



# Introduction to image processing and analysis with ImageJ / Fiji.

## Part 7

### Macros

Course by Dale Moulding



# Session 7

40 minute lecture

4 hours exercises

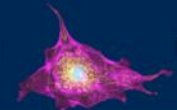
## Learning objectives:

- Record ImageJ commands in the in-built macro recorder
- Write a macro to count objects
- Use macros to process multiple files (whole folders at a time)
- Write conditional loops  
If / else, while, for...
- Save results directly from a macro
- Adapt existing Macros for your own use



## Why use macros?

- Macros further automate your analysis
- You can perform all ImageJ functions and plugins directly from a macro
- Macros can do tasks that cannot be done with simple ImageJ commands
- Once written you just run the macro and all analysis is performed without any further input
- You can make semi-automatic macros – you may need to fine tune a step of the analysis, so the macro will pause, present the image processing step for manual fine tuning, then continue



## Online guides

[https://imagej.net/Introduction into Macro Programming](https://imagej.net/Introduction%20into%20Macro%20Programming)

Fantastic intro to macros

<https://imagej.net/developer/macro/functions.html>

Dictionary of commands

[←](#)
[→](#)
[↻](#)

<https://imagej.net/developer/macro/functions.html>

[home](#) | [news](#) | [docs](#) | [download](#) | [plugins](#) | [resources](#) | [list](#) | [links](#)

Built-in Macro Functions

[A][B][C][D][E][F][G][H][I][J][K][L][M][N][O][P]

[Q][R][S][T][U][V][W][X][Y][Z]

Print List

A [ Top ]

abs(*n*)

Returns the absolute value of *n*.

acos(*n*)

Returns the inverse cosine (in radians) of *n*.

Array Functions

These functions operate on arrays. Refer to the [ArrayFunctions](#) macro for examples.

Array.concat(array1,array2) - Returns a new array created by joining two or more arrays or values ([examples](#)).

Array.copy(array) - Returns a copy of *array*.

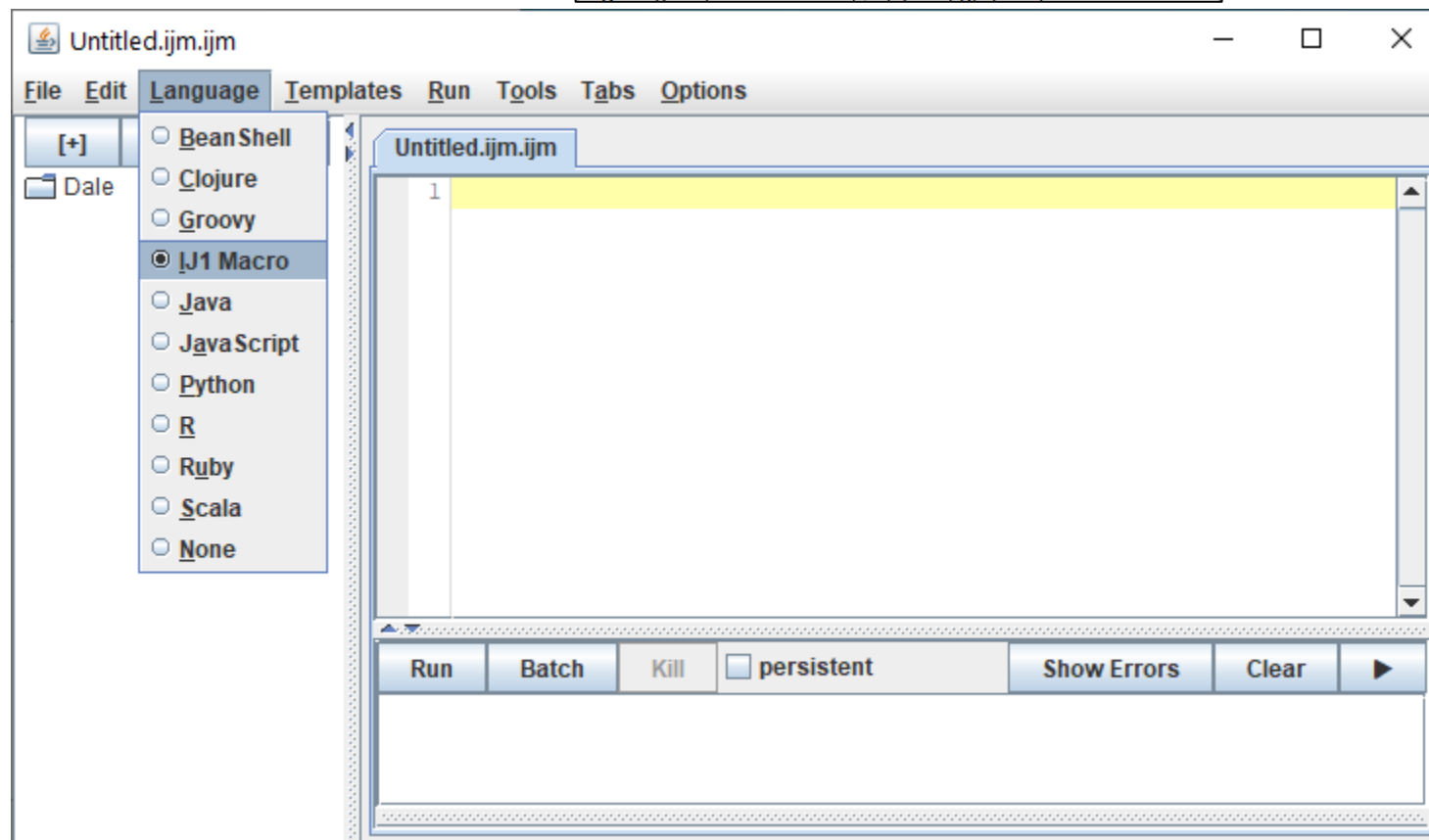
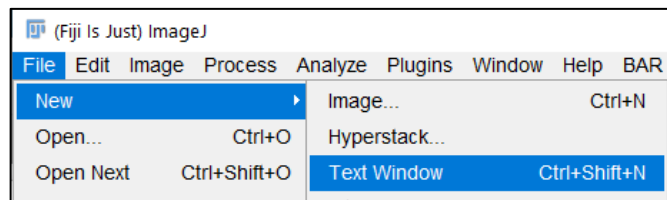
Array.deleteValue(array, value) - Returns a version of *array* where all numeric or string elements in the array that contain *value* have been deleted ([examples](#)). Requires 1.52o.

Array.deleteIndex(array, index) - Returns a version of *array* where the element with the specified index has been deleted. Requires 1.52o.



