<u>single-lif-analysis-v2</u> (Updated 11/13/24)

Code & Documentation by Tyler Johnston

Inputs: .lif file directory, final saving directory
Outputs: stacked fluorescent images (.ometif), segmented masks (.ometif), single-cell data (.csv)

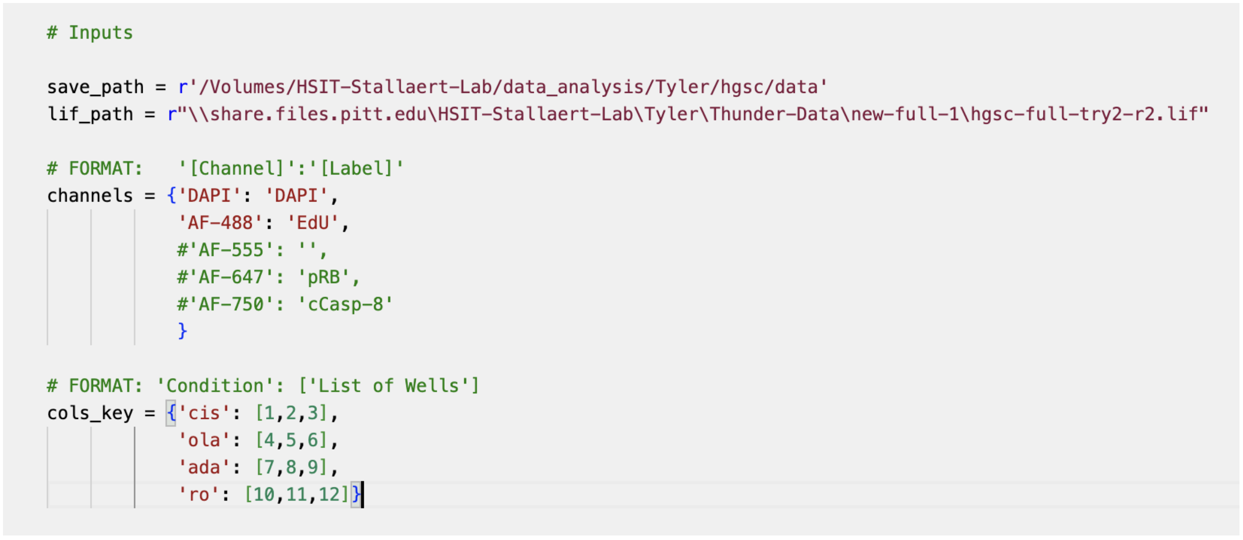
1-lif-to-ometiff.ipynb– Convert .lif data to .ometiff data

** Script works best if raw data is also saved, and images are in format of* `{Slide Name}/{Row}/{Col}`

Inputs: .lif file directory, final saving directory
Outputs: stacked fluorescent images (.ometif)

1. Complete the following inputs:
   a. `save_path`: directory where the analysis images & csv will be saved.
   b. `lif_path`: path directly to the .lif file to be analyzed.
   c. `channels`: dictionary describing channel contents of image.
      i. `channels` KEY describes the channel
      ii. `channels` VALUE describes the antibody in the given channel
      iii. If some channels are not being used, COMMENT THEM OUT.
      iv. Channels being analyzed should be in ASCENDING ORDER.
   d. `cols_key`: dictionary describing contents of each column.
      i. `cols_key` KEY describes the condition
      ii. `cols_key` VALUE lists the wells holding those conditions

Figure 1.

