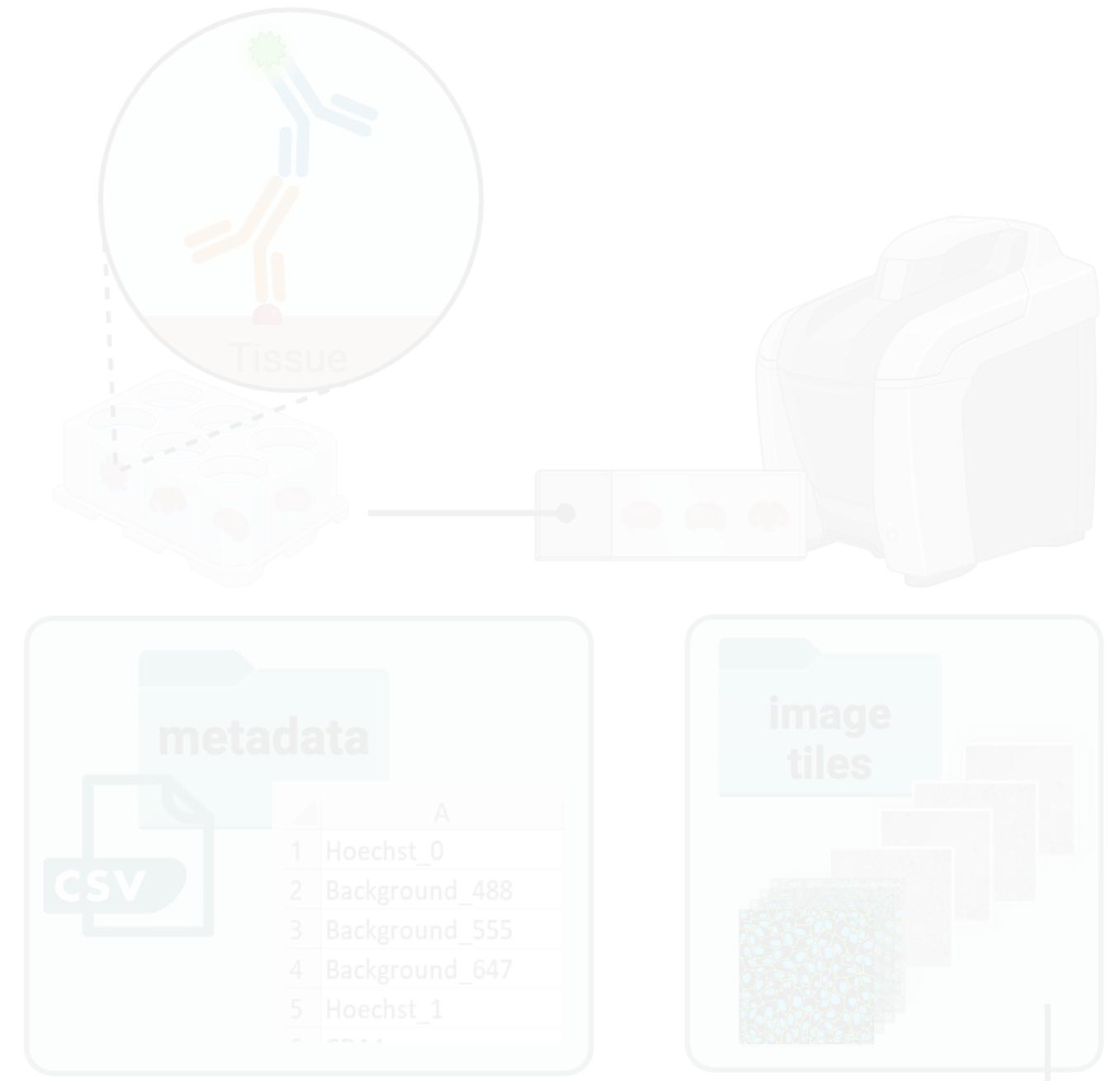
```
(base) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ conda activate stardist
(stardist) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ python stardist_segmentation.py
```

```
[165]
Loading thresholds from 'thresholds.json'.
Using default values: prob_thresh=0.479071, nms_thresh=0.3.
Image list: ['S130_i0me.ome.tif', 'S131_i0me.ome.tif']
Do you want to proceed? [y/n]y
```

It will ask you again when all images are finished. Press N to stop.

```
Image has a shape of (26226, 26624)
Calculate tiles = 12
100%|██████████| 144/144 [02:45<00:00,  1.15s/it]
Finish image ./registration/S131_i0me.ome.tif
Do you want to proceed? [y/n]n
```

