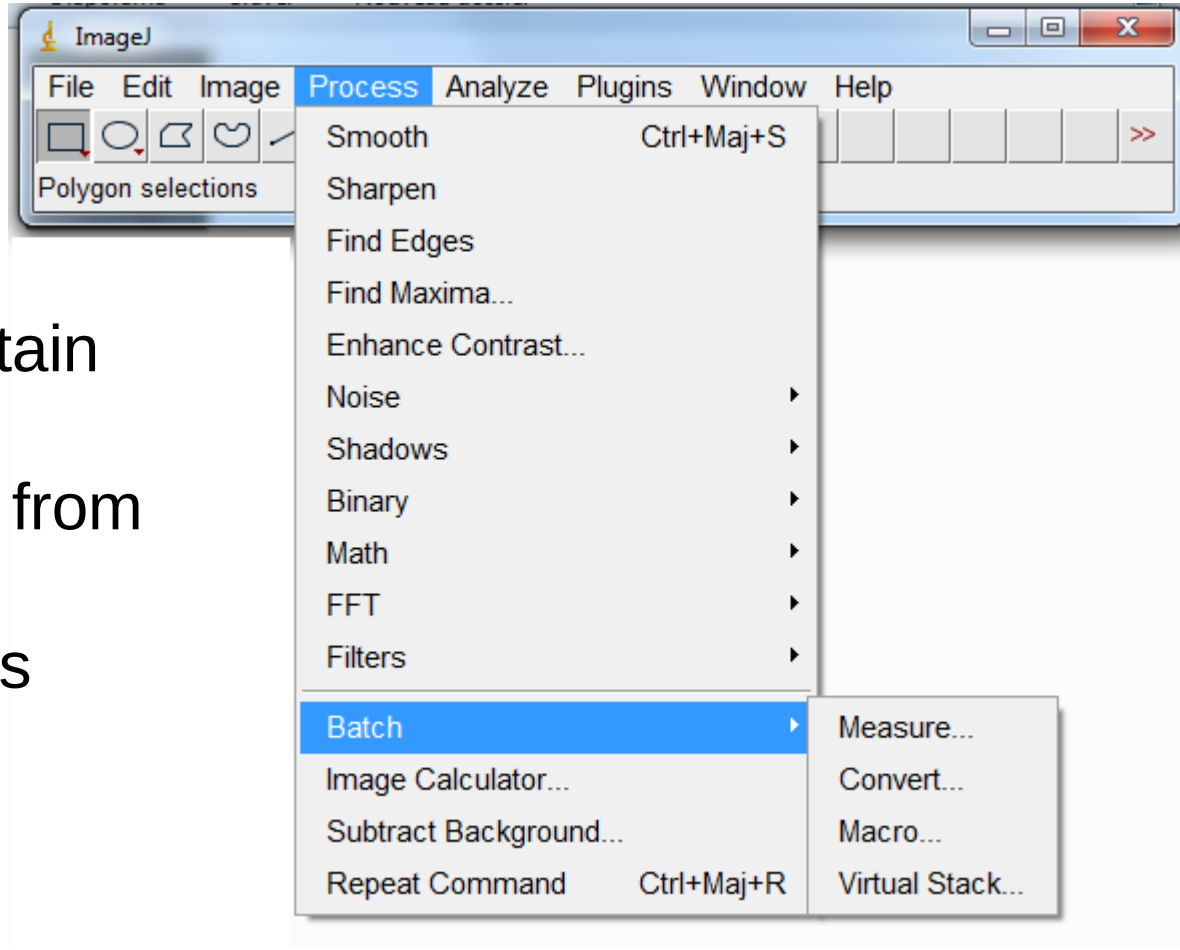
Macros avec ImageJ

Semaine 3b

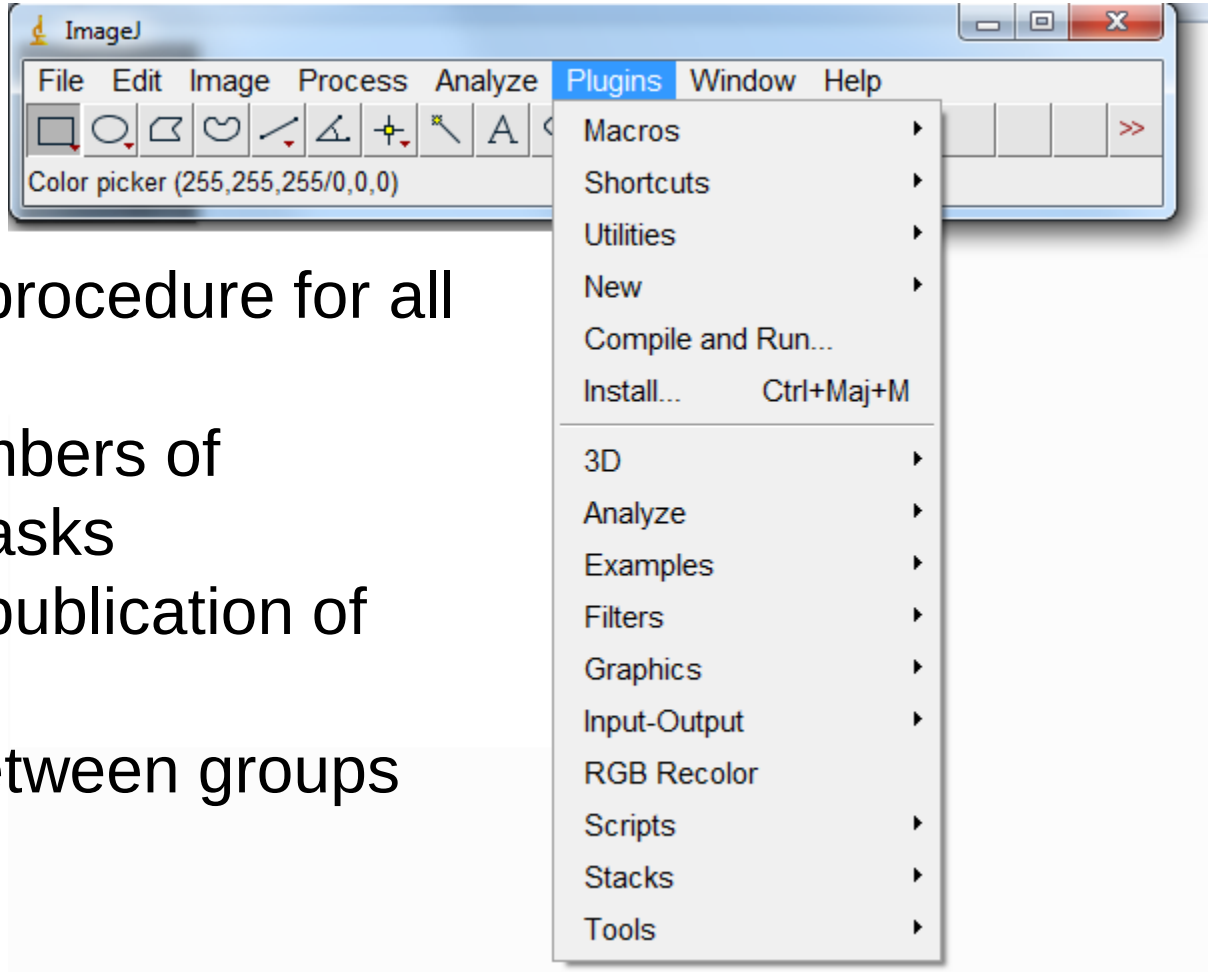
Merci à Christine Labno : University of Chicago
Confocal Light Microscopy Facility

Batch commands

- Simple
- Prewritten code for certain functions
- Operates on image file from certain directories
- Extendable with marcos



Why write a macro?



- 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, is computer code
- It does EXACTLY what you ask – it is “literal”
- Be prepared to modify the code to customize
- Often macros require specific types of image input.
- Documentation for yourself and other future users is helpful. Comments and instructions should be written right into the macro file. (“//”)
- ImageJ is is open source; it is nice to share with others!

Resources

- Fiji Introduction to Macro Programming
  http://fiji.sc/Introduction_into_Macro_Programming
- ImageJ website → Documentation (Docs) → Macro Language
  http://rsbweb.nih.gov/ij/developer/macro/macros.html
- Macros on the ImageJ website
  http://rsb.info.nih.gov/ij/macros/
- Macros on UChicago 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
- ImageJ Macro Language (Programmer's Ref. Guide v1.46d)
- Macro Workshop

Steps for Writing a Macro

Necessary steps:

- Use “Macro Recorder" to obtain basic code
- Customize code for and example application

Optional steps:

- Make the macro generic so it runs on any image
- 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]");
run("Green");
selectWindow("stack");

T=getTitle;
