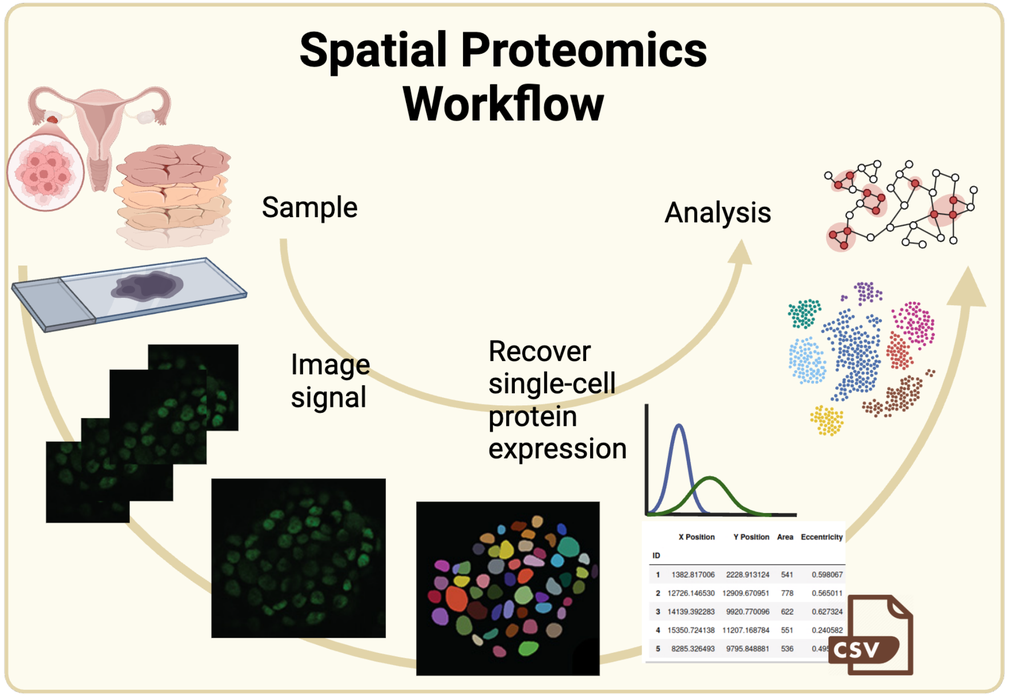
Tutorial: In-house t-CyCIF image preprocessing pipeline

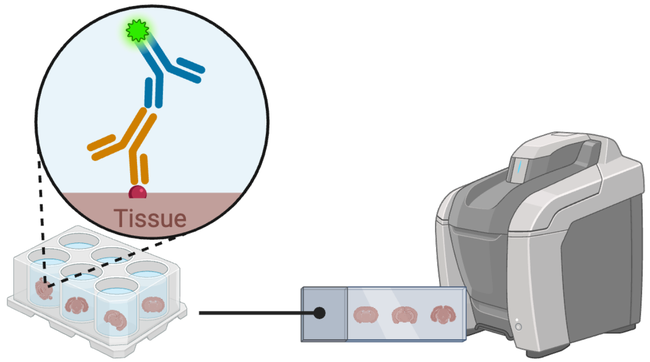
Färkkilä Lab Seminar (OncoTalks)

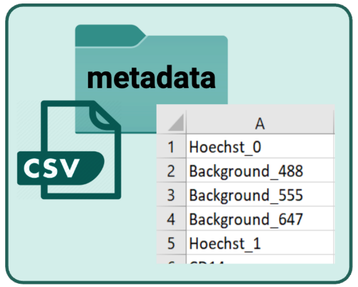
Ziqi Kang, 27.02.2025
Thanks to Lina: providing the material

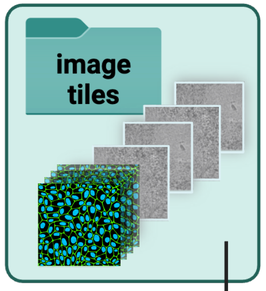
FÄRKKILÄ LABORATORY
Systems Medicine of the Tumor Microenvironment


Spatial Proteomics Workflow
Sample
Image signal
Recover single-cell protein expression
Analysis
CSV

