Data visualisation with ImageJ and Fiji

Chris Hall

ImageJ and FIJI

- Developed at the National Institute of Health in the USA
- The ImageJ application is very popular and has a large support community
- Written in Java, it will run on almost anything ☺
- It supports many file formats
  - Some inbuilt, some via 'plugins'
- It handles stacks of 2-D images better than many other applications
- It's free! (https://imagej.nih.gov/ij/)

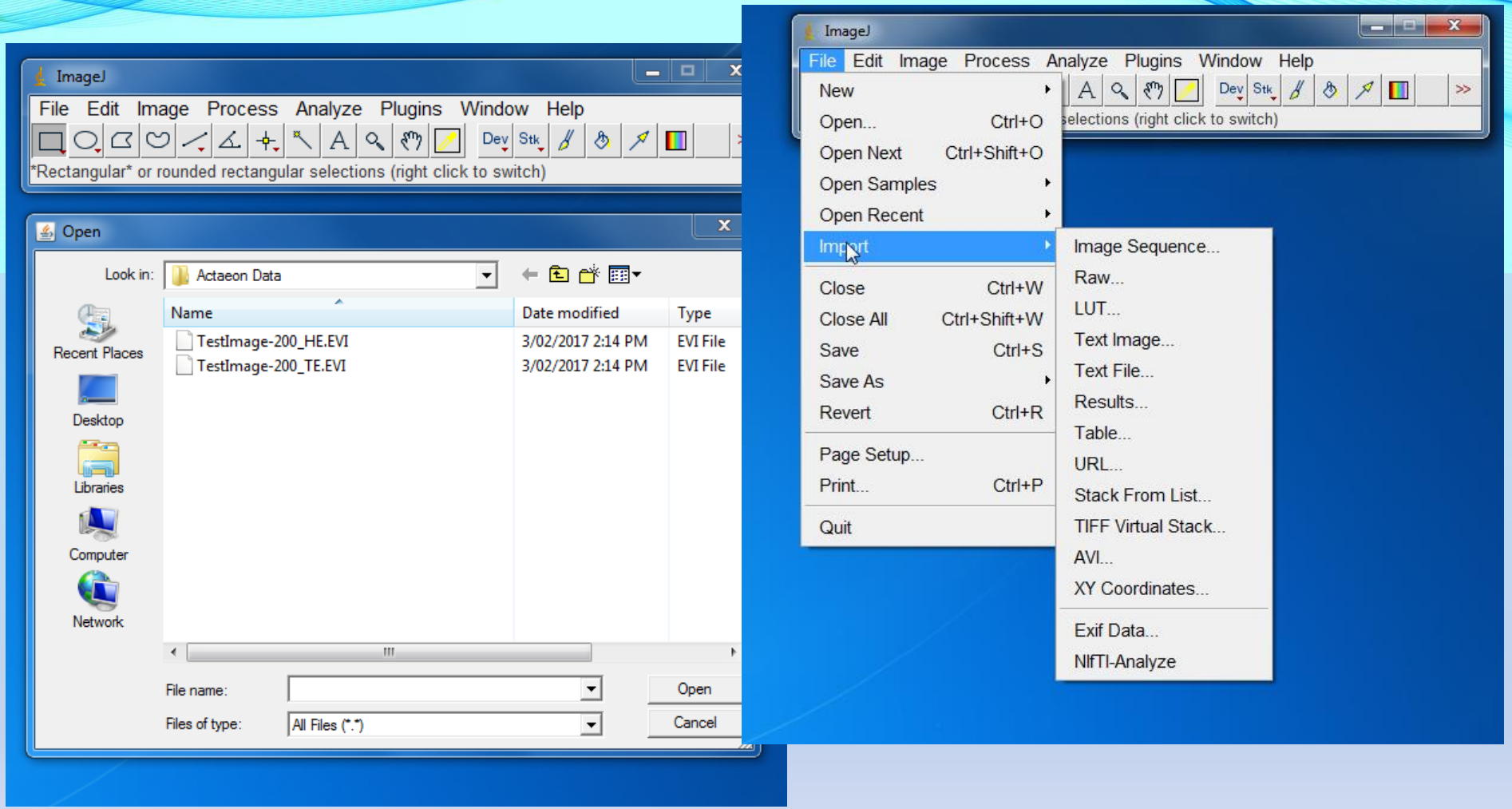
(Schneider, C. A.; Rasband, W. S. & Eliceiri, K. W. (2012), "NIH Image to ImageJ: 25 years of image analysis", Nature methods 9(7): 671-675)