run("Duplicate...", "title=[green image]");
run("Green");
selectWindow(T);
```

Note: Run part of code by highlighting that part

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

```
t=getTitle;
selectWindow(t);
run("Duplicate...", "title=[cd31 glomeruli-1.tif]");
run("Red");
selectWindow(t);

rename("stack");
run("Duplicate...", "title=[green image]");
run("Green");
selectWindow("stack");
```

```
T=getTitle;
run("Duplicate...", "title=[green image]");
run("Green");
selectWindow(T);
```

Example Macro Code – for stack

```
t=getTitle;
//selectWindow(t);
setSlice(019);
t=getTitle;
run("8-bit");
run("Duplicate...", "title=[cd31 glomeruli-1.tif]");
run("Red");
selectWindow(t);

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


T=getTitle;
run("Duplicate...", "title=[green image]");
run("Green");
selectWindow(T);
```

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

```
//opens all images in a directory, changes them to
green, (via LUT) and saves in a different folder with a
different name.
//needs 8-bit grey tiff files

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);
```

Useful Bits of Code

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

Example Macros:

- Complete list

Fun:
- Bull's Eye (lien)
- Pong (lien)

Some examples:
- Plots (lien)
- Segmentation (lien)
- Stack management(lien)
- Count and mark cells (lien)
- Image calculator (lien)

DRAWING:
- Polygon (lien)
- Random ovals (lien)
- Draw random dots (lien)←Use this for devoir 2
- Tracking movie (lien) ←Can be modified for devoir 2?