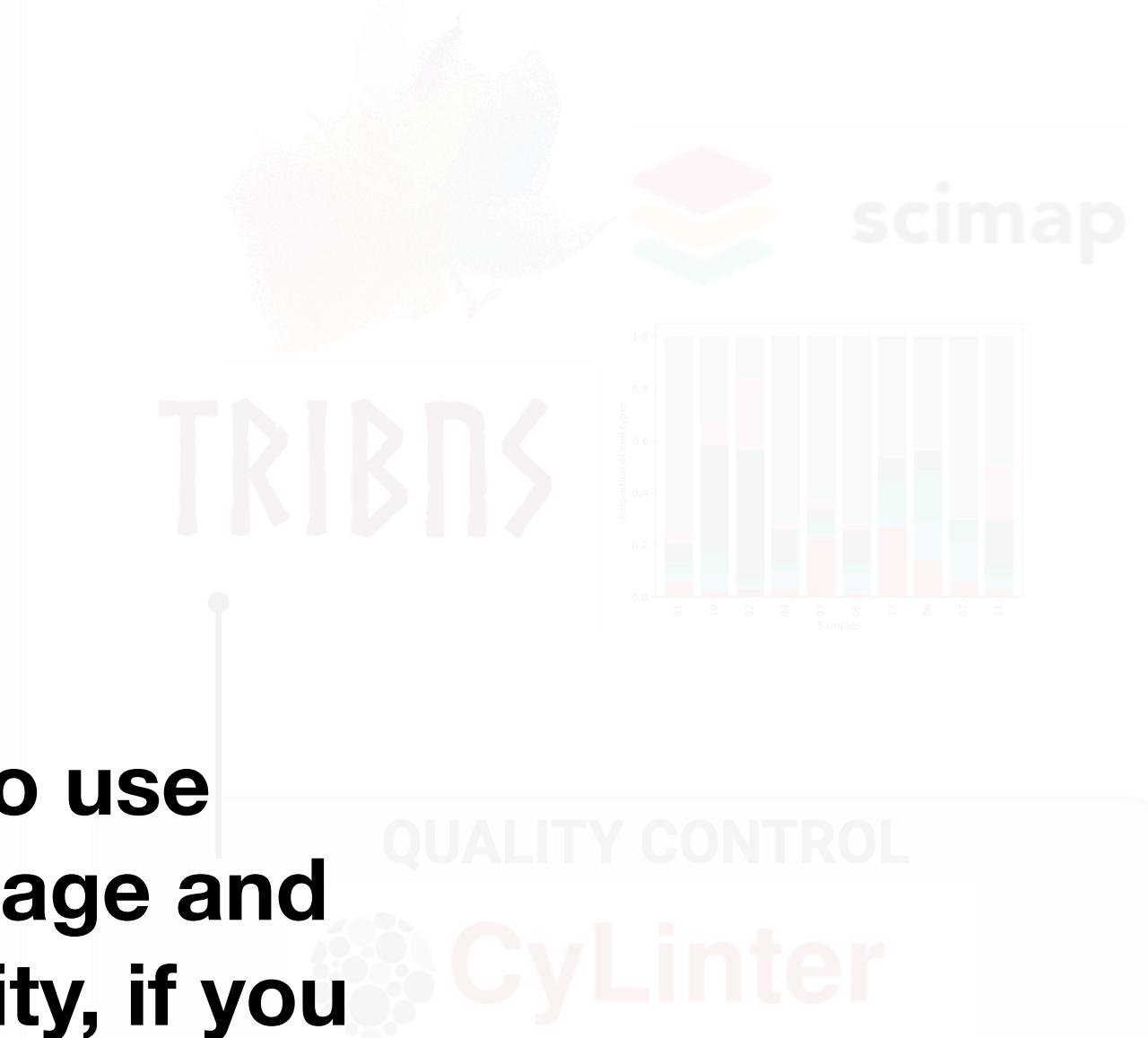
STAINING AND IMAGE ACQUISITION



Spatial Proteomics Workflow

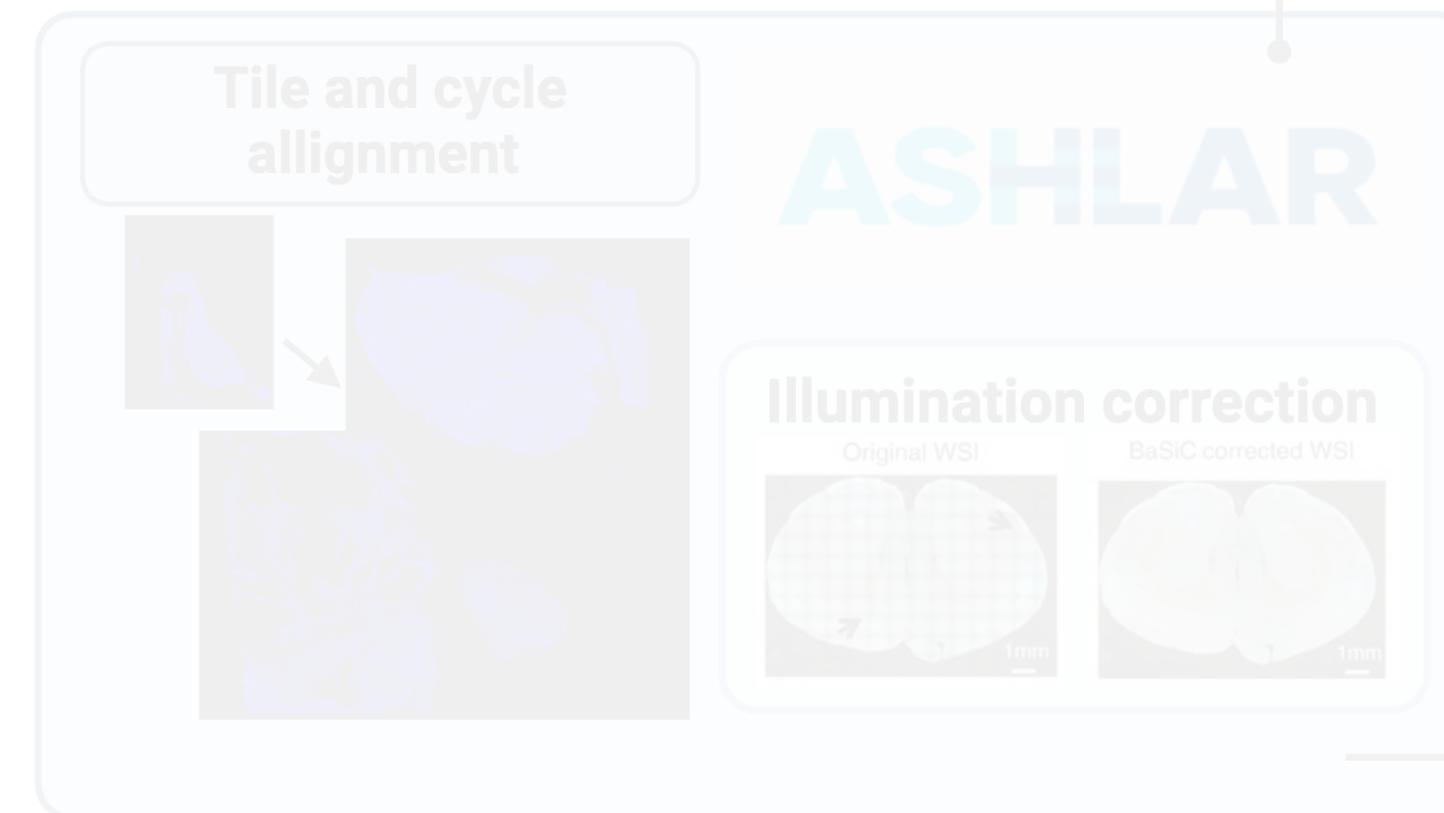


DATA ANALYSIS



You can already start to use Napari to check the image and the segmentation quality, if you wish.

