EASI-ORC Guide:

(Shahar Garin, August 2023)

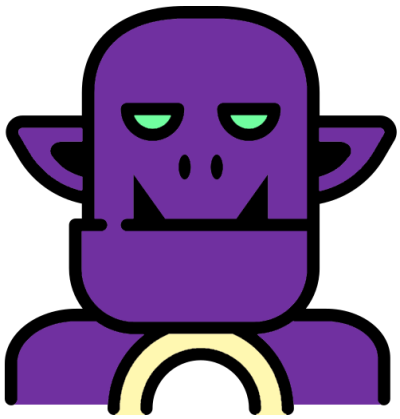


Table of Contents

Introduction

This document aims to thoroughly explain the different steps required for fluorescent image analysis using EASI-ORC and make them accessible with relatively little knowhow. It requires basic knowledge of working with computers, and how to activate scripts in Fiji (Plugins → Macros → Run…).

Prerequisites, such as file formats or needed values are explained in each section, so please read the instructions carefully.

Examples shown in this guide were mainly based on ER segmentation, but the processes described in it can also be used, for any other organelle you wish.

The guide and scripts were all written and used while using a windows PC. Use of any of the scripts on different systems may create issues.

All scripts' code can be found in the EASI-ORC GitHub (ADD LINK ONCE MADE PUBLIC). Each module is described with the steps taken by the corresponding script. These descriptions can be used to perform the steps manually, through FIJI's menus, if you wish to check each step's effect on your images yourself.

Applications you need to install before beginning:

Fiji can be downloaded for free here.

YeastMate standalone app can be downloaded for free here (download .exe file)

YeastMate plugin Fiji installation can be performed by following instructions on this page.

RS-FISH plugin Fiji installation can be performed by following instructions on this page.

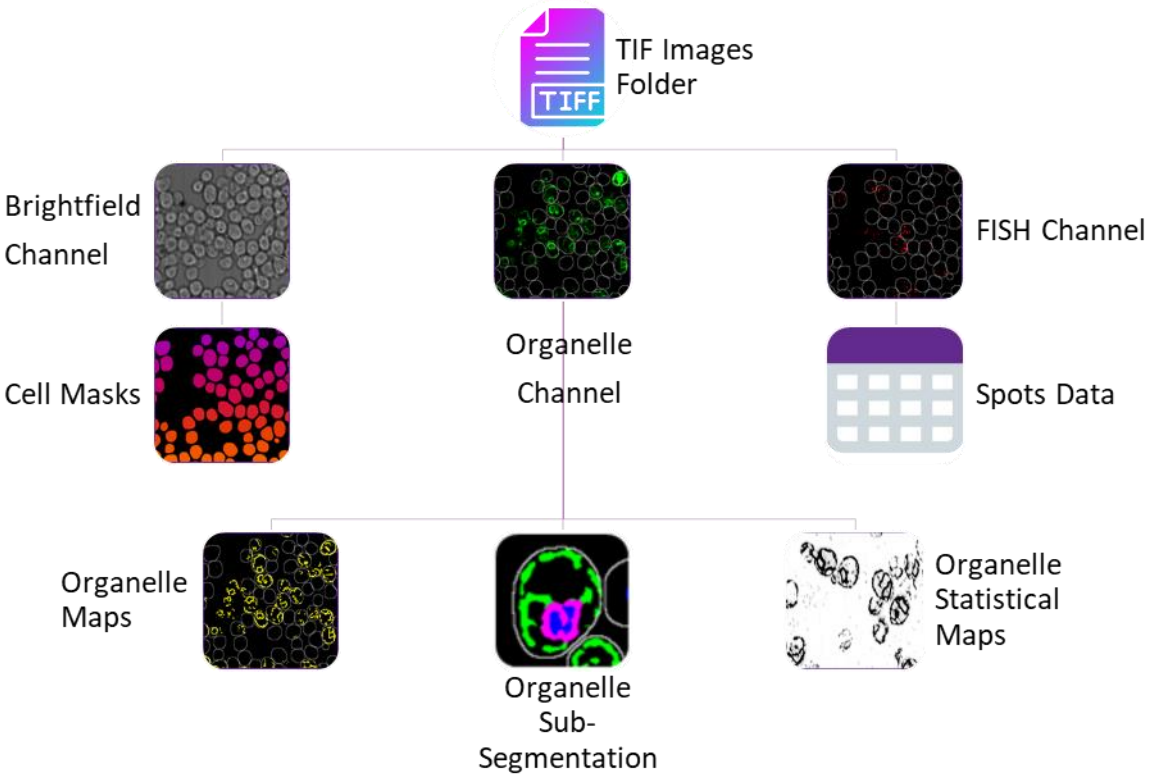
1. Requierments and Folder Structure

By and large, EASI-ORC can work on any modern system. It will work faster, the better the CPU, but it can complete its work without needing state of the art hardware. However, it is limited by RAM size. The higher the image resolution, the more RAM is required. The system was tested using 2048x2048 images and for that resolution we would recommend at least 32 GB of RAM. Anything less can result in some of the steps (specifically cell and organelle segmenation) crashing midway.

Another requierment concerns the way you capture your images. smFISH signals are identifed using data from multiple planes of capture. As such, making sure your fields are correctly focused and centered is important to maximize your yield of usable, positive cells. Specifically, signals should be least focused on the edge z-planes.

Also, EASI-ORC relies on a specific folder structure. Every image of each channel captured must be saved in a seperate folder, with nothing but images in it. The images should be in the TIF format. In the next segment we include a script that can convert the images to TIF and order them in folders according by channel. Lastly, high resolution images require a large amount of space. Before running EASI-ORC, make sure you have available space that is at least 3 times that of your raw images. After completing the scripts run and image analysis, you can remove any images you wish.

The final folder structure created by EASI-ORC is presented in the figure below.



2. How to use a Fiji Script:

Each script is a single file. To use it, simply open Fiji and drag the file to the Fiji menu widow. A new window will open, containing the script's code. Press the 'Start' button (below the code section of the window) and the script will run. Any inputs needed will appear in pop-up windows.

3. Convert Raw Images to TIF and Arrange in Folders:

Most microscope software outputs images in a format unique to the manufacturer. These file types (bio-formats) can be less accessible for many programs or processes. Accordingly, we must make sure all files are in the correct format.

This script assumes each channel and scene are saved in a seperate image file (**no multi-channel images**). File names must be unique for each channel. It can have the name of the marker (GFP, DAPI, mCherry, etc.) or an arbitrary name (c1, c2, etc.). The user must know these names and input them (**case sensetive**) once he runs the script.

Important – if your files are formatted differently (in a multi channel image, for example) this script won't work. You must first convert each channel to a seperate image. You must have each channel images in a seperate folder, in TIF format, before using the EASI-ORC scripts.

1) The user will be asked to input the directory containing the images and their file type (usually appears at the end of the file name, after a dot; tested with stk files).
2) The user will be asked to input the number of channels, as well as each channel's name. Channel names must be the same as they appear in the corrsponding file's name (**case sensative**) and they must be unique per channel.
3) Each raw image will be opened seqeuncially, and saved in its proper folder as a tiff file, with its original name.
4) The TIF images will be saved in a new folder located in the same location as the raw images. The main folder is called 'Tiff Images' and sub-folders for each of the channels will be created in it.

