

1. Usage examples

1.1. Running a macro on a list of files specified by Excel

Scenario

In a very simple application, an Excel spreadsheet is used to tabulate a series of image files to be analyzed, and an identical macro is applied to all the images, with collection of possible macro results in a common table.

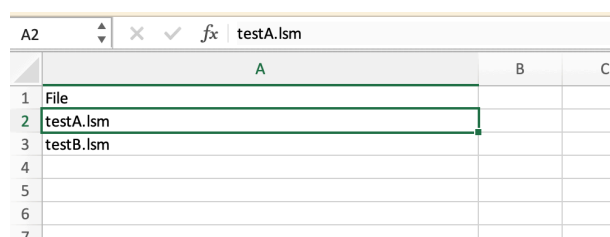
Such a scenario is implemented by the “test_file_only.xlsx” Excel file, and the two image files “testA.lsm” and “testB.lsm” located in the same folder as “test_file_only.xlsx”.

Execution synopsis

In this simplistic scenario, the PoreSizeExcel plugin reads an Excel file. It applies a user-specified macro to each of the images listed in the Excel file, and collects the results in a common table.

Pre-requisite: Excel file with image file listing

A pre-requisite for running ImageJ macros using Excel files to direct the choice of files to be analyzed is to draft an Excel file providing at least the names of the files to be handled. In this most prototypical scenario, this is exemplified by the “test_file_only.xlsx”. This spreadsheet contains a single sheet (“Sheet1”), and data only in a single column, labelled “File”. The data entries correspond to the files to be analyzed, here, “testA.lsm” and “testB.lsm”. This is shown in Figure 1.



	A	B	C
1	File		
2	testA.lsm		
3	testB.lsm		
4			
5			
6			
7			

Figure 1. Contents of “test_file_only.xlsx”.

Plugin launch

Assuming correct installation of the PoreSizeExcel plugin, the Excel batch processing can be launched in ImageJ (or equivalently, Fiji) by choosing Plugins > Microniche PoreSizeExcel > Batch Process with Excel File as shown in Figure 2.

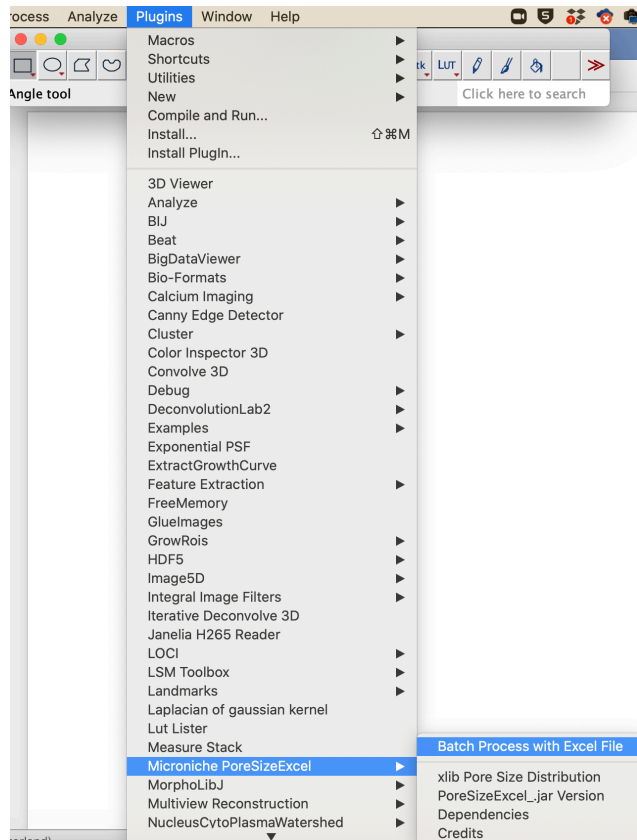


Figure 2. Plugin launch

Excel file loading

The PoreSizeExcel plugin next allows to select the Excel file for use. This is a standard file selection dialog, as shown in Figure 3. Select the appropriate Excel file, here “test_file_only.xlsx”

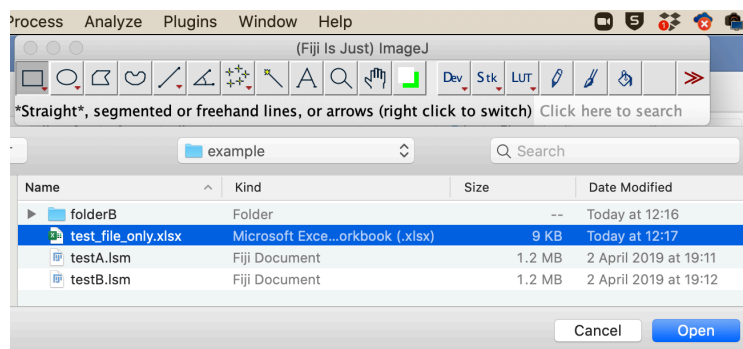


Figure 3. Excel file selection

In “test_file_only.xlsx”, there is only a single Sheet, and so the PoreSizeExcel plugin will directly proceed to the main plugin dialog, otherwise, a sheet selection dialog would be open (see more complex scenarios).