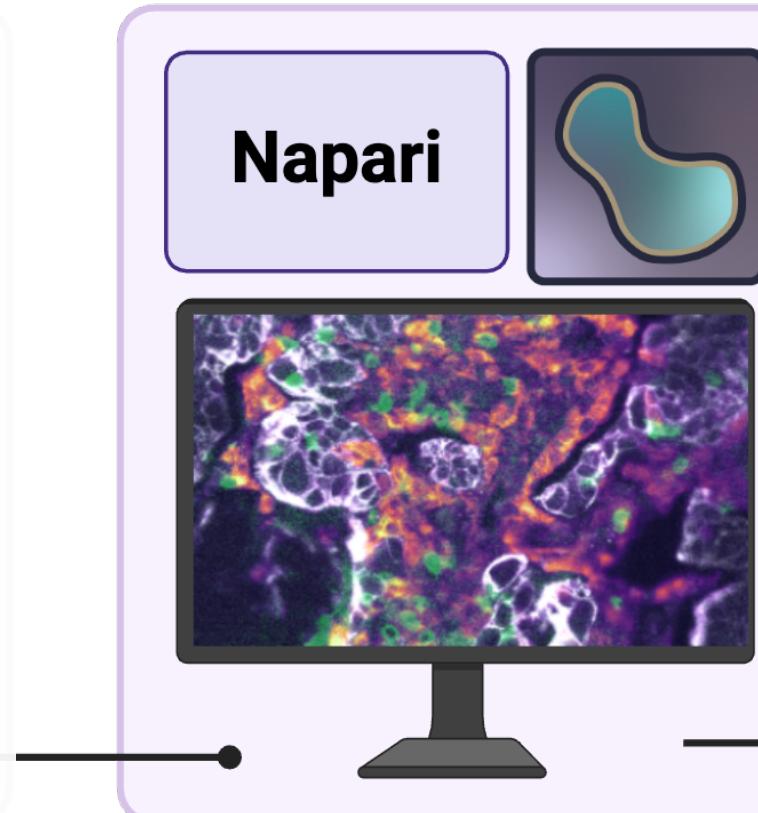
STITCHING



SEGMENTATION



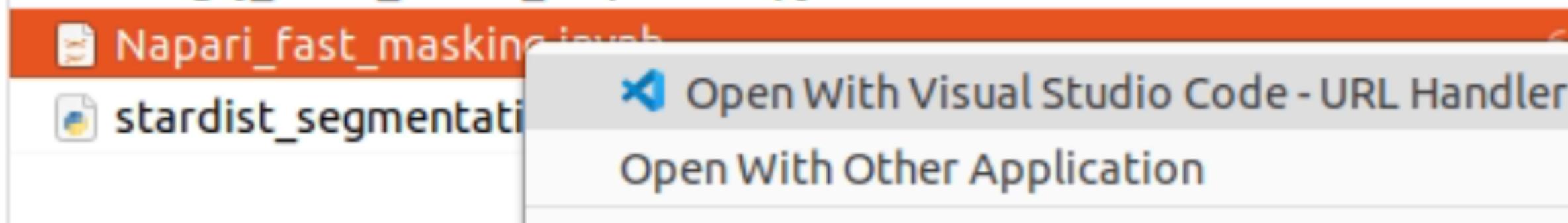
VISUALIZATION



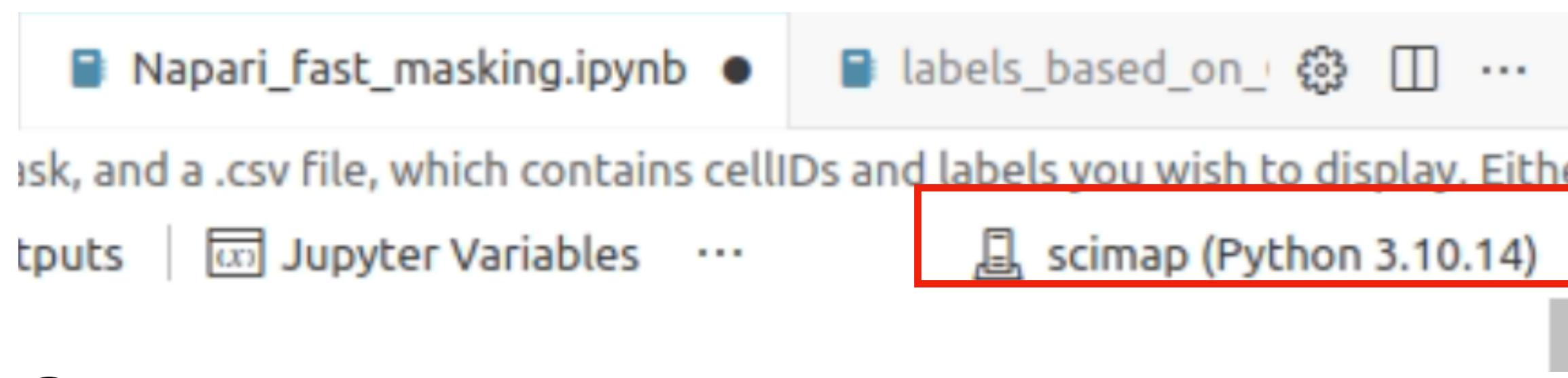
QUANTIFICATION



Download Napari_fast_masking.ipynb., right click and open with VScode.