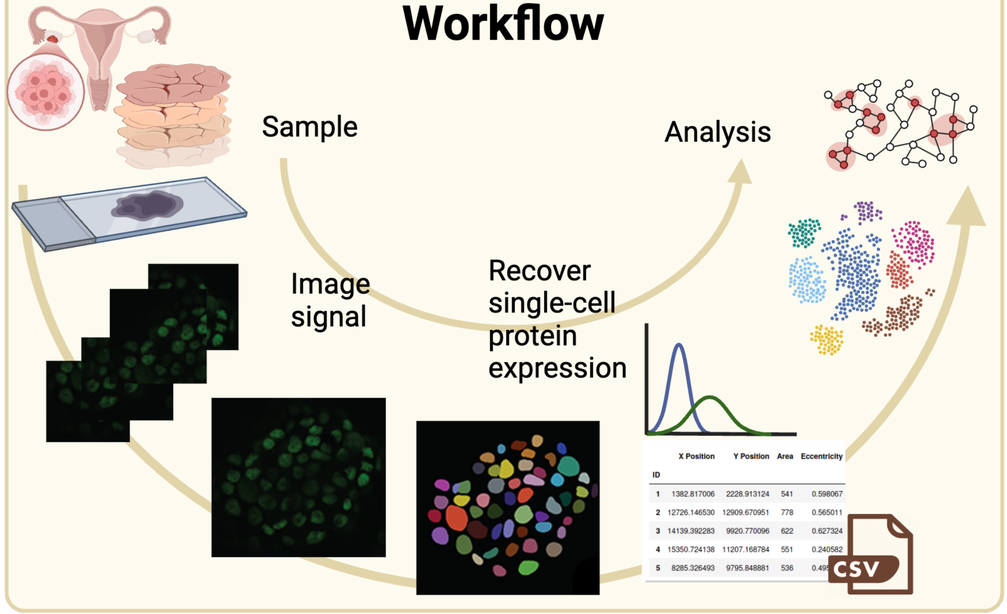
STAINING AND IMAGE ACQUISITION







Spatial Proteomics Workflow



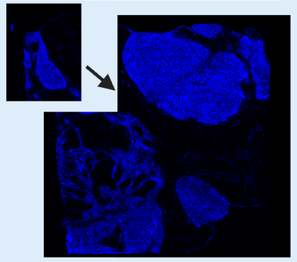
DATA ANALYSIS

QUALITY CONTROL

CyLinter
IMAGE SEGMENTATION FILTER

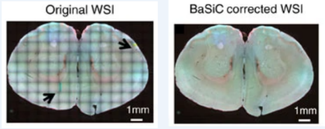
STITCHING

Tile and cycle allignment

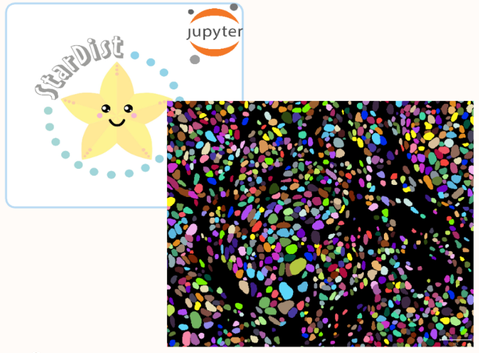


ASHLAR

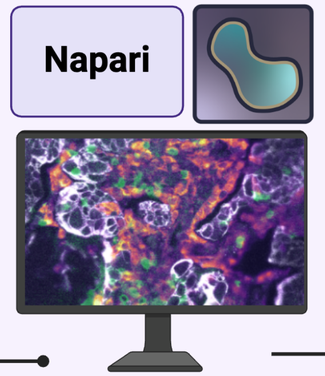
Illumination correction



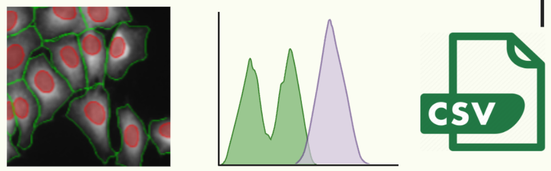
SEGMENTATION



VISUALIZATION



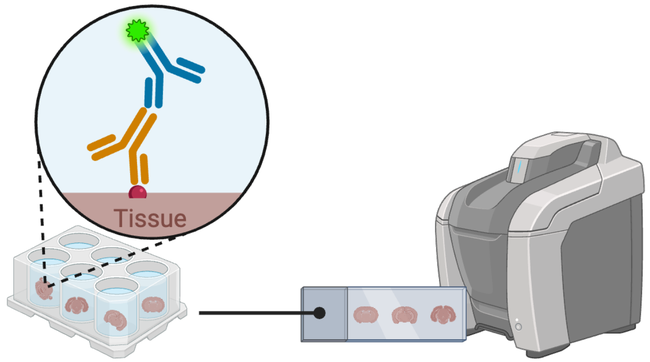
QUANTIFICATION



Marker expression per cell

ID	X Position	Y Position	Area	Eccentricity	Hoechst_0	Background_488	Background_555	Background_647	Hoechst_1	
1	1382.817006	2228.913124	541	0.598067	3116.057301	239.465804	147.199630	108.094270	4543.007394	42
2	12726.146530	12909.670951	778	0.565011	3658.253213	283.266067	161.682519	108.277635	5120.197943	46
3	14139.392283	9920.770096	622	0.627324	3616.737942	287.443730	164.199357	109.270096	5300.040193	43
4	15350.724138	11207.168784	551	0.240582	5241.809437	262.221416	158.794918	109.820327	5439.705989	48
5	8285.326493	9795.848881	536	0.495469	4188.796642	229.117537	146.197761	108.968284	6365.944030	43

STAINING AND IMAGE ACQUISITION



Wet-lab experiment: Tissue staining, scanning

Spatial Proteomics Workflow

Sample

Image signal

Analysis

Recover single-cell protein expression

DATA ANALYSIS

scimap

TRIBUS

metadata

image tiles

QUALITY CONTROL

CyLinter

IMAGE SEGMENTATION FILTER

STITCHING

Tile and cycle allignment

ASHLAR

Illumination correction

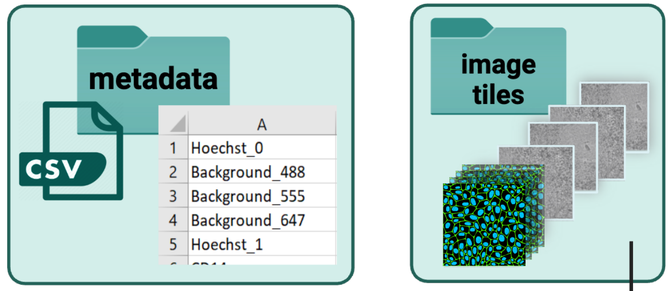
SEGMENTATION

VISUALIZATION

Napari

QUANTIFICATION

Marker expression per cell

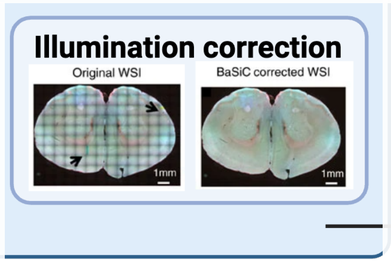


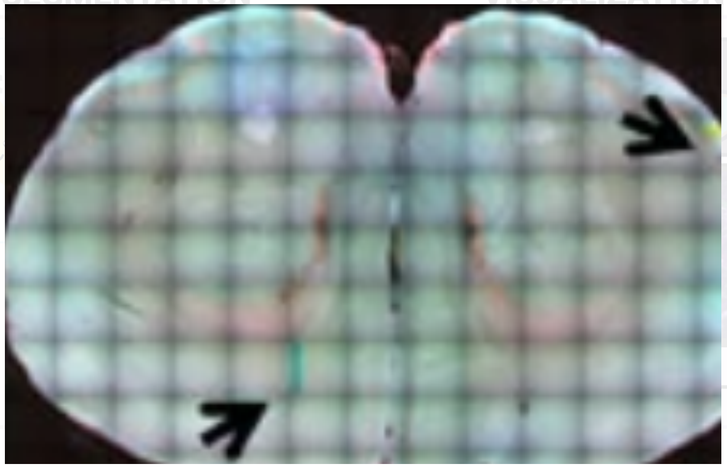
Now you have the raw images from the scanner…
Time to run the pipeline!
Go to https://github.com/farkkilab/image_processing
You can download files there.

Step 0:

Do I need to perform illumination correction?

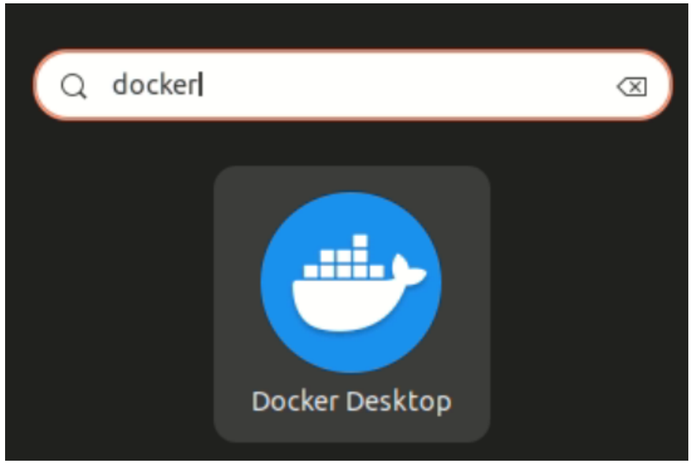
- **When you do scanning, you can use <u>Avivator</u> to quickly check the image.**
- **Did you see this?**
- **If so, you will need to run.**