Fiji is just ImageJ

- Over the years (since 1997) many plugins have been written for ImageJ by many groups and individuals
- Fiji has many of these built in. (Originally aimed at neuroscience)
- Fiji has a more sophisticated updating system than ImageJ (https://fiji.sc/)

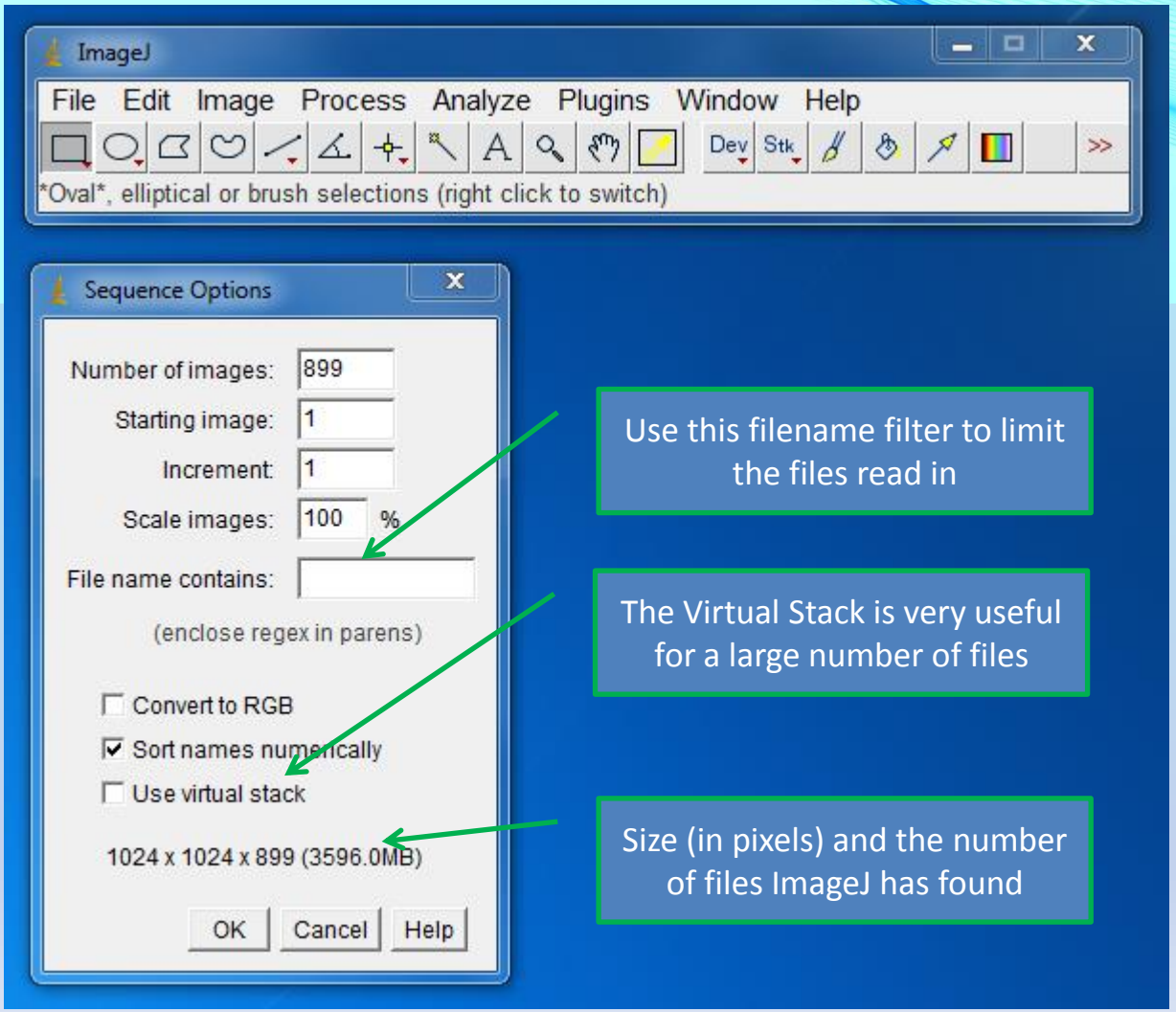
(Schindelin, J.; Arganda-Carreras, I. & Frise, E. et al. (2012), Schneider, C. A.; Rasband, W. S. & Eliceiri, K. W. (2012), "NIH Image to ImageJ: 25 years of image analysis", Nature methods 9(7): 671-675 *Nature methods* **9(7)**: 676-682)