## Macro tips – open the macro editor

*File / New > Text Window*

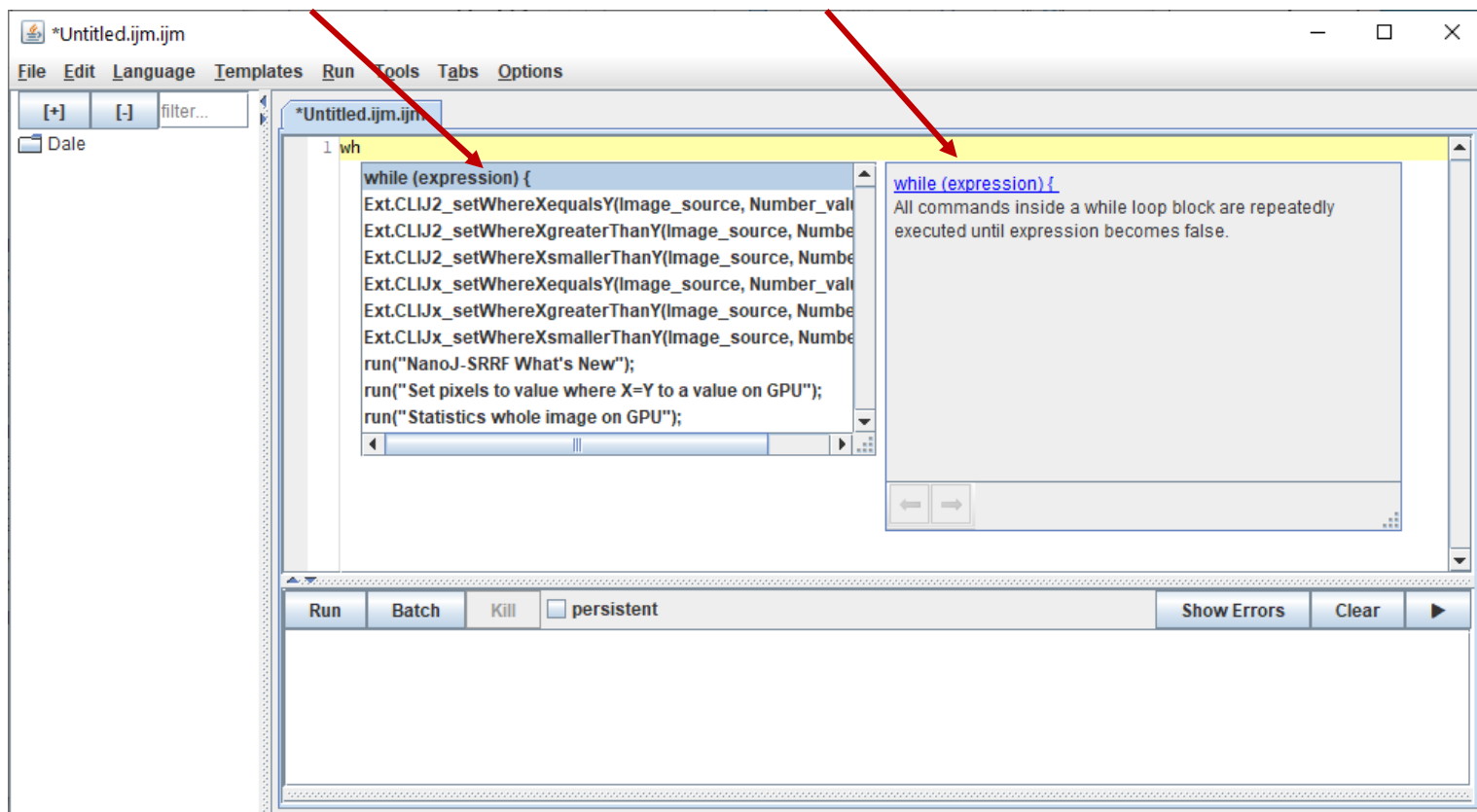




## Macro tips – built in helper tool

Potential commands

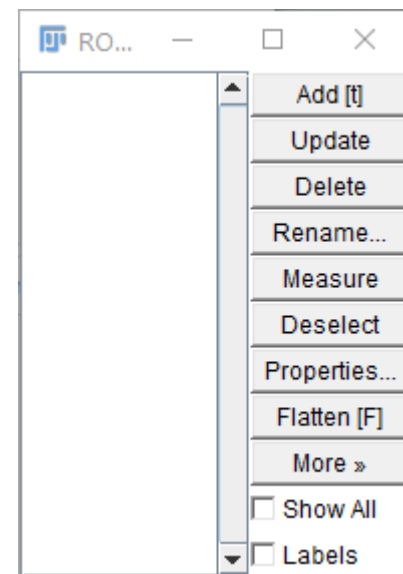
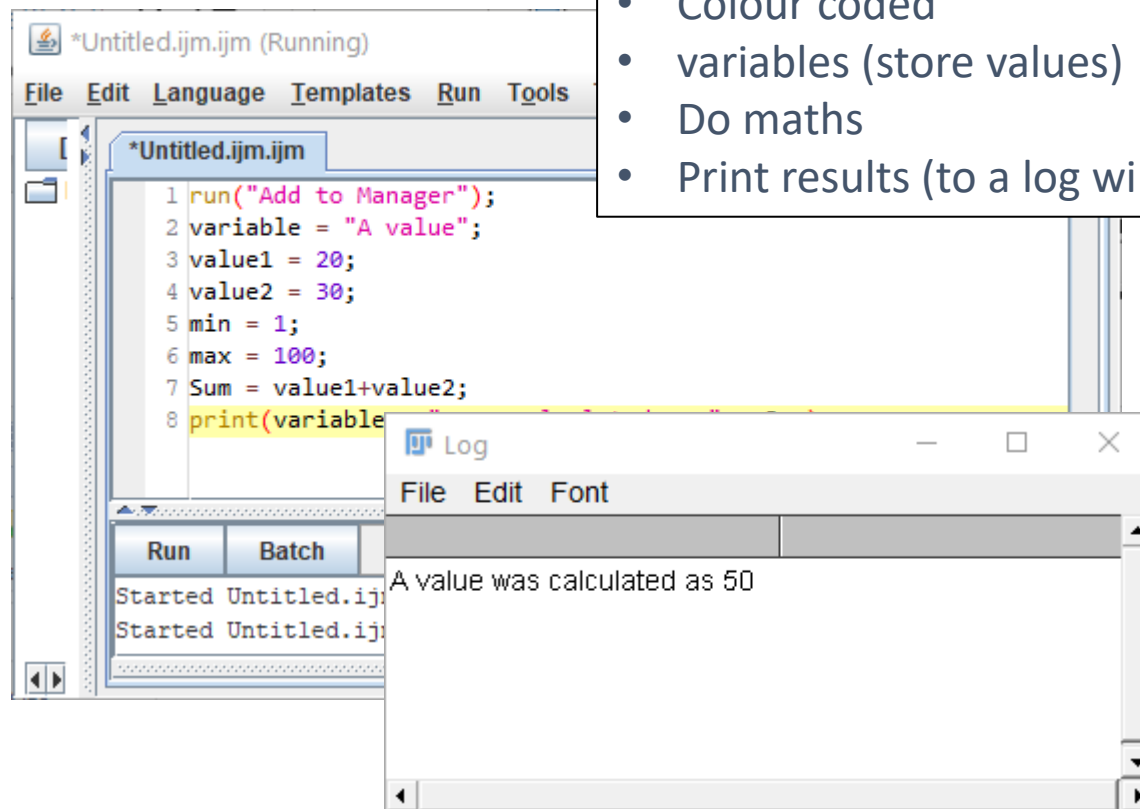
Documentation