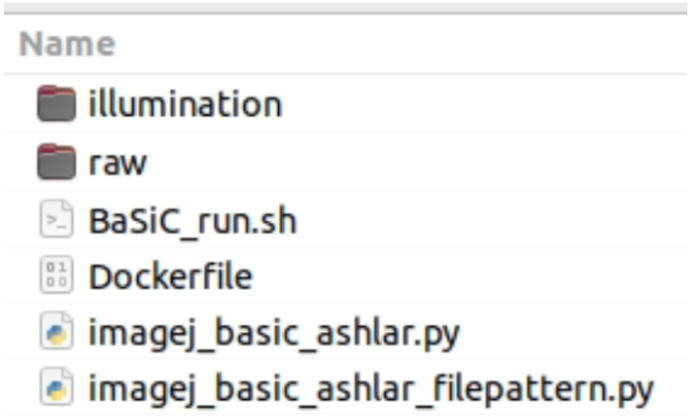


To run illumination correction (with BaSiC)

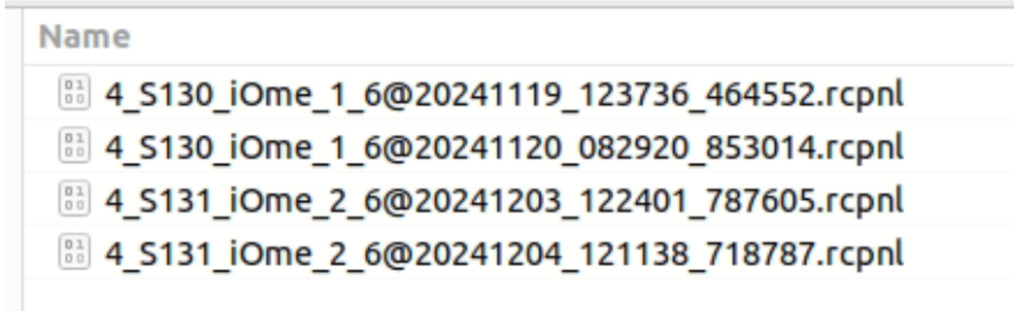
- It only works on Linux system
- Make sure you have Docker started:



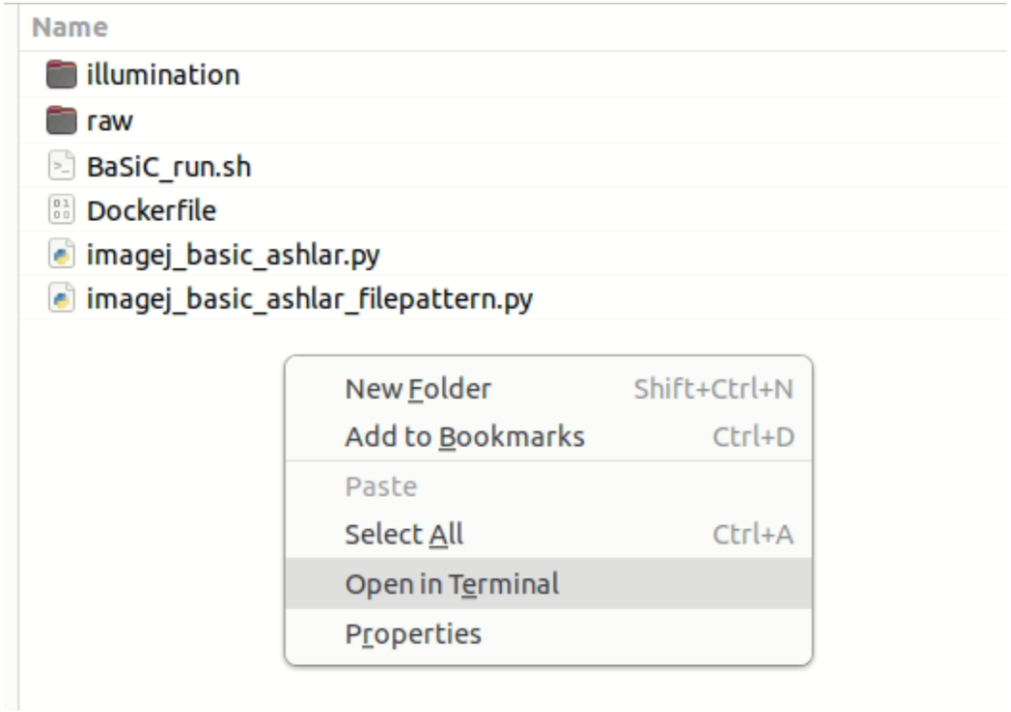
- Your folder should contain those files (you can download from GitHub:
  - **Make sure your folder is in local now (under /home), not in the remote drive!** (Docker have no access to the files in the remote drive)
  - Fold **raw** is for input.
  - Folder **illumination** is for output.



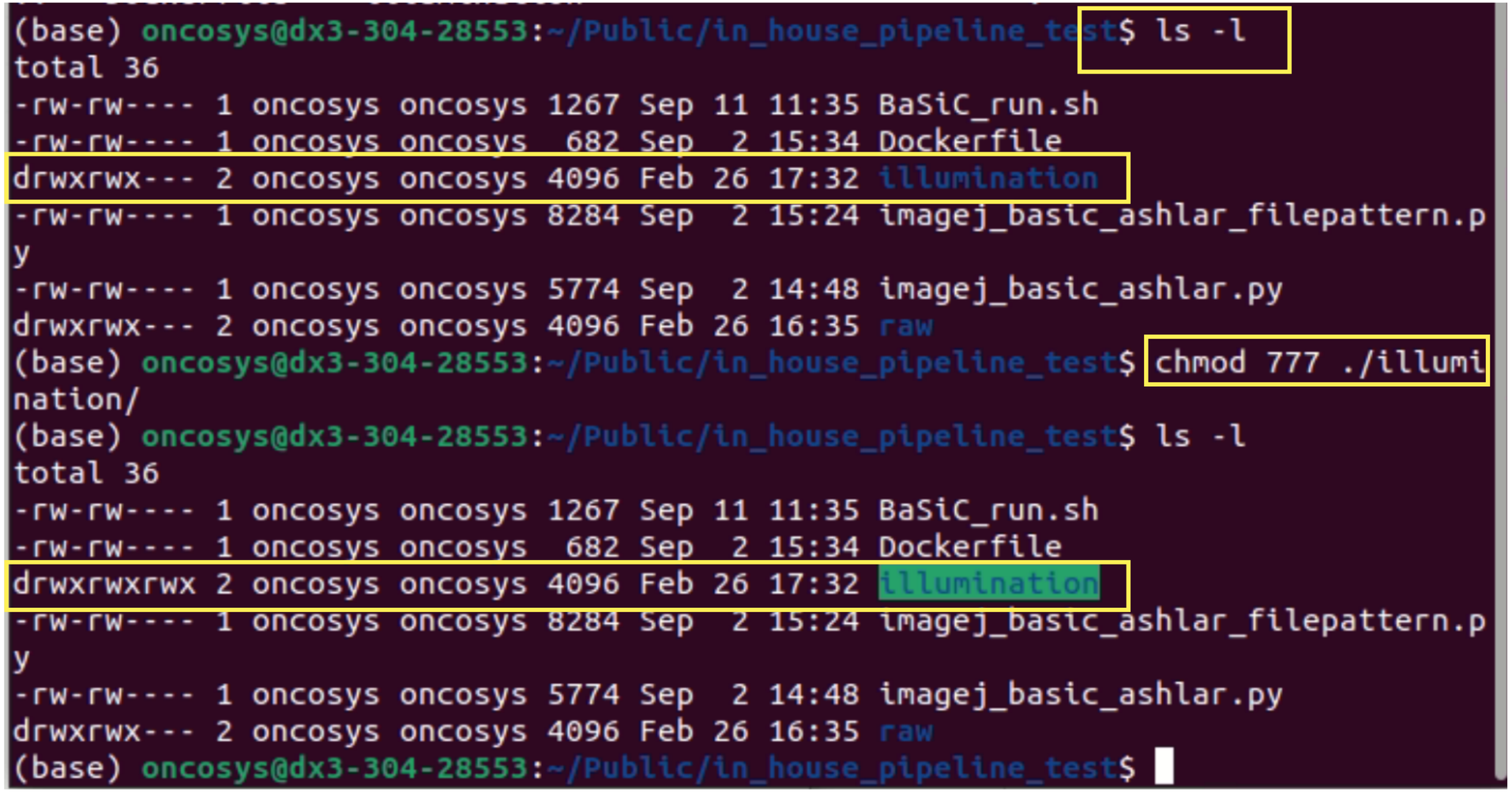
- * Notice! All rcpnls should be stored in the raw folder like this:



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



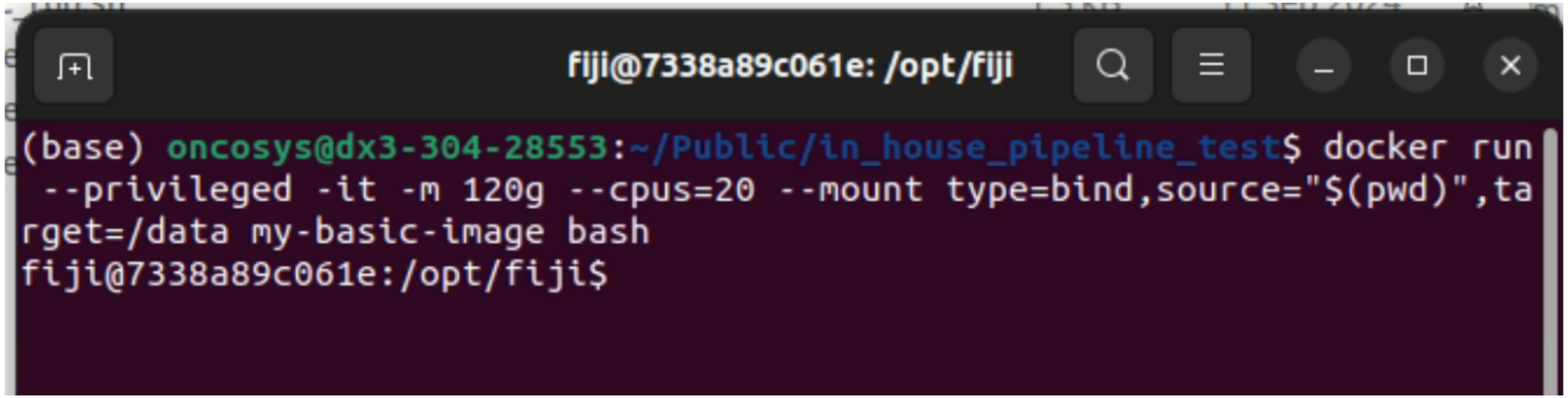
Make sure you have full rights to the output folder. Otherwise the output files cannot be saved properly.
Use `ls -l` to check.
Use `chmod 777 <path>` to ensure you have full modification rights.



- 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:



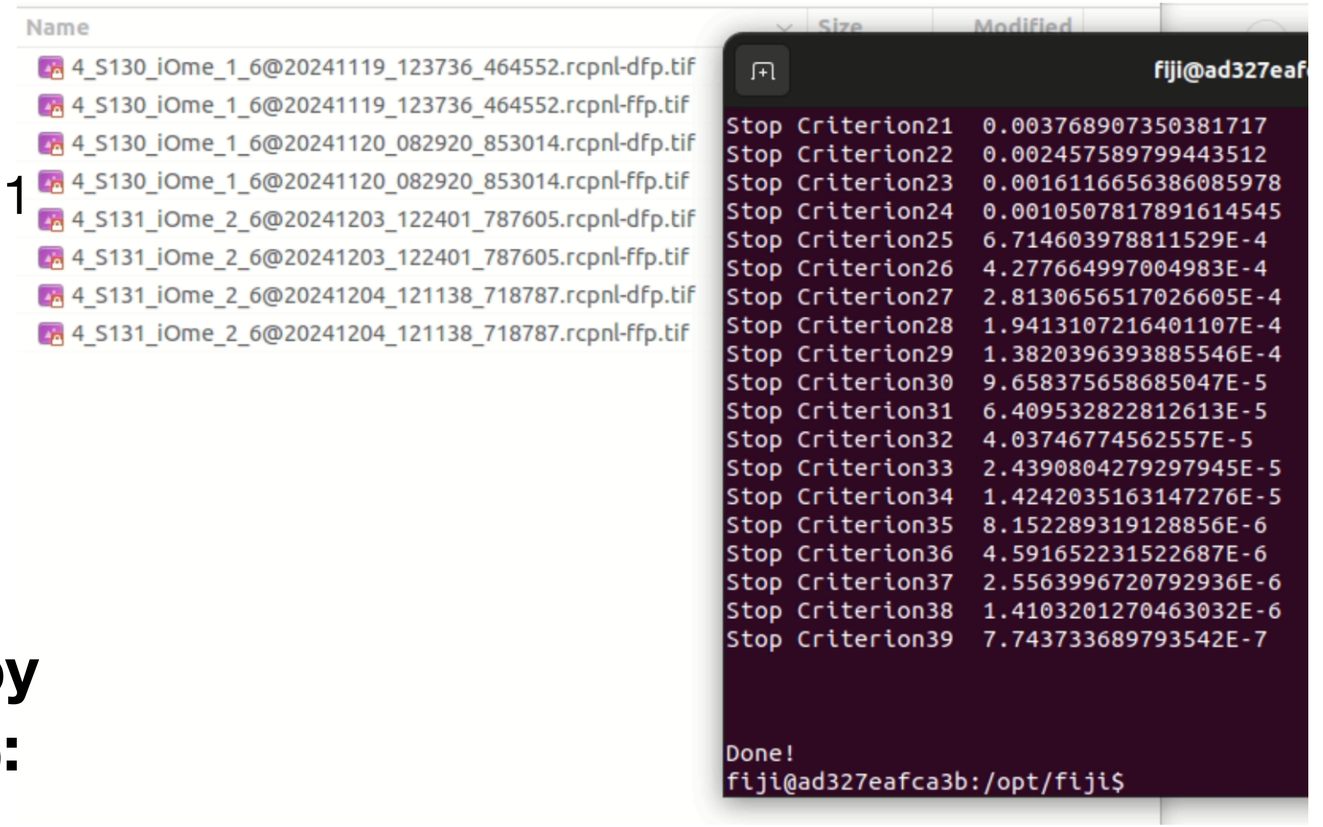
- Next:

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

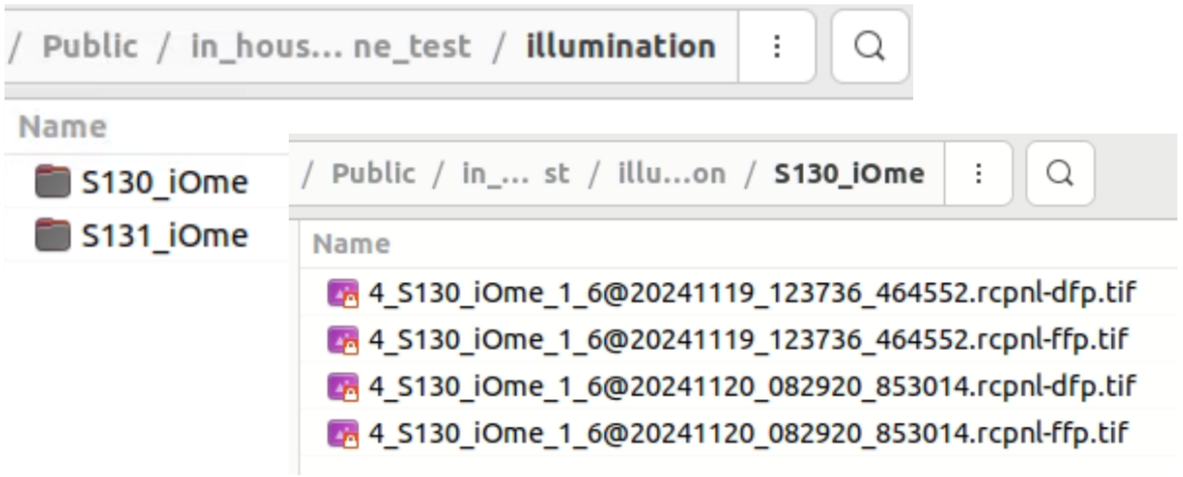
- It is running now:

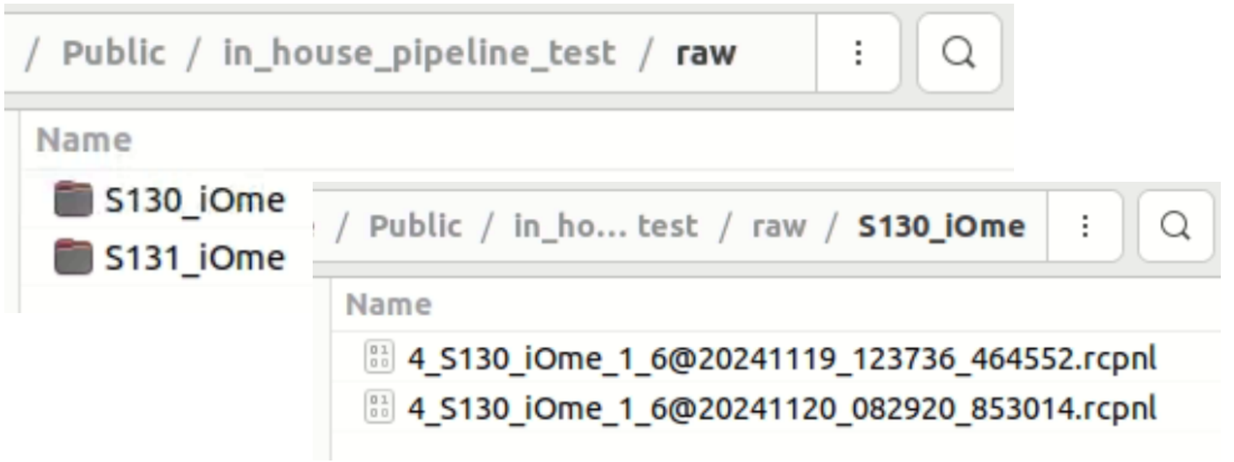
```
RCPNLReader initializing /data/raw/4_S130_iOme_1_6@20241119_123736_46455
2.rcpnl
Reading header
Populating core metadata
Reading extended header
Populating original metadata
Populating OME metadata
Reading header
Populating original metadata
Populating OME metadata



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!

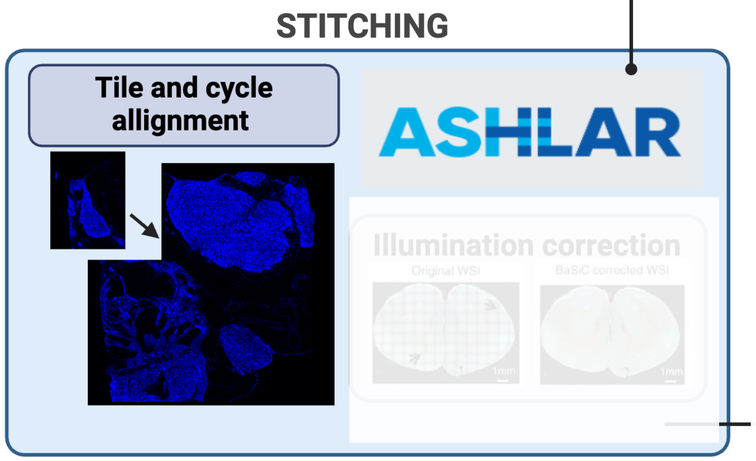


Prepare your folders. You can now copy them to the remote drive (if no space in local)**:**





Step 1: Stitching with ASHLAR. Stitch all tiles and then align the channels.



Activate the conda environment for ashlar.

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

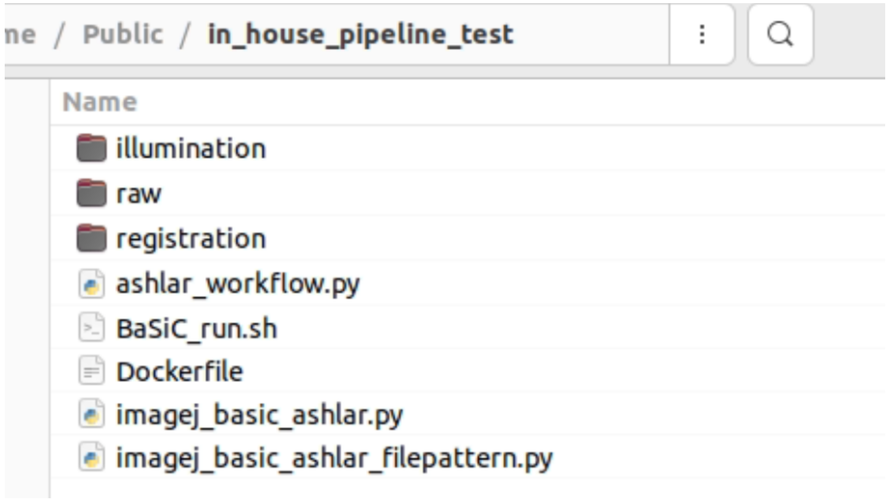
Download `ashlar_workflow.py` from GitHub.

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