Opening files



Opening several files

- If all the images are the same size. The result is an image **stack** within ImageJ (more on this later)



ImageJ Image types

- There are several ways that a 2-D array of numbers (pixels) can be shown on the screen as an image.
- A pixel value is always *mapped* to screen greyscale intensity or colour (value can be integer or real)
- Pixel values can index into a colour chart (only integers)
- RGB: Three numbers per pixel = red, green and blue intensities (only integers)

1.42	0.08	1.58	1.22
1.95	1.90	1.69	0.16
0.33	0.47	1.74	1.47
1.30	1.50	0.78	1.03

240	242	123	214
132	24	232	69
57	58	201	32
51	133	155	31

128,53,190	189,185,240	135,238,13	90,36,104
29,96,5	14,92,176	239,71,222	157,174,219
203,75,36	163,88,20	12,174,219	214,21,194
186,115,121	9,236,7	31,182,243	129,174,130

Viewing images

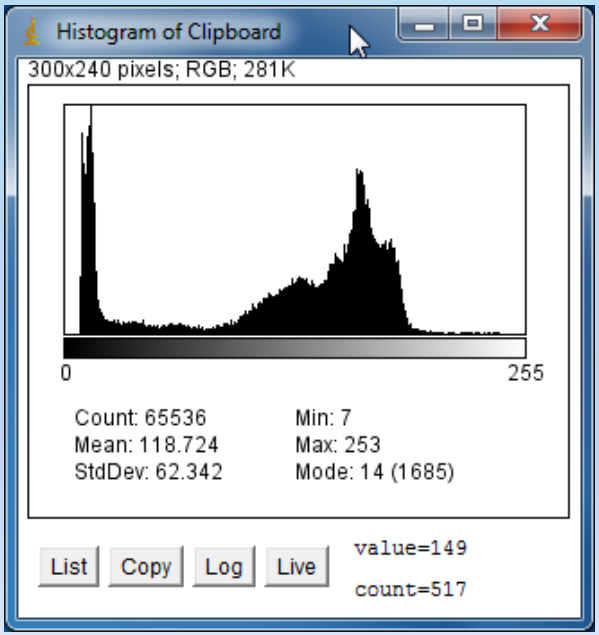
- Our screens allow ‘32 bit’ colours = three 8 bit numbers, representing the red, green and blue intensity, for each pixel.
- The mapping of the pixel values will always end up as an ‘RGB’ value for the display.
- Using a real value, then mapping it to one 8 bit number (0 to 255) determines the intensity (black to white, R=G=B)
- Real numbers (or integers) in pixels are mapped to 8 bit RGB integer(s) first through a Look Up Table (LUT)

Viewing images

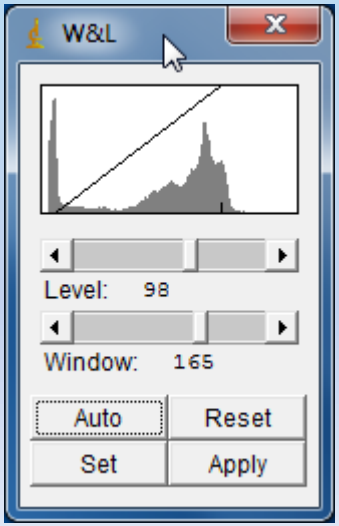
- There are several standard LUTs in ImageJ. Some are more confusing than useful!

