.

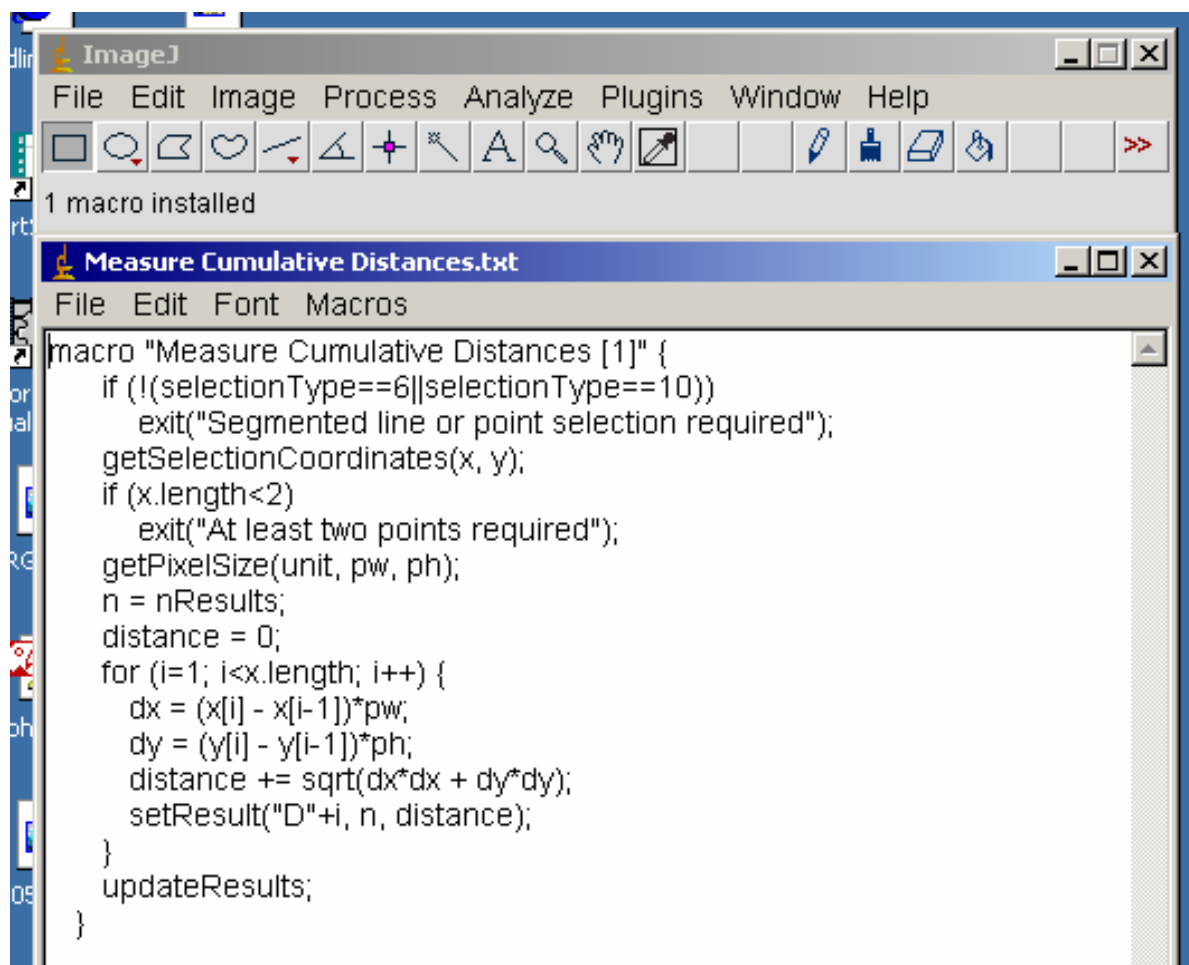# Using and writing macros in ImageJ 　　　2 August 2007

Jacqui Ross

What are macros?

Macros are strings of commands which are joined together to automate routines. For example if you want to run the same sequence of processing steps over several images, you can use a macro to do this. The sequence of commands is written into the macro file.

Macros are written in plain text and must have the extension .txt. They are located in the Macros folder.

Using existing macros

Open the macro file from the File menu, *File Open*. The macro will open up in **Macro Editor**.


```
macro "Measure Cumulative Distances [1]" {
    if (!(selectionType==6||selectionType==10))
        exit("Segmented line or point selection required");
    getSelectionCoordinates(x, y);
    if (x.length<2)
        exit("At least two points required");
    getPixelSize(unit, pw, ph);
    n = nResults;
    distance = 0;
    for (i=1; i<x.length; i++) {
        dx = (x[i] - x[i-1])*pw;
        dy = (y[i] - y[i-1])*ph;
        distance += sqrt(dx*dx + dy*dy);
        setResult("D"+i, n, distance);
    }
    updateResults;
}
```

