

MIAP Workshop 04/09/2018

The presentation and a PDF version of the workshop are available at <https://downloads.openmicroscopy.org/presentations/2018/MIAP-Basel/>

Software versions used for this workshop:

- OMERO: 5.4.7
- OMERO.iviewer: 0.5.0
- OMERO.figure: 4.0.1
- OMERO.parade: 0.1.1
- OMERO.FPBioimage: 0.2.0
- OMERO training scripts: 0.3.0
- OMERO training notebooks: 0.3.0
- Bio-Formats: 5.9.1
- Fiji/ImageJ: 2.0.0-rc-68/1.52e
- TrackMate: 3.8.0
- Orbit: 3.05
- CellProfiler: 3.1.3

Content

- Introductory talk about OMERO
- OMERO.insight import for another user
- CLI Import
- In-place import
- Bulk import
- Render CLI
- Metadata CLI
- Analysis using TrackMate
- Plotting data using R in Jupyter notebook
- Orbit - Manual and Scripted analysis
- Java API - e.g. create map annotations
- CellProfiler in Jupyter notebook
- Python API in Jupyter notebook - simple FRAP analysis
- OMERO.figure JavaScript scripting
- Data Management using CLI

CLI Import

Description

In this first part, we show how to import data for another user into OMERO using various import strategies. The user importing the data needs to have some admin (or restricted-admin) privileges. More information about restricted privileges can be found at

<https://docs.openmicroscopy.org/latest/omero/sysadmins/restricted-admins.html>

For this workshop, we will mainly use the Command Line Interface (CLI) to import and manipulate data.




The import will be done by a facility manager *importer1*.

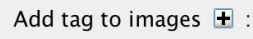

Setup

- The Desktop client OMERO.insight has been installed on the local machine. The installation instructions can be found at <http://help.openmicroscopy.org/getting-started-5.html#installing>.
- Client libraries from the OMERO.server have to be installed on the client to import images using CLI. The installation instructions can be found at <https://docs.openmicroscopy.org/latest/omero/users/cli/installation.html>.
- To use the cli-render plugin, OMERO.py has to be installed. This is included with OMERO.server but if you are not planning to import using CLI, you can **only** install OMERO.py.
- We assume that you have pip already installed.
- To install the cli-render plugin, run:
 - a. `$ pip install omero-cli-render`

Desktop client Import (not covered during this session)

In this example, we show how to import data for another user. A facility manager *importer1* with restricted privileges imports the data for *user-1*.

1. Launch OMERO.insight.
2. In the OMERO login dialog, click the wrench icon  and then add the server address in the dialog i.e. **outreach.openmicroscopy.org**. Click *Apply*.
3. Enter your *Username* and *Password* and click *Login*.
4. Click on the Importer Icon  in the toolbar.
5. Use the *File Chooser* to locate the data to be imported. We will import the DICOM files listed in <https://github.com/ome/training-repos/blob/master/data.md> under “**Lionheart MRI DICOM**”.
6. Select the image data, click on the right Add arrow .

- a. Optional: Create a Project **medical**.
7. (Optional) Go to the *Options* tab
 - a. Click on  to bring the Tag selection dialog.
 - b. Select the tag(s) on the left-hand side.
 - c. Click  to move the tag(s) to the right-hand side.
 - d. Click *Save*.
8. In the Import Location dialog, select:
 - a. the Group
 - b. the User to import data for
 - c. Optionally, select the target Project and/or Dataset
9. Click *Add to the Queue*.
10. Click *Import*.
11. Check that the images have been imported for the specified user.

Advantages:

- Users can validate that import worked.
- Failed imports can be repeated and/or reported to QA etc..
- Users do not have to wait for import to be scheduled.

Limitations

- Installation of a Desktop application to import data.
- The number of files that can be imported at the same time is limited. The default is 2000. The option is configurable client-side. Note that from OMERO version 5.4.8, there will be no limit.
- Import can be slow due to the data transfer to OMERO via the client.
- The person doing the import **must** be in the same group as the user. This is a limitation of the Desktop client.

More information about import options using the Desktop client can be found at <http://help.openmicroscopy.org/importing-data-5.html>.

In-place Import CLI

In this example, we show how to import data for another user and how to avoid data duplication by doing a so-called *in-place* import. A facility manager *importer1* with restricted privileges imports the data for *user-1*. The facility manager has been given the ability to import for others. For this training, the path to the OMERO.server is `/opt/omero/server`. The facility manager is also a UNIX user.

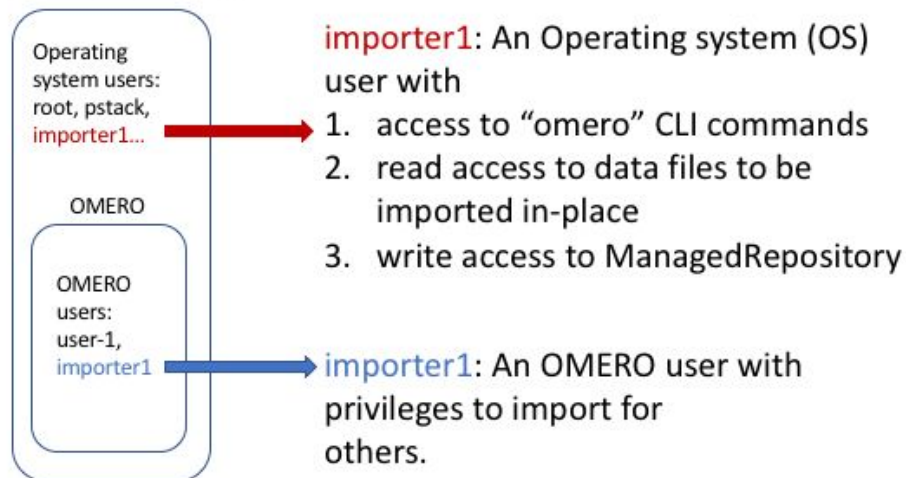
Important:

Someone wanting to perform an in-place import **MUST** have:

- a regular OMERO account

- an OS-account with access to `bin/omero`
- read access to the location of the data
- **write** access to the ManagedRepository or one of its subdirectories. More information about the ManagedRepository can be found at <https://docs.openmicroscopy.org/latest/omero/developers/Server/FS.html>

Unix machine with installed OMERO.server



1. Open a terminal and connect to the server as **importer1** using ssh.
2. Go to /OMERO and look at the subdirectories.
3. Go to /OMERO/in-place-import i.e. `cd /OMERO/in-place-import`.
4. The **importer1** user logs in as *user-1*:
 - a. `$ /opt/omero/server/OMERO.server/bin/omero --sudo importer1 -u user-1 login`
5. Create a Dataset **import_one** as *user-1*:
 - a. `DID=$(/opt/omero/server/OMERO.server/bin/omero obj new Dataset name=import_one)`
6. Import the data “*in-place*” in the newly created Dataset. The option for performing an in-place transfer is `--transfer`:
 - a. `$ /opt/omero/server/OMERO.server/bin/omero import -d $DID --transfer=ln_s /OMERO/in-place-import/svs/77917.svs`
7. Check that the image is successfully imported and open it in OMERO.figure.

More information about import options using the CLI import can be found at <https://docs.openmicroscopy.org/latest/omero/sysadmins/in-place-import.html> and <https://docs.openmicroscopy.org/latest/omero/users/cli/import.html>

Advantages:

- All in-place import scenarios provide non-copying benefit. Data that is too large to exist in multiple places, or which is accessed too frequently in its original form to be renamed, remains where it was originally acquired.
- The person doing the import **does not need to** be in the same group as the user.


Limitations:

- Only available on the OMERO server system itself
- Do not edit or move the files after an in-place import. OMERO may no longer be able to access them if you do.

Bulk Import CLI

In this example, we show how to combine several import strategies using a configuration file. This is a strategy heavily used to import data to <https://idr.openmicroscopy.org/>.

We import one study (set of images) named *idr0021*. For this training, the path to the OMERO.server is `/opt/omero/server`.