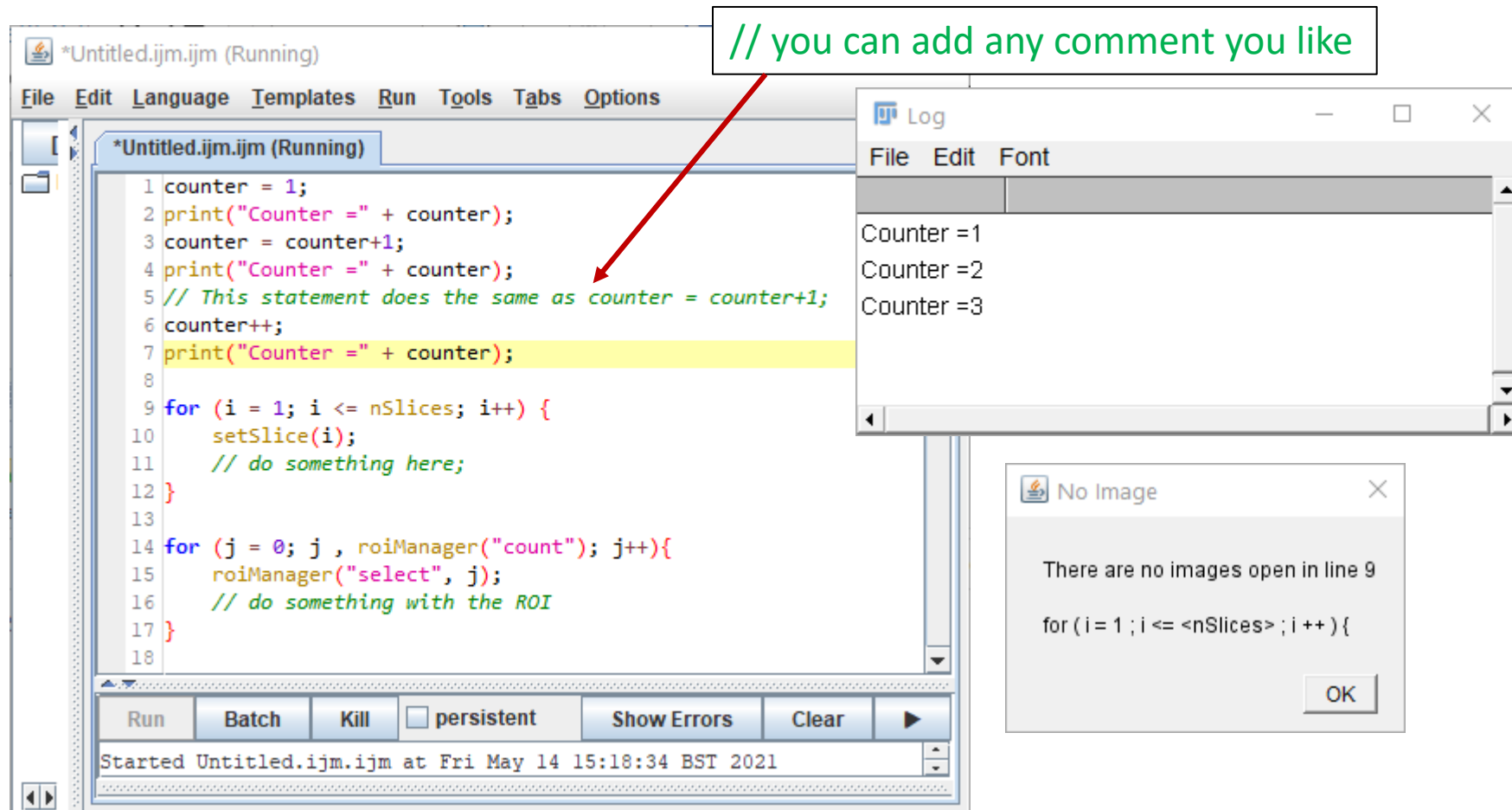
## Macro tips – variables, maths, print

- Colour coded
- variables (store values)
- Do maths
- Print results (to a log window)





## Macro tips – counters, loops & comments



The screenshot shows the Fiji macro editor with a macro named `*Untitled.ijm.ijm (Running)`. The macro code is as follows:

```

1 counter = 1;
2 print("Counter =" + counter);
3 counter = counter+1;
4 print("Counter =" + counter);
5 // This statement does the same as counter = counter+1;
6 counter++;
7 print("Counter =" + counter);
8
9 for (i = 1; i <= nSlices; i++) {
10     setSlice(i);
11     // do something here;
12 }
13
14 for (j = 0; j , roiManager("count"); j++){
15     roiManager("select", j);
16     // do something with the ROI
17 }
18

```

A red arrow points from a green text box to the comment on line 5:

`// you can add any comment you like`

The Log window shows the output of the macro:

```

Counter =1
Counter =2
Counter =3

```

The No Image dialog box is also visible, indicating an error in the macro:

```

There are no images open in line 9

for (i = 1 ; i <= <nSlices> ; i ++ ) {

```

The status bar at the bottom indicates the macro was started on Fri May 14 15:18:34 BST 2021.





## Macro tips – measurements, loops, modify results tables

BlobsDemov001.ijm (Running)

File Edit Language Templates Run Tools Tabs Options

\*Untitled.ijm BlobsDemov001.ijm


```

1 run("Blobs (25K)");
2 setOption("BlackBackground", true);
3 run("Convert to Mask");
4 run("Set Measurements...", "area feret's redirect=None decimal=
5 run("Analyze Particles...", "display clear summarize add");
6   for (i = 0; i < nResults(); i++) {
7       v1 = getResult('Area', i);
8       v2 = getResult('Feret', i);
9       setResult('Area / Feret', i, v1/v2);
10  }
11 updateResults();
12

```

blobs.gif

256x254 pixels; 8-bit; 64K



Results

File Edit Font Results

	Area	Feret	FeretX	FeretY	FeretAngle	MinFeret	Area / Feret
1	433	36.620	15	30	55.008	17.814	11.824
2	185	21.378	53	0	169.216	11.000	8.654
3	658	33.015	102	27	54.866	27.000	19.930
4	434	27.203	149	22	53.973	23.000	15.954
5	477	31.780	243	29	65.854	19.000	15.009
6	285	21.840	194	27	74.055	18.000	13.049
7	81	12.042	133	26	41.634	9.000	6.727
8	278	23.087	216	39	72.350	17.000	12.042
9	231	19.209	39	34	51.340	16.000	12.025
10	30	14.036	0	20	94.086	3.000	2.137
11	501	27.459	167	45	56.889	24.684	18.245
12	660	35.000	62	54	53.130	26.386	18.857
13	99	13.416	5	51	63.435	10.000	7.379
14	228	19.313	231	57	68.749	16.000	11.805
15	448	26.926	137	42	105.068	22.000	16.638
16	401	35.468	190	76	68.499	15.696	11.306

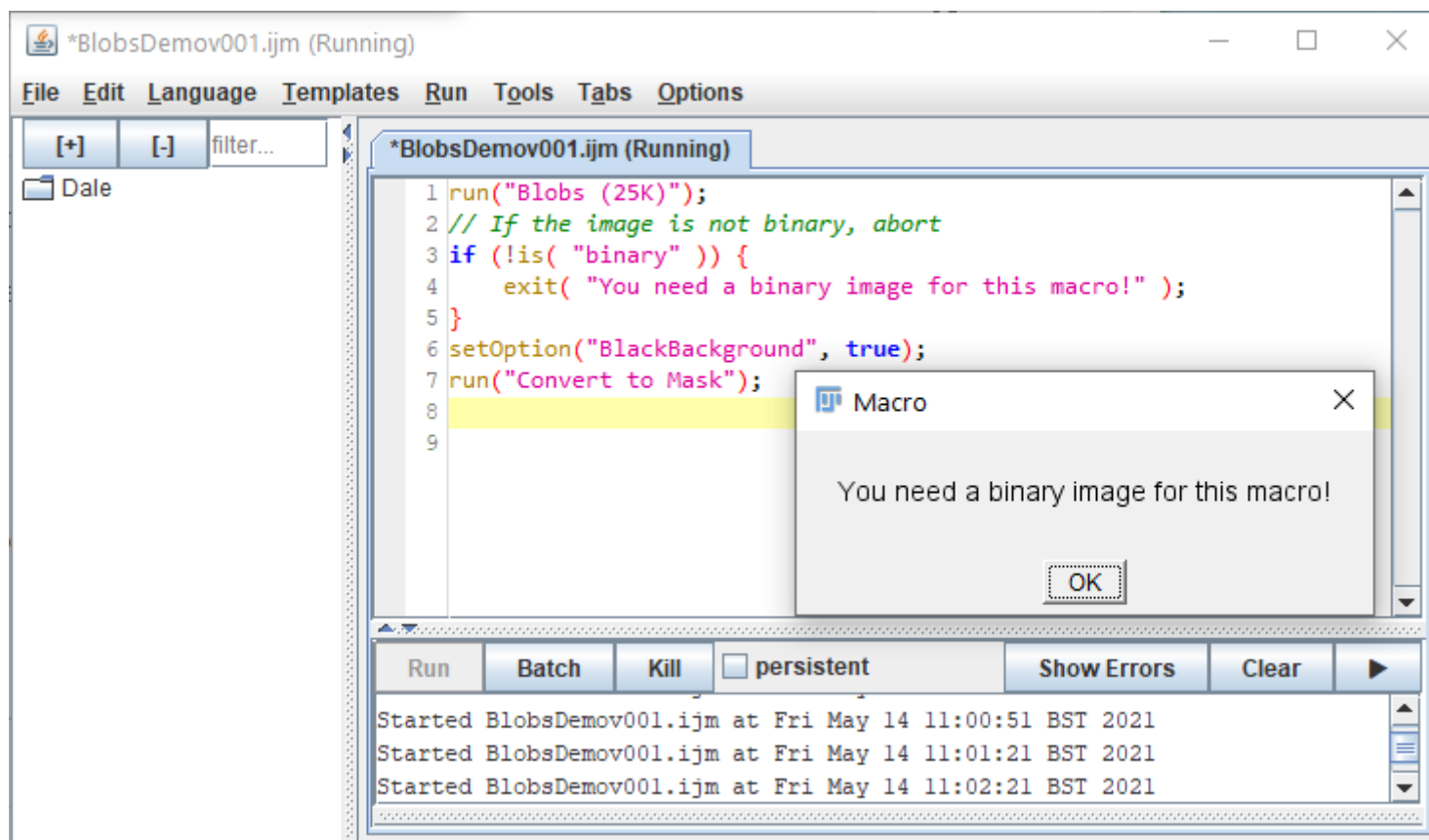
Summary

File Edit Font

Slice	Count	Total Area	Average Size	%Area	Feret	FeretX	FeretY
blobs.gif	64	22243	347.547	34.207	24.875	123.828	



