Image J Tutorials – Basic Macro Writing

By Christine Labno
University of Chicago
Confocal Light Microscopy Facility
When is it Worthwhile to Write a Macro?

When you need one or more of the following:

- Consistency – same procedure for all images
- Speed – for large numbers of images or repetitive tasks
- Documentation – for publication of novel methods
- Sharing – within or between groups
Things to Know Before You Start

• A macro, like all computer code, is unforgiving and literal.
• Often macros require specific types of image input. Copy carefully and be prepared to modify.
• Documentation for yourself and other future users is helpful. Comments and instructions can (should?) be written right into the macro file.
• ImageJ is meant to be open source. This means if you create a macro, it’s nice to share with others.
Places to Go For Help

- Fiji Introduction to Macro Programming
  http://fiji.sc/Introduction_into_Macro_Programming
- ImageJ website → Documentation (Docs) → Macro Language
- Macros on the ImageJ website
  http://rsb.info.nih.gov/ij/macros/
- Macros on our website
  http://digital.bsd.uchicago.edu/%5Cimagej_macros.html
- ImageJ mailing list (subscription) or the list archives at the ImageJ Website → List
  http://rsbweb.nih.gov/ij/list.html
Steps for Writing a Macro

Necessary steps:
• Macro Recorder to build the basic framework
• Make the macro generic so it runs on any image

Optional steps:
• Adding steps to “loop” the process, having it work automatically through a group of images
• Adding areas of user input with pauses and user input tables
The Macro Recorder (Command Listener)

- Plugins → Macros → Record
- An easy way to get started

- Writes out exact code for *most* functions as you work though your procedure
- Mistakes and re-takes also get recorded, so either work out your procedure beforehand or be prepared to do a lot of code cleanup
Making Your Macro “Generic”

• The command listener writes down exactly what you do, right down to the names of the images you use. To use a macro on more than one image, specific names must be replaced with either generic names or variables. For example: rename(“stack”); then selectWindow(“stack”); or use t=getTitle; then selectWindow(t);

• Names must be typed / copied EXACTLY in order to work. “Stack” is different from “stack” and also from “ Stack”
Example Macro Code

run("Duplicate...", "title=[cd31 glomeruli-1.tif]");
run("Green");
selectWindow("cd31 glomeruli.tif");

rename("stack");
run("Duplicate...", "title=[green image]" func="saveAs" param="cd31 green image.tif");
run("Green");
selectWindow("stack");

T=getTitle;
run("Duplicate...", "title=[green image]" func="saveAs" param="stack green image.tif");
run("Green");
selectWindow(T);
Making Your Macro “Generic”

• Remember that ImageJ works on whatever image / slice is selected (i.e. on the top of the pile).

• If you are processing multiple images or a stack of images, find a way to make sure the macro knows which image is which, and which image you want it to work on at a given time. For example: `selectWindow("name");` or `setSlice(002);`
Example Macro Code

rename("stack");
run("Duplicate...", "title=[green image]");
run("Green");
selectWindow("stack");
setSlice(2);
run("Duplicate...", "title=[blue image]");
run("Blue");

rename("stack");
run("Duplicate...", "title=[green image]");
run("Green");
selectWindow("stack");
//setSlice(2);
run("Duplicate...", "title=[blue image]");
run("Blue");
Repeat a Process Automatically

- You can make a macro repeat a process for a whole stack of images or for a whole folder full of images

- Loops like this are opened and closed with `{   }`

- You can set up a file path to automatically save images and / or data to a particular file path – be careful you’re not saving over raw data or running your output images back through the macro
Example Macro Code

dir = getDirectory("Choose a Directory to PROCESS");
list = getFileList(dir);
dir2 = getDirectory("Choose a Directory for SAVING");

//setBatchMode(true);
for (f=0; f<list.length; f++) {
    path = dir+list[f];
    if (!endsWith(path,"/")) open(path);
    if (nImages>=1) {
        if (endsWith(path,"f")) {
            t=getTitle();
            s=lastIndexOf(t, '.');
            t=substring(t, 0,s);
            t=replace(t," ","_");
            t2= t + ' processed';
            run("Green");
            rename(t2);
            saveAs("Tiff", dir2 + t2 + ".tif");
            run("Close");
        }
    }
}
}
Adding User Input

• If you need to do something custom for each run of the macro (i.e. adjust a threshold manually) there is a `waitForUser(" ");` command. This will pause the macro indefinitely and let you do what you want. The macro resumes when you hit “OK.”

• Variables can be used to create user input tables at the beginning of a macro. The information put into this table can be used over the course of the macro.

• Command keys can be added to code. You can have a macro with several parts and assign a key (F6, for example) that a user can hit when they want to run that part of the macro.
Example Macro Code

Dialog.create("Which color for which slice?");

Dialog.addNumber("Slice number for green image:", 0);
Dialog.addString("Name for the green image:", "green");
Dialog.addNumber("Slice number for blue image:", 0);
Dialog.addString("Name for the blue image:", "blue");

Dialog.show();

grn_chan = Dialog.getNumber();
g = Dialog.getString();
blu_chan = Dialog.getNumber();
b = Dialog.getString();

rename("stack");
setSlice(grn_chan);
run("Duplicate...", "title=");
rename(g);
run("Green");
selectWindow("stack");
setSlice(blu_chan);
run("Duplicate...", "title=");
run("Blue");
rename(b);
Testing your macro

- You will need to test your macro (probably several times) to make sure it runs the way you expect it to.

- // in front of a line of code will “comment it out” or keep it from being read as a command. This can be useful for taking out a line of code without erasing it.

- To run only part of a macro, highlight the code you want to run and then Macro → Run macro. Be aware that if you use a variable assigned earlier in the code, the macro won’t have this information and will break down.
Troubleshooting Suggestions

• When making large-scale changes to a macro, create a working copy called name v.2 or some such. If you break your code and need to back up, you still have a functional copy of the original macro.

• Test your macro in stages. First make sure it will process one image the way you expect, then expand to auto-processing a folder with three or four test images, THEN run your macro on all of your images. Keeping a temporary backup copy of that folder is a good idea.
Useful Bits of Code

• Get the name of an image: \texttt{t=getTitle;}
• Select a particular image: \texttt{selectWindow("imagename");}
• Select a particular slice in a stack: \texttt{setSlice(#);}
• Pause and wait: \texttt{waitForUser("instructions here");}
• To run a block of code on a number of stacks: \texttt{for (i=1; i<=nSlices; i++) \{ insert lines of code \}}
• To assign a macro to a start key: \texttt{macro "name [key]" \{ insert lines of code \}} Note that the function keys (F1-F12) are popular keys to assign to a start function.
• To close a window: \texttt{run("Close"); or close();}
• To create a user input box: \texttt{Dialog.create("nameofdialog");}
• To add choices to a user input box: \texttt{Dialog.addChoice("textforuser:", newArray(t, t1, t2, t3, t4, t5))};