1. Open a terminal and connect to the server as *importer1* using ssh.
2. Description of the files used to set up the import, the files are in the directory `idr0021-scripts` under `/OMERO/in-place-import`:
 - a. **idr0021.tsv**: this file has at least two columns where the first column is the name of the target Dataset and the second one is the path to the file to import. We will import the whole study, which consists of ~400 files in 10 Datasets.
 - b. **bulk.yml**: this file defines the various import options: transfer option, checksum algorithm, format of the `.tsv` file, etc. Note that setting the `dry_run` option to true allows to first run an import in `dry_run` mode and copy the output to an external file. This is useful when running an import in parallel.
3. Go to `/OMERO/in-place-import` i.e. `cd /OMERO/in-place-import`.
4. The **importer1** (Facility Manager with ability to import for others) user logs in as *user-1*:
 - a. `$/opt/omero/server/OMERO.server/bin/omero --sudo importer1 -u user-1 login`
5. Import the data using the `--bulk` command:
 - a. `$/opt/omero/server/OMERO.server/bin/omero import --bulk idr0021-scripts/bulk.yml`
6. Go to the webclient during the import process to show the newly created dataset.
7. Select an image.
8. In the right-hand panel, select the *General* tab to validate:
 - a. Click on  to show the import details.
 - b. Validate that In-place import is indicated `Imported with --transfer=In_s`.

Advantages:

- Large amount of data imported using one import command.
- Reproducible import.

Limitations:

- Preparation of the .tsv file.

More information about import options using the CLI import can be found at <https://docs.openmicroscopy.org/latest/omero/users/cli/import.html>.

Render data using CLI

In this section, we will now change the rendering settings of the images imported in OMERO in two different Datasets in a bulk manner. The cli-render plugin is available from <https://pypi.org/project/omero-cli-render/>.

This section assumes that the plugin has already been installed.

1. On your local machine, open a terminal.
2. Go to the place where you can run `bin/omero` commands e.g. in the `OMERO.py-xxx/` directory.
3. Description of the files used to set up the rendering. The files can be found under <https://github.com/ome/training-scripts/blob/v0.1.0/maintenance/scripts/>:
 - a. `renderingdef.yml` and `renderingdef2.yml` are files defining the various rendering parameters, such as the channel color, channel name and minimum and maximum values.
 - b. `renderingMapping.tsv`: this file has two columns, in the left-hand one there are the images to be rendered and in the right-hand side column points to the appropriate `renderingdef.yml` for that image. This file is consumed by the bash script (see below) and thus is not used in the workshop itself.
4. Change the rendering of images in one Dataset:
 - a. Run `bin/omero render set Dataset:$ID local_path/to/renderingdef.yml` where the `$ID` is the ID of the **CDK5RAP2-C** Dataset.
 - b. This will modify the rendering settings of each of the Image included in **CDK5RAP2-C** Dataset.
5. Verify the change in the browser.
6. Change the rendering of images in two Datasets:
 - a. Run `bin/omero render set Dataset:$ID1 Dataset:$ID2 renderingdef2.yml` where the `$ID1` and `$ID2` are the IDs of **CDK5RAP2-C** Dataset and **CENT2** Dataset respectively.
 - b. This will again change the rendering settings of Images in **CDK5RAP2-C** Dataset as well as in **CENT2** Dataset.

7. It is also possible to modify the rendering settings in batch using, for example, a shell script such as https://github.com/ome/training-scripts/blob/v0.3.0/maintenance/scripts/apply_rnd_settings_as.sh
8. which uses `HQL` to find the Images IDs in OMERO and deliver them to the `omero-cli-render` plugin. The script reads the Datasets in which the Images are located in OMERO are listed from a `renderingMapping.tsv` file, such as <https://github.com/ome/training-scripts/blob/v0.3.0/maintenance/preparation/renderingMapping.ts>
9. **(not covered today)** Go to the folder when the script is located.
10. Run the bash script with the default parameters:
 - a. `$ sh apply_rnd_settings.sh`
 - b. The script could be run by a facility manager on behalf of other users.

Metadata Import CLI

In this example, we show how to add Map Annotations to the `idr0021` dataset. This plugin is not yet available on PyPI but it can be used as part of the default installation. We are currently migrating it to its own repository, see <https://github.com/ome/omero-cli-metadata>.

1. Log in to webclient on the target server and create a Project: **in-place-idr0021**.
2. Drag and Drop the ten Datasets of `idr0021` you just imported, into the Project: **in-place-idr0021**. Also, note the ID of the Project: **in-place-idr0021**.
3. On your local machine, open a terminal.
4. Go to the place where you can run `bin/omero` commands e.g. in the `OMERO.py-xxx/` directory.
5. There is a CSV file which is a table with one row per image describing what metadata will be added to each Image in a given Project. The CSV table is converted into an `OMERO.table` (an HDF5 table) using the command 8b below. The `OMERO.table` has an additional column (when compared to the original CSV file), which lists the ID of the Image in OMERO.
6. In the second step, a **bulkmap-config.yml** file specifies the categories of the Map Annotations (Key-Value pairs) such as Gene or Phenotype. In the command 8c below the `OMERO.table` on the Project: **in-place-idr0021** is read and the values are written into Key-Value pairs on each Image. The Key-Value pairs are categorized according to the `bulkmap-config.yml` file.
7. Below is the description of the files used to set up the metadata population:
 - a. **idr0021-annotation.csv** describes the setup of the future `OMERO.table` created in the command 8b listed below.
 - b. **idr0021-bulkmap-config.yml** specifies the categories of the Map Annotations (Key-Value pairs).
8. Run:
 - a. `$ export OMERO_DEV_PLUGINS=1`
 - b. `$ bin/omero metadata populate --report --batch 1000 --file idr0021-subset-annotation.csv Project:<project_id> (enter value)`

```
c. $ bin/omero metadata populate --context bulkmap --cfg
idr0021-bulkmap-config.yml --batch 100 Project:<project_id>
```

Client-side scripting using ImageJ, R, Orbit, CellProfiler

Fiji-TrackMate example

In this example we open all images from a dataset in OMERO in Fiji and use TrackMate plugin of Fiji. For more details about TrackMate, go to <https://imagej.net/TrackMate>. We first find the tracks of the moving objects. Afterwards, these tracks (polygon ROIs), as well as the segmented objects ROIs will be saved to the same images in OMERO. Also, an OMERO.table will be produced from the measurement results of TrackMate and saved as an attachment to the Project containing the images in OMERO. To determine the parameters suitable to analyse the images, we will first go through a manual TrackMate workflow, then later we will use the parameters in a scripted TrackMate workflow.

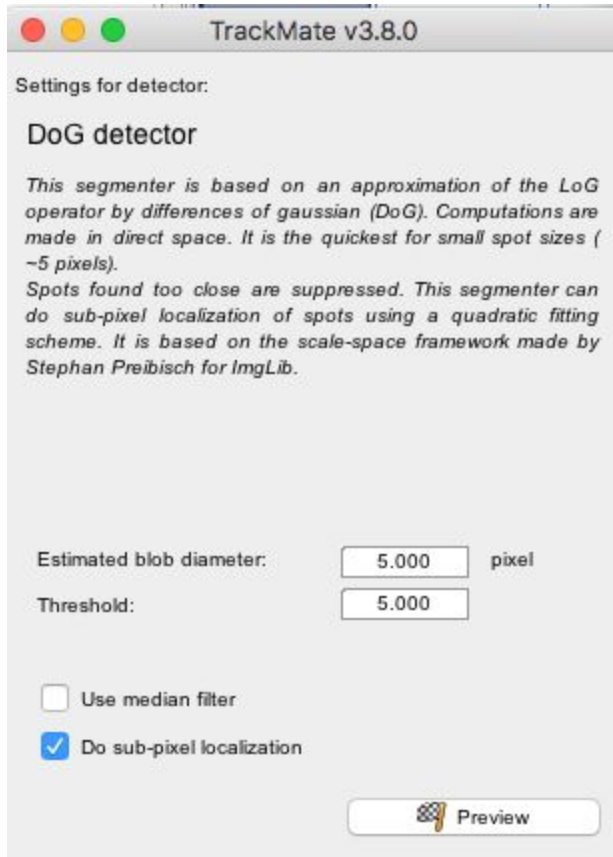
Setup

- Fiji has been installed on the local machine with the OMERO.insight-ij plugin. The installation instructions can be found at <http://help.openmicroscopy.org/imagej.html#plugins>.
- Installing the OMERO plugin also adds the dependencies required to connect to OMERO using the Script Editor of Fiji.