Brightness and contrast

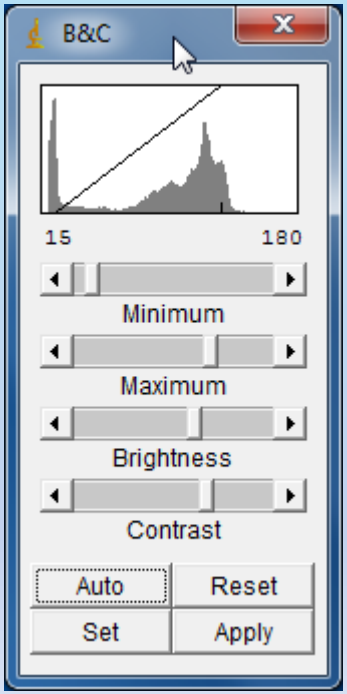
- Two best ways to adjust: *brightness and contrast*, or *window and level*.
- Changes the mapping between the pixel value and the greyscale (or pallet).



Cntrl-H



Cntrl-Shift-C

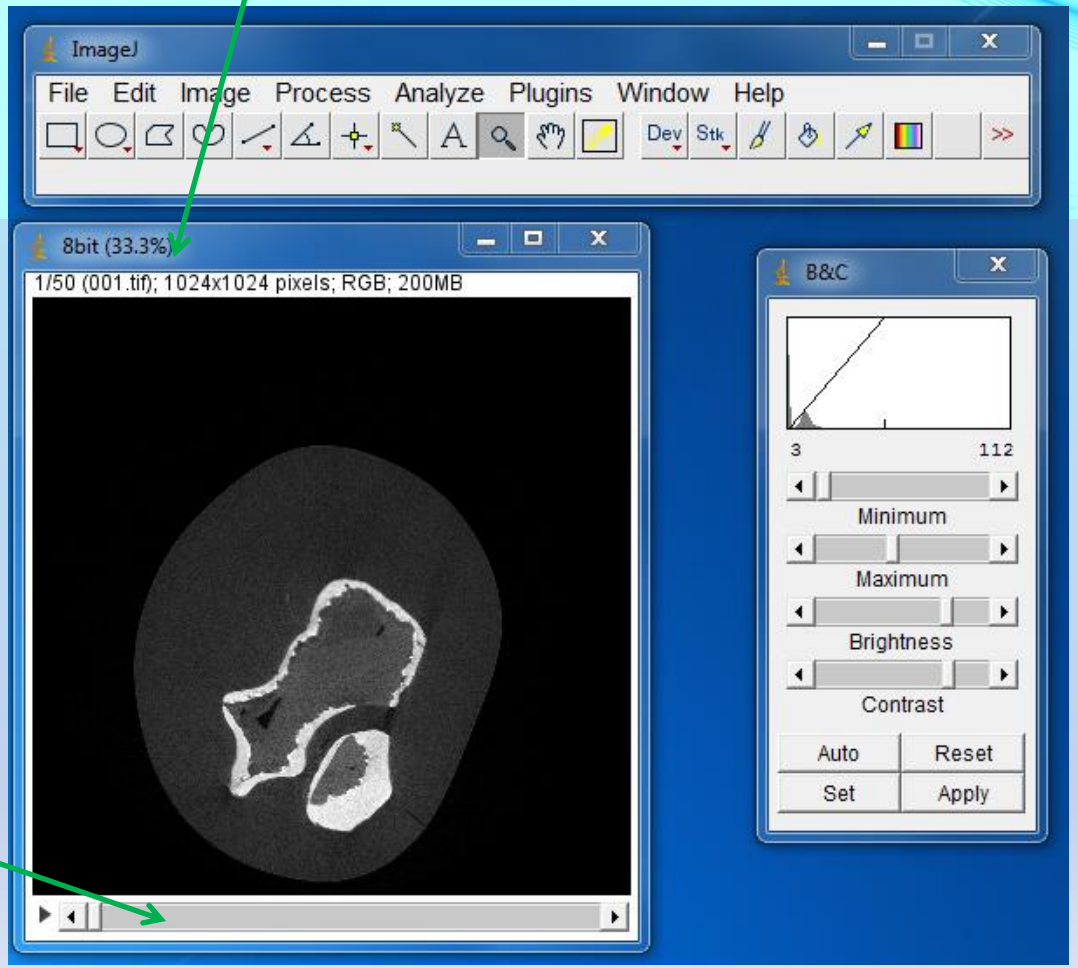


Making calculations on images

- 'Process' operations perform calculations on a single image (e.g. add a number to all pixel values)
- 'Image Calculator' operates on *two images* (e.g. add together corresponding pixels values in each image)
- E.g. A rough flat field correction can be made using 'divide' (Rough because of dark pedestal $\frac{I_a+P}{I_b+P} \neq \frac{I_a}{I_b}$)