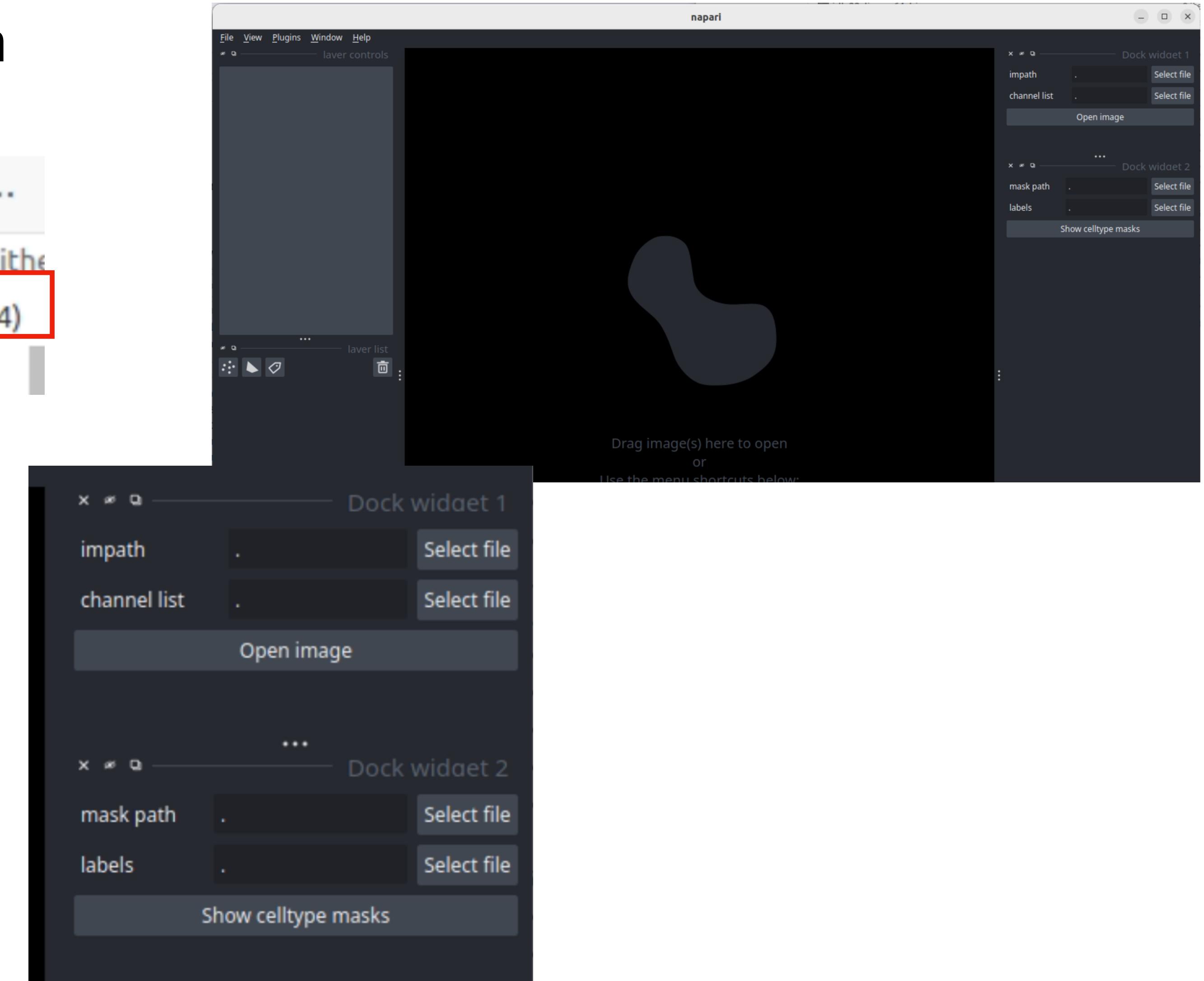
Choose the conda environment here in the vscode.



Click and run every chunk in the notebook.

A Napari window will open:

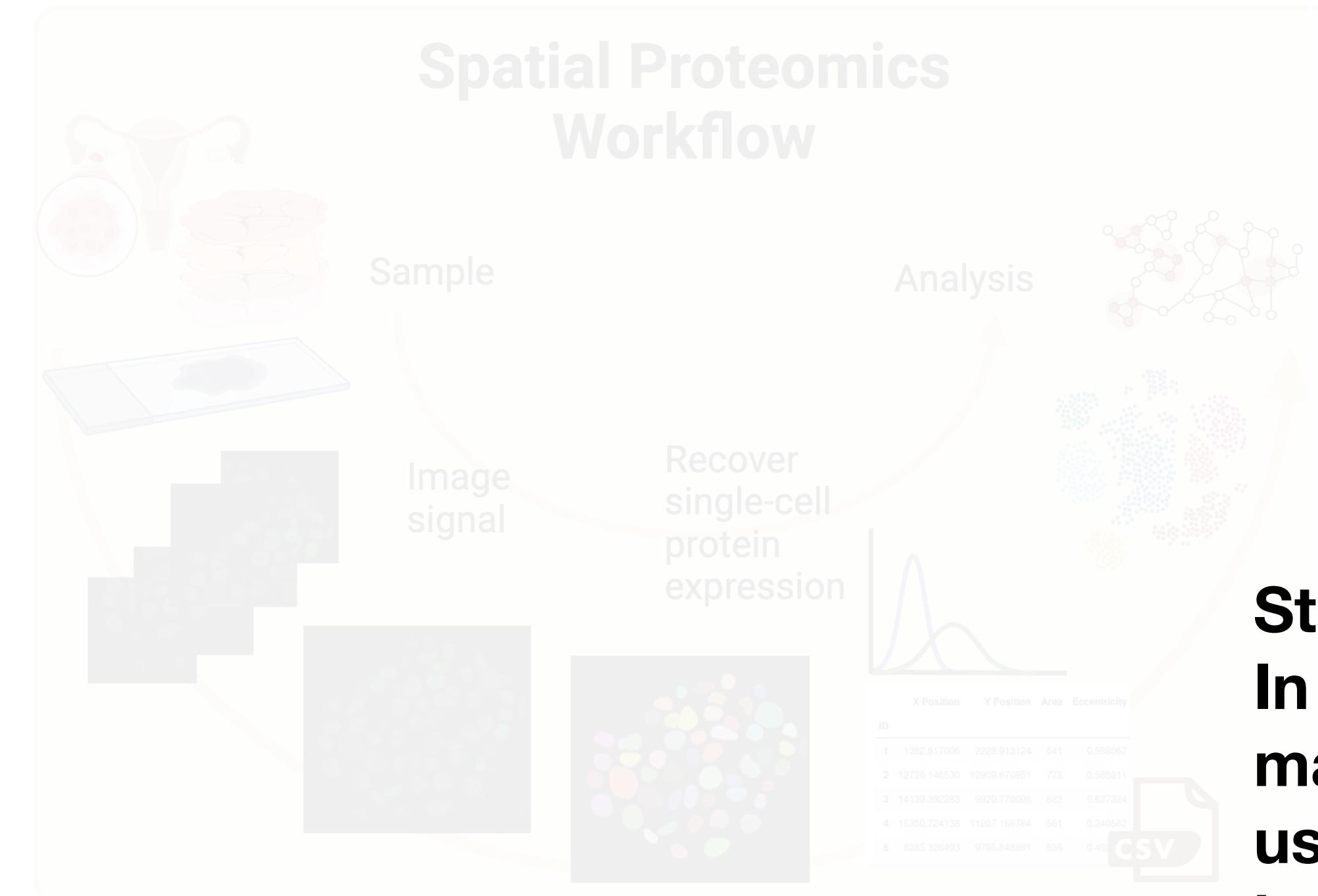
You can then select your tiff image, channel list, mask and cell type labels.