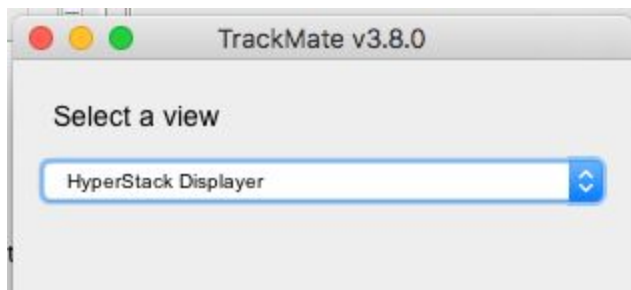
Manual

Each of you can work on your own data. We will use this manual workflow to explain the steps of the script below.

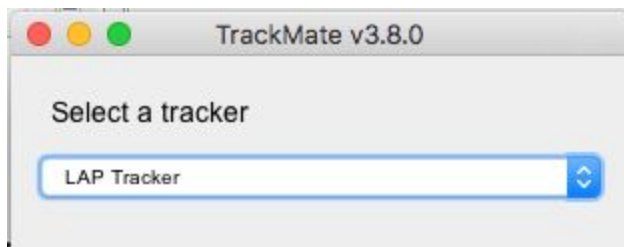
1. Launch Fiji
2. Go to *Plugins > OMERO > Connect to OMERO*
3. Enter the username and password provided for you.
4. Browse the Dataset **artificial-trackmate**
5. Double-click on the **FakeTracks.tif** Image to open it in Fiji. Make sure it is opened using the Hyperstack viewer (This option is dependent on what you did last time when using your Fiji. If you last time used Fiji for running the scripts from Day 1 of this workshop, you should be fine.)
6. Go to *Plugins > Tracking > TrackMate*.
7. Click *Next* in the first dialog that pops up.
8. A new dialog pop ups indicating to select a *detector*. Select the *DoG detector*. Click *Next*.
 - a. Set the *Estimated blob diameter* to 5.0
 - b. Set the *Threshold* to 5.0



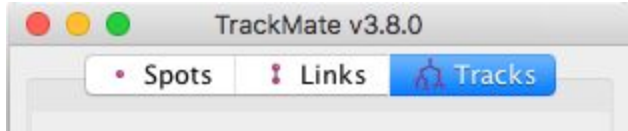
9. Click the *Preview* button to find spots. 4 spots should be found
10. Click the *Next* button few times, until you get to the *Select a view* dialog.
11. Select the *HyperStack Displayer* view



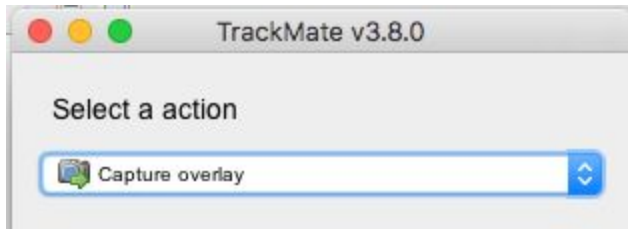
12. Click the *Next* button twice, until you get to the *Select a tracker* window. Select the *LAP Tracker*



13. Click the *Next* button few times (5), until a dialog with three tabs pops up, see screenshot below.
14. Select the *Tracks* tab to display the Tracks.



15. Click Next until you get to *Select a action* dialog.
 - a. Select the option *Capture overlay*
 - b. Click *Execute*.

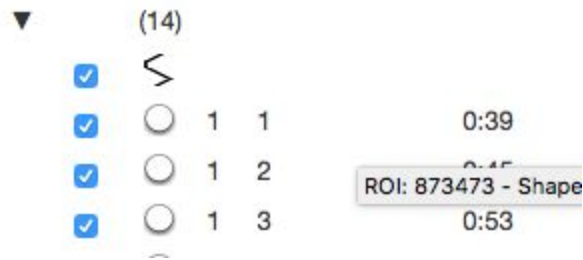


16. Click OK in the following dialog, leaving the defaults (the whole stack will be captured).
17. New image appears. Select it. This new Image can then be imported as on OME-TIFF using the option *Plugins > OMERO > Save Image(s) to OMERO*.

Programmatically

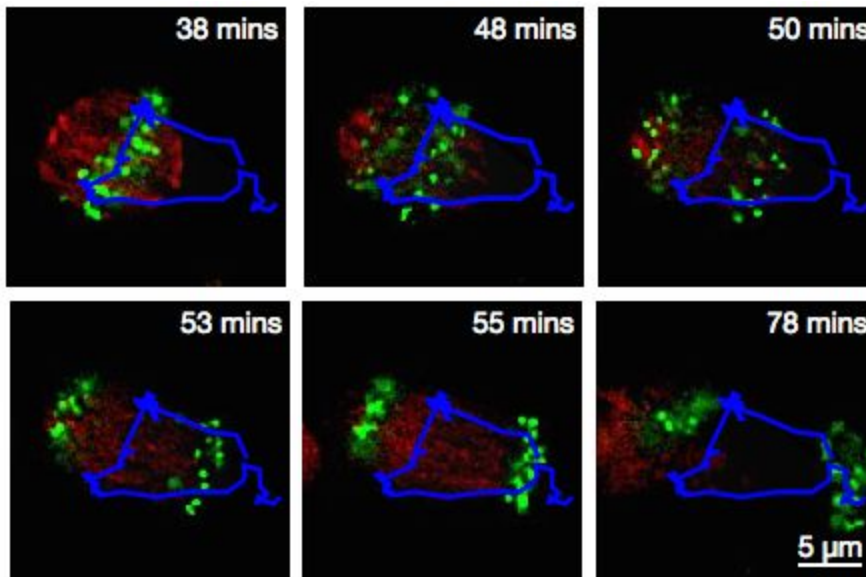
This option saves the ROIs to the server, we do not generate a new Image.

1. Go to <https://outreach.openmicroscopy.org/webclient/>.
2. Enter the username and password provided.
3. Make sure you are working with **your own data**.
4. Create a Project: **trackmate**.
5. Drag and Drop the Dataset **dv-iain-multiTZ** into the Project: **trackmate**. Note the ID of the Project.
6. Launch Fiji, open *File > New > Script...* and select *Python* as language.
7. Copy the content of the script <https://raw.githubusercontent.com/ome/training-scripts/v0.3.0/practical/jython/tracker.jy> (to be replaced after tagging) into the script window of Fiji.
8. Edit the credentials to connect to the server and change the ID of the Project in the script window of Fiji, replacing it with the ID of your Project **trackmate**.
9. Study the script step-by-step.
10. Click *Run*.
11. After the script has finished, go to the images in the webclient and open the second image in OMERO.iviewer.
12. Click on the ROI tab and observe that you now have ROIs under which there are Shapes. Each ROI is a collection of shapes. The ROI corresponds to a Track in Trackmate. There is always one polyline shape in each ROI which represents the track. The other, elliptical shapes in the same ROI represent the tracked spots.



13. Play the timelapse video in OMERO.iviewer.

14. Go to the *Info* tab, and in the *Open with:* line click on *OMERO.figure*. In *OMERO.figure*, add the Tracks and ellipses to the panel by selecting the appropriate ROIs in the *Labels* tab of *OMERO.figure*.



R-idr0021 follow-up example

For this section, we will use a notebook since the setup is a bit more complex. It is necessary to install several dependencies. We use JupyterLab, a dedicated computational environment.

In the following example, we will generate a plot in R and some basic statistics from the output of the *idr0021.groovy* script which was run at Day1 of this workshop on IDR data. The script created an *OMERO.table* which will be consumed by R. R will create a plot based on these data. Also, the statistical tests will be run on the data in R. The plot and the results of the tests will be uploaded back to OMERO.

1. Go to <https://idr-analysis.openmicroscopy.org/training>
2. In the *Files > notebooks > R* folder, select the notebook *idr0021_rois.ipynb*.
3. Select the first Step and click on the *Run* button to execute each step in turn.
4. The first step is to find the **idr0021** Project, the attached table and load the *Dataset name* and corresponding *Bounding box* data directly as R dataframe. (Right-click -> Enable Scrolling for output, to collapse the long output).
5. Then plot the data. First the datasets (Proteins) are ordered from lowest to highest *Bounding box* to match the figure in the paper.