You can then run the macro by going to *Macros – Run macro* inside the **Macro Editor** or by typing **ctrl-R**.
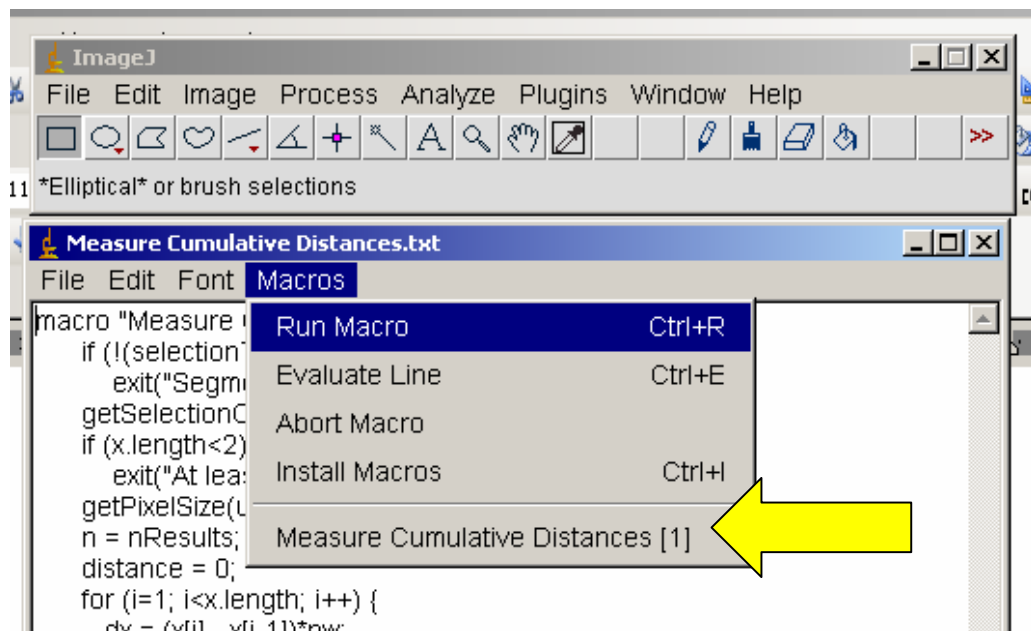

The macro then runs through the sequence of commands contained in it. Note that the name of the macro also appears at the bottom of the window (indicated by arrow).

This macro (**Measure Cumulative Distances**) allows you to measure cumulative distances of a segmented line which has been drawn using the segmented line tool.
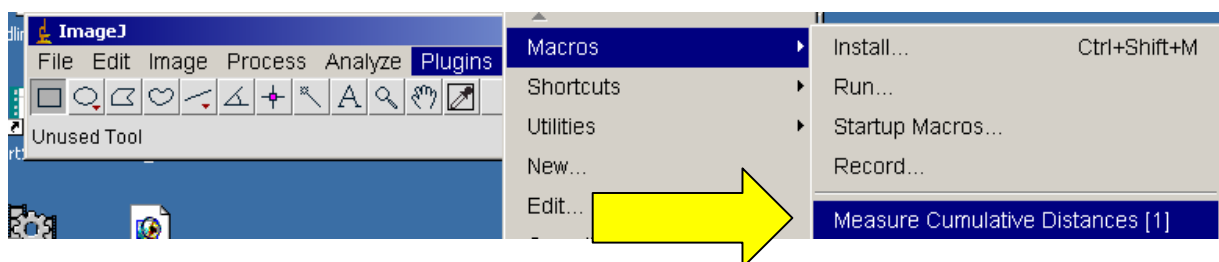
You can also open macro files directly by just dragging and dropping them on the program icon (Mac) or on the main ImageJ window (Windows).

Installing macros

You can install macros in the **Plugins – Macros** submenu by going to *Plugins – Macros – Install*.
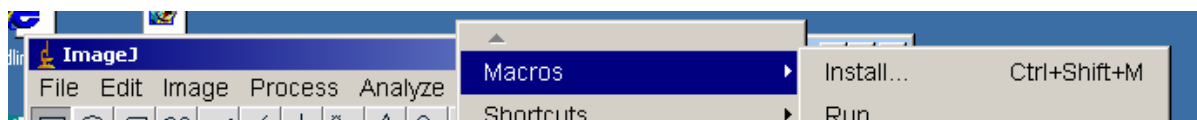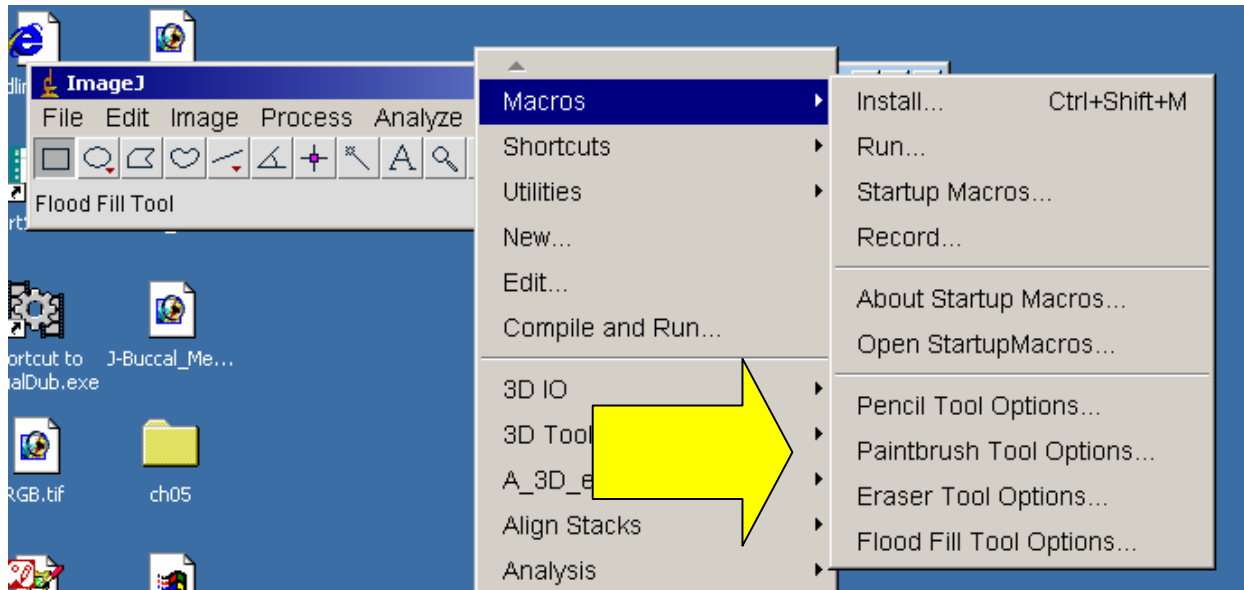