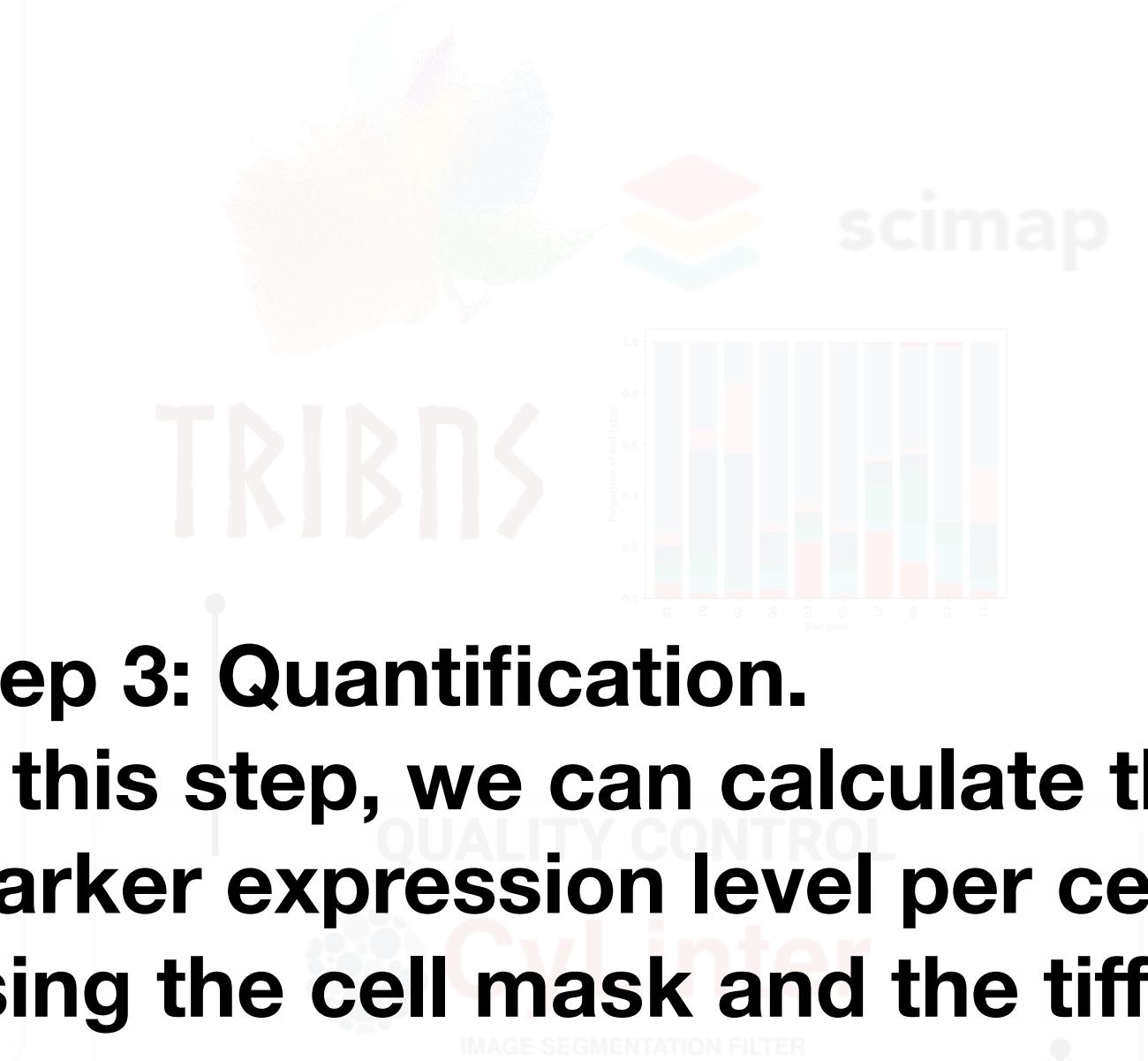
STAINING AND IMAGE ACQUISITION



Spatial Proteomics Workflow



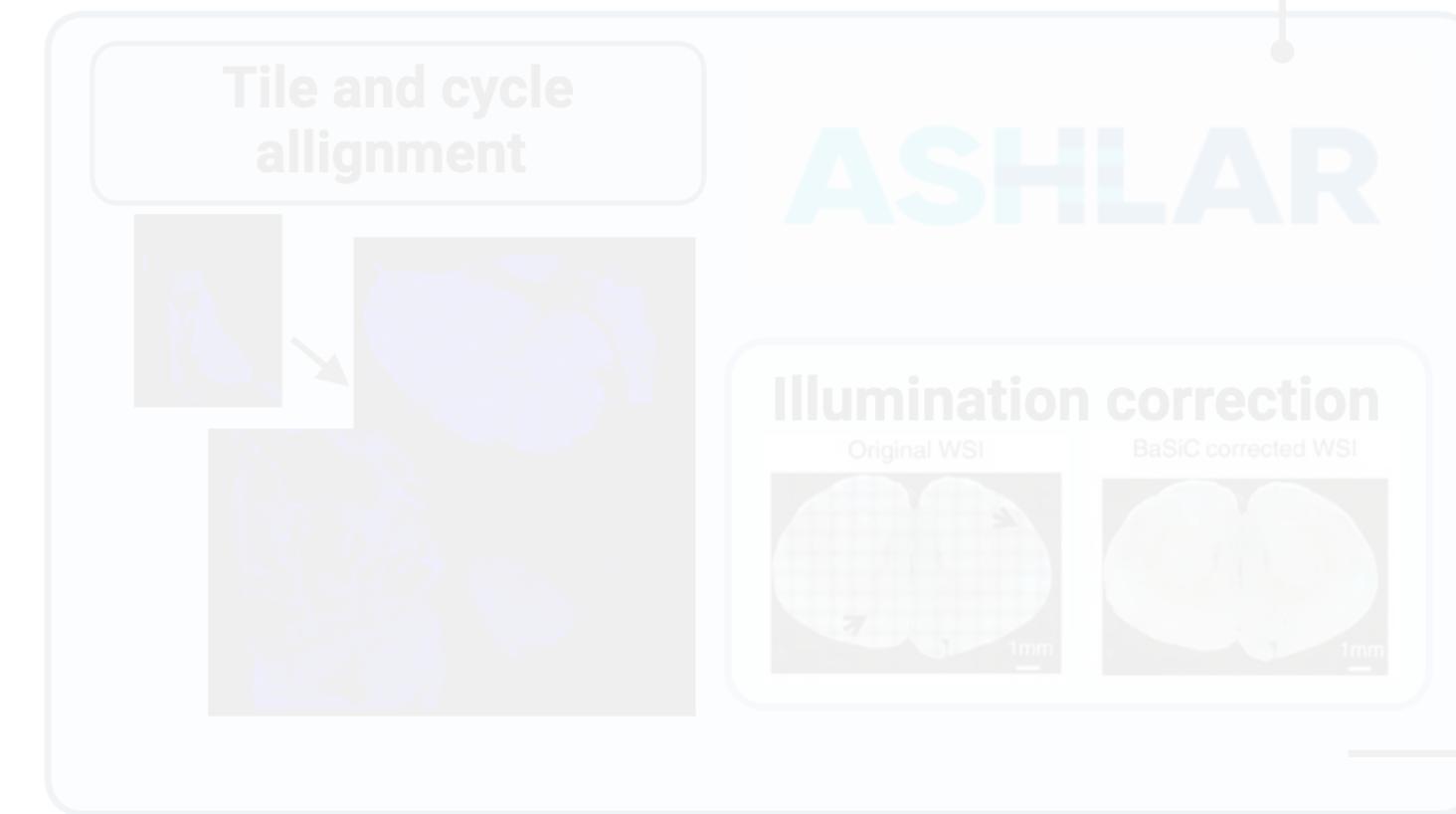
DATA ANALYSIS