6. Check if there is a significant difference between the Proteins with respect to the Bounding box. This is done as one-way analysis of variance. The value of $2e-16$ indicates that there is a difference. One-way analysis of variance is questionable because it assumes normal distributed data and similar variances between the groups (this is not given, see plot). Therefore also do a 'Mann-Whitney' test to check the differences between each possible pairwise combination of Proteins. -> There is a significant difference between most of the groups, not for some of the groups.
The results show a similar picture as the paper, but the details are different, because we used a simpler approach to estimate the toroid diameter, whereas the authors of the paper used a more accurate method to measure the toroid diameter.
7. Finally the plot and statistical tests results are saved back to OMERO and attached to the project.

Orbit workflow: Manual training and segmentation

Description:

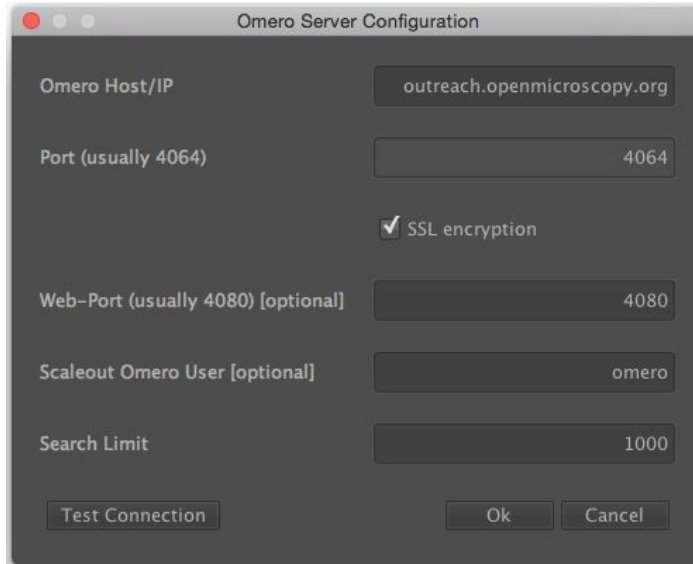
Orbit Image Analysis is a free open-source software with the focus on quantification of big images such as whole slide scans. It offers sophisticated image analysis algorithms. Of those, tissue quantification using machine learning techniques, object / cell segmentation, and object classification are the basic ones. For more details, go to <http://www.orbit.bio/>.

Setup

- Orbit has been installed on the local machine.
1. Launch Orbit, click Yes in the first dialog box:



2. Then enter server details in the next dialog. Only the Host/IP field is essential here and should be set to **outreach.openmicroscopy.org**.

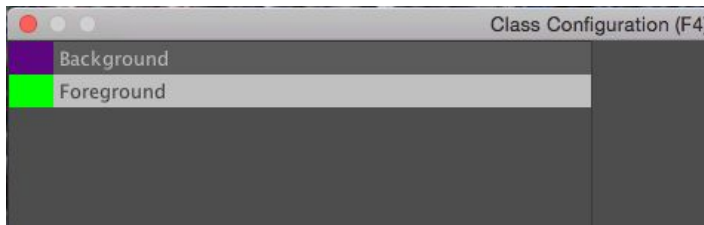


3. Then login to OMERO with the username and password provided:



4. Orbit will show data from OMERO in the left-hand panel. Click *show only my assets* to filter by data you own.
5. Select the group **Lab1**.

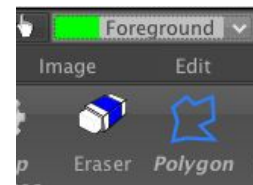
14. You should now have two classes named *Background* and *Foreground*:



15. Click *OK* to close the dialog.

16. Now we will construct the model by defining regions of *Foreground* and *Background* on the Image.

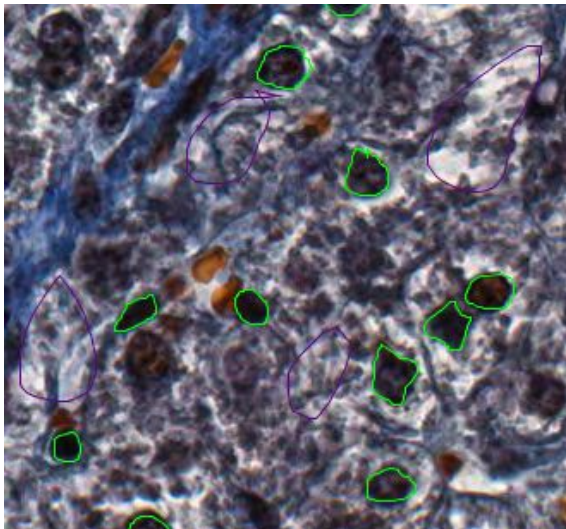
17. Click the *Object Detection* tab, select the *Polygon* tool and choose the *Foreground* class from the



chooser at the top-left of the screen.

18. You can now draw around a number of cell nuclei on the Image. The more accurately you draw and the higher number of objects you define will improve the performance of the segmentation, but about a dozen should be sufficient.

19. Now switch to the *Background* class and draw around several background regions:



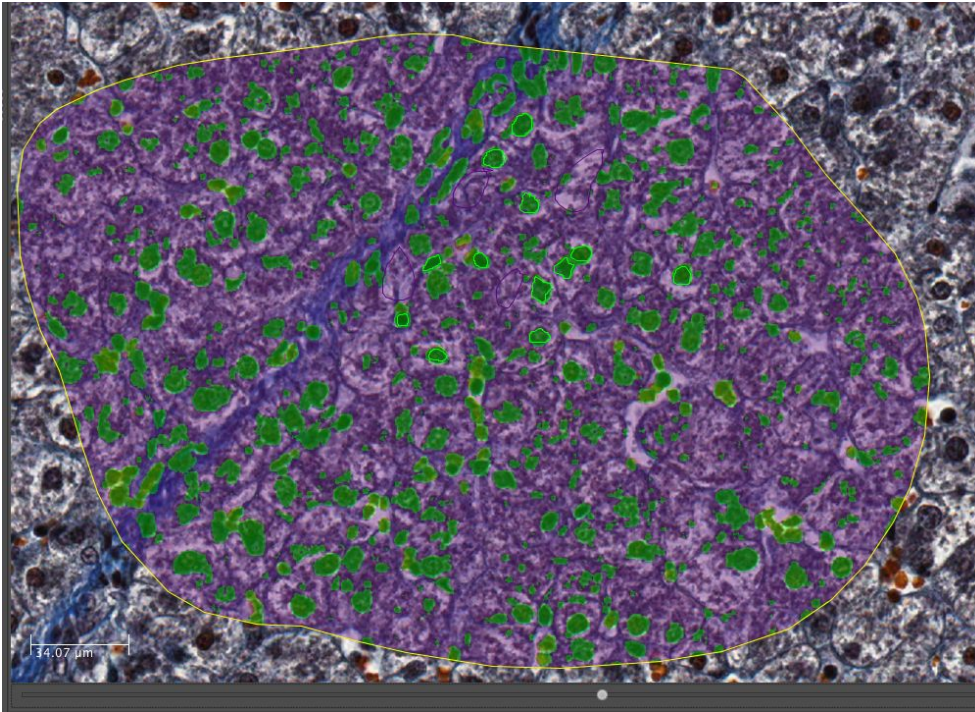
20. We can then train the model by clicking the *Train* button or press F7. You will see a progress bar in the right-hand panel.

21. To see how this model classifies objects within a region, click the *Define ROI* button and draw around a region of the image. Then click *Classify*. If no ROI is drawn, Orbit will attempt to classify



the whole Image which can be very time-consuming.

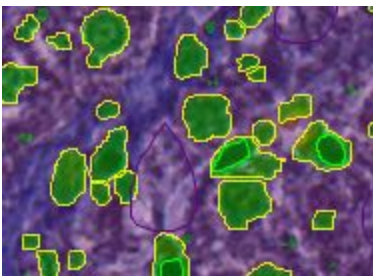
22. Once the classification is complete, a notification window pops up. Close it and view the results on the Image by dragging the slider below the Image to the right:



23. To segment the Image using this classification, click *Set Primary Segmentation Model* and then



Object Segmentation.



24. Click the *Model* tab and *Save Model On Server*, enter a name to save the model to OMERO. Note that you can also use *Save Model as...* to save the model to your local drive.