## Macro tips – conditional code: if



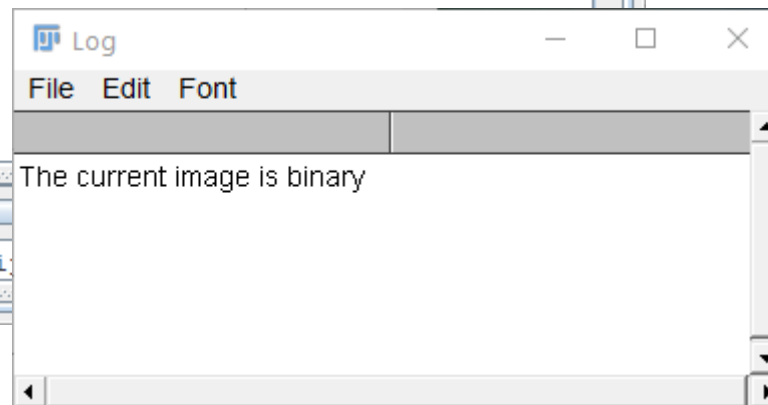


## Macro tips – conditional code: if / else

```
*BlobsDemov001.ijm (Running)
File Edit Language Templates Run Tools Tabs Options

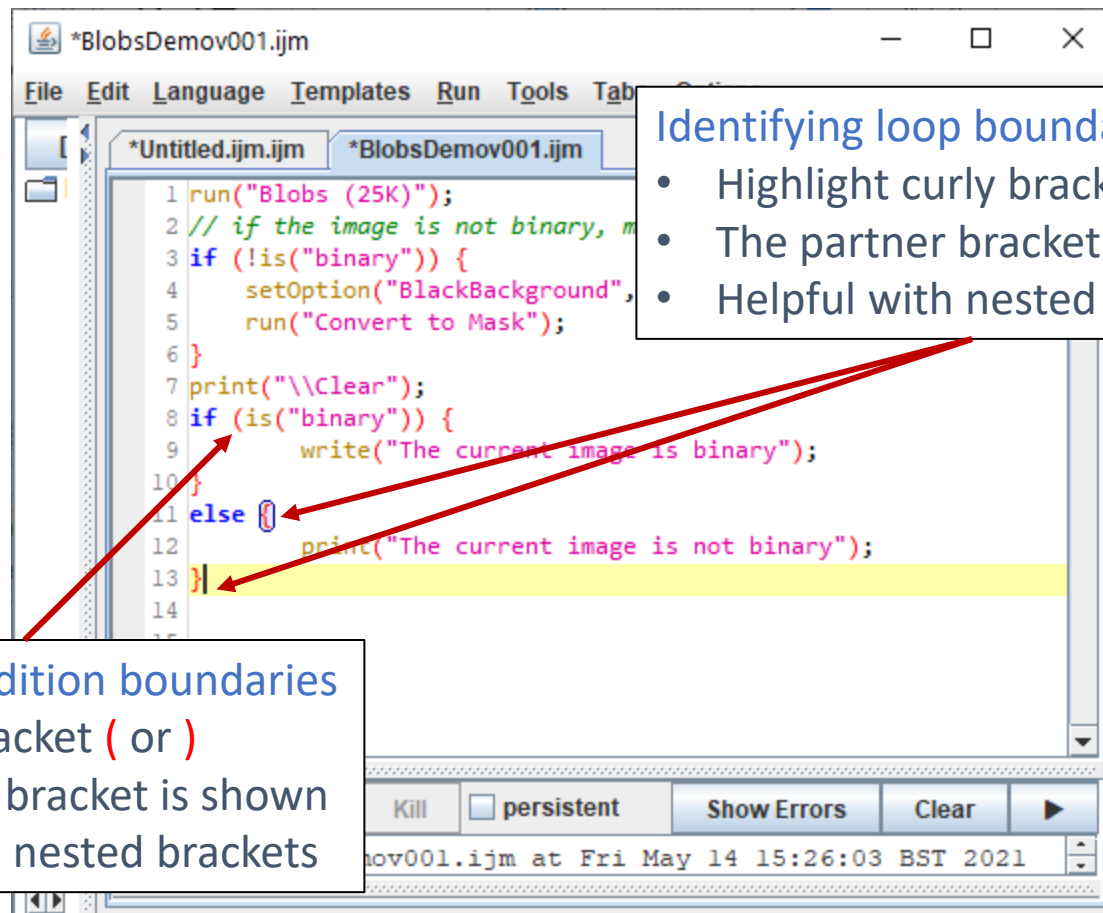
*Untitled.ijm.ijm *BlobsDemov001.ijm (Running)

1 run("Blobs (25K)");
2 // if the image is not binary, make it binary
3 if (!is("binary")) {
4     setOption("BlackBackground", true);
5     run("Convert to Mask");
6 }
7 print("\\Clear");
8 if (is("binary")) {
9     write("The current image is binary");
10 }
11 else {
12     print("The current image is not binary");
13 }
14
15
16
17
18
19
20
21
22
23
24
25
```





## Macro tips – conditional code: if / else



The screenshot shows the IJ Macro Editor window titled '\*BlobsDemov001.ijm'. The code is as follows:

```

1 run("Blobs (25K)");
2 // if the image is not binary, m
3 if (!is("binary")) {
4     setOption("BlackBackground",
5     run("Convert to Mask");
6 }
7 print("\\Clear");
8 if (is("binary")) {
9     write("The current image is binary");
10 }
11 else {
12     print("The current image is not binary");
13 }
14

```

Annotations include:

- A red box highlights the `else {` line (line 11).
- Red arrows point from the `else {` line to the 'Identifying loop boundaries' box.
- Red arrows point from the `else {` line to the 'Identifying condition boundaries' box.

### Identifying loop boundaries

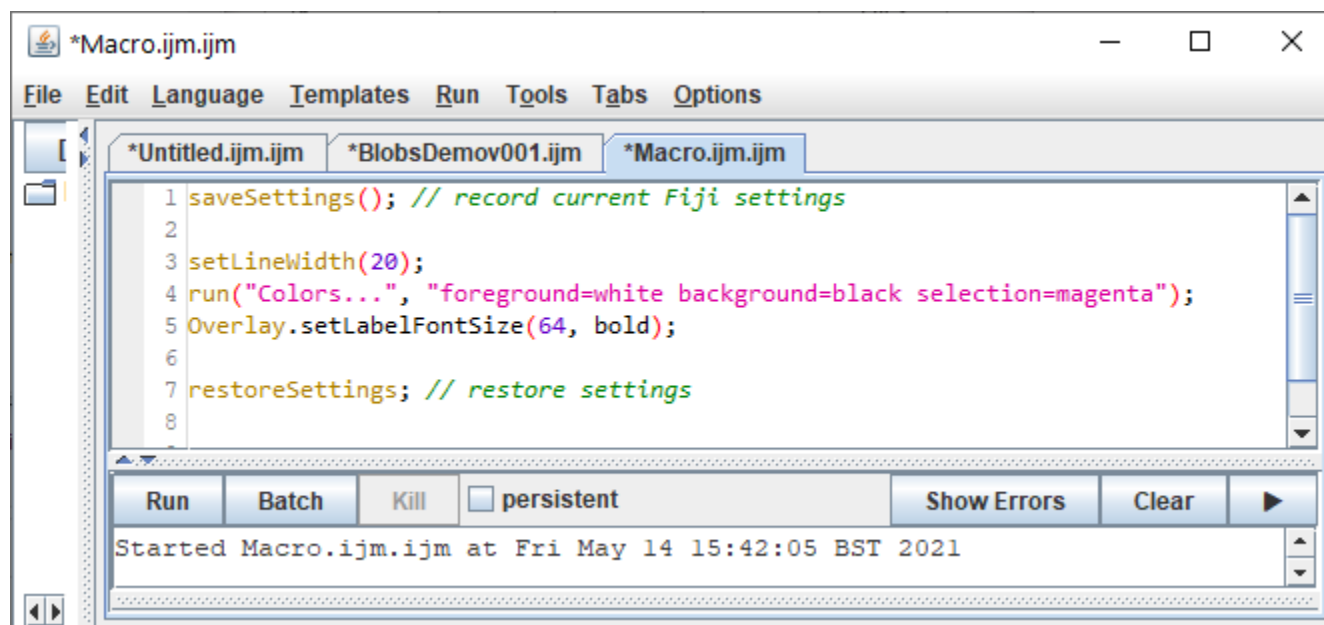
- Highlight curly bracket { or }
- The partner bracket is shown
- Helpful with nested loops