4. Use YeastMate to Segment Yeast Cells in an Image:

One of the first challenges in image analysis is identifying and defining each cell in the image. When it comes to images of yeast cells, YeastMate is a Fiji plugin that provides fast and trustworthy results, using a pre-trained, machine learning approach. The script uses the apps default settings, with a change to the threshold value to 0.6, to maximise the number identified cells.

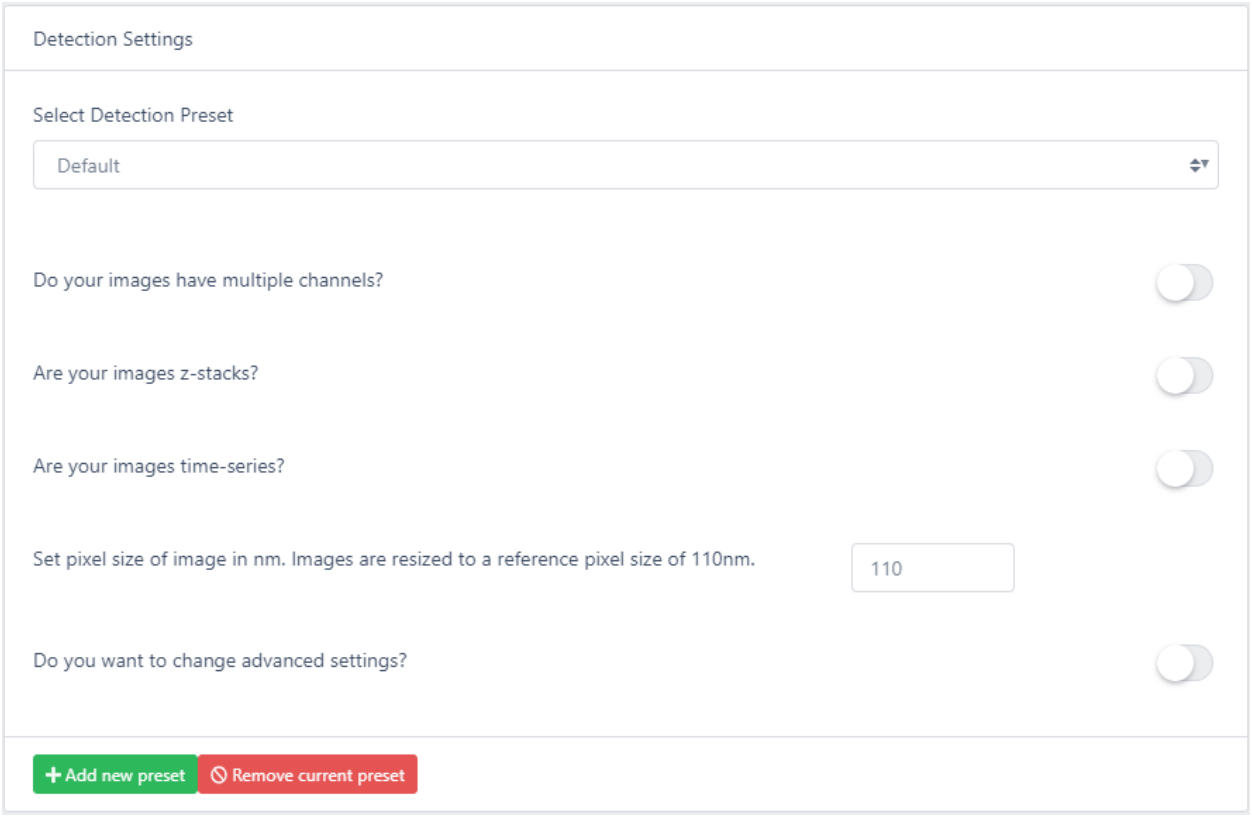
Crucially, to use the YeastMate plugin, you must install its standalone app, and use it to start its backends. Failing that, the script described here will not work. The plugin itself must also be installed to Fiji. See the introduction for links to download the app and the plugin.

YeastMate works on single plane images. To choose the best plane from a multiplane image, masks of each plane will be created and the one containing the largest amount of cells will be chosen to be outputted as the final image.

In addition, **YeastMate must run in a computer without interruption**. It can take a long time to get a mask image when using high resolution images with many slices. If you have many images to process, take that into consideration. For example, when processing 50 images with 12 slices each, at 2048x2048 resolution, the script needed ~4 hours to complete. Timing will vary greatly according to available computing resources and image size.

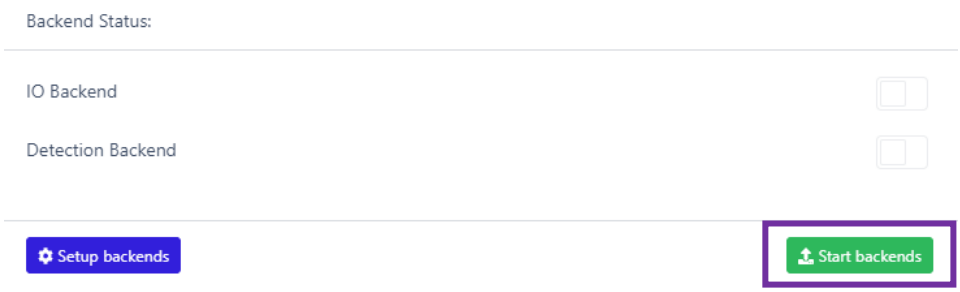
1) First, choose the folder with the cell images in visible light (brightfield or DIC).
2) Open YeastMate. The program will start in 'Start a new job' tab (this and the following stage can be performed before you run the script).

3) Press the ‘Detection’ button under settings. Your settings should look all be turned off, as below:



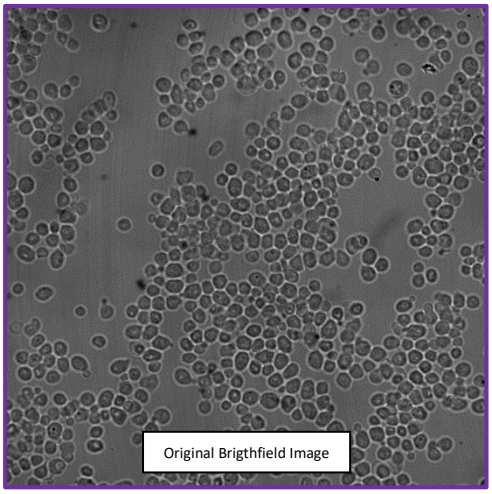
You may change the defualt pixel size according to your specific image paramters, but as long as its not significantly different, YeastMate will work well.

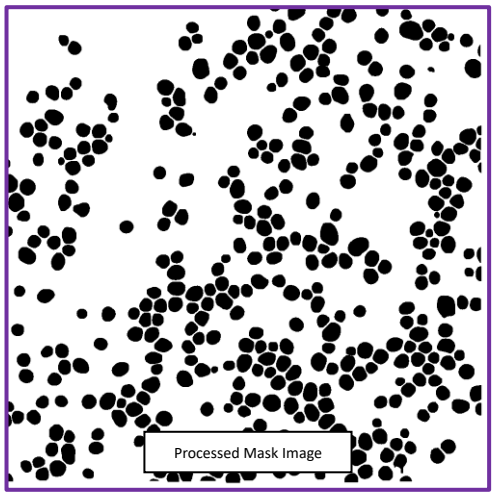
4) Back in the ‘Start new job’ page, press the ‘Start Backends’ Button. Two console windows will open. Both IO Backend and Detection Backend markers will turn green. Once they are green you may press ‘Ok’ on the popup message. The script will start running over the images in the folder.