Orbit workflow: Scripted segmentation and saving to OMERO

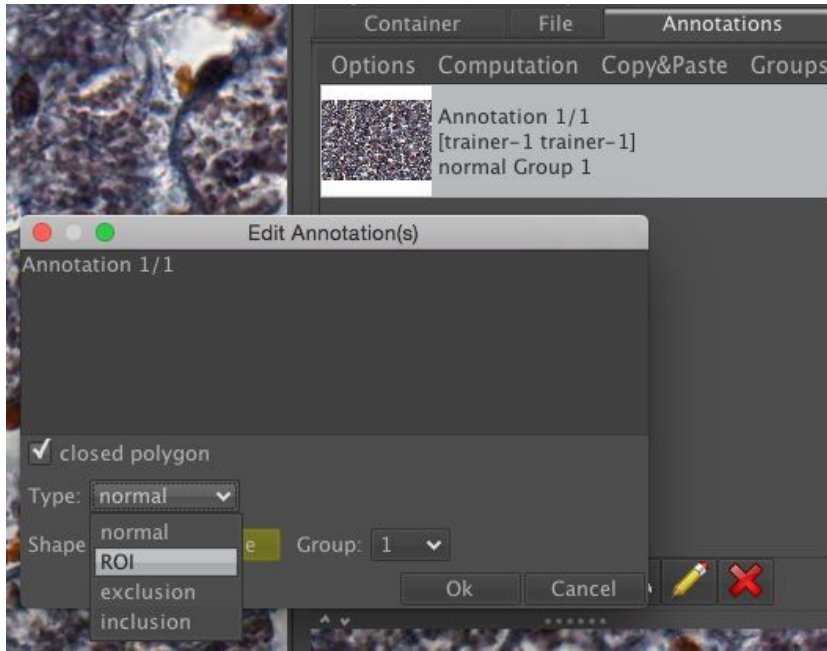
Description: We will use the model created in the last step above to repeat the segmentation, using a script which allows us to save the results back to OMERO. This will use a saved ROI Annotation instead of a temporary ROI as in the manual workflow.

1. Re-open the same image **77928.svs [Series 1]** to clear the ROIs and in the right-hand panel select the *Annotations* tab.

2. Pan the Image to a region you wish to analyse, select the *Add Polygon* button and draw around a region.



3. Select this *Annotation* from the list in the right panel and click *Edit* (pencil icon).
4. In the dialog, set the *Type* to *ROI*.



5. Click *Ok*. This will save the ROI as an annotation on this image in OMERO.
6. Click on *Tools > Script Editor* to open a scripting window.
7. Copy the script from training-scripts:
<https://raw.githubusercontent.com/ome/training-scripts/v0.3.0/practical/orbit/segmentation.groovy>
and replace the existing code in the script window.
8. Update the username and password
9. The script will load the Orbit model and the ROI that we saved to OMERO, segment the image within the ROI and save the segmented shapes as Polygons to OMERO.
10. Click *Run*.
11. When complete, you can use OMERO.iviewer to see the ROIs created in OMERO.

Using the OMERO Java API to create Annotations

1. We can use the Orbit's script editor as a convenient way to explore the OMERO Java API.
2. Orbit interacts with OMERO (as above) using the Java class
<https://github.com/mstritt/image-provider-omero/blob/master/src/main/java/com/actelion/research/orbit/imageprovider/ImageProviderOmero.java>
3. We will use the "**Create a Map Annotation**" example from
<https://docs.openmicroscopy.org/latest/omero/developers/Java.html> to add an annotation to the Image we have just analysed.

4. We will reuse the segmentation script that we have just run, but instead of saving Polygons we will create a Map Annotation to summarise some parameters of the analysis.
5. Find the “**Create a Map Annotation**” section on the OMERO Java documentation page above.
6. Copy this code and go to the Orbit script editor, with the segmentation script.
7. Edit the script to remove the `for` loop that saves Polygons `for (Shape shape: res.shapeList) {`, replacing it with the code we have just copied.
8. We can update the Keys and Values to e.g.:
 - Analysis: Orbit
 - ROIs: Roi count
 - Model: Model name
9. We also need to link the Annotation to an Image instead of a Project, using the `omeroImageId`.
10. For example:

```
57
58 List<NamedValue> result = new ArrayList<NamedValue>();
59 result.add(new NamedValue("Analysis", "Orbit"));
60 result.add(new NamedValue("ROIs", "" + res.shapeList.size()));
61 result.add(new NamedValue("Model", model.getName()));
62 MapAnnotationData data = new MapAnnotationData();
63 data.setContent(result);
64 //Use the following namespace if you want the annotation to be editable
65 //in the webclient and insight
66 data.setNameSpace(MapAnnotationData.NS_CLIENT_CREATED);
67 DataManagerFacility fac = gateway.getFacility(DataManagerFacility.class);
68 fac.attachAnnotation(ctx, data, new ImageData(new ImageI(omeroImageId, false)));
69
```

11. Run the script again and then check in the webclient for the Map Annotation on the Image.

CellProfiler

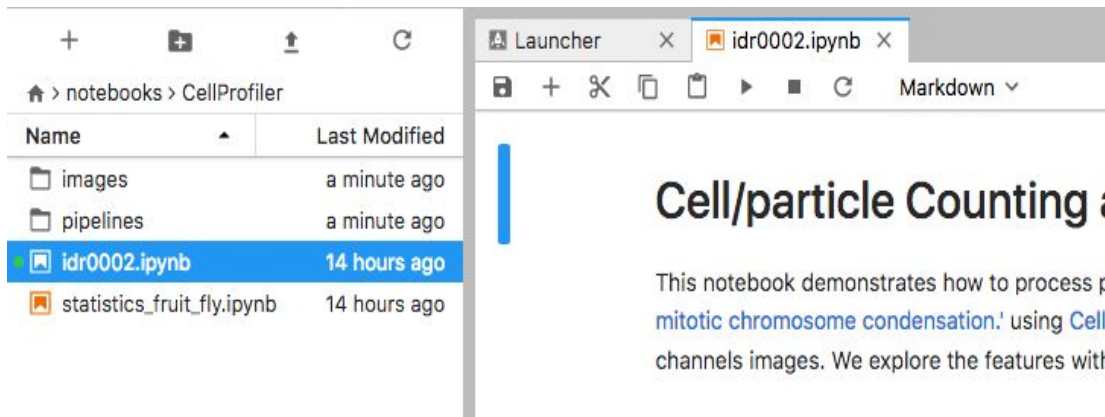
CellProfiler is a free open-source software for quantitative analysis of biological images. For more details, go to <http://cellprofiler.org/>.

Setup:

CellProfiler **version 3.1.3** has been installed in a Jupyter notebook using docker as described at <https://github.com/ome/training-notebooks/releases/tag/v0.3.0>. We will use the Python API only.

1. We will use CellProfiler to analyse a Plate of Images in OMERO. We will run CellProfiler version 3.1.3 “headless” via a Jupyter notebook.
2. We will use the example pipeline from <http://cellprofiler.org/examples/#PercentPositive> to analyse RNAi screening data from IDR <https://idr.openmicroscopy.org/webclient/?show=screen-102>
3. First, open the webclient and find the Plate belonging to *trainer-1* named *plate1_1_013*.

4. Go to <https://idr-analysis.openmicroscopy.org/training> and look under *Notebooks > CellProfiler* for *idr0002.ipynb*.