Plugin dialog

The PoreSizeExcel plugin next provides the main plugin options dialog, shown in Figure 4.

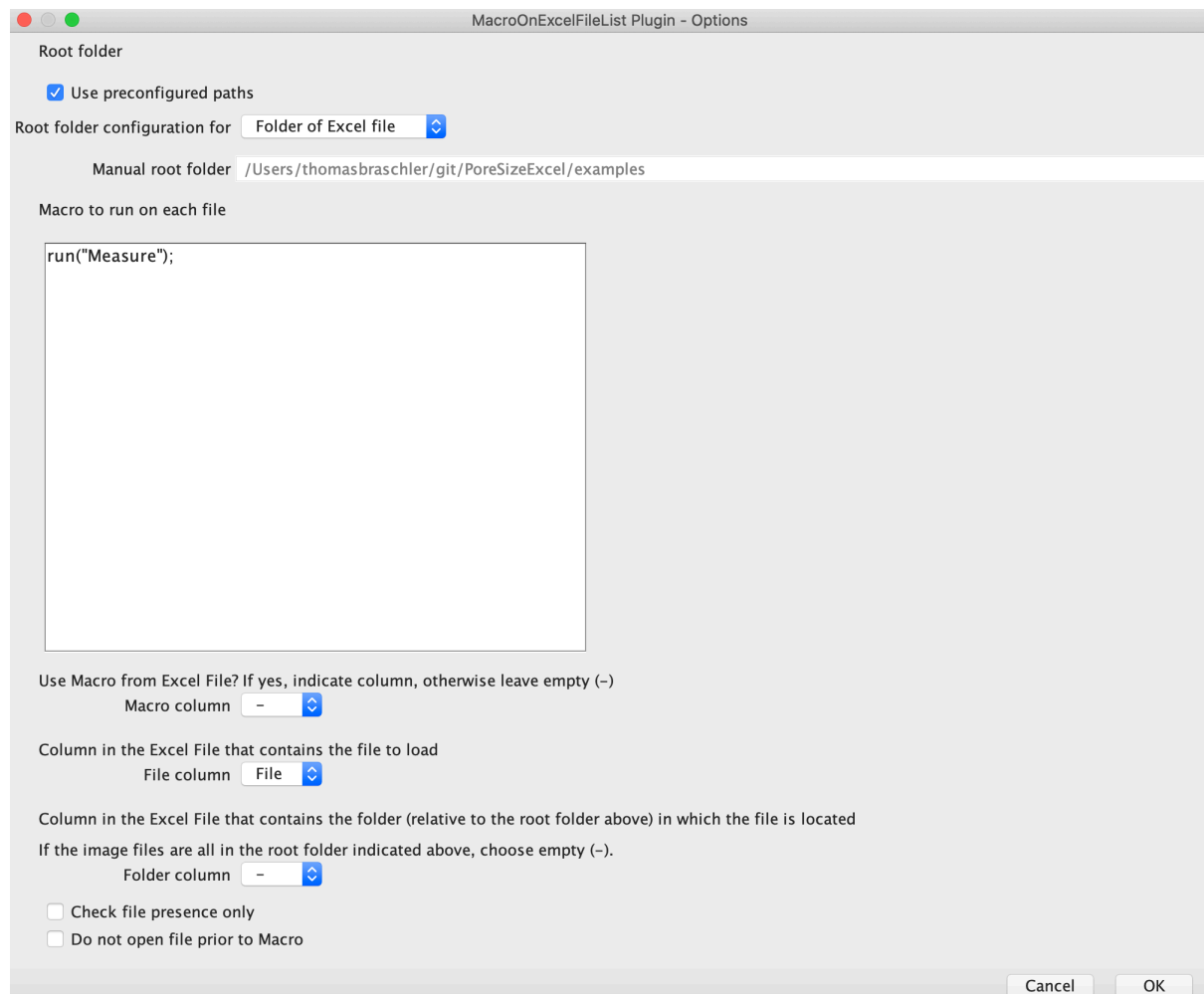


Figure 4. Plugin dialog

In the main plugin dialog, several options are available. First, the root folder must be configured. This is the folder relative to which images are located. The plugin offers either standard options (by default, the folder of the Excel file, but also programmatically configurable standard locations), or fully manual entry of the path. In Windows, the entry of the paths can be tricky. Either on your system, forward slashes are accepted, or you need to use backslashes (“\”), but then they must be doubled in order not to count as escape sequences. So it would be something like C:\\\\some\\folder.