5) For each image in the folder:
    I. Image is opened.

II. For each slice:
    i. Yeastmate will create a mask image. Masks will be transforemed to binary images (using MinError tresholding, to capture all cells).
    ii. Using Analyze → Analyze Particles, cells are added to the Roi Manager and counted.
III. The plane with the largest number of identifed cells is chosen.
IV. The chosen plane is duplicated: Image → Duplicate n number of original planes.
V. The images are converted to binary images: Process → Binary → Convert to Mask.
VI. A new, stacked binary image is created: Image → Stacks → Images to Stack.
VII. The image is saved in the same folder as the original images, under the 'Cells Masks' folder.
VIII. For each image you will get a binary image, with a white background and the shape of the cells in black. This will be used downstream to identify each cell's location.





Unsolved Bugs or issues:

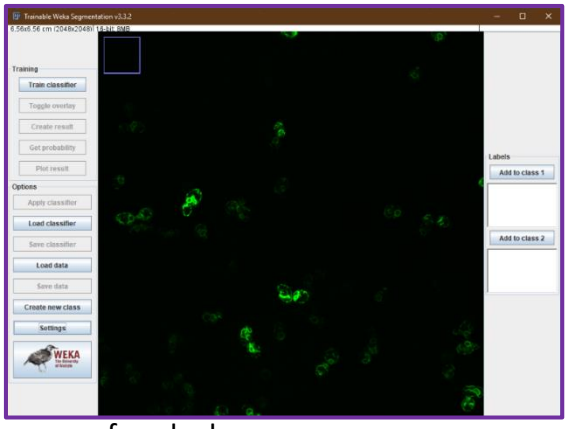
Low quality images, where no cells are identified may occasionally lead to YeastMate crushing for one or more of the slices, resulting in the script crushing as well. In such a case the image cannot be used and must be removed from all folders (meaning, every channel) before running the script again. Before running the script, make sure to delete all the folders added in its disrupted run.

Once the script is done, the Fiji 'Console' window may open, presenting an error. It can be ignored.

5. Use Trainable Weka Segmentation to Train a Model For Sub-cellular Segmentation:

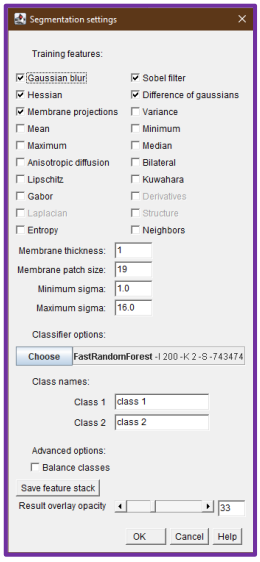
Identfying and defining sub-cellular structures is a more challenging step, the less regular the structure's shape. In case of ER, for example, its shapes varies considerably from cell to cell. The solution from Weka, in the form of a Fiji plugin, allows a user to manually train an algorithm fairly easily, and using the results of the training (a file called a classifier or a model), to perform segmentations of all similar signals in many different images. Here follows a detailed explanation of the training procces.

1) Open an image of the appropriate channel in Fiji.
2) If you wish to save time, make sure to only use a single *z*-plane image for this stage (in a multiple stacks image: Image → Stacks → Stacks to images).



2

3) Press Plugins → Segmentation → Trainable Weka Segmentation. The pictured window will open.
4) To train the classifier, you must mark specific areas of each class (your organelle, background etc.). You may add classes by pressing the 'Create new class' button. Name each class by pressing 'Settings', at the bottom left. The pictured window will open.

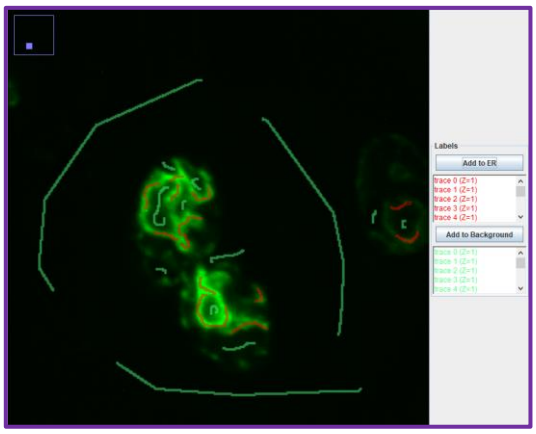


3

5) Change class names to fit your experiment (for example, 'ER' and 'Background').
6) Under the settings window, you may also change different features that can alter the behavior of the classifier. Notice adding more training features will result in longer training and identification times, and require more computational resources, but feel free to experiment.
7) Now you can mark different classes on the image using the mouse. It is recommended to adjust brightness/saturation and zoom in for this stage (adjusting brightness won't affect the classifier).

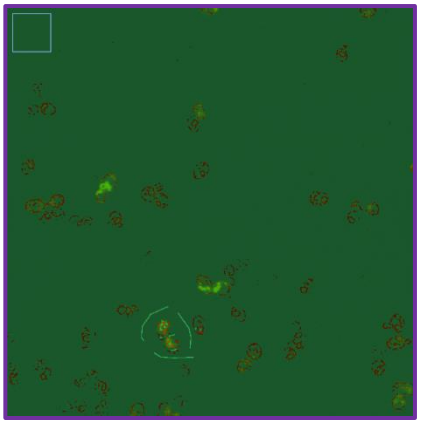
8) Mark a small area of the image each time, and press ‘Add to Class’ button. The markings will change color and be added to the list under the class name (pictured).

7



9) Avoid adding a marking to a class, if the marking is inaccurate. You cannot remove a marking once it’s added. You can start marking without saving, and the previous marking will disappear.
10) After marking and adding a few times for each class (no need to add more than 5-6), press ‘Train classifier’.
11) After training will end, you will be able to see how the different classes are marked on the entire image (pictured).

10



12) Repeat these steps until you’re happy with the segmentation results.
13) Save the classifier by pressing ‘Save classifier’.
14) Test and train your classifier on different images of the same channel/signal. By pressing ‘Get probability’, you can look at a probability map resulting from your classifier. Make sure to look at it next to the original image, and see that the signals match.

6. Perform Sub-cellular Segmentation Using a Trained Model:

The following script allows the use of a trained classifier on multiple images in a user inputted folder.

Script assumes a classifier was already created (as desceribed above).

Script assumes a folder containing only images of a channel corresponding with chosen classifier.