```
16 my_path = "./raw"
17 output_path = "./registration"
18 subfolders = [ f.path for f in os.scandir(my_path) if f.is_dir() ]
19 file_type = 'rcpnl' # 'nd2' 'rcpnl'
20
21 illumination = 'Y' # 'N'
22 illumination_folder = "./illumination/"
```

Create a folder to store the output.
Here I named it “registration”.

Your folder should be like this before run ASHLAR:

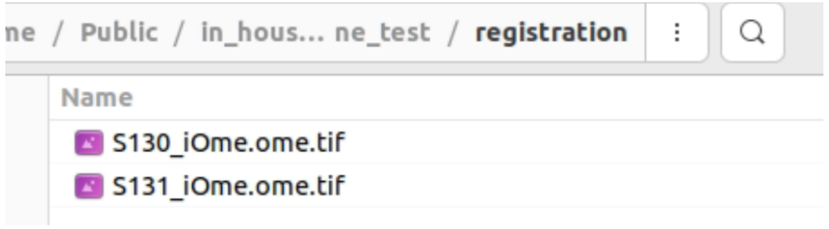


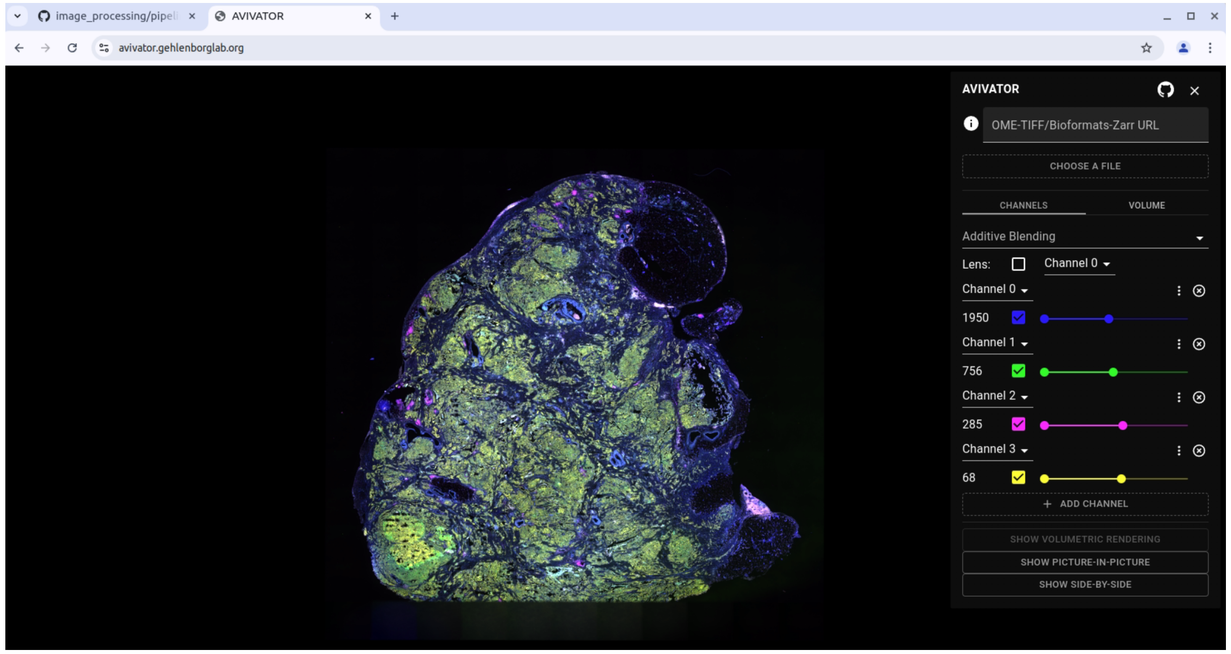
Run:

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_iOm
e_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_iOm
e_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
```

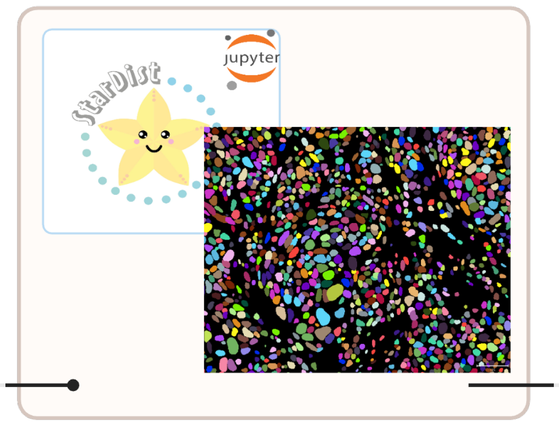
Done! Use Avivator to quickly check the image quality:





Step 2: Segment cells with Stardust.

This step generates cell masks, allow us to retrieve single-cell level information from images.

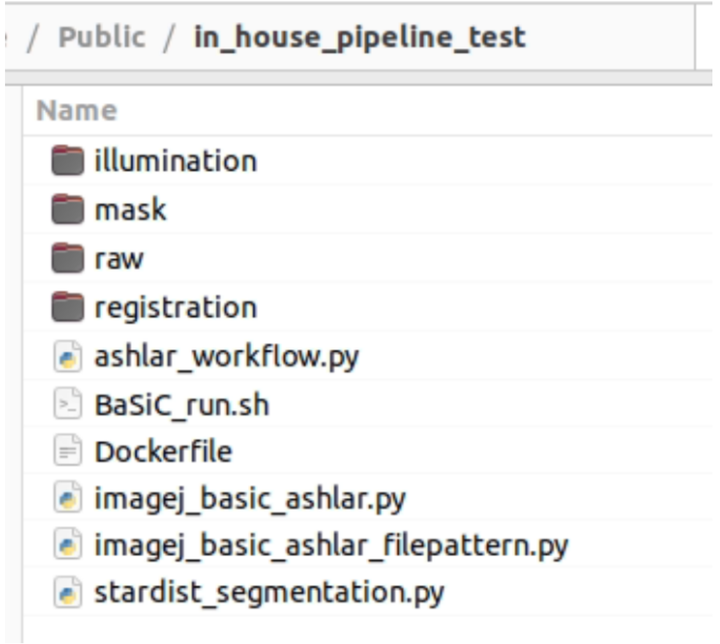


Download `stardist_segmentation.py`

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.