### Identifying condition boundaries

- Highlight bracket ( or )
- The partner bracket is shown
- Helpful with nested brackets



## Save and restore Fiji settings in a macro





## Macro tips – opening files in folders & saving results

```
dir1 = getDirectory("Input folder"); //select an input folder  
dir2 = getDirectory("Choose a folder to save to"); //select an output  
list = getFileList(dir1); //make a list of the filenames
```

Ask for folders of files and  
where to save the results



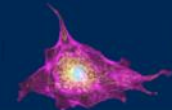
## Macro tips – opening files in folders & saving results

```
dir1 = getDirectory("Input folder"); //select an input folder
dir2 = getDirectory("Choose a folder to save to"); //select an output
list = getFileList(dir1); //make a list of the filenames
```

Ask for folders of files and  
where to save the results

```
// repeats the macro for every file in the folder using a for loop.
for (i=0; i<list.length; i++) {
    showProgress(i+1, list.length);
    filename = dir1 + list[i];
    open(filename);
}
```

Loop the macro to measure  
every file



## Macro tips – opening files in folders & saving results

```
dir1 = getDirectory("Input folder"); //select an input folder
dir2 = getDirectory("Choose a folder to save to"); //select an output
list = getFileList(dir1); //make a list of the filenames
```

Ask for folders of files and  
where to save the results

```
// repeats the macro for every file in the folder using a for loop.
for (i=0; i<list.length; i++) {
    showProgress(i+1, list.length);
    filename = dir1 + list[i];
    open(filename);
}
```

Loop the macro to measure  
every file

```
ImageName = File.nameWithoutExtension;
```

Keep a record of the current filename

Macro performs measurements...





## Macro tips – opening files in folders & saving results

```
dir1 = getDirectory("Input folder"); //select an input folder
dir2 = getDirectory("Choose a folder to save to"); //select an output
list = getFileList(dir1); //make a list of the filenames
```

Ask for folders of files and where to save the results

```
// repeats the macro for every file in the folder using a for loop.
for (i=0; i<list.length; i++) {
    showProgress(i+1, list.length);
    filename = dir1 + list[i];
    open(filename);
```

Loop the macro to measure every file

```
ImageName = File.nameWithoutExtension;
```

Keep a record of the current filename

Macro performs measurements...

```
saveAs("Tif", dir2+ImageName+"-cilia");//saves an image of the detected
```

```
selectWindow("Results");
saveAs("Results", dir2+ImageName + "-Results.csv"); // saves results
roiManager("Save", dir2+ImageName + "-RoiSet.zip"); // saves the ROIs
```

```
selectWindow("Summary");
saveAs("Results", dir2+"Summary.csv");// save the summary as a .csv.
exit("Cilia measured in "+i+" images");// close the macro and display a window
```

### Save

- Images
- Results tables
- ROIs in the ROI manager
- Results summary
- Close the macro



## Macro tips – naming & installing macros

```

HelloWorldMacro.ijm
1 macro "Hello World Macro" {} // give it a name here
2   write("Hello, world!");
3 }
    
```

Named **macro** must be enclosed in curly brackets { }

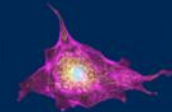
To appear as a Fiji menu item the saved macro must have an underscore in the name  
i.e. Hello\_World.ijm

Save your macro in a "Plugins" subfolder of  
./Fiji.app/scripts/

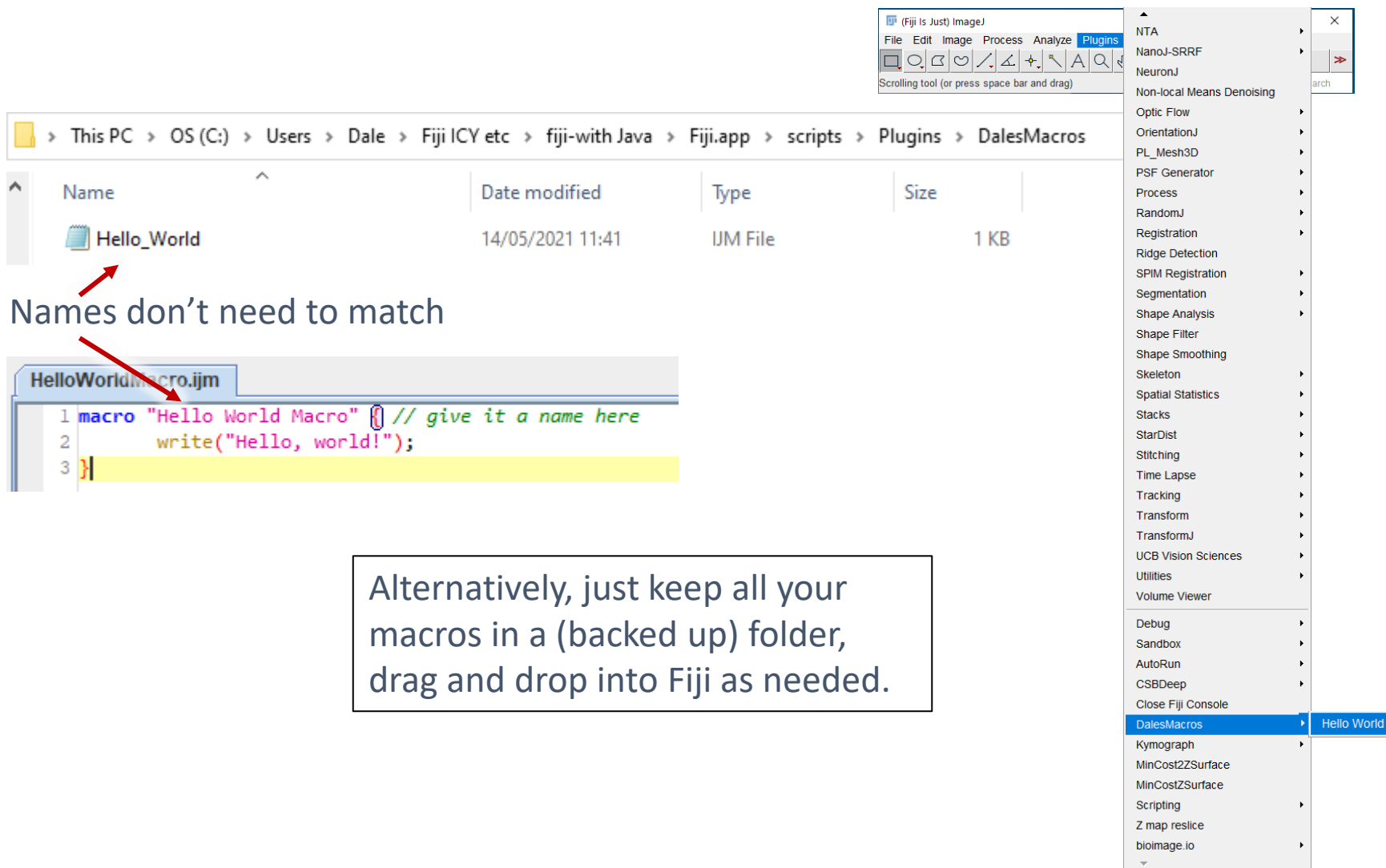
(e.g. ./Fiji.app/scripts/Plugins/MyScripts/My\_Macro.ijm),