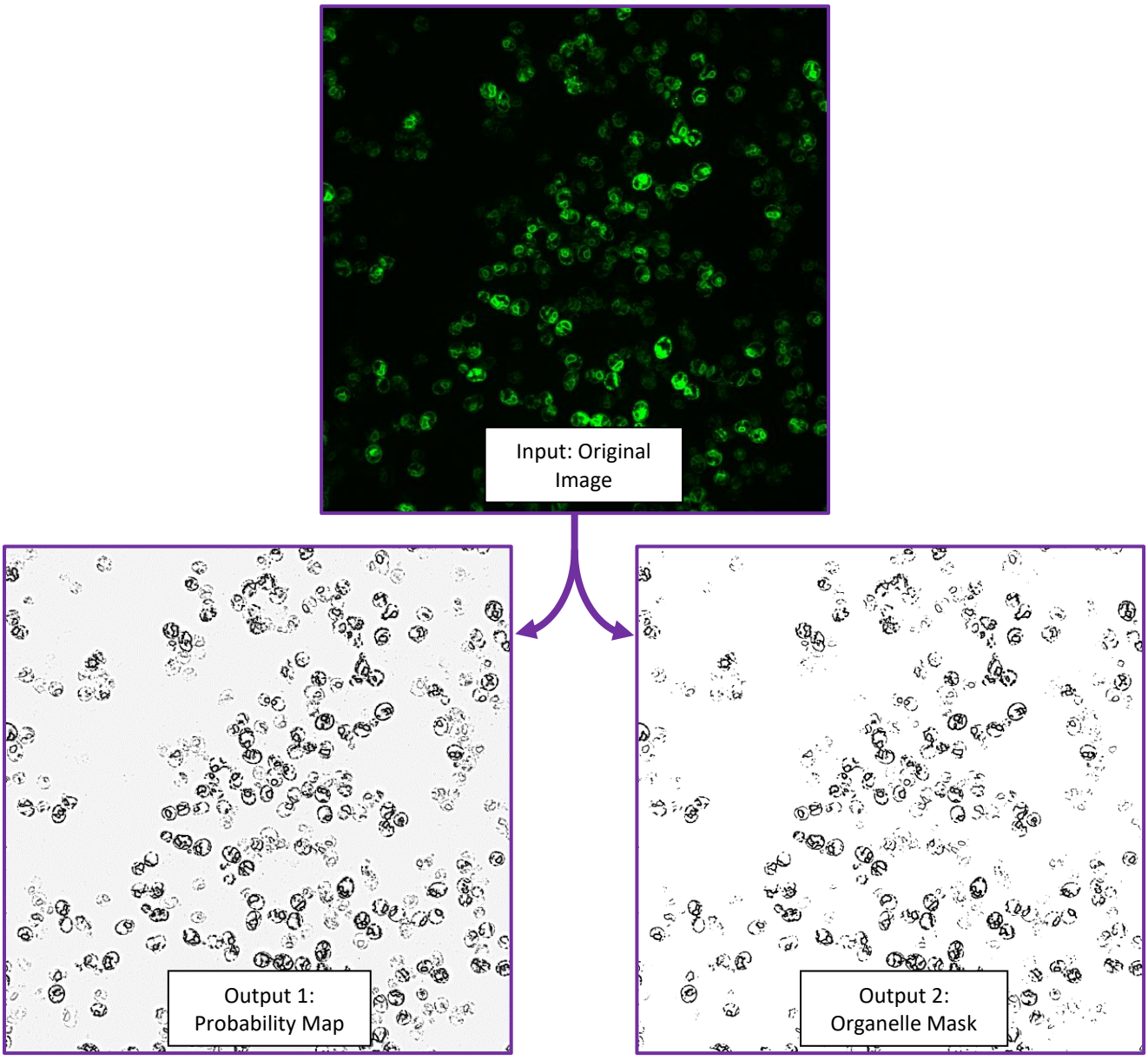
The script saves both the organelle mask and statistical map files. The statistical maps are not used downstream by other script, but can be important for troubleshooting. After going over a few of them to make sure organelle segmentation was performed well, they can be deleted.

This step is very time consuming. When using a classifier for the first time, it is recommended to test it on a few imges (3-4) and examine the resulting masks, to make sure the results are of high quality.

1) The user is asked to input the folder containing images of the channel they wish to segment and the classifier.
2) It creates subfolders to save the probability maps and mask images, in the same folder as the images.
3) For each image:
    I. Each z-plane is saved as a seperate file (in 'seperated_planes' folder).
    II. Go over the seperate *z*-plane images, one at a time and runs a macro that performs the following actions:
        i. Run Trainable Weka Segmentation plugin.
        ii. Load classifier.
        iii. Create probability map.
        iv. Save probability map in subfolder (in 'temp_maps' folder).
        v. Close all.
    III. After all planes from one image have a probability map, open them all.
    IV. Stack to a single image (under the same name + '_map').
    V. Save probability map image with multiple *z*-planes in probability maps folder.
    VI. Create a mask file based on the probability map (using Minimum thresholding) and save it.

VII. Close all.

VIII. Delete unnecessary image files from ‘seperated_planes’ and ‘temp_maps’ folders.



7. Organelle Sub-segmentation:

Yeast ER is usually divided to cortical (cER) and perinuclear (nER) ER. The first is proximal to the nuclear membrane, and the second is distal to it. When checking for colocalization with the ER, it is common to calculate colocalization with both sub-sections separately. EASI-ORC can separate the two sub-sections, by using an image of the nucleus as a proximity marker. The nucleus is a specific case of a proximity marker that allows the sub-segmentation of an organelle. The process can be performed with any other proximity marker you choose, to sub-segment any signal you've previously made a mask for.

This script assumes the Trainable Weka Segmentation process was performed on both organelle and proximity marker signals and masks for each exist. Each must be saved in their own folders (without any subfolders) and the files must be ordered in the same manner. If the files were created using the scripts in this guide, all the conditions will be met.

Important note: In cells or *z*-planes without a proximity marker signal, there will be no way of identifying the sub-sections of the organelle that are close or far from the marker. As such, these cells, in these planes, must be filtered out in the colocalization calculations (the EASI-ORC colocalization script takes that into account).