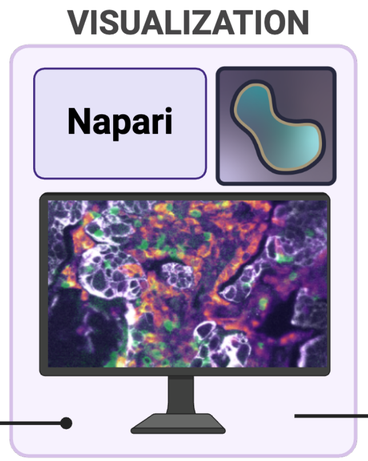
```
(base) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ conda activate sta
rdist
(stardist) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ python stardis
t_segmentation.py
```

```
165]
Loading thresholds from 'thresholds.json'.
Using default values: prob_thresh=0.479071, nms_thresh=0.3.
Image list: ['S130_iOme.ome.tif', 'S131_iOme.ome.tif']
Do you want to proceed? [y/n]y
```

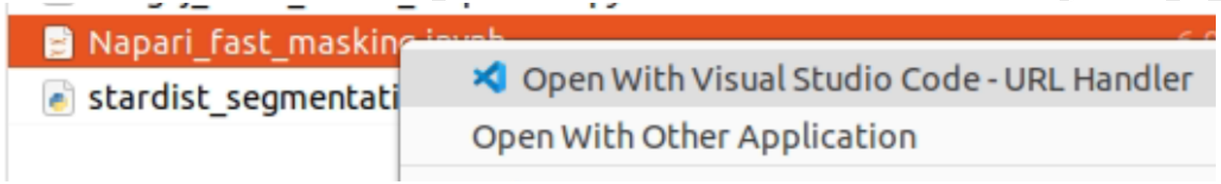
It will ask you again when all images are finished. Press N to stop.

```
Image has a shape of (26226, 26624)
Calculate tiles = 12
100%|                                                  | 144/144 [02:45<00:00,  1.15s/it]
Finish image ./registration/S131_iOme.ome.tif
Do you want to proceed? [y/n]n
```

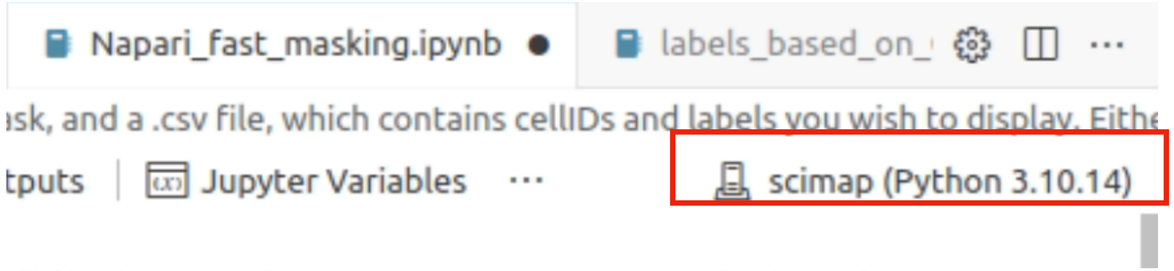
You can already start to use Napari to check the image and the segmentation quality, if you wish.



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

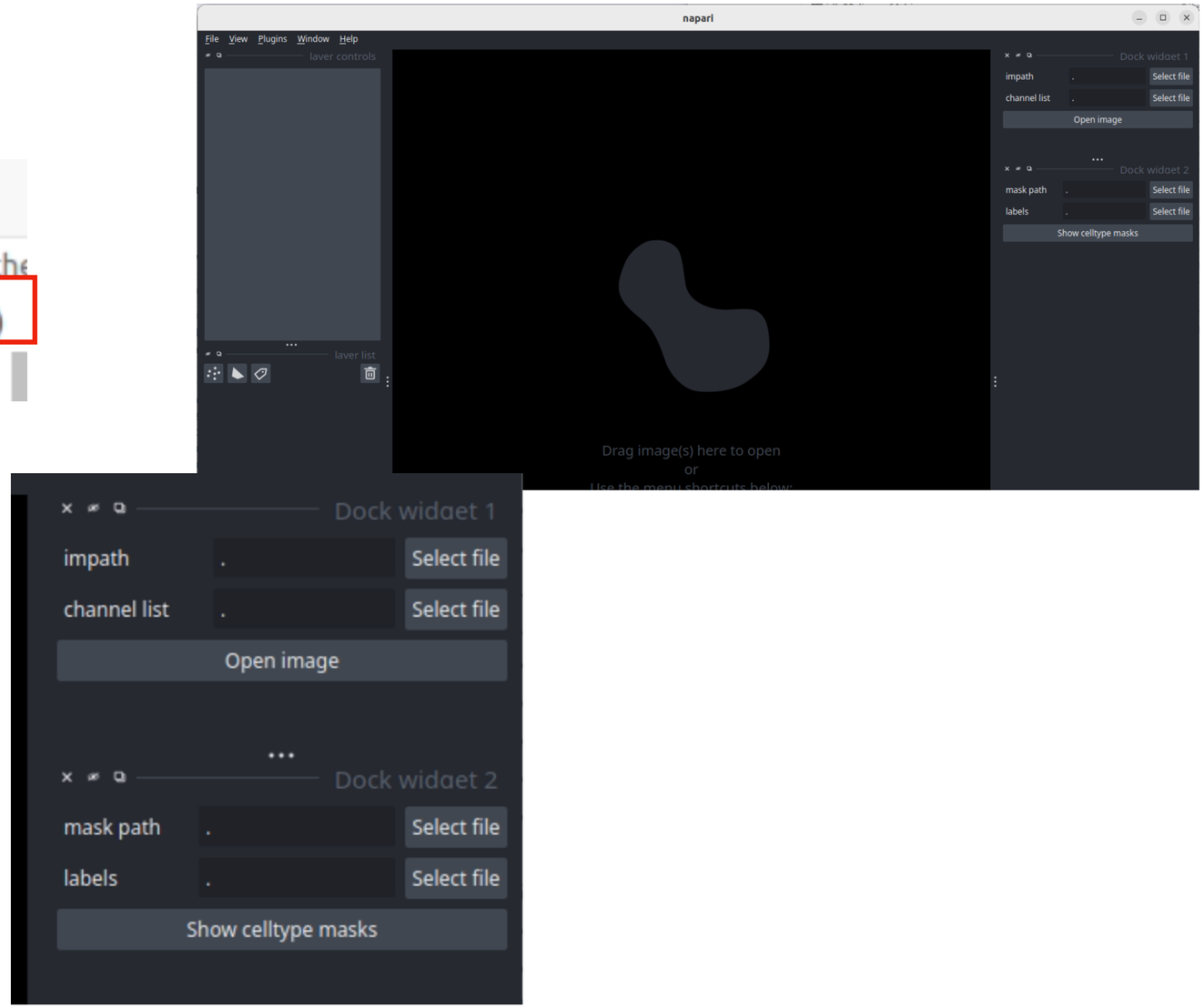


Choose the conda environment here in the vscode.