and it will appear in the respective menu

(e.g. *Plugins > MyScripts > My Macro*) upon restart of Fiji



## Macro tips – naming & installing macros



Names don't need to match

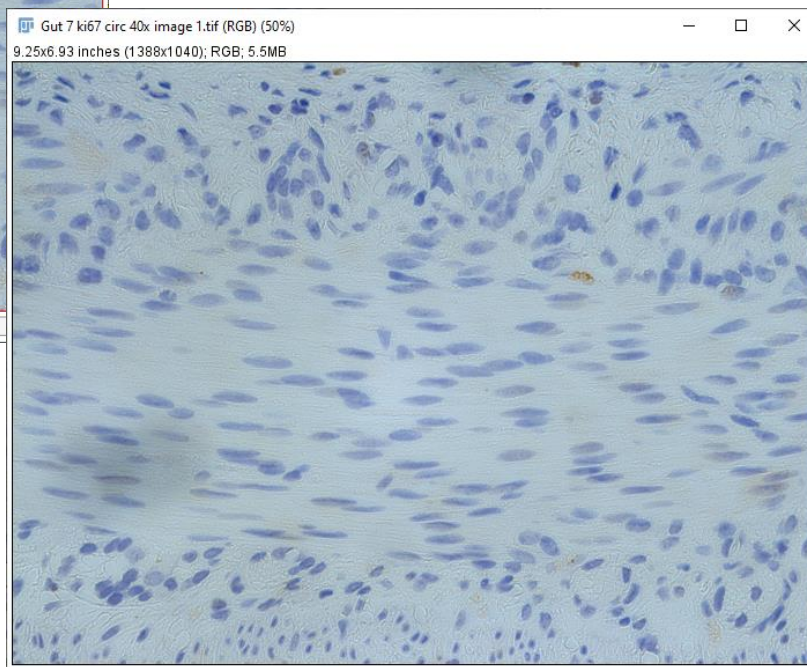
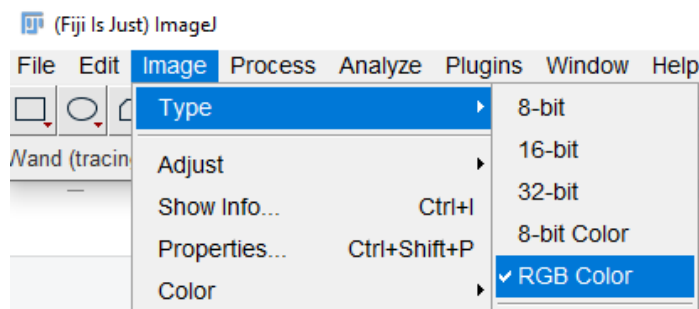
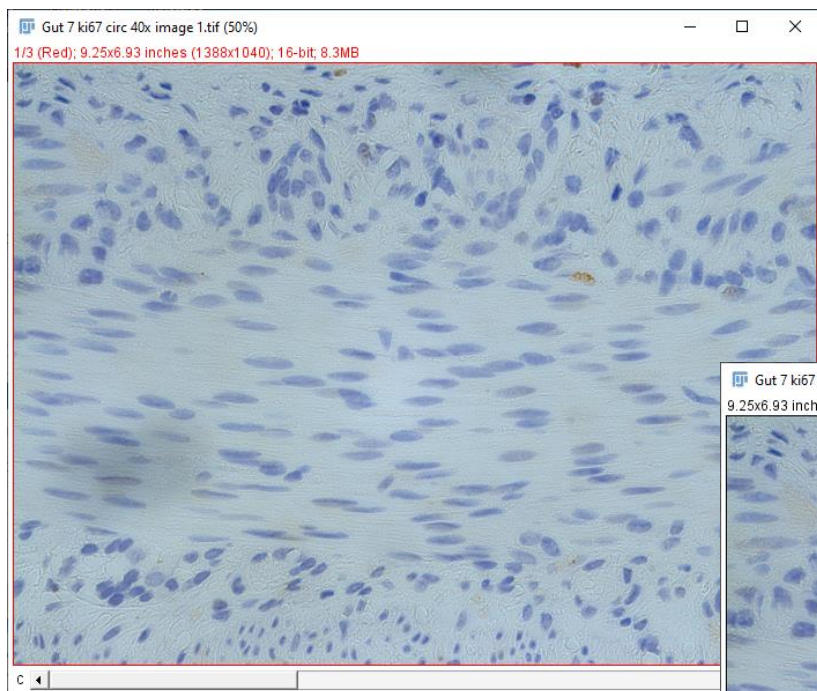
```

1 macro "Hello World Macro" // give it a name here
2   write("Hello, world!");
3 }
  
```

Alternatively, just keep all your macros in a (backed up) folder, drag and drop into Fiji as needed.



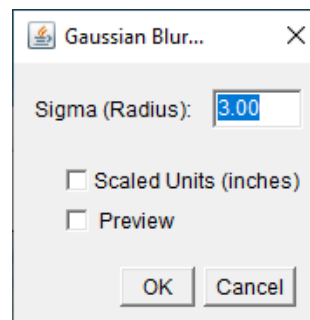
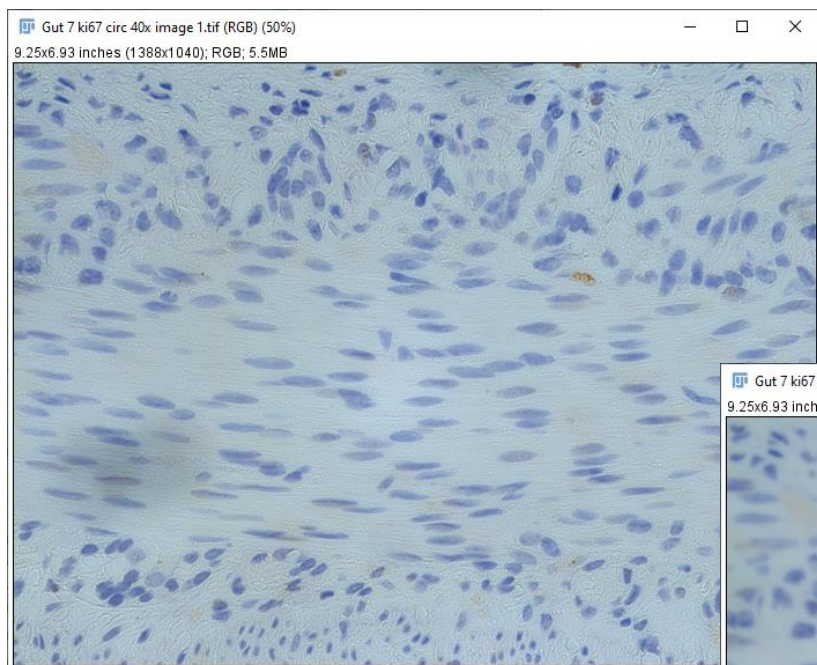
## My first macro...