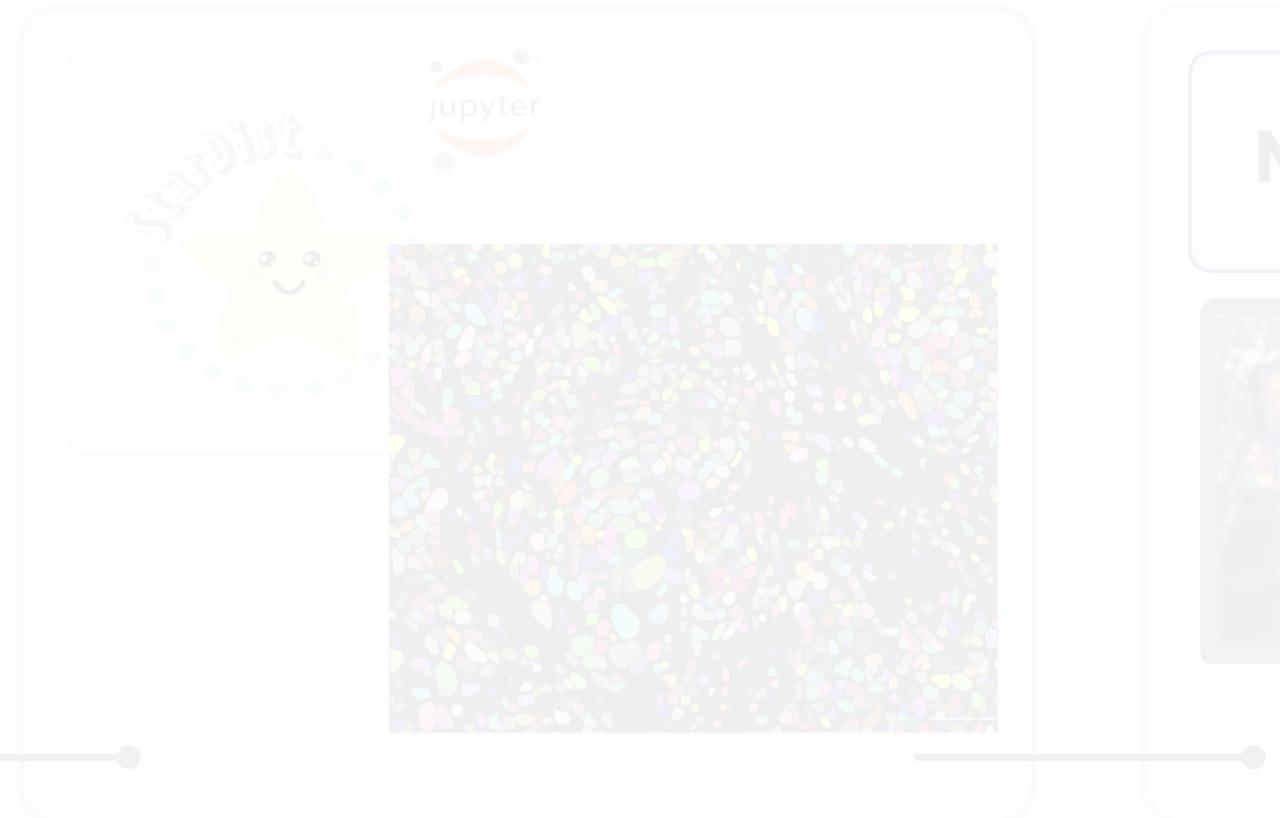
Step 3: Quantification.

In this step, we can calculate the marker expression level per cell, using the cell mask and the tiff image.