5. Select the first Step and click on the *Run* button to execute each step in turn.
6. For the connection to OMERO, you will be asked to enter your login details when running the OMERO credentials cell.
7. Select the plate in the webclient, find the Plate ID in the right-hand panel and copy this into the

Fetch OMERO Plate

```
[ ]: # To be modified  
plate_id = 422
```

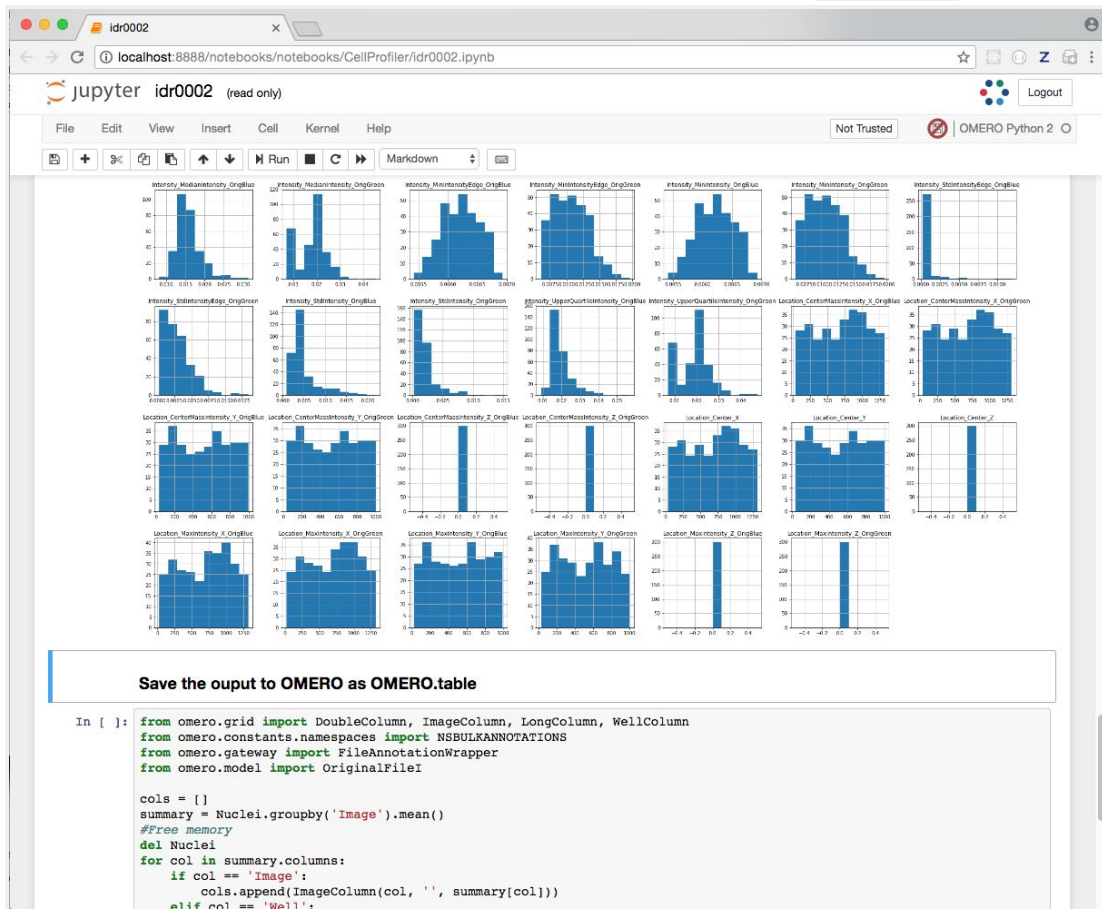
plate_id variable in the next step of the notebook.

8. The following step loads the example pipeline and modifies it to remove the modules that are normally used for loading images from disk.
9. These modules are replaced by the *InjectImage* modules, using numpy planes loaded from OMERO Images.
10. The pipeline is run on a 2-Channel image from each Well in the 96-well plate, generating a CSV file containing rows for different objects identified in the image and columns for various parameters measured.
11. Note that to save time during the workshop, we can run on a subset of all Wells in the plate. To use the first 5 wells, add this line `wells = wells[0:5]` as shown:

```
# Create list from generator  
wells = list(plate.listChildren())  
wells = wells[0:5]  
well_count = len(wells)
```

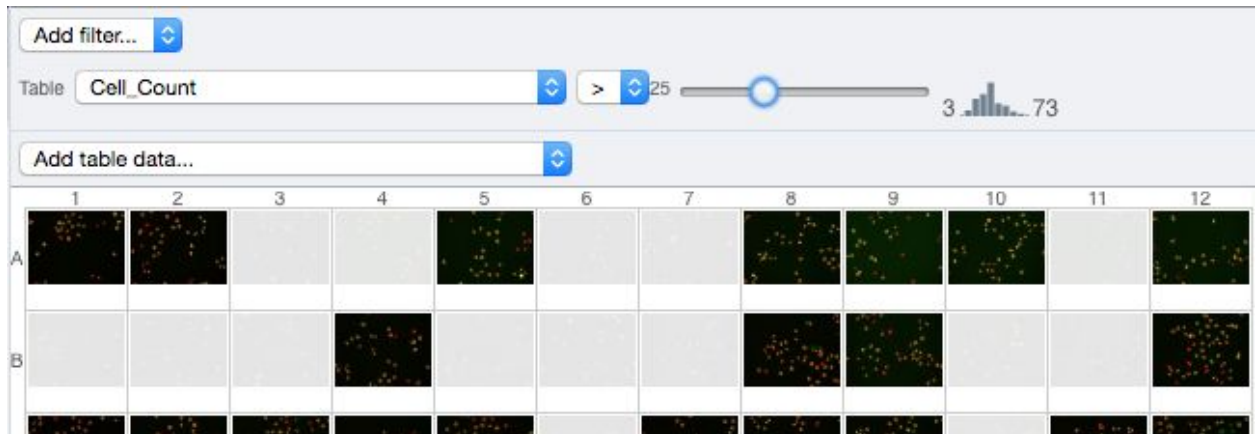
12. CSV files are read into a Dataframe for each image. We add the Image ID and Well ID, as well as the total number of Objects, *Cell_Count*, to each Dataframe.
13. All the Dataframes are then concatenated into a single Dataframe.

14. We can visualise the data as histograms for each column with `df.hist()`



15. Finally, the Dataframe rows are grouped by Image to give an average value per Image of each parameter (column) in the table.
16. This data is saved back to OMERO as an HDF5-based table attached to the Plate, which can be read by other clients.
17. Return to the webclient and select the Plate.
18. Select a Well and open the *Tables* pane in the *General* tab in the right-hand panel. This will show all the CellProfiler values for this Well.
19. In the *Thumbnails* chooser at the top-right of the centre panel, select the *Parade* plugin.
20. At the top-left of the centre panel choose *Add filter... -> Table* to filter Wells by the data from CellProfiler.
21. Change the filter from *ImageNumber* to *Cell_Count* (at the bottom of the list).

22. Now you can use a slider to filter Wells by Cell Count.



OMERO.py scripting - Simple FRAP

Setup

- The following OMERO.scripts have been uploaded to the server for this workshop:
 - https://github.com/ome/training-scripts/blob/v0.3.0/practical/python/server/simple_frap.py
 - https://github.com/ome/training-scripts/blob/v0.3.0/practical/python/server/simple_frap_with_figure.py

We will use Jupyter notebooks to introduce the OMERO Python API. See docs for OMERO Python at <https://docs.openmicroscopy.org/latest/omero/developers/Python.html>.

We will use various code examples from that page to begin a very simple FRAP analysis.

All the examples use the BlitzGateway connection wrapper “conn” to access data in OMERO.

- Go to <https://idr-analysis.openmicroscopy.org/training>
- Load the notebook *Files > notebooks > Python > OMERHelloWorldNotebook.ipynb*. The first step is already loaded for you:

```
from omero.gateway import BlitzGateway
conn = BlitzGateway(raw_input("Username: "), getpass("OMERO Password: "),
host="outreach.openmicroscopy.org", port=4064)
conn.connect()
```

1. Go to webclient, and make sure you are working on your own data.
2. Find the Dataset **FRAP**.
3. Note the ID of the first image in this Dataset.
4. Double-click on the first Image from this Dataset to open it in OMERO.iviewer.
5. Play the timelapse in OMERO.iviewer.
6. Draw in OMERO.iviewer an ellipse ROI on the spot which was bleached.
7. We will use an Ellipse saved on the Image in OMERO.iviewer **without T index set** (spans all T).