This will make the macro available in the macro menu as shown below:

Startup macros

These macros are automatically installed when the program is launched and appear in the **Plugins – Macros** submenu, e.g. Pencil Tool Options, etc.


Built-in macro functions

Some functions are built into ImageJ using macros. These appear automatically as part of the program and the commands can be used in other macros. The built-in macro functions are available at http://rsb.info.nih.gov/ij/developer/macro/functions.html listed in alphabetical order. You can copy and paste the command you need directly from the list into your new macro.

For example, you can add a Region Of Interest (ROI) to the ROI Manager by including **roiManager("Add");** in the macro. If the ROI Manager is not open, the macro will also open it. Other functions of the ROI Manager can be used by replacing "Add" with another command.

Creating macros from scratch

Open a text window in ImageJ, *File – New – Text Window*.
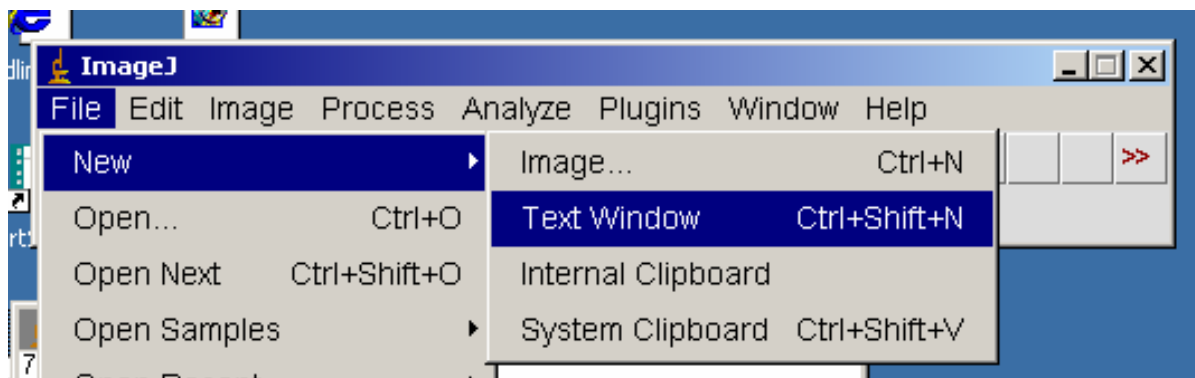
Or go to *Plugins – New*. Give the file a name and select **Macro**.

Note that if you use an <u>underscore</u> in the name, then the macro will be available through the **Plugins** menu as long as it is located in the **Plugins** folder or a **Plugins subfolder**. This is convenient if you use the macro regularly since you don't then need to install it every time. The underscore is converted to a space in the name when it appears in the **Plugins** menu.
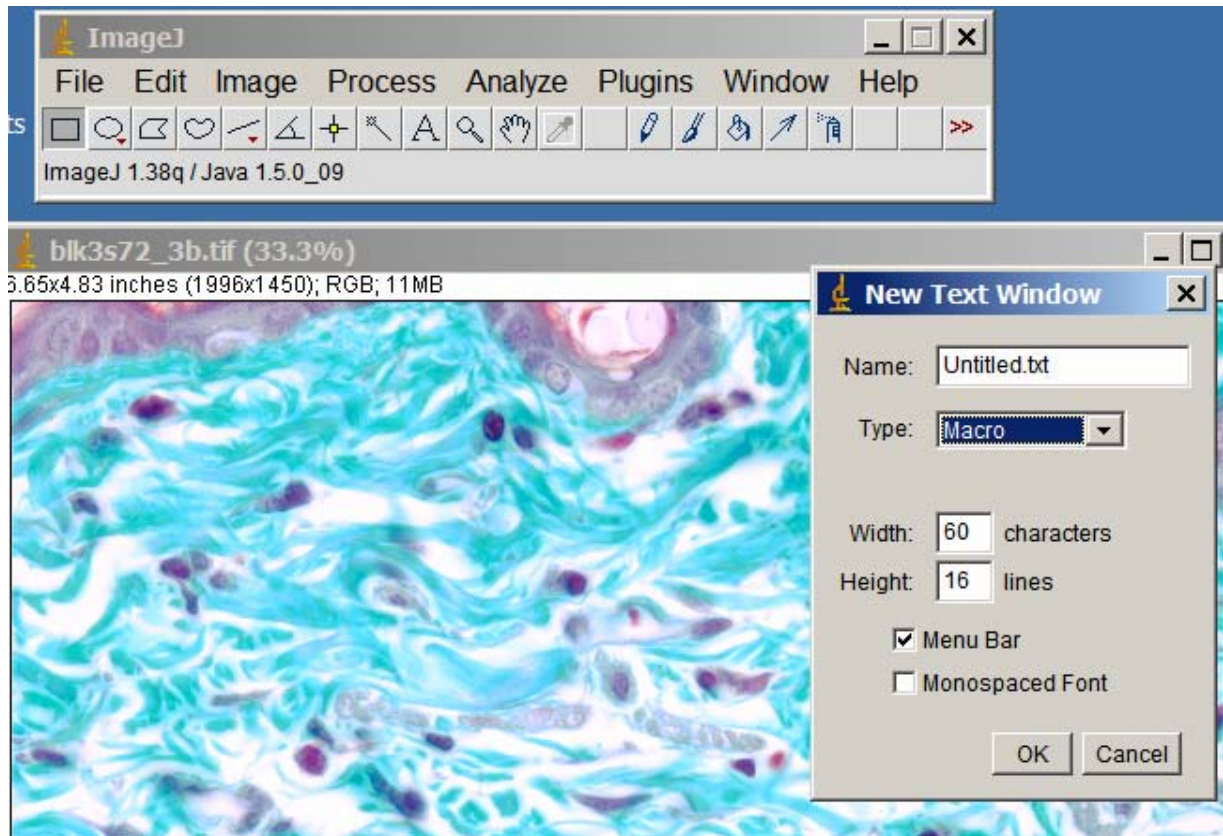
Select the number of characters and lines that you think you might need for the macro.