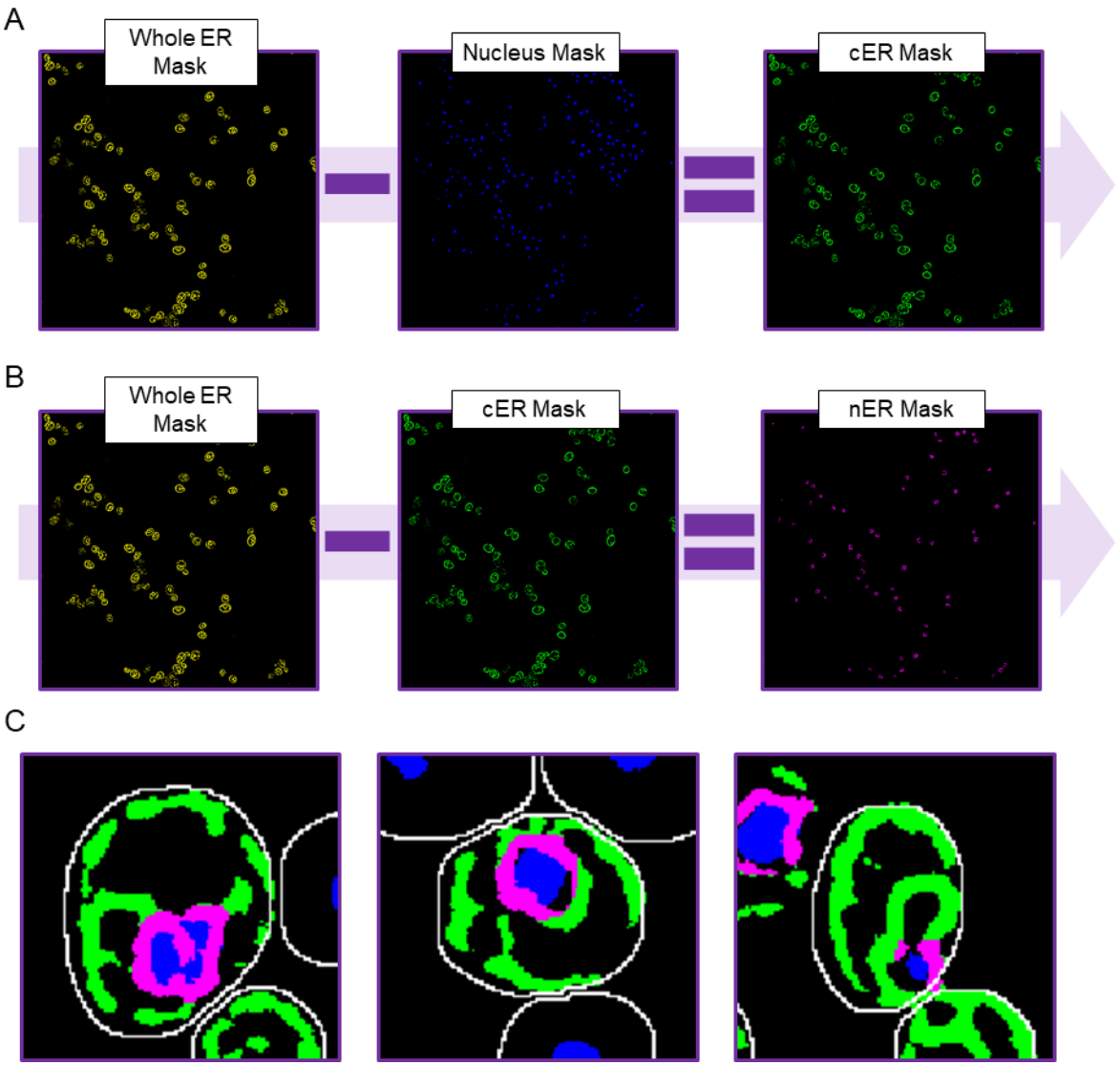
1) The user is asked for the folder of the proximity marker and organelle masks.
2) The user is asked for the number of dilation steps (adding 1 row of pixels near its edges) to be performed on the proximity marker signal.

   ***Note:** It is recommended the user tests this manually with several values, to make sure the proximity marker is of satisfactory size to cover the proximal sub-section (for ER sub-segmentation using a nuclear marker, we found 3 steps works well). Too check for yourself, simply open a proximity marker mask and go to the menu Process → Binary → Dilate. You can then merge the proximity marker image with the organelle image and see if the coverage is sufficient.
3) For each proximity marker image:
   I. Process → Image Calculator… Performs subtraction of the proximity marker mask image from the organelle mask image, outputting and saving a mask image containing only the distal sub-section.

II. Process → Image Calculator… Performs subtraction of the produced distal sub-section mask image form the original organelle image, outputting and saving a mask image containing only the proximal sub-section.

All images are saved in the original organelle mask folder.

The process and the effect of different quality of proximal markers are shown below.



8. Manual run of RS-FISH for threshold identification:

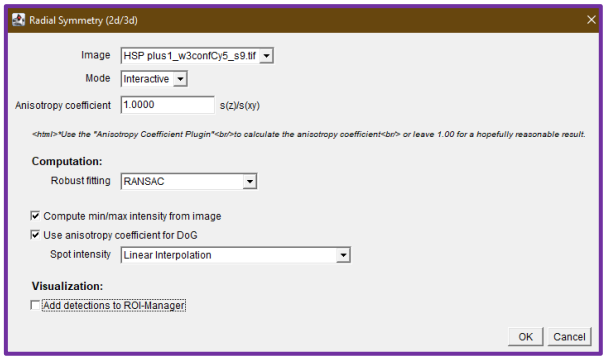
smFISH signals are a relatively simple signal, that can be identified using specific plugins. RS-FISH, is one such plugin. It's fast, accurate and takes into account signal intensity between different cross sections. Crucially, it can be automated using Fiji scripts.

Important: To properly identify FISH spots, RS-FISH requires a signal threshold, which will be different between experimental procedures and different probes. So, before automation, RS-FISH must be used manually on a few images for each signal type you wish to identify.

A step by step explanation is provided here, but to gain a full understanding of the process, it is recommended to read the documentation provided with the plugin. Make sure to follow the link provided in the introduction to install RS-FISH before continuing.

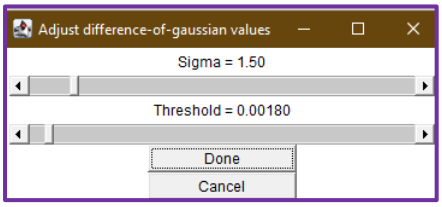
1) Open an mRNA channel image and activate the plugin (Plugins → RS-FISH → RS-FISH).
2) A window will open. Under 'Mode', choose 'Interactive. Under 'Robust fitting', choose RANSAC. Mark both tick boxes and finally, under 'Spot intensity', choose 'Linear Interpolation' (it should look like it does in the image). Click ok.

2



3) Several windows will pop up. Move the additional image window side by side with the original image.
4) If you haven't done so before, change the image's brightness and contrast so you can clearly see signals. Drag and move the square that appeared on your image so it'll mark an area with signals in it. You can also manipulate its size and shape to your liking, though it shouldn't be too large. Zoom in to it on both images.
5) Now, when you move across different slices, you can see the signals are marked by a red circle. For each spot, across all planes, only the most intense is marked.

6) In the ‘Adjust difference-of-gaussian values’ window, you can change the Threshold value. Change it several times and test to see the best value, which allows you to capture as many signals as possible, without marking noise. **Always prefer to miss some signals (a false negetive) than to mark noise as signal (false positive).**



7) Once an appropriate value is found, you may check it by repeating the process on a second and third image with the same signal source.
8) The threshold value can be used on all images with the same signal source (a specific probe or other marker), who were subjected to the same capturing method (same channels, same microscope, same exposure time, etc.).

9. smFISH spots identification using RS-FISH:

Once you have an appropriate threshold, you can use the automation script, to collect data regarding the locations of all spots in your images:

1) The user is asked for the location of the cells masks images directory.
2) The user is asked for the location of the smFISH channel images directory.
3) The user is asked for the spots identification threshold value.
4) Once these are inputted, the script performs the following steps on each image:
    I. Opens cells mask file.
    II. Dilates once to allow identification of signals at the edge of the cell (Process → Binary → Dilate) and watershed to separate close cells (Process → Binary → Watershed).
    III. Selects all cells and adds each to roi manager.
    IV. Opens smFISH image.
    V. Selects all cells location and removes anything outside of them.
    VI. Run RS-FISH, using the provided spots threshold.
    VII. Sort results by '*z*' value (the *z*-slice where it was found).
    VIII. Save results using current image name.