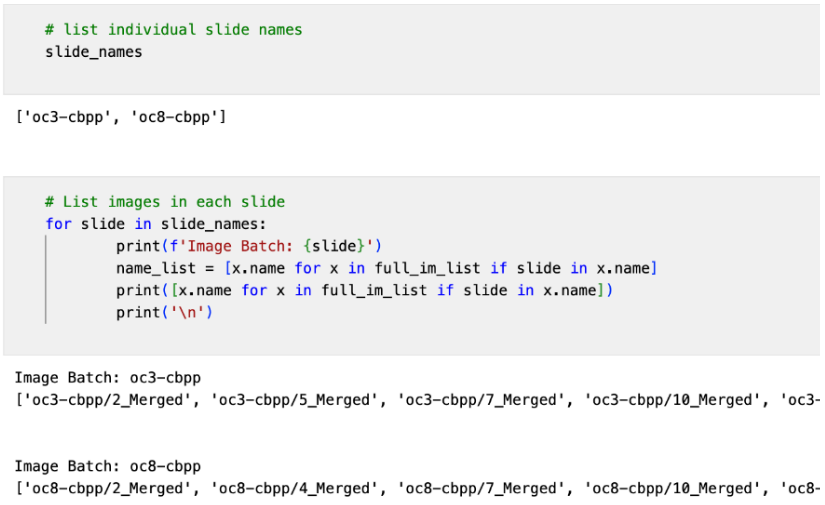


```
# Inputs

save_path = r'/Volumes/HSIT-Stallaert-Lab/data_analysis/Tyler/hgsc/data'
lif_path = r"\\share.files.pitt.edu\HSIT-Stallaert-Lab\Tyler\Thunder-Data\new-full-1\hgsc-full-try2-r2.lif"

# FORMAT:   '[Channel]':'[Label]'
channels = {'DAPI': 'DAPI',
            'AF-488': 'EdU',
            #'AF-555': '',
            #'AF-647': 'pRB',
            #'AF-750': 'cCasp-8'
            }

# FORMAT: 'Condition': ['List of Wells']
cols_key = {'cis': [1,2,3],
            'ola': [4,5,6],
            'ada': [7,8,9],
            'ro': [10,11,12]}
```

DAPI and AF-488 are being used– the non-utilized channels are commented out so they will not be included in the naming scheme. Wells 1-3 have been treated with cisplatin ('cis'), wells 4-6 have been treated with olaparib ('ola'), etc.

2. Review contents of .lif file, including names of slides & individual images.

Figure 2.


```
# list individual slide names
slide_names
```

```
['oc3-cbpp', 'oc8-cbpp']
```

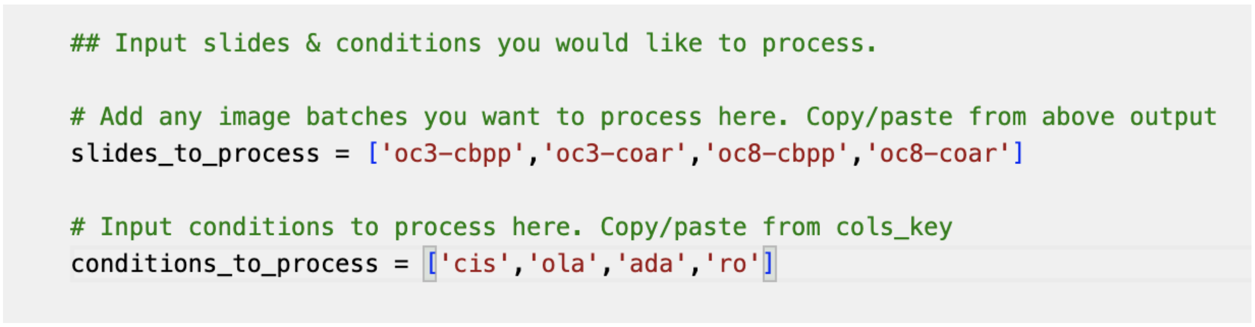
```
# List images in each slide
for slide in slide_names:
        print(f'Image Batch: {slide}')
        name_list = [x.name for x in full_im_list if slide in x.name]
        print([x.name for x in full_im_list if slide in x.name])
        print('\n')
```

```
Image Batch: oc3-cbpp
['oc3-cbpp/2_Merged', 'oc3-cbpp/5_Merged', 'oc3-cbpp/7_Merged', 'oc3-cbpp/10_Merged', 'oc3-


Image Batch: oc8-cbpp
['oc8-cbpp/2_Merged', 'oc8-cbpp/4_Merged', 'oc8-cbpp/7_Merged', 'oc8-cbpp/10_Merged', 'oc8-
```

Slide names available to process are ‘oc3-cbpp’ and ‘oc8-cbpp’. Image names within each slide are listen under ‘Image Batch: [Slide Name]’

3. Complete inputs of what images you would like to analyze. ONLY these selections will be converted to .ometif files and will be usable in subsequent scripts.
    a. `slides_to_process`: list describing which slides will be converted to .ometif format.
        i. Inputs MUST match the name of an image batch– see output to copy/paste.
    b. `conditions_to_process`: conditions from cols_key to include from each slide.
        i. Inputs MUST match a condition from cols_key.

