OMERO guide CellProfiler Documentation

Release 0.2.0

Open Microscopy Environment

May 20, 2022

CONTENTS

1 Contents 3

2 Contribute 11

CellProfiler is a free open-source software for quantitative analysis of biological images. We demonstrate how to integrate CellProfiler and OMERO using the CellProfiler Python API and the OMERO Python API.

For more details, visit the CellProfiler website.

Follow the instructions in the `README.md` file to run the notebooks either locally or on mybinder.org.

CHAPTER

ONE

CONTENTS

1.1 Install CellProfiler and OMERO Python bindings

In this section, we show how to install CellProfiler in a Conda environment. We will use the CellProfiler API to analyze data stored in an OMERO server.

CellProfiler currently runs on Python 2.7. It does not yet support Python 3.

1.1.1 Setup

We recommend to install the dependencies using Conda. Conda manages programming environments in a manner similar to virtualenv. You can install the various dependencies following the steps below (Option 1) or build locally a Docker Image using `repo2docker` (Option 2). When the installation is done, you should be ready to use the CellProfiler API and OMERO, see *Getting started with CellProfiler and OMERO*.

The installation below is needed to run the scripts and/or notebooks. If you wish to start your own environment without the scripts/notebooks, copy locally into an `environment.yml` file the content of binder/environment.yml, remove or add the dependencies you need and run the commands below to create a conda environment.

Option 1

- Install Miniconda if necessary.
- If you do not have a local copy of the omero-guide-cellprofiler repository, first clone the repository:

```
$ git clone https://github.com/ome/omero-guide-cellprofiler.git
```

- Go into the directory:

```
$ cd omero-guide-cellprofiler
```

- Create a programming environment using Conda:

```
$ conda create -n cellprofiler python=2.7
```

- Install CellProfiler, its dependencies and `omero-py` in order to connect to an OMERO server using an installation file:

```
$ conda env update -n cellprofiler --file binder/environment.yml
```

- Activate the environment:

```
$ conda activate cellprofiler
```

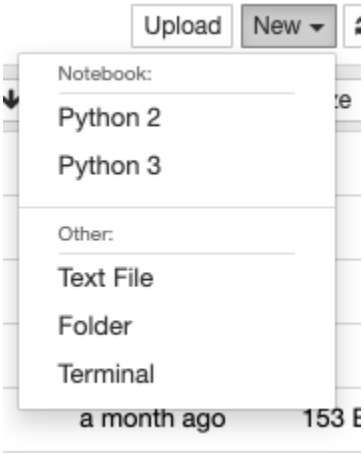
Option 2

Alternatively you can create a local Docker Image using `repo2docker`, see `README.md`:

```
$ repo2docker .
```

When the Image is ready:

- Copy the URL displayed in the terminal in your favorite browser
- Click the `New` button on the right-hand side of the window
- Select `Terminal`



- A Terminal will open in a new Tab
- A Conda environment has already been created when the Docker Image was built
- To list all the Conda environment, run:

```
$ conda env list
```

- The environment with CellProfiler and the OMERO Python bindings is named `kernel`, activate it:

```
$ conda activate kernel
```

1.2 Getting started with CellProfiler and OMERO

1.2.1 Description

We will use a Python script showing how to analyze data stored in an OMERO server using the CellProfiler API.

We will show:

- How to connect to server.
- How load images from a Plate using the OMERO API.

- How to run CellProfiler using its Python API.
- How to save the generated results and link them to the Plate.

1.2.2 Resources

We will use a CellProfiler example pipeline to analyse RNAi screening data from the Image Data Resource (IDR).

- Example pipeline from the CellProfiler website: Cell/particle counting and scoring the percentage of stained objects.
- Images from IDR idr0002.

For convenience, the IDR data have been imported into the training OMERO.server. This is only because we cannot save results back to IDR which is a read-only OMERO.server.

1.2.3 Setup

We recommend to use a Conda environment to install CellProfiler and the OMERO Python bindings. Please read first *Install CellProfiler and OMERO Python bindings*.