In both cases, the **Macro Editor** window opens up as shown below:
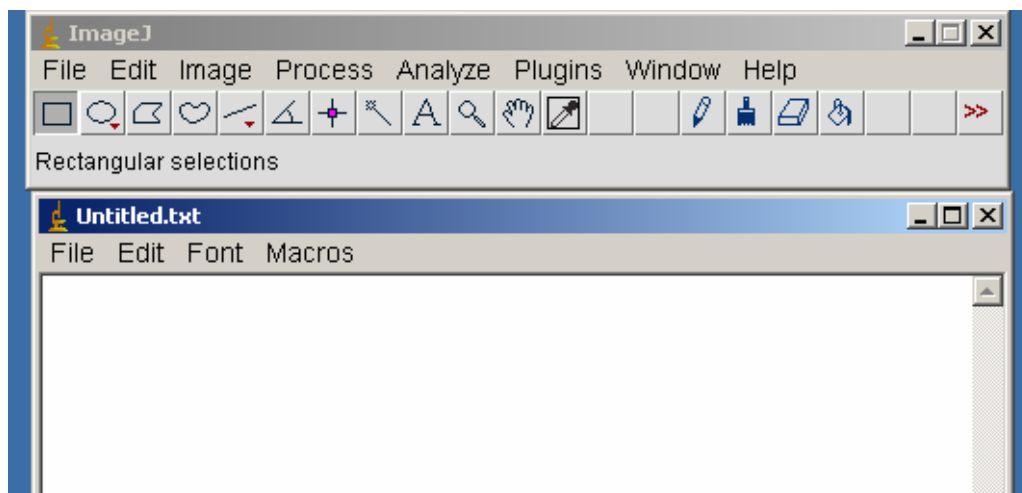
Now you can type in directly any commands that you want, or copy and paste commands from the **Built-in Macro** page. Another option is to open a macro which is similar to what you want and copy and paste portions of that macro into your own version.
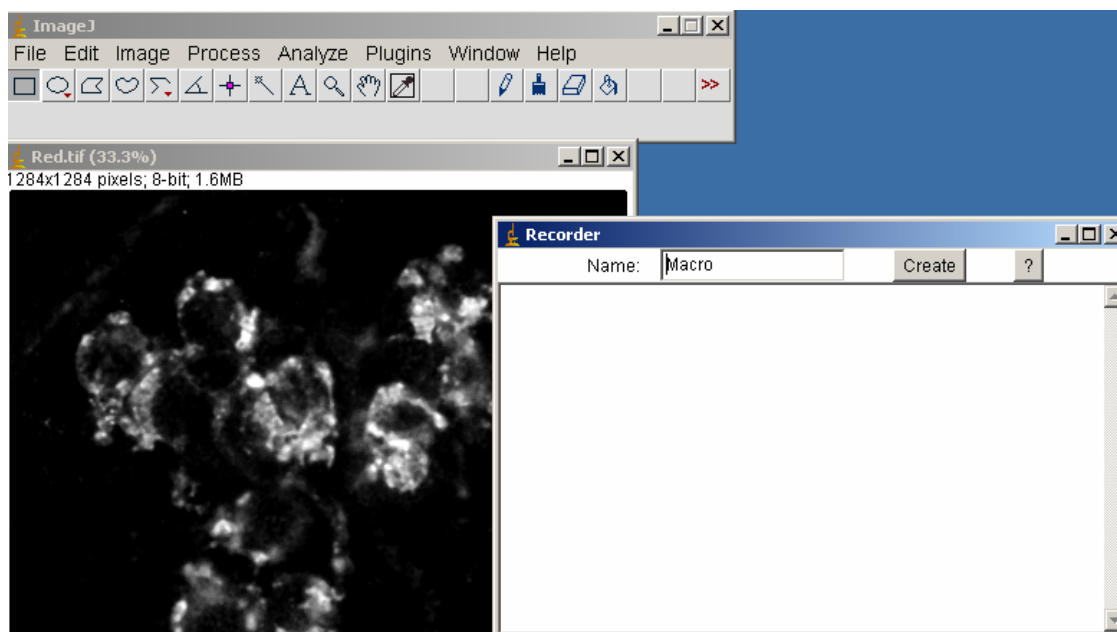
Macro Recorder
This is the easiest way to create a macro and also to learn about the macro language.