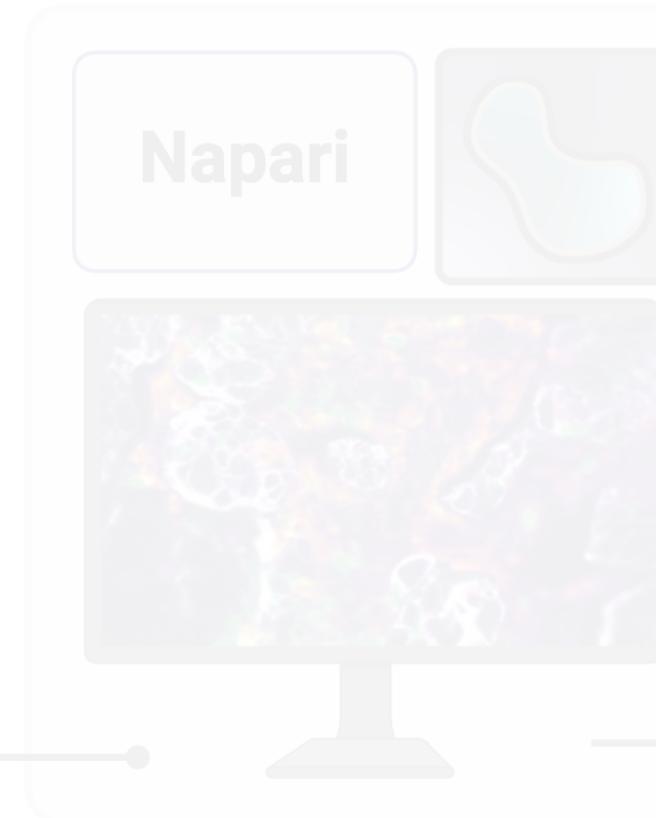
STITCHING



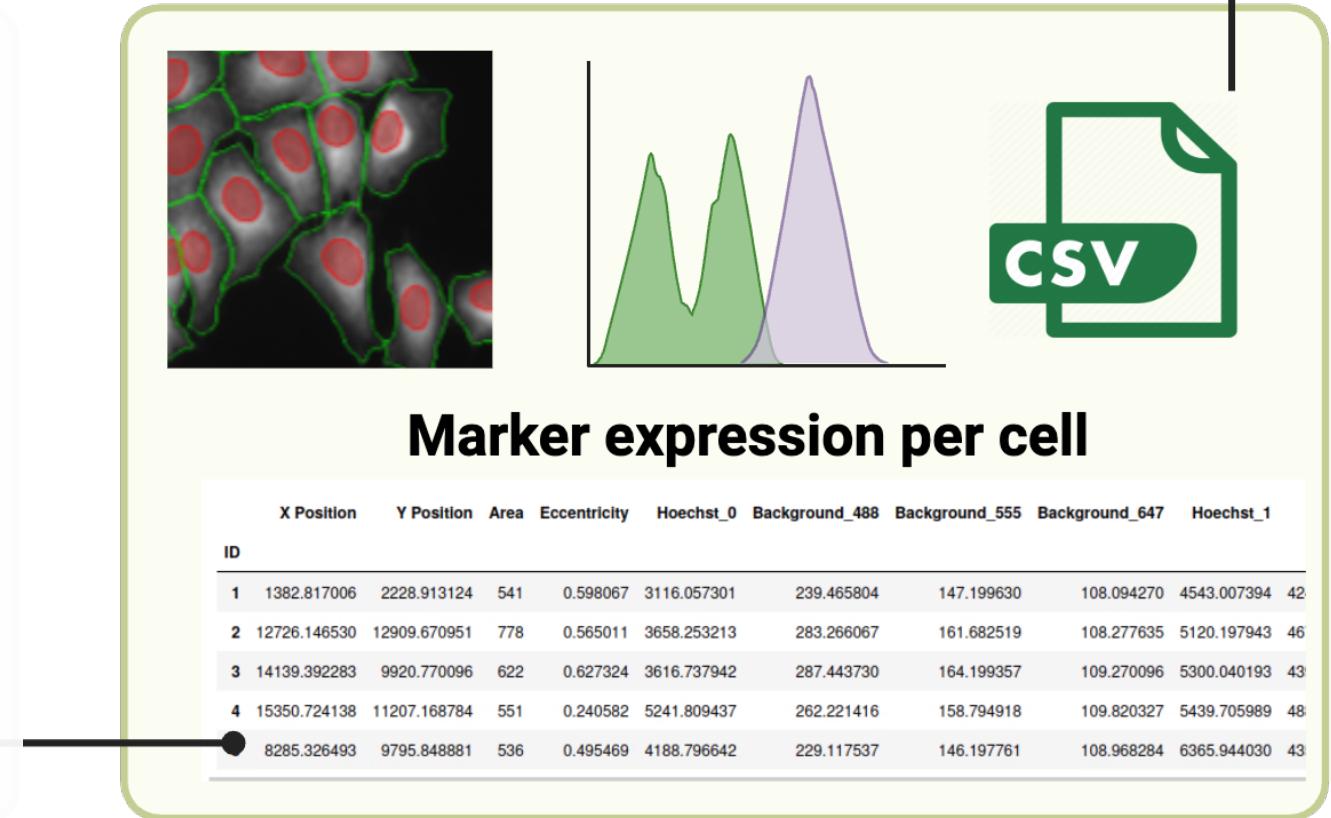
SEGMENTATION



VISUALIZATION



QUANTIFICATION



Download quantification_workflow.py

Create new folder named csv.