1.2.4 Step-by-Step

In this section, we go over the various steps required to analyse the data. The script used in this document is `idr0002_save.py`.

When running CellProfiler **headless**, it is important to set the following:

```
import cellprofiler_core.preferences as cpprefs
# Important to set when running headless
cpprefs.set_headless()  # noqa
```

Connect to the server:

```
def connect(hostname, username, password):
    conn = BlitzGateway(username, password,
                        host=hostname, secure=True)
    conn.connect()
    return conn
```

Load the plate:

```
def load_plate(conn, plate_id):
    return conn.getObject("Plate", plate_id)
```

A CellProfiler pipeline usually expects the files to be analyzed to be available locally. This is not the case here. So we first need to remove some modules from the pipeline so we can then inject data retrieved from the OMERO server:

```
def load_pipeline(pipeline_path):
    pipeline = cpp.Pipeline()
    pipeline.load(pipeline_path)
    # Remove first 4 modules: Images, Metadata, NamesAndTypes, Groups...
    # (replaced by InjectImage module below)
    for i in range(4):
        print('Remove module: ', pipeline.modules()[0].module_name)
        pipeline.remove_module(1)
    print('Pipeline modules:')
    for module in pipeline.modules():
        print(module.module_num, module.module_name)
    return pipeline
```

We are now ready to analyze the plate:

```
def analyze(plate, pipeline):
    warnings.filterwarnings('ignore')
    print("analyzing...")
    # Set Cell Output Directory
    new_output_directory = os.path.normcase(tempfile.mkdtemp())
    cpprefs.set_default_output_directory(new_output_directory)

    files = list()
    wells = list(plate.listChildren())
    wells = wells[0:5]  # use the first 5 wells
    for count, well in enumerate(wells):
        # Load a single Image per Well
        image = well.getImage(0)
        print(image.getName())
        pixels = image.getPrimaryPixels()
        size_c = image.getSizeC()
        # For each Image in OMERO, we copy pipeline and inject image modules
        pipeline_copy = pipeline.copy()
        # Inject image for each Channel (pipeline only handles 2 channels)
        for c in range(0, size_c):
            plane = pixels.getPlane(0, c, 0)
            image_name = image.getName()
            # Name of the channel expected in the pipeline
            if c == 0:
                image_name = 'OrigBlue'
            if c == 1:
                image_name = 'OrigGreen'
            inject_image_module = InjectImage(image_name, plane)
            inject_image_module.set_module_num(1)
            pipeline_copy.add_module(inject_image_module)
        pipeline_copy.run()

        # Results obtained as CSV from Cell Profiler
        path = new_output_directory + '/Nuclei.csv'
        files.append(path)
    print("analysis done")
```

(continues on next page)

(continued from previous page)

```
    return files
```

Let's now save the generated CSV files and link them to the plate:

```
def save_results(conn, files, plate):
    # Upload the CSV files
    print("saving results...")
    namespace = "cellprofiler.demo.namespace"
    for f in files:
        ann = conn.createFileAnnfromLocalFile(f, mimetype="text/csv",
                                              ns=namespace, desc=None)
        plate.linkAnnotation(ann)
```

When done, close the session:

```
def disconnect(conn):
    conn.close()
```

In order to use the methods implemented above in a proper standalone script: **Wrap it all up** in an `analyze` method and call it from `main`:

```
def main():
    # Collect user credentials
    host = input("Host [wss://workshop.openmicroscopy.org/omero-ws]: ") or 'wss://
↪workshop.openmicroscopy.org/omero-ws'
    username = input("Username [trainer-1]: ") or 'trainer-1'
    password = getpass("Password: ")
    plate_id = input("Plate ID [102]: ") or '102'
    # Connect to the server
    conn = connect(host, username, password)

    # Read the pipeline
    pipeline_path = "../notebooks/pipelines/ExamplePercentPositive.cppipe"
    pipeline = load_pipeline(pipeline_path)

    # Load the plate
    plate = load_plate(conn, plate_id)

    files = analyze(plate, pipeline)

    save_results(conn, files, plate)
    disconnect(conn)
    print("done")


if __name__ == "__main__":
    main()
```

1.2.5 Exercises

1. Modify the script above to analyze images in a dataset (`Solution`).
2. Modify the script to link the generated results to the corresponding image (`Solution`).
3. Modify the script to aggregate the result in an OMERO.table and link the output to the plate (`Solution`).