Result tables will be saved as csv files, under the same directory as the smFISH channel images, in a new directory called 'smFISH Spots'.

The output table for each image will include data on each point (from left to right): *x*, *y*, *z* coordinates (width, height and *z*-plane, respectively), time (only relevant for time lapse images), channel (not relevant with single channel images) and intensity.

x	y	z	t	c	intensity
1519.123	65.0544	1.1473	1	1	252.9637
1203.276	48.6669	1.1954	1	1	120.9883
1588.631	85.913	1.2808	1	1	357.8814
211.1326	1167.16	1.3055	1	1	246.1136
582.0259	364.5312	1.4322	1	1	221.7589

10. Colocalization of smFISH signals and organelle:

Once the organelle is well defined in mask files and smFISH spots data is obtained via RS-FISH, colocalization can be examined. First, this script will prompt the user for the locations of the cells mask images, spots csv files created by RS-FISH (if you wish to identify spots using another method, provide a csv file with the same structure as the RS-FISH output (presented above).

Next the script will ask the user whether the organelle was sub-segmented (as described in section 5 of this guide). If it was, choose 'Yes'. If it wasn't, choose 'No'. According to the user's answer, they will be prompted to provide locations of organelle and proximity marker masks (only in case organelle sub-segmentation was performed). Make sure you input the binary mask files' directories, and not the statistical map files' locations.

The user will be asked where they want to save the results tables and lastly, for the value of the smFISH signal diameter, in pixels. We recommend using 1, as this will only look for signals 1 pixel away from the spot. If you wish to choose a different value, you can examine several images closely, by zooming into several isolated signals and count the number of pixel making their diameter. Colocalization will be measured according to the diameter chosen. A large diameter may mark signals far from the organelle as colocalized, and a small one will result in less signals identified as associated with the organelle. Take that into consideration when choosing this value. Make sure to use the same signal diameter when examining the same signal source (the same FISH probe, for example).

This script is quite complex, so a step-by-step description of its workings won't be provided. The general description of the script's process is as follows:

After the user inputs the aforementioned values and folders, the script goes over each image, each plane within that image, each cell and each mRNA. Each mRNA is assigned to its specific cell, as well as given a colocalization assignment to the organelle. Colocalization is defined by having **more than 0** organelle signal at a distance smaller than the diameter provided by the user. In case both sub-section signals are present, the one with a greater total signal (i.e., more ER signal pixels) is chosen. In case of a tie between the signals, the check will be repeated sequentially, with bigger diameters (starting from 1), until the tie is broken. The check is done by creating a circle with the provided radius around the smFISH spot's maxima and measuring the amount of

organelle signal inside that circle (using basic ImageJ commands). Note that since RS-FISH uses data from multiple slices to identify a spot's location, a spot's '*z*' value can be smaller than 0 or bigger than the actual number of cross sections in the image (meaning, it's located at a theoretical point, beyond the limits of the image). In these cases, the spot will be ignored, since there is no organelle data for that location. The number of spots of this nature, if any exist, is very small. In addition, the *z*-plane chosen for colocalization measurement will be the nearest one to each spot.

The script filters out cells according to a few features: edge cells (containing pixels found in the 2% nearest to the edges), cells with less than 5% relative area of proximity marker signal, per plane **(if you wish to change this value, you can do so by changing the 'dapi_cutoff' variable in the script, to another value between 0 and 1)**.

For each image, a CSV file is created in the folder the user inputted. The results will be saved in the following format: Cell #, total number of mRNAs in cell, number of mRNAs of each localization type (organelle near/far, non-organelle), a string with coordinates for each spot within that cell (X, Y values, plane, spot intensity, localization determined) and the organelle coverage value for each z-plane in the image.

Cell #	Total mRNA per Cell	Total Colocolized With Organelle Near	Total Colocolized With Organelle Far	Total Not Colocolized with Organelle	Spots Coordinates Intensity and Colocalization (Far, Near or Not Colocolized)	Organelle Signal Size Cross Section 1	Organelle Signal Size Cross Section n
85	1	0	0	1	[[199.4,19.8,558.8,NC],]	0.22461	0.14793
77	2	0	2	0	[[860.2,1.14,1,2.2,Far], [75.1,18.74,5,36.8,Far],]	0	0
36	3	1	1	1	[[10.3,92.5,1,24.3,Near] ,[12.35,92.5,2,22.1,NC], [11.16,95.1,6,14.7,Far],]	42.37867	51.78698

These files can be analyzed by any means you wish, but we recommend the use of our analysis tool-set. A user guide for it can be found in the next section. If you choose to analyze the results yourself, please take note the spots coordinates are written in a list syntax (for readability) but it is of the string data type. The first part of the data analysis tools we developed will save these data as a list, making further analysis easier.

Crucial notes:

The script must run on folders containing only the images described above and no other file or folder. Using the previous scripts in the guide to create these images will make sure all files are saved and ordered appropriately. Lastly, the script cannot run in batch mode. Meaning, it must open each image it uses and should not be interrupted while it works. **It is recommended you do not use the computer while it runs** (it's fairly quick, but if you have many images, try to run it on 2-3 of them to evaluate how long you need to leave your computer).

Unresolved Bugs or issues:

At times, an empty results table window remains open after the script is done. It can be ignored and closed.

The log will be filled with lines only containing '1'. This is an output of the files deletion function, and can be ignored.

11. Colocalization data analysis tool:

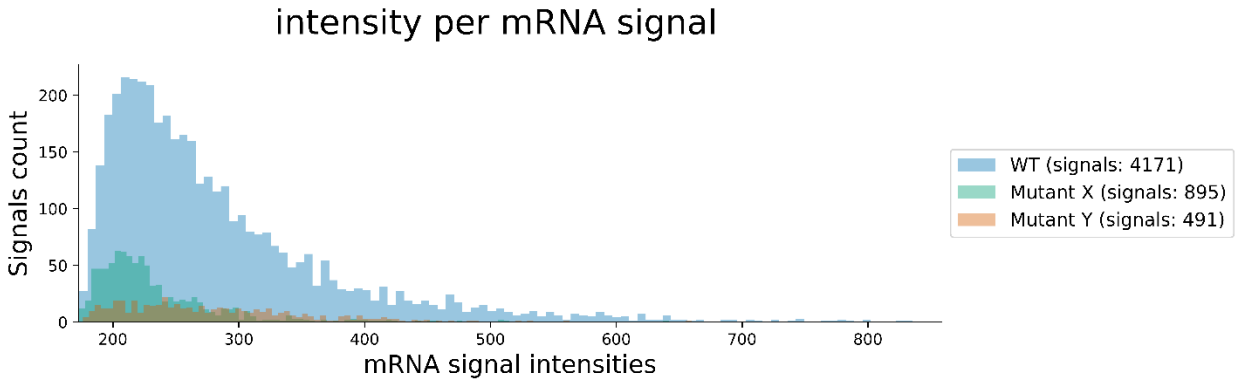
To allow easy analysis of the output files, we developed and provided several tools written in python, which can be easily used online, via Google Colab. The Colab notebook also includes directions for each step. Here, we will explain what the analysis tool provides. Specific instruction can be found in the analysis tool notebook itself. For every plot, the user can manipulate header names, axis names, styling, and font sizes of different sections within the graph. They may decide on the quality of the saved image (by DPI) and the image format.