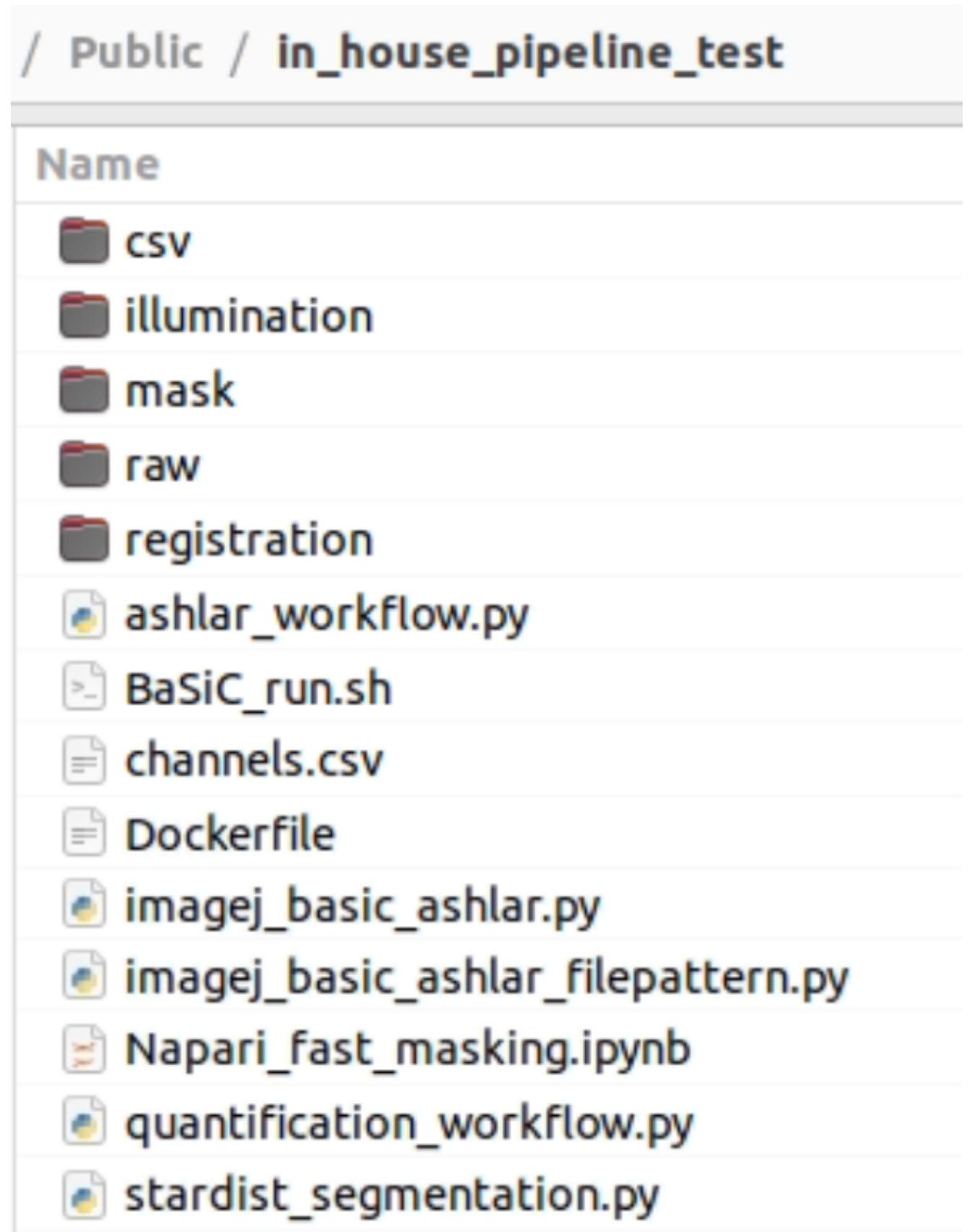
-->

Modify the input and output path in the quantification_workflow.py

```
2 dir2 = './mask' # replace it with your own mask dir
3 dir1 = './registration' # replace it with your own image dir
```

Activate python environment, run the script.

```
(quantification) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ conda activate quantification
(quantification) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ python quantification_workflow.py -o ./csv -ch ./channels.csv -c 1
Channels are ['DNA1', 'CD163', 'PanCK', 'CD8a', 'Vimentin', 'DNA2', 'CD4', 'FAP', 'PAX8']
imagePath is ./registration/S130_iOme.ome.tif
maskpath is ./mask/S130_iOme.ome.tif
m name is S130_iOme
Current mask is [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
Length of channels: 9
channelQuantification step, channel names file is ./channels.csv, read in channel 0
```



Note: you can specify larger number for parameter -c, to enable parallel computation.

Keep an eye on the system monitor if you set a large c!

Finished! Now you get the csv files (data table) and you can perform downstream analysis on it.

```
Sample S130_iOme quantified in 68.81 seconds
imagePath is ./registration/S131_iOme.ome.tif
maskpath is ./mask/S131_iOme.ome.tif
m name is S131_iOme
Current mask is [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
Length of channels: 9
channelQuantification step, channel names file is ./channels.csv, read in channel 0
channelQuantification step, channel names file is ./channels.csv, read in channel 1
channelQuantification step, channel names file is ./channels.csv, read in channel 2
channelQuantification step, channel names file is ./channels.csv, read in channel 3
channelQuantification step, channel names file is ./channels.csv, read in channel 4
channelQuantification step, channel names file is ./channels.csv, read in channel 5
channelQuantification step, channel names file is ./channels.csv, read in channel 6
channelQuantification step, channel names file is ./channels.csv, read in channel 7
channelQuantification step, channel names file is ./channels.csv, read in channel 8
./registration/S131_iOme.ome.tif
Sample S131_iOme quantified in 253.91 seconds
(quantification) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$
```

Name	Size	Modified	
S130_iOme.csv	52.3 MB	20:40	☆
S131_iOme.csv	173.3 MB	20:43	☆

STAINING AND IMAGE ACQUISITION

Here we already reach the end of the image-preprocessing in this short seminar.

Sample Analysis

Next steps are quality control and cell phenotyping.

For quality control with CyLinter, they have very detailed instructions:

<https://labsyspharm.github.io/cylinter/>

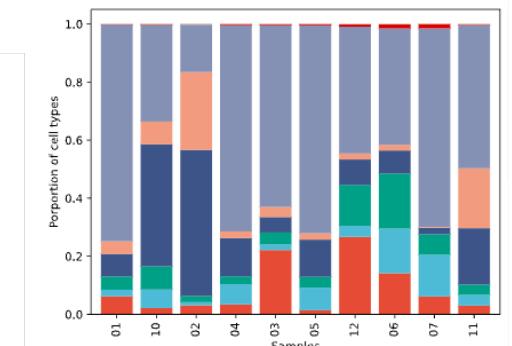
If there are still many questions, we might need another seminar to introduce...

For cell phenotyping, you can use SCIMAP gating (https://scimap.xyz/tutorials/md/scimap_phenotyping/) or Tribus (<https://github.com/farkkilab/tribus>).

DATA ANALYSIS



TRIBUS



QUALITY CONTROL

CyLinter

IMAGE SEGMENTATION FILTER

QUANTIFICATION



ID	X Position	Y Position	Area	Eccentricity	Hoechst_0	Background_488	Background_555	Background_647	Hoechst_1
1	1380.817096	2228.913124	541	0.598607	3116.057301	239.455904	147.198030	108.094279	4543.007394
2	12720.146930	12909.670951	778	0.569011	3058.255213	203.266967	181.692519	108.277630	5100.197943
3	14139.392203	9920.770996	822	0.627324	3618.737942	287.443730	164.199357	109.270090	5300.040190
4	15055.734196	11207.168794	551	0.540582	3541.094327	252.221416	158.794918	109.820327	5429.795862
5	8289.726493	4705.840981	936	0.495409	4198.795862	229.117537	140.107701	108.095294	4205.040703

Thanks for listening!

Questions?