Figure 3.


```
## Input slides & conditions you would like to process.

# Add any image batches you want to process here. Copy/paste from above output
slides_to_process = ['oc3-cbpp','oc3-coar','oc8-cbpp','oc8-coar']

# Input conditions to process here. Copy/paste from cols_key
conditions_to_process = ['cis','ola','ada','ro']
```

`slides_to_process` *include slide names listed above (if a given input is not a valid image batch, it will be ignored). conditions_to_process match with* `cols_key` *keys.*

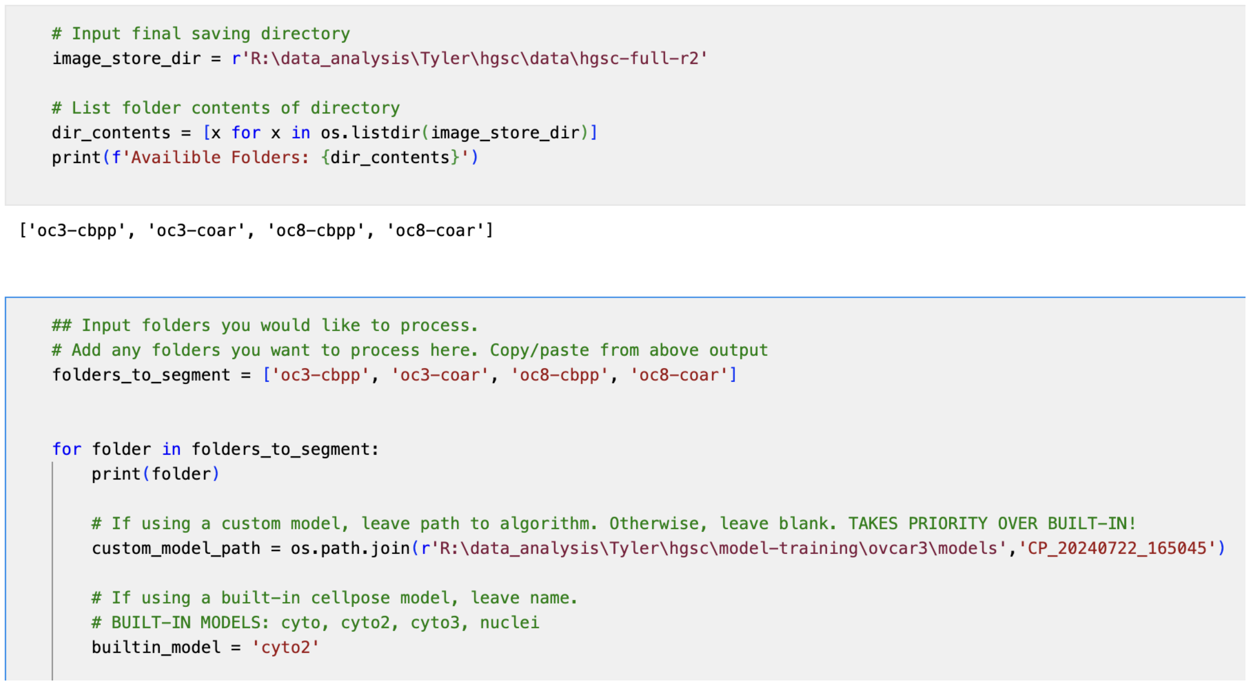
4. Final cell processes the images.

2-segment.ipynb– Produce segmented masks for each image
** If processing more than one batch of images, consider running this code in multiple windows, each segmenting a different experiment/folder.*

Inputs: path to final saving directory
Outputs: segmented masks (.ometif)

1. Complete the following inputs:
    a. `image_store_dir`: path to image analysis directory.
    b. `folders_to_segment`: list describing which folders will be segmented.
        i. Inputs MUST match the name of a folder– see output to copy/paste.
    c. `custom_model_path` OR `builtin_model`:
        i. `custom_model_path`: pathway to a custom Cellpose model.
        ii. `builtin_model`: name of a valid built-in Cellpose model.
            1. Official options include `cyto`, `cyto2`, `cyto3`, and `nuclei`.
        iii. `custom_model_path` takes priority over `builtin_model`– if you are NOT using a custom model, leave `custom_model_path` blank.

Figure 4.



```
# Input final saving directory
image_store_dir = r'R:\data_analysis\Tyler\hgsc\data\hgsc-full-r2'

# List folder contents of directory
dir_contents = [x for x in os.listdir(image_store_dir)]
print(f'Availible Folders: {dir_contents}')
```

```
['oc3-cbpp', 'oc3-coar', 'oc8-cbpp', 'oc8-coar']
```

```
## Input folders you would like to process.
# Add any folders you want to process here. Copy/paste from above output
folders_to_segment = ['oc3-cbpp', 'oc3-coar', 'oc8-cbpp', 'oc8-coar']


for folder in folders_to_segment:
    print(folder)

    # If using a custom model, leave path to algorithm. Otherwise, leave blank. TAKES PRIORITY OVER BUILT-IN!
    custom_model_path = os.path.join(r'R:\data_analysis\Tyler\hgsc\model-training\ovcar3\models','CP_20240722_165045')

    # If using a built-in cellpose model, leave name.
    # BUILT-IN MODELS: cyto, cyto2, cyto3, nuclei
    builtin_model = 'cyto2'
```

`image_store_dir` is the folder within `save_dir` that contains analysis data. `custom_model_path` is the path to a custom-trained model using Cellpose API. `builtin_model` will not be used if `custom_model_path` is not `False`.