Go to *Plugins – Macro – Record*.
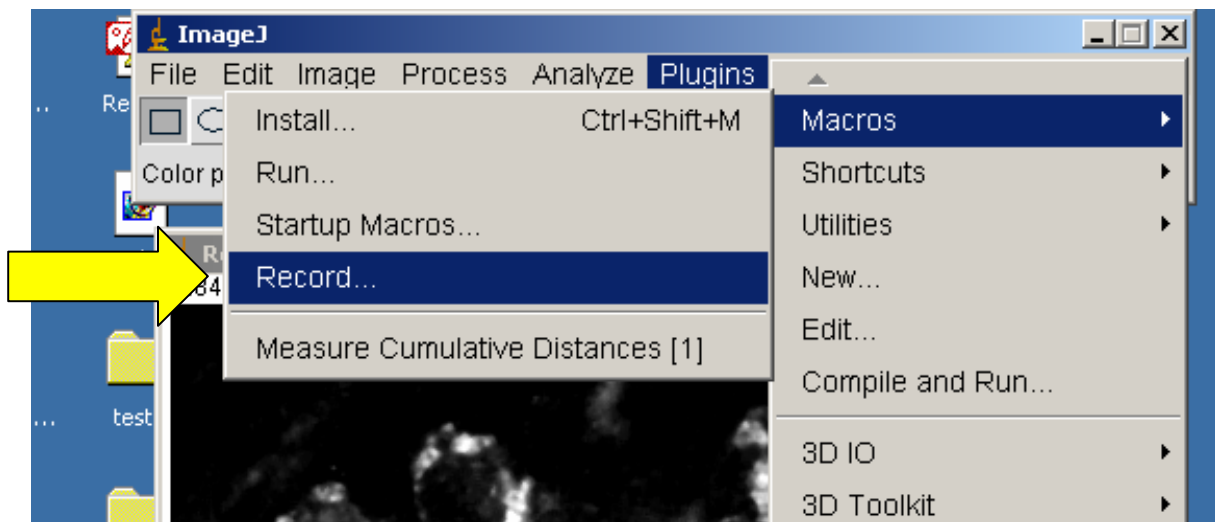
The **Recorder** window opens up as shown below:


Give the macro an appropriate name.

Now every action that you perform will be recorded in the **Recorder** window.

Once you have completed all the steps that you need for your macro, click on **Create**. A new **Macro Editor** window will appear with the name you have given to your macro. You can then run the macro to test it, or evaluate it line by line (**Evaluate Line**). Once you are happy with it, save it into the Macros folder or into the Plugins folder if you want it to be available on startup (remember the underscore in the name if you want to do this).
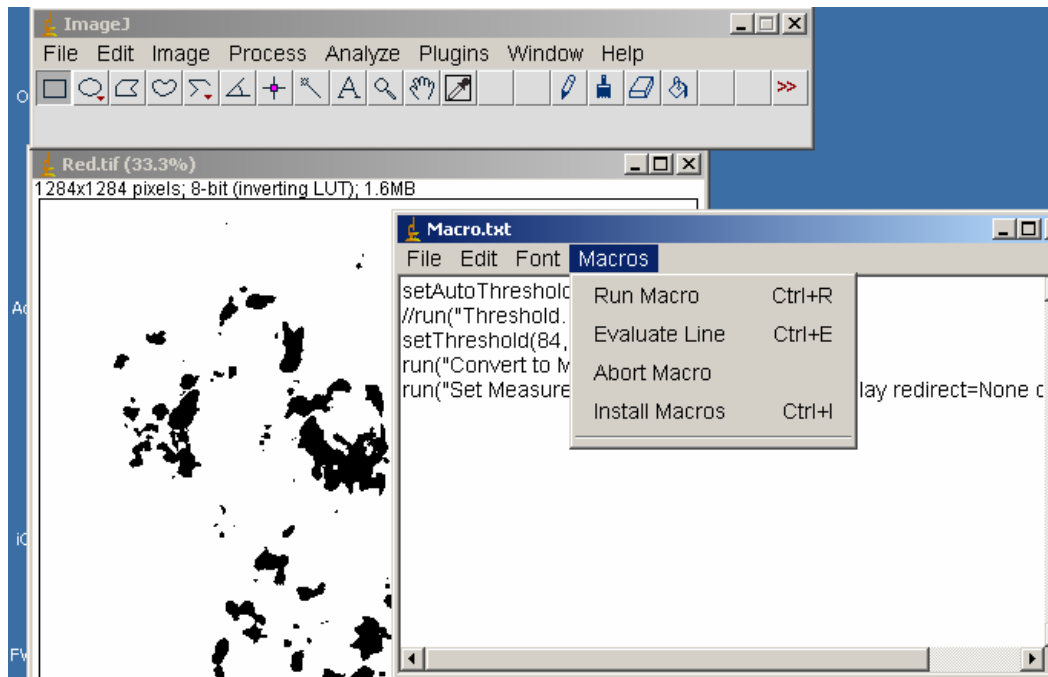
Shortcuts