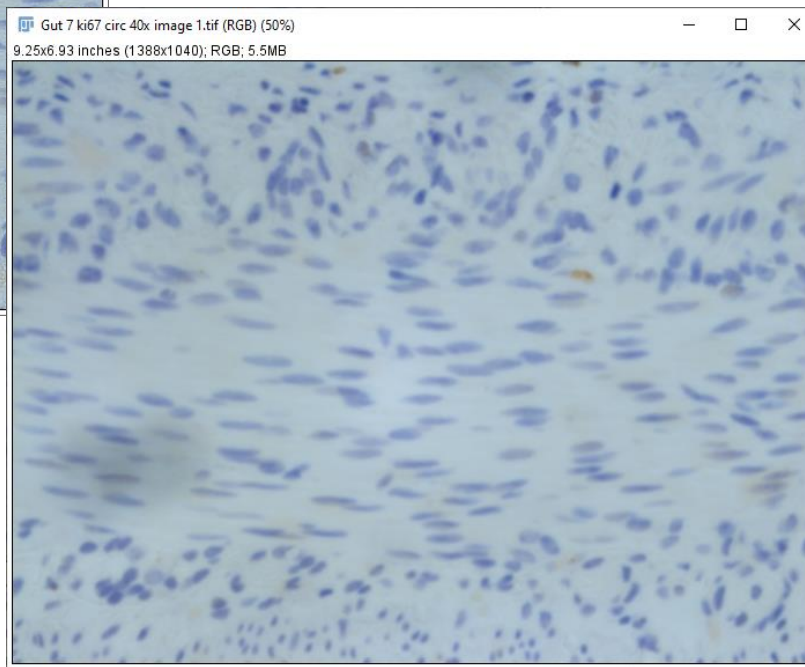
**1) Image / Type / RGB Color**



## My first macro...



**2) Process / Filters > Gaussian**







## My first macro...

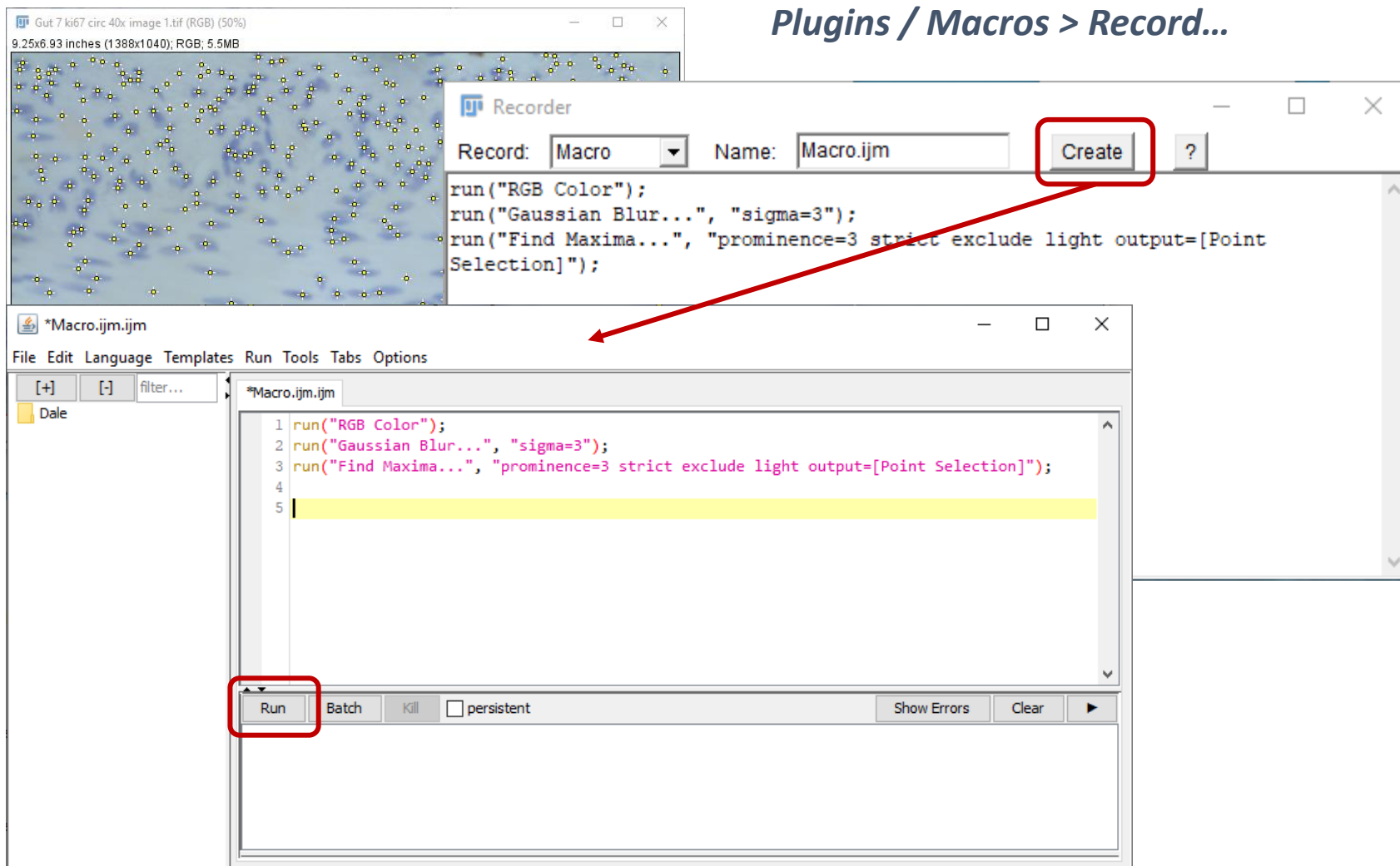


**3) Process / Find maxima...**



## My first macro...

*Plugins / Macros > Record...*



The Recorder window shows the following macro code:

```
run("RGB Color");
run("Gaussian Blur...", "sigma=3");
run("Find Maxima...", "prominence=3 strict exclude light output=[Point Selection]");
```

The Macro Editor window shows the same code in a text editor. The 'Run' button is highlighted with a red box.



## My first macro...

Lets test it. Open the next image, run the macro... Cells counted, saves 3 steps.







## My first macro...

```
1 run("RGB Color");
2 run("Gaussian Blur...", "sigma=3");
3 run("Find Maxima...", "prominence=3 strict exclude light output=[Point Selection]");
4
```

```
1 run("RGB Color");
2 run("Gaussian Blur...", "sigma=3");
3 do("Find Maxima...", "prominence=3 strict exclude light output=[Point Selection]");
4 doCommand("Command");
5 doWand(x, y);
  doWand(x, y, tolerance, mode);
  #@ double(value=25, min=0, max=1, style="spinner") realNumber;
  Array.fourier(array, windowType);
  Ext.CLIJ2_downsample2D(Image_source, Image_destination, Number);
  Ext.CLIJ2_downsample3D(Image_source, Image_destination, Number);
  Ext.CLIJ2_downsampleSliceBySliceHalfMedian(Image_source, Image_destination, Number);
  Ext.CLIJ2_setRandom(Image_source, Number_minimumValue, Number_maximumValue);
  Ext.CLIJ2_thresholdOtsu(Image_input, Image_destination);
  Ext.CLIJ_downsample2D(Image_source, Image_destination, Number);
```

doCommand("Command");

Runs an ImageJ menu command in a separate thread and returns immediately. As an example, *doCommand("Start Animation")* starts animating the current stack in a separate thread and the macro continues to execute. Use *run("Start Animation")* and the macro hangs until the user stops the animation.



## My first macro...

```
1 run("RGB Color");  
2 run("Gaussian Blur...", "sigma=3");  
3 run("Find Maxima...", "prominence=3 strict exclude light output=[Point Selection]");  
4
```

```
1 run("RGB Color");  
2 run("Gaussian Blur...", "sigma=3");  
3 doCommand("Find Maxima...");  
4
```

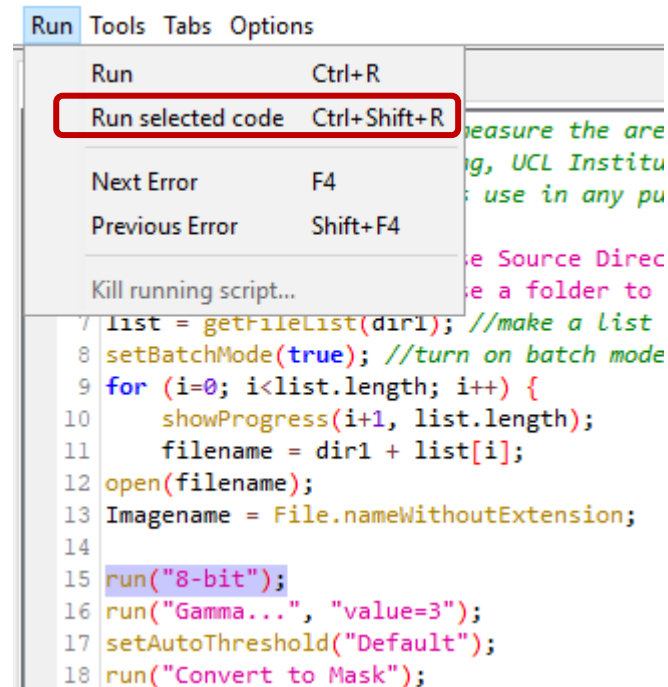
**doCommand** – doesn't complete the action, it opens the window, then you can fine tune the parameters