The analysis tool can be divided into four sections:

1) Table consolidation.
2) Data Filtering.
3) Data Visualization.
4) Statistical Analysis.

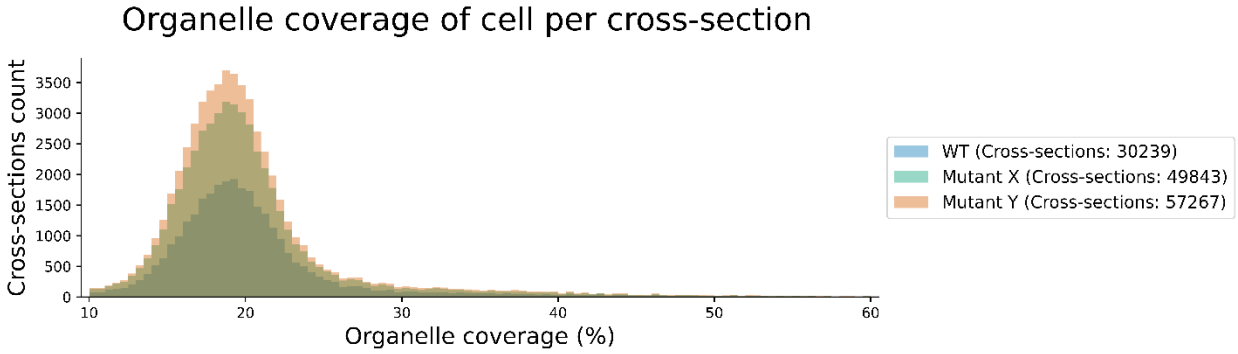
Section 1 asks the user how many different treatments/strains are present in their assay. Accordingly, it will prompt the user to upload the files for each treatment and name each one. It will then create a consolidated table for each treatment, which can be used in the next sections, as well as downloaded by the user.

Section 2 provides several filtering steps, presented visually and updated as you input new values for them. The filters must be performed in the order in which they appear (if you change one, you must also change the next ones). First, a filter of smFISH spots by spot intensity. Second, a filter the removes cross sections with organelle signal which is too great or too small (no organelle signal will lead to non-colocalized designation to all FISH spots, while too much signal will likely result in bias to designate all signals as localized). Lastly, you may filter the data by the number of FISH signals per cell. This section also creates a new table for each treatment, of the data with the filters applied to it. This too, can be downloaded by the user.



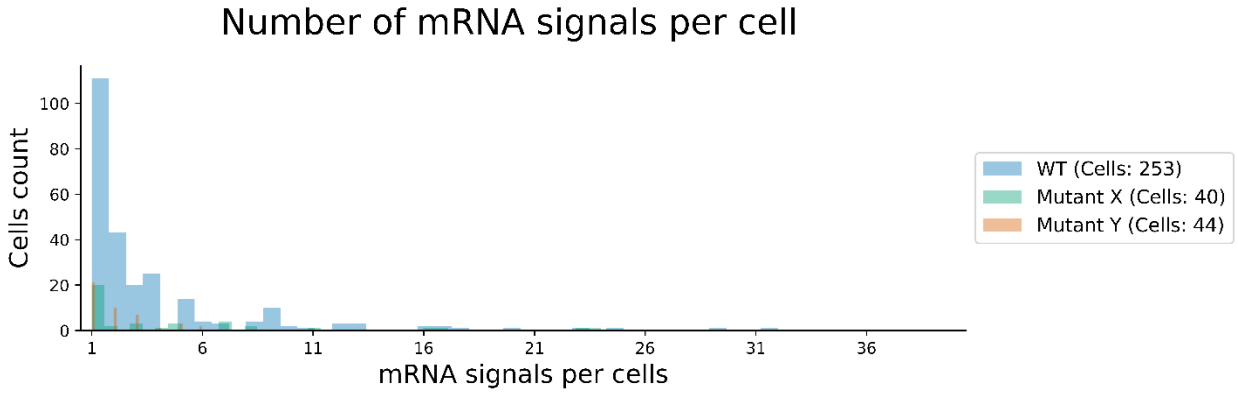
Total signals presented : 5557
Total signals omitted: 27