You can create a shortcut by going to *Plugins – Create Shortcut* as shown below:
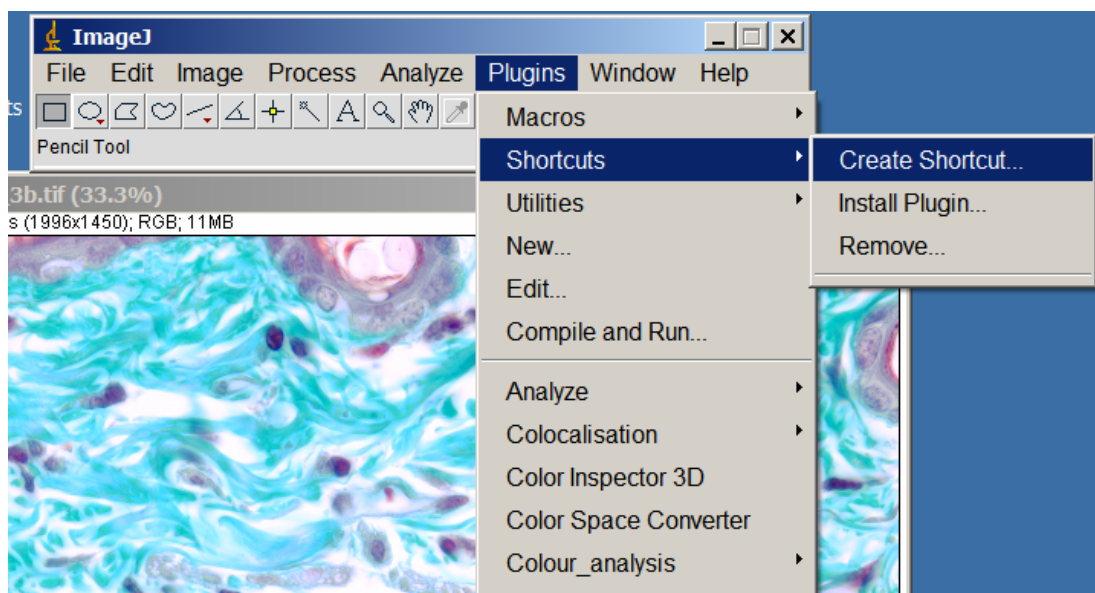
You can use Function keys [f1] as well as numeric [n0] and other keys. You need to have the NumLock key on if you have a PC for this to work. Alphabetic keys won't work if Plugins are already using them.

Multiple macros

You can have multiple macros written into one macro file and run them separately by installing them using the Macro editor.

For example, two macros have been written sequentially into a text window as given below.
The first macro "8bit" converts the image into 8 bit and the second macro ("Binarise") converts it to a binary image. The braces {} are used to break up the two macros.
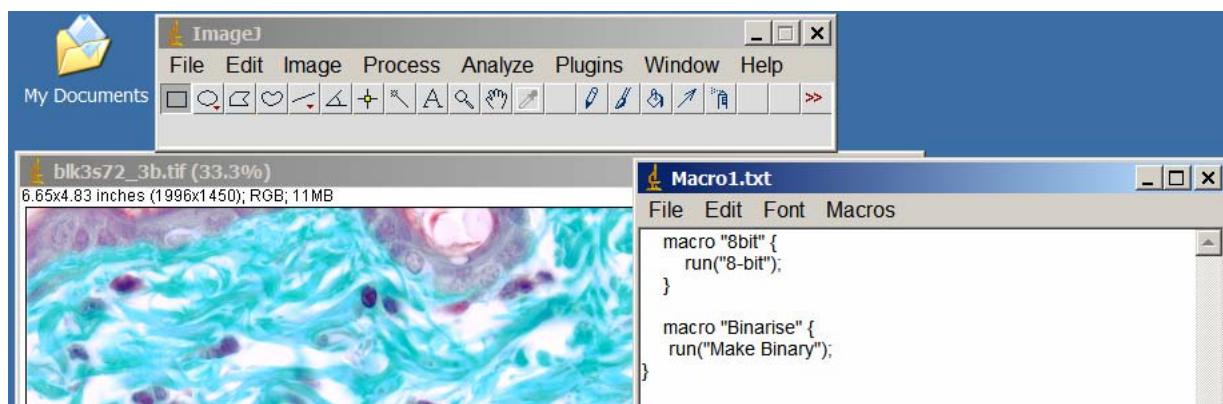
Go to *Macros – Install Macros*.
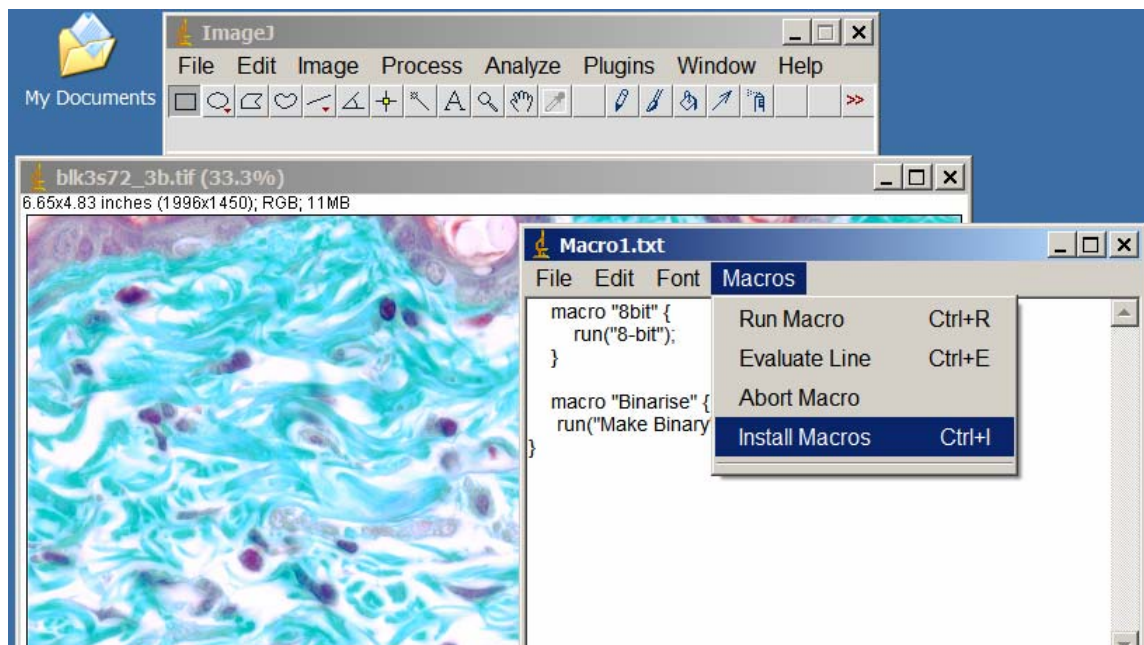
Both macros will then appear in the **Macros** list:

They will also appear under the **Plugins** menu. If you want to use them again, you must save the file.
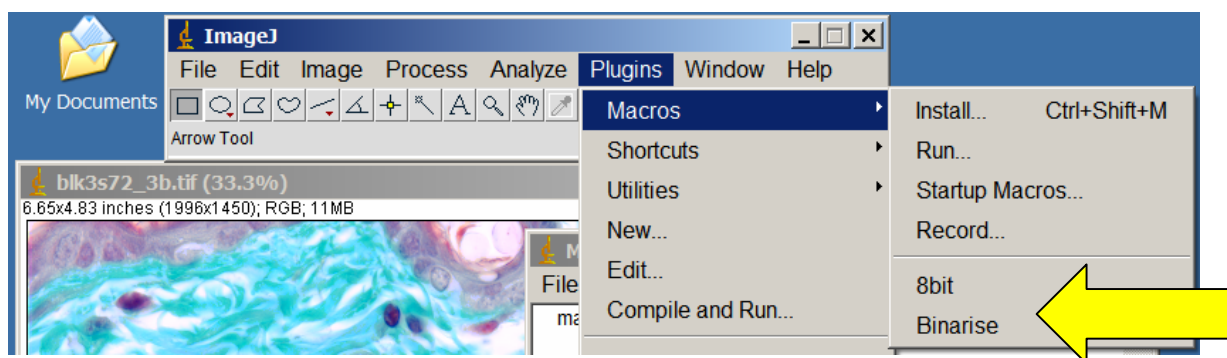
Batch Mode

Batch mode allows you to get the macro to run much faster.

setBatchMode(arg)
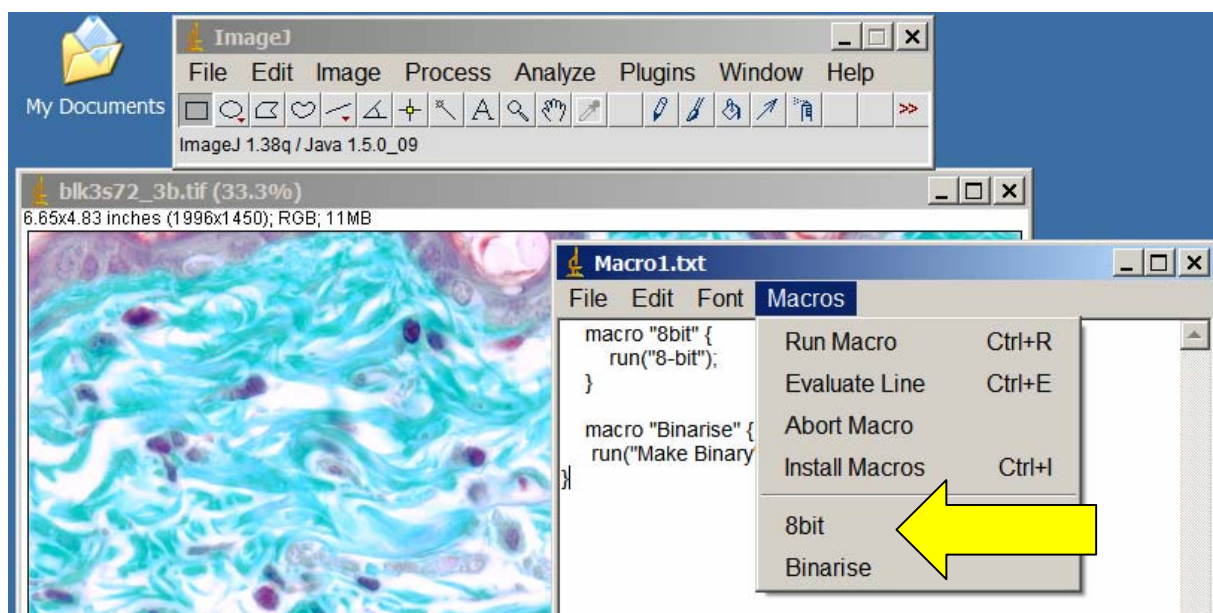
There are two options for the argument (arg), **true** or **false**. If arg is **true**, then the macro runs much faster (up to 20x) since only the final image or results are displayed, i.e. you don't see the intermediate steps.