8. Click on the lock next to the T Box to unlock _____.
9. Save the ROI.
10. Go back to the Jupyter notebook. First list the ROIs and Shapes saved on the Image, using the `image_id` of the image you just worked with in iviewer.
11. Enter your username and password when requested.
12. You may wish to 'Cut' (remove) the remaining steps from the *HelloWorld* example before adding your own.
13. Add a new step with this code, setting the `image_id` to the Image you just edited.

```
image_id = 123
roi_service = conn.getRoiService()
result = roi_service.findByImage(image_id, None)
shape_id = None
for roi in result.rois:
    for s in roi.copyShapes():
        shape_id = s.id.val
print shape_id
```

14. With the `shape_id`, we can get pixel intensity stats for the first channel across all time points and make a list of mean intensity values. Add a new step with this code:

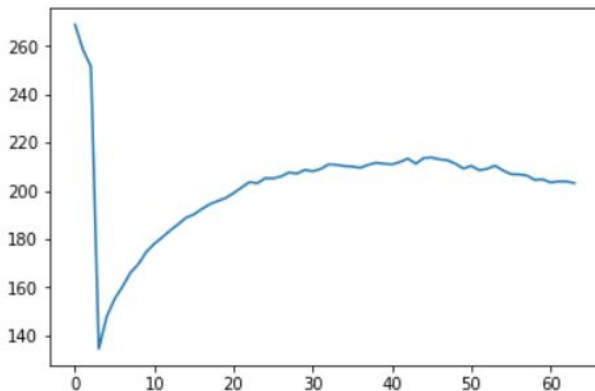
```
the_c = 0
image = conn.getObject('Image', image_id)
size_t = image.getSizeT()
meanvalues = []
for t in range(size_t):
    stats = roi_service.getShapeStatsRestricted([shape_id], 0, t, [the_c])
    meanvalues.append(stats[0].mean[the_c])
print meanvalues
```

15. We can add these values as a Map Annotation (Key-Value pairs) on the image:

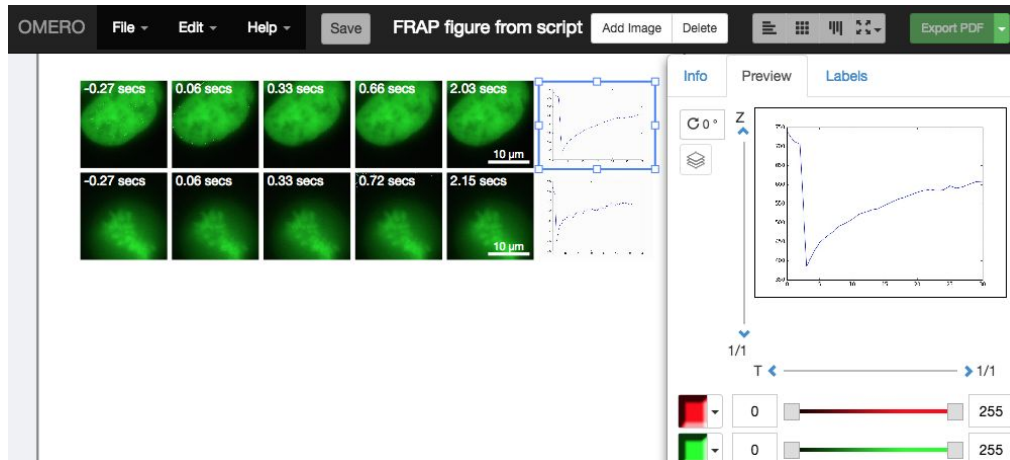
```
import omero
key_value_data = [[str(t), str(meanvalues[t])] for t in range(size_t)]
map_ann = omero.gateway.MapAnnotationWrapper(conn)
namespace = "demo.simple_frap_data"
map_ann.setNs(namespace)
map_ann.setValue(key_value_data)
map_ann.save()
image.linkAnnotation(map_ann)
```


16. Plot values with Matplotlib:

```
import matplotlib
matplotlib.use('Agg')
from matplotlib import pyplot as plt
fig = plt.figure()
plt.subplot(111)
plt.plot(meanvalues)
fig.canvas.draw()
fig.savefig('plot.png')
pil_img = Image.open('plot.png')
pil_img.show()
```



17. Go to the webclient, select the Image you have drawn the ROI on.
18. Note the new Map Annotation (Key-Value pairs).
19. If desired, draw an Ellipse as above (T unset) on additional FRAP Images to analyse.
20. Select the image(s) and Open the script *workshop_scripts > simple frap...*
21. Click on the *View script* link in the bottom-left corner and inspect the script - basically, the script does what you have done in the Jupyter notebook up till now.
22. Run the script, then observe that a new Map Annotation appeared on the image with the FRAP values.
23. Now select the script *workshop_scripts > simple frap with figure...*
24. Inspect this script as well, it is basically an extension of the *simple frap* script with the creation of `OMERO.figure` at the end.
25. Run the script on the image. Note in *Activities*, the ID of the newly created figure.
26. Open `OMERO.figure`, and go to *File > Open...* to open the newly created figure (named **FRAP Figure from script**).



OMERO.figure scripting

We can use JavaScript in the browser console to script changes to a figure. This is an experimental feature and not documented. To see the data model for the current file, go to *File > Export as JSON...* The `figureModel` variable is accessible in the console. We can use AJAX to load JSON data. In this example we will load the FRAP intensities from the Map Annotations on these images.

1. Open the browser console by *right-click > Inspect Element (Firefox)* or *right-click > Inspect (Chrome)* and click on the *Console* tab.
2. Copy the code from https://raw.githubusercontent.com/ome/training-scripts/v0.3.0/practical/javascript/figure_frap_map_annotation_label.js
3. Drag to select the FRAP movie images in the figure.
4. Paste the code into the console. **Do not hit enter yet.**
5. Inspect the code. It will iterate through each of the selected panels, make an AJAX call to load the map annotations with the namespace that we created from FRAP values above.
6. The FRAP values are a list of [key, value] pairs and we can get the value for the current T index of the panel `values[theT][1]` and use this to create a label.
7. Edit the 'position' of the label to 'bottomleft' (can change size and color too if desired) and hit Enter to run the code on selected panels.
8. The labels should be added. Note that you can undo and redo these changes in the UI as normal.
9. Try out other examples in <https://github.com/ome/training-scripts/tree/v0.3.0/practical/javascript>

Administrate Groups and Users

Administrate using the Web Interface

Most of following tasks below can only be done by users with some administrator privileges. The following steps will be done by the trainer. Below are some useful links to know more about permissions and administrator privileges:


- <https://docs.openmicroscopy.org/latest/omero/sysadmins/server-permissions.html>
- <https://docs.openmicroscopy.org/latest/omero/sysadmins/restricted-admins.html>

1. Go to <https://outreach.openmicroscopy.org/>
2. Login using the username and password





3. In the toolbar, click the *Admin* button


A dark grey rectangular button with the word "Admin" written in white text.

4. Managing Groups:

- a. Click on the *Groups* tab.
- b. You can search for groups if desired
- c. To create a new Group:
 - i. Click on the *Add new Group* button
 - ii. The *Name* and *Permissions* fields are mandatory.
 - iii. Click *Save*
- d. To edit a Group:
 - i. Click on the Pencil button 
 - ii. You can add or remove members or group's owners. Change permissions etc..
 - iii. Make sure that the data owned by a user is moved or transferred to another user before removing the user from the group
 - iv. Click *Save*

5. Managing Users:

- a. Click on the *Users* tab
- b. You can search for users if desired
- c. How to identify users:
 - i. Users with administrators privileges with 
 - ii. Active users with 
 - iii. Inactive users with 
 - iv. LDAP users with 
- d. To create a new User:
 - i. Click on the *Add new User* button

- ii. Mandatory fields are highlighted in red
 - iii. You can select the role of the user
 - iv. Click *Save*
 - e. To edit a User:
 - i. Click on the Pencil button 
 - ii. You can add or remove the user to/from a group. Modify the roles etc..
 - iii. Click *Save*
 - f. Creating an administrator privileges allow to give some limited rights to some trusted users e.g. to allow a facility manager to import data for other users. It is currently preferable to create users with such role via the Web Interface.
6. Note that a user can manage his/her settings.
- a. In the top-right corner of the webclient, click on your name
 - b. In the dropdown menu, click on *User settings*
 - c. In the dialog that pops up, the user can change password, default group i.e. the group he/she will log in by default etc.