Working with stacks

- Input as an ‘Image sequence’
- Or a single file containing multiple images

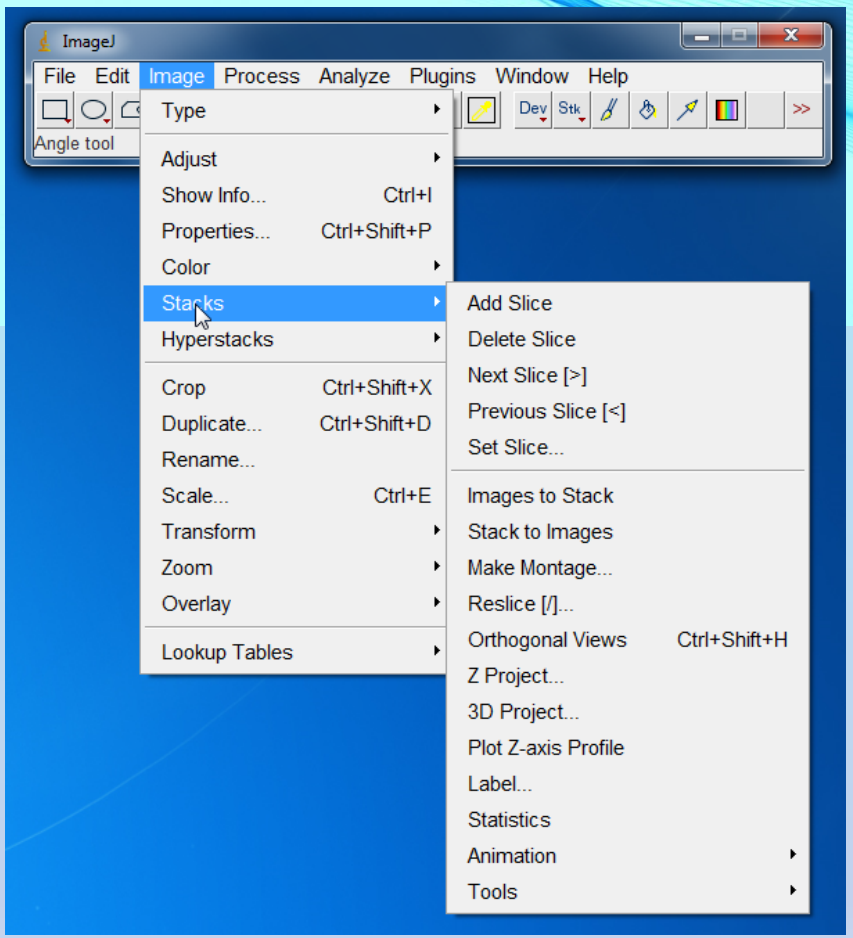


Stack operations

- Many ways to manipulate and view the stack
- Note: Some plugins for 3-D viewing are rather slow
- Use:

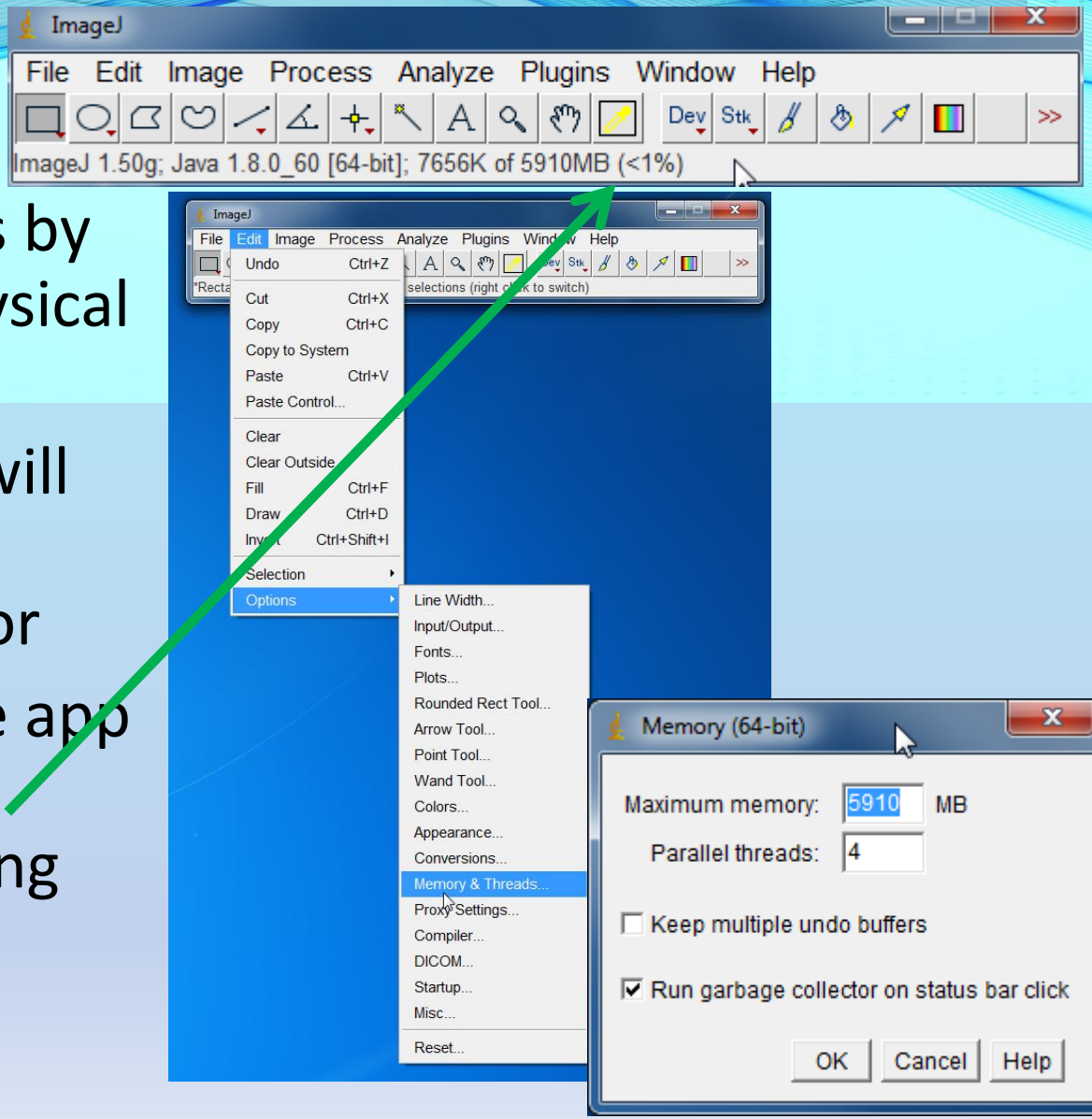
  Image>Stacks>Plot Z-Axis profile
- Use:

  Plugins>Stacks>Measure Stack



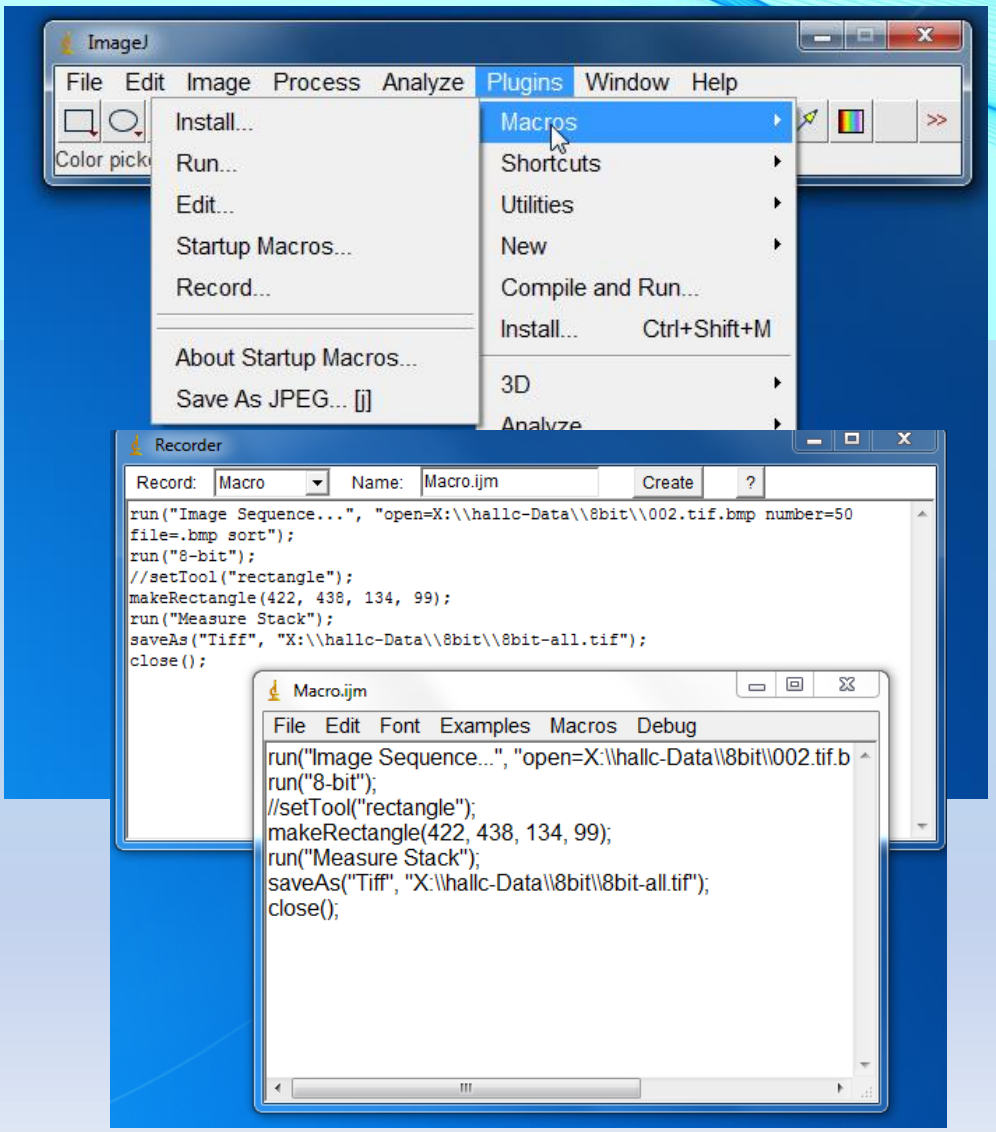
Altering ImageJ's memory

- Allocated memory is by default ~ 75% of physical memory
- Trying to use more will results in an 'OutOfMemory' error
- The status bar in the app tells you how much memory you are using (and tidies it up if clicked)



Scripting (macros)

- A sequence of operations can be captured and saved in an edit box
- Then saved as a text macro file (.ijm)



Rich macro language

Many built-in functions

Conditional flow control (if-then-else)

Simple print to log or console

Conditional flow control (for loop)

'Background' mode for increased speed

DisplayWindowTitles.ijm

File Edit Font Examples Macros Debug

```
// Display Window Titles
//
// Displays the titles of non-image and image windows.

list = getList("window.titles");
if (list.length==0)
   print("No non-image windows are open");
else {
   print("Non-image windows:");
   for (i=0; i<list.length; i++)
      print("   "+list[i]);
}

print("");
if (nImages==0)
   print("No images are open");
else {
   print("Image windows:");
   setBatchMode(true);
   for (i=1; i<=nImages; i++) {
      selectImage(i);
      print("   "+getTitle);
   }
}
print("");
```

Many other features to explore

- Filtration
- Segmentation
- Maths macro
- Calibration and measurements
- Batch macro
- Stitching (Fiji)
- Etc.