If arg is false, then it exits batch mode and displays the active image. ImageJ will exit batch mode at the end of the macro if there is no setBatchMode(false) call. With ImageJ 1.37j or later, set arg to "exit and display" to exit batch mode and display all open batch mode images.

Example: ReplaceRedWithMagenta

Batch Processing

Batch processing can be done by using the macro to create a file list, path, etc. The macro will run until all the files in a selected folder have been processed.

Example1 :   Batch Convert to Binary

```
// This macro converts all the files in a folder to binary. The binary
// files are saved in the same folder with a "-bin.tif" suffix.

  macro "Batch Convert to Binary" {
      requires("1.33s");
      dir = getDirectory("Choose a Directory ");
      list = getFileList(dir);
      setBatchMode(true);
      for (i=0; i<list.length; i++) {
         path = dir+list[i];
         open(path);
         run("8-bit");
         setAutoThreshold();
         run("Apply LUT");
         dotIndex = lastIndexOf(path, ".");
         if (dotIndex!=-1)
            path = substring(path, 0, dotIndex); // remove extension
         save(path+"-bin.tif");
         close();
      }
  }
```

Example2 : BatchConvert

This macro allows all files in a folder to be converted and then saved in a different directory.

```
macro "Batch Convert to BMP" {convert("bmp");}
function convert(format) {
requires("1.33s");
dir1 = getDirectory("Choose Source Directory ");
dir2 = getDirectory("Choose Destination Directory ");
list = getFileList(dir1);
setBatchMode(true);
for (i=0; i<list.length; i++) {
showProgress(i+1, list.length);
open(dir1+list[i]);
if (startsWith(format, "8-bit")) convertTo8Bit();
saveAs(format, dir2+list[i]);
close();
}
}

function convertTo8Bit() {
if (bitDepth==24)
run("8-bit Color", "number=256");
else
run("8-bit");
 }
```

Some Basics

;   A semi-colon is required at the end of each line of the macro.

//   If you want to add a comment within the macro for yourself or to explain something to other users, then use //. This indicates that the text which follows is not a command but just a comment so ImageJ will ignore the rest of that line.

+   Can be used as an operator to connect 2 strings
E.g. distance = rate * time;

v   To initialise <u>local</u> variables that will be used in a single function. Can be numerical, text or arrays, don't have types. E.g. **v = 3.79;**

var   To initialise <u>global</u> variables that will be used in more than one function or macro.

=   Used in assignment statements for initialization of variables as above. (Don't need to be declared as in Java!)

==   Used for comparisons

run   Commands which are derived from menu items can be used by including them within brackets and quotation marks.
          E.g. **run("Select All");** creates a selection of the entire image.
          E.g. **run("Despeckle")**; runs the despeckle filter.
          E. g. **run("Duplicate...", "title=yellow.tif");** duplicates the image and names it yellow.tif
          E.,g. **run("Red");** changes the LUT to red.

set   Processes which require input values require a set command.
          E.g. **setThreshold(x,y);** allows you to set particular threshold values.
          setAutoThreshold(); sets the threshold levels based on the histogram of the current image or selection. This is the same as using "Auto" under *Image>Adjust>Threshold.*

          However, if you want to <u>set</u> measurements, you need to run that as a command
          E.g. **run("Set Measurements...", "area standard display redirect=None decimal=3");**

get   Allows you to get information (i.e. values) required by the macro such as size, pixel values, coordinates, etc.
          e.g. **getMinAndMax(min, max);** Gets the minimum and maximum displayed pixel values.

close();   Closes the window

saveAs   To save to a particular format, need to give path as well.

           e.g **saveAs("Gif", "C:\\Documents and Settings\\jros013\\Desktop\\Red.gif");**

update Results()   Updates the results window

print   Shows text and result in a log window.

           e.g. **print("The image title is " + getTitle);**

String   string of one or more text characters between quotation marks, can be joined together

           using **+** as shown above **("The image title is")**

If/else   Allows a statement to be executed if a certain condition is met.

Loop   Allows repetition of macro (dependent on conditions, e.g. **for**, **while** or **do…while**)

References

ImageJ Macro Language

http://rsb.info.nih.gov/ij/developer/macro/macros.html

Good overview of the macro language in ImageJ.

Plugins Menu
http://rsb.info.nih.gov/ij/docs/menus/plugins.html#macros

Useful information about the macros submenu

Index of ImageJ macros

http://rsb.info.nih.gov/ij/macros/

Macros are available here indexed alphabetically. It's sometimes worth checking here first to see if a suitable macro has already been written or if there is one (or more) that might be able to be modified for your purposes.

Java/Javascript Programming Books

David J. Eck, Introduction to Java Programming http://math.hws.edu/javanotes/

Michael Moncur, Sams Teach Yourself JavaScript In 24 Hours, Sams Publishing, 2000.

Walter Savich, Absolute Java, Pearson Education, 2004.