Next, the plugin offers to execute macros on the image files listed in the Excel file. In the simplest configuration, an identical macro (here: `run(“Measure”);`) is executed on each image file. In more sophisticated scenarios (see below), it is also possible to execute specific macros on each image, for example if the file name needs to be used explicitly; this happens via the Excel file and is also explained below. Here, no macro information is taken from Excel, and so in the “Macro Column” choice field, “-” is selected.

In order to know which images need to be treated, the PoreSizeExcel plugin reads the Excel file. In the “File column” choice field, the column containing the file names can be selected.

Optionally, it is also possible to read the folder localization from the Excel as well. This is useful when the images resides in different subfolders, but is not needed for this simple example, and so no folder column ("-") is selected in the choice field.

For debugging purposes, it is possible to only check for the file presence, without running the actual macro. In this case, the checkbox "Check file presence only" should be checked.

Finally, the basic idea is that the PoreSizeExcel plugin opens the image, to then apply the macro. However, sometimes the image opening is more involved and needs to be handled as part of the macro. In this specific case, one would have to check the box "Do not open prior to Macro". However, this is not necessary for the simple test scenario here.

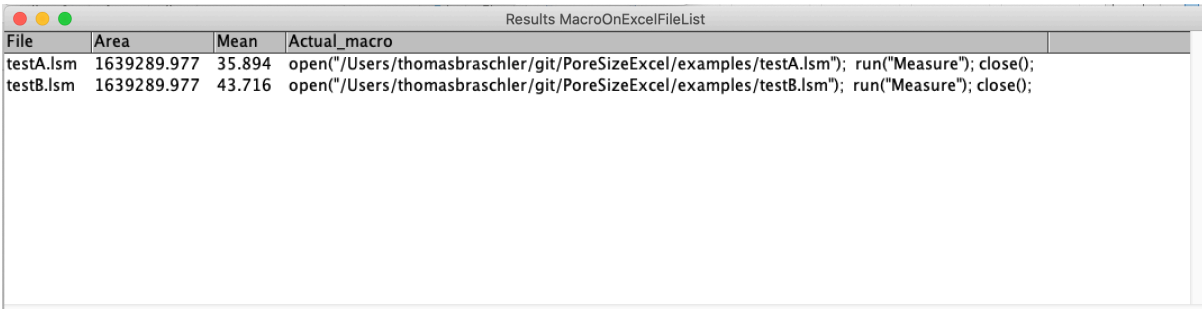
Running the plugin

After hitting "OK" in the plugin dialog, the PoreSizeExcel plugin runs through the list of files provided in the Excel file. Each file is opened. This happens by default using the ImageJ "open" macro command (i.e. box "Do not open prior to Macro" unchecked in Figure 4). Then, the macro is applied to the image. In the scenario here, this is the macro specified in the "Macro to run on each file" textbox (see Figure 4) textbox. In the concrete example is the measure command: `run("Measure")`. After execution of the macro, the image window is automatically closed.

Result collection

The PoreSizeExcel plugin keeps track of the macros executed and reports them as an ImageJ result table (Figure 5). At the very least, this table contains a copy of the columns in the original Excel file (i.e., the "File" column in Figure 5) as well as the actual macro that was run for each image (i.e. the "Actual macro" column in Figure 5). The actual macro comprises usually an "open" command, the user-specified macro, followed by a "close" command.

In addition to these minimal columns, the PoreSizeExcel plugin reports output generated during the user-defined macro. Indeed, some ImageJ commands, such as the `run("Measure")` command yield text output in the form of an ImageJ Results table, and this information is collected by the PoreSizePlugin. With the default measurement settings used here, this additional information corresponds to the area and mean grey value. These values are reported as shown in Figure 5 for each image. Hence, in the simplistic scenario presented here, user-defined characteristics of each image as specified in the Excel file are read and reported along with the macro used in an overview results table.



File	Area	Mean	Actual_macro
testA.lsm	1639289.977	35.894	<code>open("/Users/thomasbraschler/git/PoreSizeExcel/examples/testA.lsm"); run("Measure"); close();</code>
testB.lsm	1639289.977	43.716	<code>open("/Users/thomasbraschler/git/PoreSizeExcel/examples/testB.lsm"); run("Measure"); close();</code>

Figure 5. PoreSizeExcel output.