Administrate using the Command Line Interface

1. Managing Groups:

- a. By default when creating a group, its permissions level is set to private. To create a new **read-annotate** group *Basel*, run:

```
$ bin/omero group add Basel --type=read-annotate
```

Or

```
$ bin/omero group add Basel --perms='rwra--'
```

- b. To list all the groups and save the output for example in a CSV file:


```
$ bin/omero group list --style csv > groups.csv
```
- c. To add an existing user *user-1* to the *Basel* group and make him/her a group owner (the option `--as-owner` is not needed when adding a member), run:


```
$ bin/omero group adduser user-1 --name=Basel --as-owner
```

- d. Let's add *trainer-1* as an owner of the group too:

```
$ bin/omero group adduser trainer-1 --name=Basel --as-owner
```

- e. To remove *user-1* from the list of owners (*user-1* will still be a member of the *Basel* group):

```
$ bin/omero user leavegroup Basel --name=user-1 --as-owner
```

- f. To remove *user-1* from the *Basel* group, run:

```
$ bin/omero group removeuser user-1 --name=Basel
```

- g. To edit the *Basel* group:

- i. First determine its ID:

```
$ bin/omero group info --group-name Basel
```

id	name	perms	ldap	# of owners	# of members
653	Basel	rwra--	False	0	0

- ii. Change its name:
`$ bin/omero obj update ExperimenterGroup:653 name='Basel-1'`
- iii. Let's reset the name back to "Basel" to simplify the rest of the workflow
- iv. Change groups permissions to **read-write**:
`$ bin/omero group perms --perms='rwrw--' --name='Basel'`

2. Managing Users:

- a. Create a new user *user-basel* and add the user to the *Basel* group:
`$ bin/omero user add user-basel Jan Purkyne --group-name Basel`
- b. Let's now add the user to another group:
`$ bin/omero user joingroup Lab1 --name=user-basel`
- c. To edit the user and for example add an email address
 - i. First determine the user's ID:
`$ bin/omero user info --user-name user-basel`
 - ii. Add an email address:
`$ bin/omero obj update Experimenter:123 email=jpurkyne@demo.co.uk`
- d. Make a user inactive:
 - i. User cannot be deleted but it is possible to prevent a user from logging in. For that we need to remove the user from the *user* group (internal OMERO group).
`$ bin/omero user leavegroup user --name=user-basel`
- e. To reactivate the user:
`$ bin/omero user joingroup user --name=user-basel`
- f. LDAP authentication, for information only:
 - i. It is possible to convert non LDAP users to LDAP authentication using the command `bin/omero ldap setdn`
 - ii. When using LDAP as an authentication backend, user when they log in will be added to the internal OMERO group called `default` unless they have already been added to a given group. To add a user before they have ever logged in to OMERO, run:
 - 1. First create the user
`$ bin/omero ldap create enoether`
 - 2. Then add the user to the *Basel* group
`$ bin/omero group adduser enoether --name=Basel`

Search data

In order to move data, change ownership, etc., the first step is to find the data. Depending on the complexity of the search, this can be achieved either one of the commands:

- `$ bin/omero search`
- `$ bin/omero hql`

As an administrator, let's find the Projects starting with **idr** and retrieve only their IDs:

```
$ bin/omero search Project "idr*" --ids-only
```

The `bin/omero hql` command is usually used for more complex query. The number of items is by default set to 25 to increase the number, the option `--limit` must be specified.

In an institution, administrators with restricted privileges (restricted administrators) will typically be imaging facility managers, image analysts, or anybody who needs to organize users and data of others in OMERO.

The following commands will be run by a restricted administrator *trainer-1* for another user.

Create data

For more information <http://docs.openmicroscopy.org/latest/omero/users/cli/containers-annotations.html>

1. The user *trainer-1* now logs in as *user-basel*, the data will be owned by *user-basel*:

```
$ bin/omero --sudo trainer-1 -u user-basel login
```

2. Create a Project:

```
$ bin/omero obj new Project name="Training-Project"
```

3. Create a new Dataset:

```
$ bin/omero obj new Dataset name="Training-Dataset"
```

4. To update the Dataset:

```
$ bin/omero obj update Dataset:501 description="Basel Day 2"
```

5. Link the Project and the Dataset:

```
$ bin/omero obj new ProjectDatasetLink parent=Project:101 child=Dataset:501
```

6. Create a Map Annotation:

- `$ bin/omero obj new MapAnnotation name=used-antibodies`
- `$ bin/omero obj map-set MapAnnotation:301 mapValue host rabbit`

7. Link the Map Annotation to the Dataset:

```
$ bin/omero obj new DatasetAnnotationLink parent=Dataset:501
```

```
child=MapAnnotation:301
```

Delete data

For more information <http://docs.openmicroscopy.org/latest/omero/users/cli/delete.html>

1. The delete command removes **entire graphs of data** based on the ID of the top node.
2. Run the command as *trainer-1*.
3. It is recommended to first run the command with the `--dry-run` option and `--report` to list the objects that will be deleted.
4. To delete the Project/Dataset graph:

```
$ bin/omero delete Project:101 --report --dry-run
```
5. To delete the graph and not delete the map annotation:

```
$ bin/omero delete Project:101 --exclude MapAnnotation --report --dry-run
```

Move data between groups

The owner of the data will usually use OMERO.web or OMERO.insight to move data between groups. This is not covered by this workshop and information can be found at <https://help.openmicroscopy.org/sharing-data#moving>

For more information <https://docs.openmicroscopy.org/latest/omero/users/cli/chgrp.html>

1. The `chgrp` command moves **entire graphs of data** based on the ID of the top node.
2. Run the command as *trainer-1*.
3. It is recommended to first run the command with the `--dry-run` option and `--report` to list the objects that will be moved.
4. First determine the ID of the target group. We want to move data from the *Lab1* group to the *Basel* group:

```
$ bin/omero group info --group-name Basel
```
5. To move the Project/Dataset/Image graph to *Basel*:

```
$ bin/omero chgrp Group:123 Project:123 --report --dry-run
```
6. To move the graph and not move the map annotation for example:

```
$ bin/omero chgrp Group:603 Project:101 --exclude MapAnnotation --report --dry-run
```
7. To move several containers at once:

```
$ bin/omero chgrp Group:603 Project:101,103,104 --report --dry-run
```

Change data ownership

For more information <https://docs.openmicroscopy.org/latest/omero/users/cli/chown.html>

Change ownership is currently not available neither in OMERO.web nor OMERO.insight. The command can be run by administrators, restricted administrators or group owners.

1. The `chown` command transfer ownership of **entire graphs of data** based on the ID of the top node.
2. Run the command as *trainer-1*.
3. To transfer the ownership of the Project/Dataset graph to *trainer-1*:

```
$ bin/omero chown trainer-1 Project:101 --report --dry-run
```

Upload data

1. It is possible to upload local files to the OMERO server and link them to an Object. This could be for example the output of an analysis.
 - a. Upload, for example, a CSV file and attach to a Project:

- ```
$ bin/omero upload analysis.csv
```
- b. The ID of the original file is returned, create a FileAnnotation:

```
$ bin/omero obj new FileAnnotation file=OriginalFile:123
```
  - c. Link it to the Project:

```
$ bin/omero obj new ProjectAnnotationLink parent=Project:101
child=FileAnnotation:123
```
2. Administrators or restricted administrators can also upload a script to the server so it can be used by all the users. It is important to be in the correct directory when uploading so that the script is uploaded with the path that will show up in the UI e.g. `base1/Channel_Offsets.py`
    - a. Go to the parent directory of the `base1` folder. Then run:

```
$ bin/omero script upload base1/Channel_Offsets.py --official
```

### Exercise:

Use bash to automate the workflows. Include the following steps:

- Log in as a restricted administrator and perform task for another user.
- Create a dataset.
- Import an image into the dataset.
- Upload a file.
- Create a file annotation and link it to the imported Image.

One solution:

[https://github.com/ome/training-scripts/blob/v0.3.0/practical/bash/import\\_as\\_create\\_annotation.sh](https://github.com/ome/training-scripts/blob/v0.3.0/practical/bash/import_as_create_annotation.sh)

## Server Installation

1. System requirements:
  - a. <https://docs.openmicroscopy.org/latest/omero/sysadmins/system-requirements.html>
2. Ansible way. CentOS 7 only
  - a. Playbook example  
<https://github.com/ome/ansible-examples-omero/blob/master/public-user/playbook.yml>
3. “Standard” way
  - a. <https://docs.openmicroscopy.org/latest/omero/sysadmins/unix/server-installation.html>
  - b. <https://docs.openmicroscopy.org/latest/omero/sysadmins/unix/install-web/web-deployment.html>
4. CLI-plugin
  - a. <https://pypi.org/project/omero-cli-render/>
5. Web apps
  - a. <https://pypi.org/project/omero-figure/>
  - b. <https://pypi.org/project/omero-iviewer/>

- c. <https://pypi.org/project/omero-fpbioimage/>
  - d. <https://pypi.org/project/omero-parade/>
6. LDAP
- a. <https://docs.openmicroscopy.org/latest/omero/sysadmins/server-ldap.html>
  - b. <https://docs.openmicroscopy.org/latest/omero/developers/Server/Ldap.html>