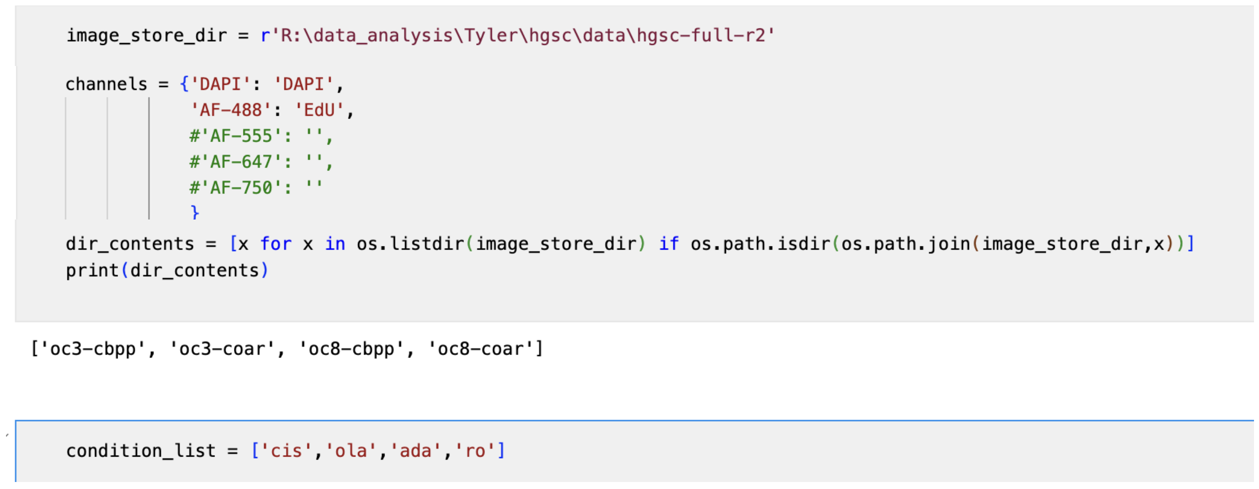
2. Final cell segments the images.

3-make-csv.ipynb– Use stacked fluorescence images & masks to generate single-cell csv.

Inputs: path to final saving directory
Outputs: 1 single-cell signature data (.csv) per drug/condition

1. Complete the following inputs:
    a. `image_store_dir`: path to image analysis directory.
    b. `channels`: dictionary describing channel contents of image.
        i. `channels` KEY describes the channel
        ii. `channels` VALUE describes the antibody in the given channel
        iii. If some channels are not being used, COMMENT THEM OUT.
        iv. Channels being analyzed should be in ASCENDING ORDER.
        v. This should be the SAME as in Script 1.
    c. `condition_list`: list describing which condition groups will be quantified.
        i. Each entry should correspond to a key in `cols_key` *(from script 1)*.

Figure 5.



```
image_store_dir = r'R:\data_analysis\Tyler\hgsc\data\hgsc-full-r2'

channels = {'DAPI': 'DAPI',
            'AF-488': 'EdU',
            #'AF-555': '',
            #'AF-647': '',
            #'AF-750': ''
            }

dir_contents = [x for x in os.listdir(image_store_dir) if os.path.isdir(os.path.join(image_store_dir,x))]
print(dir_contents)
```

```
['oc3-cbpp', 'oc3-coar', 'oc8-cbpp', 'oc8-coar']
```

```
condition_list = ['cis','ola','ada','ro']
```

`image_store_dir` is the folder within `save_dir` that contains analysis data– this should be the same input as Figure 4. `channels` is the exact same dictionary in Figure 1. `condition_list` contains values from Figure 1. Note that each folder listed in the first cell will be quantified given they have related fluorescence images and segmented masks.

2. Final cell generates single-cell signature data in .csv format.
    a. Spreadsheets are generated per individual drugs/conditions (listed in `condition_list`).
        i. i.e. if you have 4 drugs, you will get 4 spreadsheets.