Filter 1 – Intensity per mRNA signal



Total cross-sections presented : 137349
Total cross-sections filterd: 91051

Filter 2 – Organelle coverage of cell per cross-section

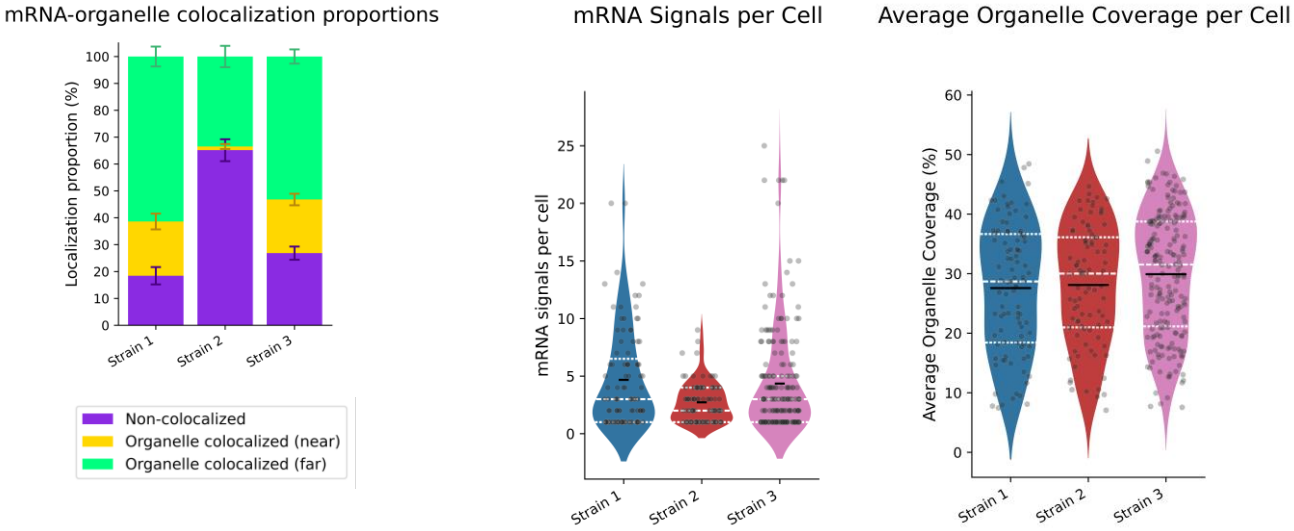


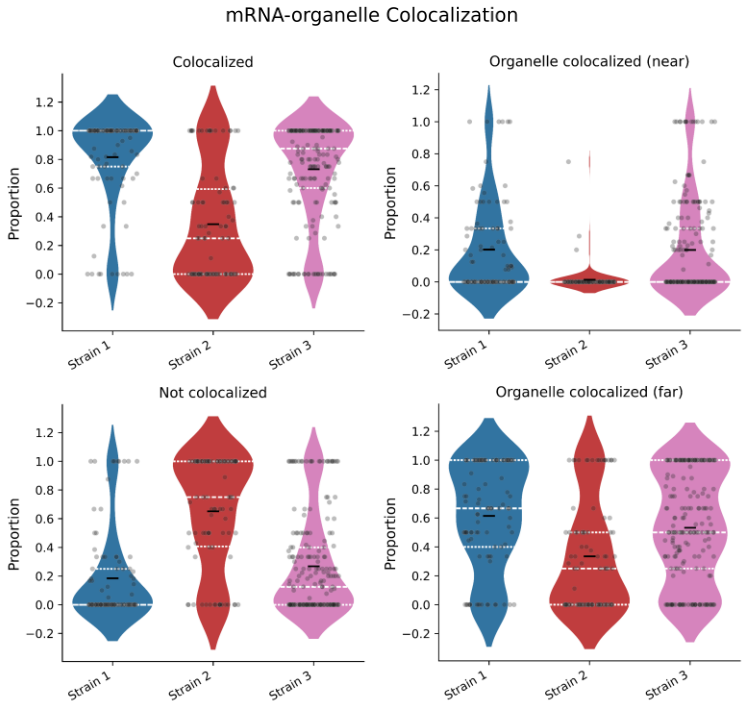
Total cells presented : 337
Total cells filterd: 3453

Filter 3 – Number of mRNA signals per cell

Section 3 provides several graphs for colocalization analysis: A proportion graph presenting organelle localization (or the near and far sub-segments) percentages, violin graphs of each

colocalization sub-population (proximal, distal, non-colocalized and total colocalized), and violin graph presenting the number of mRNA signals per cell, per treatment and average organelle coverage (calculated only from the *z*-planes remaining after filtering for organelle coverage).



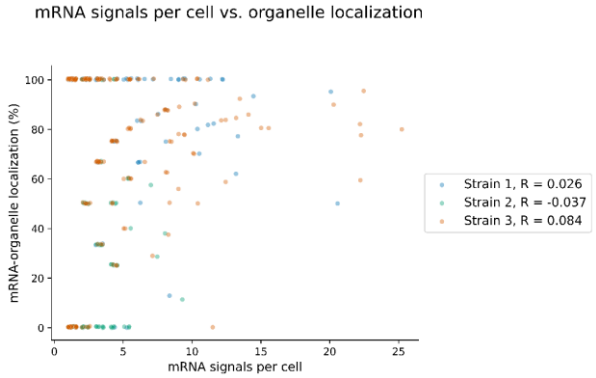


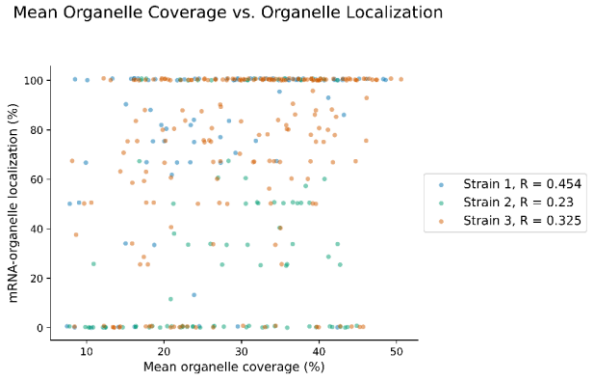
Graphs outputed by the analysis code.

In addition, two graphs describing the correlation between mRNA-organelle colocalization proportion and either number of mRNA signals per cell or average organelle coverage (the

average is calculated between all *z*-planes that were not filtered out). These are useful data, to make sure colocalization doesn't stem from a technical bias.

mRNA-organelle Colocalization correlation plots





Finally, section 4 provides a table with statistical tests performed for and between each treatment. Each treatment's data set is compared via t-test to every other treatment, to provide statistical significance of differences between each treatment.

Every graph and table produced by the analysis tool is available for download individually or together, in a zip file.

Example of Statistical Summary Table

Feature	Strain 1	Strain 2	Strain 3
Number of cells before filtering	1164	690	785
Number of cells after filtering	179	90	99
Percentage of positive cells	15.378007	13.043478	12.611465
mean proportion of non-organelle localization	0.3424406	0.650754	0.3331566
standard deviation proportion of non-organelle localization	0.3311617	0.3849421	0.303338
standard error proportion of non-organelle localization	0.0248216	0.0408038	0.0306418
independent t test with Strain 1 (non-organelle localization) p-Value	1	1.29E-09	0.8140946
independent t test with Strain 2 (non-organelle localization) p-Value	1.29E-09	1	3.70E-09
independent t test with Strain 3 (non-organelle localization) p-Value	0.8140946	3.70E-09	1
mean proportion of proximal organelle localization	0.109664	0.0137302	0.1941591
standard deviation proportion of proximal organelle localization	0.1912716	0.0860985	0.2813304
standard error proportion of proximal organelle localization	0.0143364	0.0091264	0.0284187
independent t test with Strain 1 (proximal) p-Value	1	4.25E-08	0.0088044
independent t test with Strain 2 (proximal) p-Value	4.25E-08	1	1.80E-08
independent t test with Strain 3 (proximal) p-Value	0.0088044	1.80E-08	1
mean proportion of distal organelle localization	0.5478953	0.3355159	0.4726843
standard deviation proportion of distal organelle localization	0.3397583	0.3771238	0.3316722
standard error proportion of distal organelle localization	0.025466	0.039975	0.033504
independent t test with Strain 1 (distal) p-Value	1	1.40E-05	0.075378
independent t test with Strain 2 (distal) p-Value	1.40E-05	1	0.0092901
independent t test with Strain 3 (distal) p-Value	0.075378	0.0092901	1
mean number of mRNA per cell	9.7988827	2.7444444	10.808081
median number of mRNA per cell	6	2	6
standard error number of mRNA per cell	0.7717869	0.189329	1.2844853
independent t test with Strain 1 (Total mRNA per Cell) p-Value	1	4.06E-16	0.5015701
independent t test with Strain 2 (Total mRNA per Cell) p-Value	4.06E-16	1	1.15E-08
independent t test with Strain 3 (Total mRNA per Cell) p-Value	0.5015701	1.15E-08	1
mean of organelle coverages	26.090056	28.06861	25.65967
standard error of organelle coverages	0.7614677	1.0505075	0.89201
independent t test with Strain 1 (filtered organelle coverages mean) p-Value	1	0.1312486	0.7245669
independent t test with Strain 2 (filtered organelle coverages mean) p-Value	0.1312486	1	0.0804421
independent t test with Strain 3 (filtered organelle coverages mean) p-Value	0.7245669	0.0804421	1
Pearson correlation mean organelle coverages vs organelle localizations	0.3475328	0.2341923	0.4330672
Correlation pvalue mean organelle coverages vs organelle localizations	1.87E-06	0.0263049	7.56E-06
Pearson correlation #mRNA signals in cell vs organelle localization	0.1075409	-0.036583	0.0379133
Correlation pvalue #mRNA signals in cell vs organelle localization	0.1518925	0.7321083	0.7094643