1.3 Analyze OMERO data using a Jupyter Notebook

1.3.1 Description

We will demonstrate how to integrate CellProfiler and OMERO using the CellProfiler Python API and the OMERO Python API. We will use a Jupyter notebook to demonstrate the integration.

We will show:

- How to adjust an existing CellProfiler pipeline so that it can be used with OMERO.
- How load images from a Plate using the OMERO API.
- How to run CellProfiler using its Python API.
- How to plot the results.
- How to save the generated results back to OMERO as OMERO.table so they can be used later on by OMERO.parade.

1.3.2 Resources

We will use a CellProfiler example pipeline to analyse RNAi screening data from the Image Data Resource (IDR).

- Example pipeline from the CellProfiler website: Cell/particle counting and scoring the percentage of stained objects.
- Images from IDR idr0002.
- Notebook `idr0002_save.ipynb`.

For convenience, the IDR data have been imported into the training OMERO.server. This is only because we cannot save results back to IDR which is a read-only OMERO.server.

1.3.3 Step-by-Step

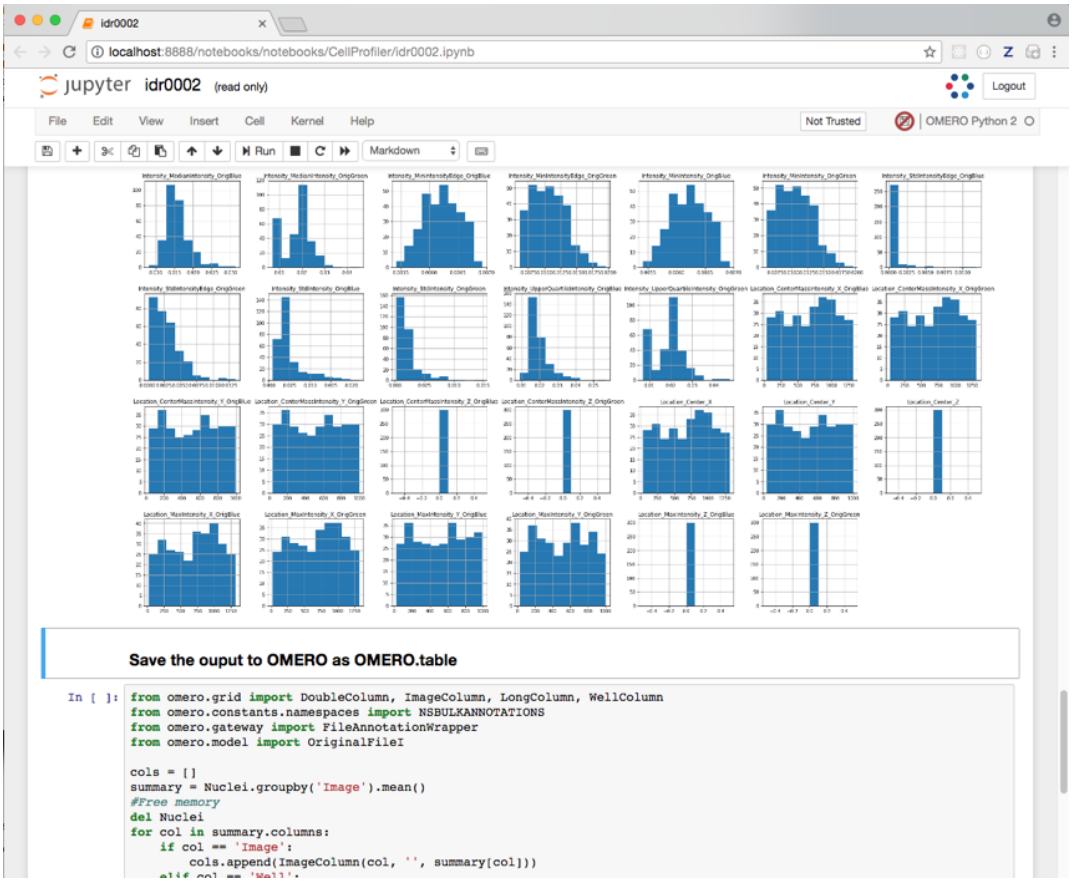
The pipeline is run on all 2-Channel images from each Well in the 96-well plate (each Well contains one image), generating a CSV file containing rows for different objects identified in the image and columns for various parameters measured.