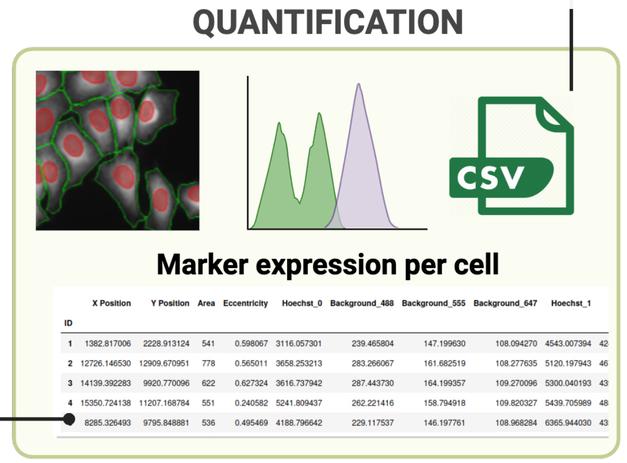
Click and run every chunk in the notebook.

A Napari window will open:



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

Step 3: Quantification.
In this step, we can calculate the marker expression level per cell, using the cell mask and the tiff image.



ID	X Position	Y Position	Area	Eccentricity	Hoechst_0	Background_488	Background_555	Background_647	Hoechst_1	
1	1382.817006	2228.913124	541	0.598067	3116.057301	239.465804	147.199630	108.094270	4543.007394	42
2	12726.146530	12909.670951	778	0.565011	3658.253213	283.266067	161.682519	108.277635	5120.197943	46
3	14139.392283	9920.770096	622	0.627324	3616.737942	287.443730	164.199357	109.270096	5300.040193	43
4	15350.724138	11207.168784	551	0.240582	5241.809437	262.221416	158.794918	109.820327	5439.705989	48
	8285.326493	9795.848881	536	0.495469	4188.796642	229.117537	146.197761	108.968284	6365.944030	43

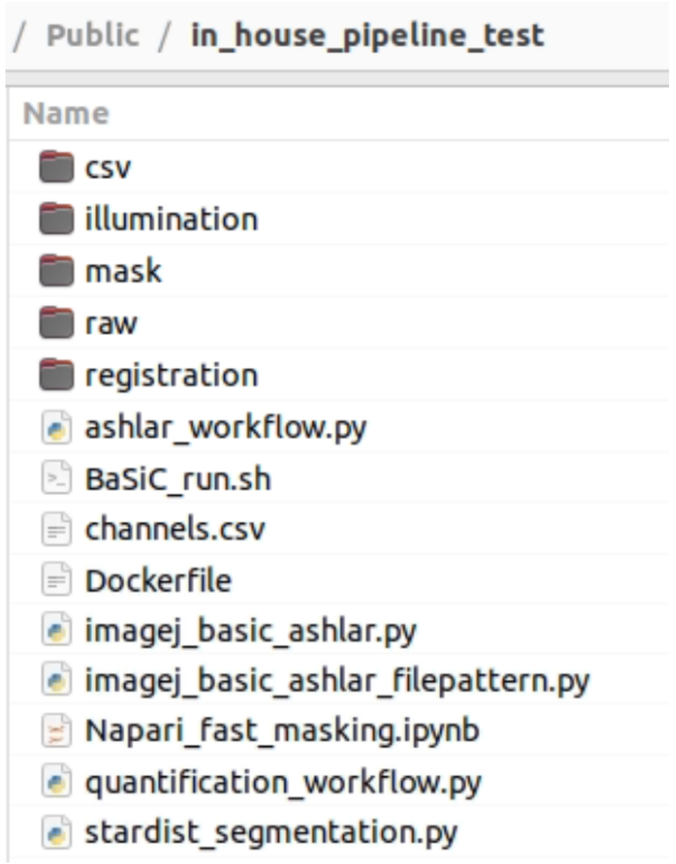
Download `quantification_workflow.py`

Create new folder named `csv`.

Modify the input and output path in the `quantification_workflow.py`

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

—>



Activate python environment, run the script.

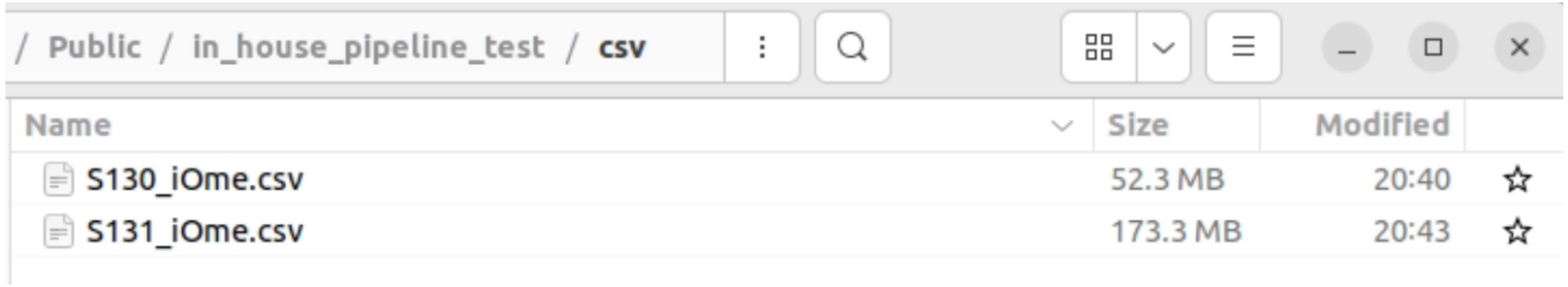
```
(quantification) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ conda ac
tivate quantification
(quantification) oncosys@dx3-304-28553:~/Public/in_house_pipeline_test$ python q
uantification_workflow.py -o ./csv -ch ./channels.csv -c 1
Channels are ['DNA1', 'CD163', 'PanCK', 'CD8a', 'Vimentin', '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 channe
l 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$
```

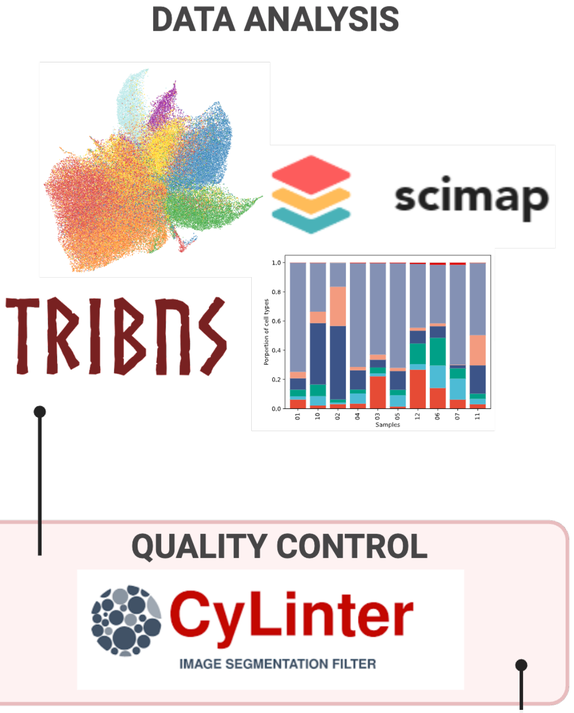


Here we already reach the end of the image-preprocessing in this short seminar.

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).



Thanks for listening!

Questions?