1. First, open the webclient and find the Plate belonging to trainer-1 named plate1_1_013.
2. Launch the idr0002_save.ipynb notebook in mybinder.org.
3. Select the first Step and click on the Run button to execute each step in turn.
4. For the connection to OMERO, you will be asked to enter your login details when running the OMERO credentials cell.
5. Select the plate in the webclient, find the Plate ID in the right-hand panel and copy this into the plate_id variable in the next step of the notebook.

6. The following cell loads the example pipeline and modifies it to remove the modules that are normally used for loading images from disk.
7. These modules are replaced by the InjectImage module, using numpy planes loaded from OMERO Images. This allows to pass data from OMERO to CellProfiler.
8. Note that to save time, we run on a subset of all Wells in the plate. We run it on the first 5 wells

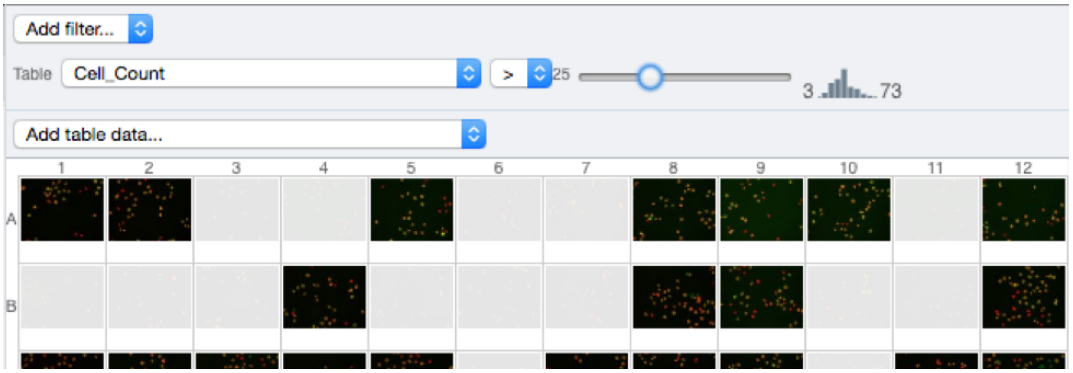
```
# create list from generator
wells = list(plate.listChildren())
wells = wells[0:5]
well_count = len(wells)
```

9. The generated CSV file is 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.
10. All the Dataframes are then concatenated into a single Dataframe.
11. We visualize the data as histograms for each column with `df.hist()`.



12. Finally, the Dataframe rows are grouped by Image to give an average value per Image of each parameter (column) in the table.
13. This data is saved back to OMERO as an HDF5-based table attached to the Plate, which can be read by other clients.
14. Return to the webclient and select the Plate named plate1_1_013_previously_analysed.

15. Select a Well in the central pane and open the Tables harmonica in the General tab in the right-hand pane. This will show all the CellProfiler values for this Well.
16. In the Thumbnails dropdown menu at the top-right of the centre panel, select the Parade plugin.
17. At the top-left of the centre panel choose *Add filter...* > *Table* to filter Wells by the data from CellProfiler.
18. Change the filter from ImageNumber to Cell_Count (at the bottom of the list).
19. Now you can use the slider to filter Wells by Cell Count.



CHAPTER

TWO

CONTRIBUTE

Changes to the documentation or the materials should be made directly in the omero-guide-cellprofiler repository.