**run("Find Maxima...");**  
This now does the same as **doCommand**



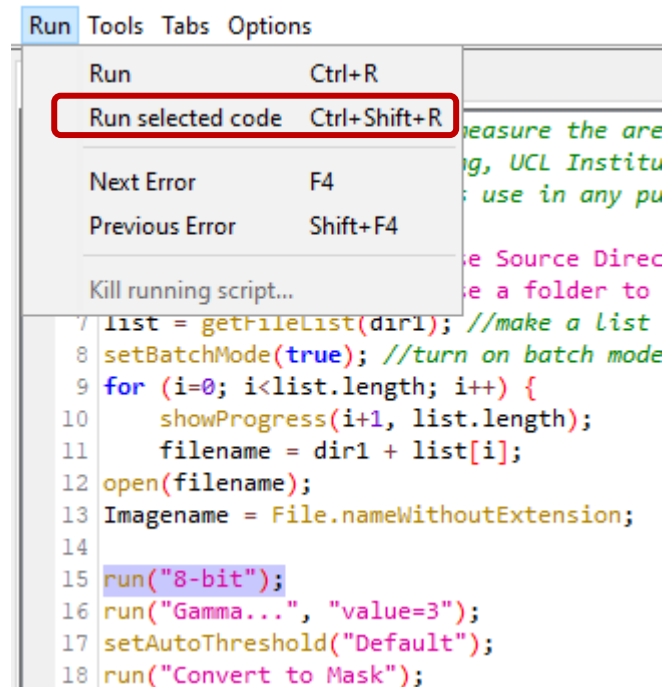
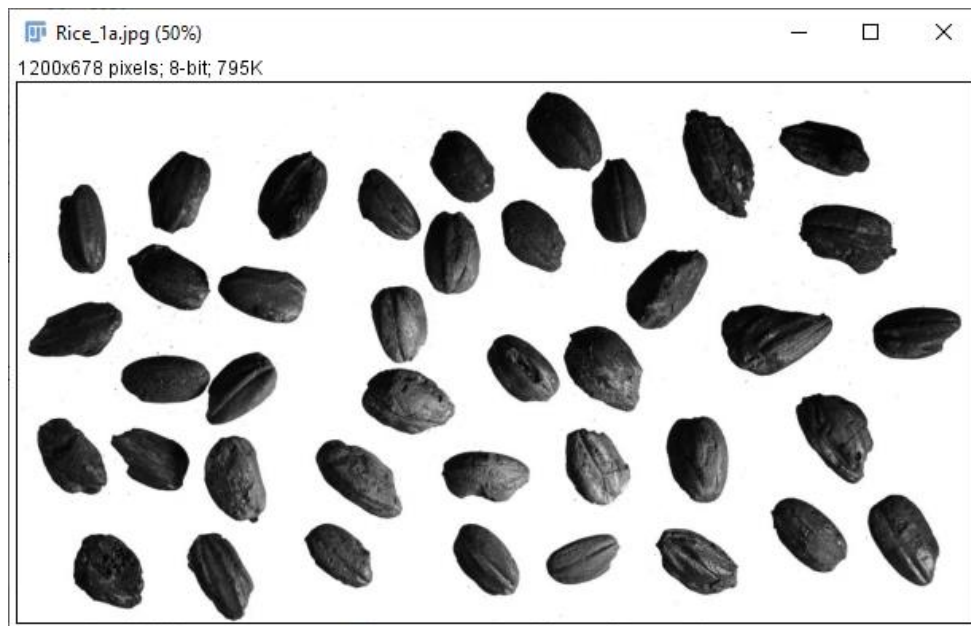


## Checking your macro one step at a time



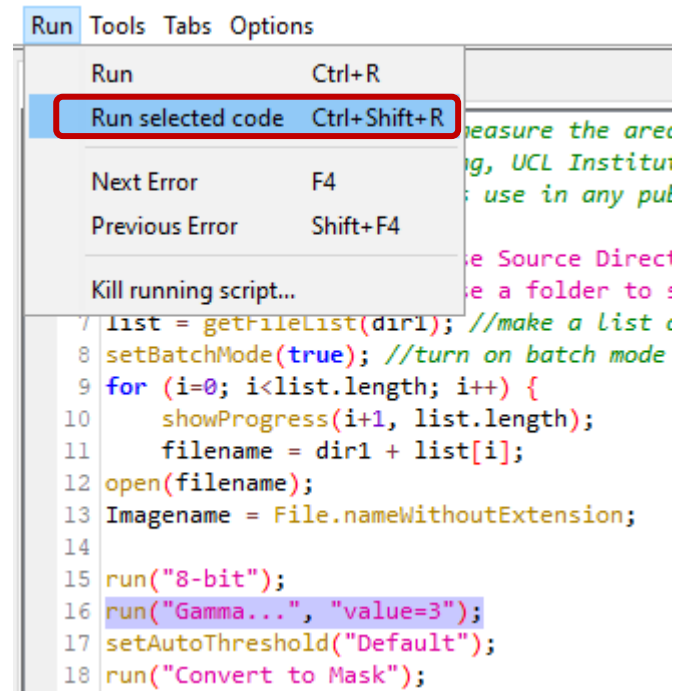
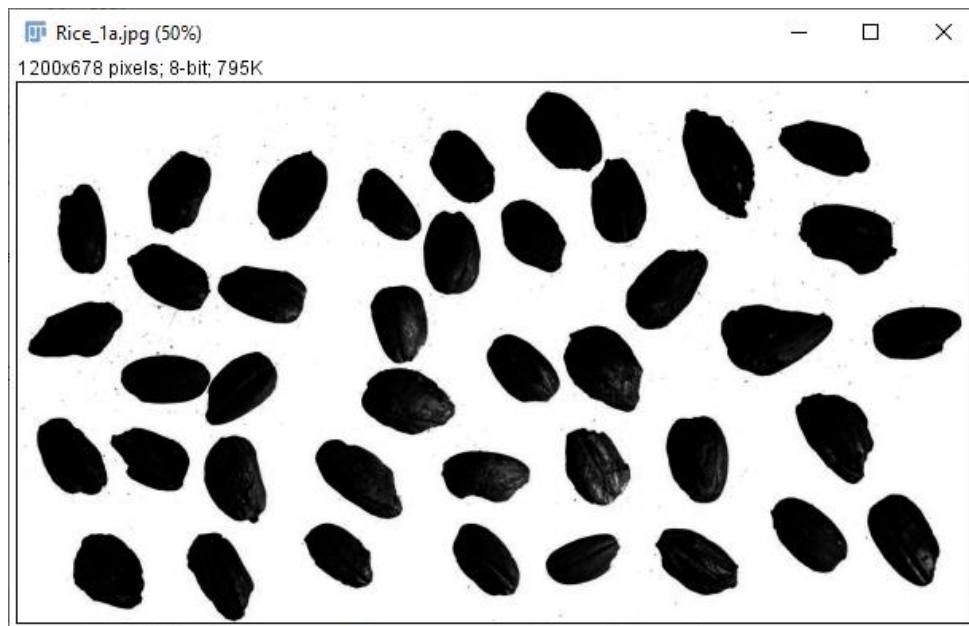


## Checking your macro one step at a time





## Checking your macro one step at a time





## Checking your macro one step at a time

```

1 // macro to identify and measure the area, length and width of grains
2 // Written by Dale Moulding, UCL Institute of Child Health Imaging Facility, August 2015
3 // Please acknowledge it's use in any publications.
4
5 dir1 = getDirectory("Choose Source Directory "); //select an input folder
6 dir2 = getDirectory("Choose a folder to save to"); //select an output folder.
7 list = getFileList(dir1); //make a list of the filenames
8 setBatchMode(true); //turn on batch mode
9 for (i=0; i<list.length; i++) {
10     showProgress(i+1, list.length);
11     filename = dir1 + list[i];
12     open(filename);
13     Imagename = File.nameWithoutExtension;
14
15     run("8-bit");
16     run("Gamma...", "value=3");
17     setAutoThreshold("Default");
18     run("Convert to Mask");
19     run("Set Measurements...", "area fit shape redirect=None decimal=3");
20     run("Analyze Particles...", "size=100-30000 show=Ellipses display clear include add in_situ");
21     run("Input/Output...", "jpeg=85 gif=-1 file=.csv use_file copy_column copy_row save_column save_row");
22     // runs the Edit>Input/Output command and adds saving the column name to the results table
23     saveAs("Results", dir2+Imagename + ".csv"); // saves results as a table for excel
24     roiManager("Save", dir2+Imagename + "RoiSet.zip"); // saves the ROIs
25 }
--

```

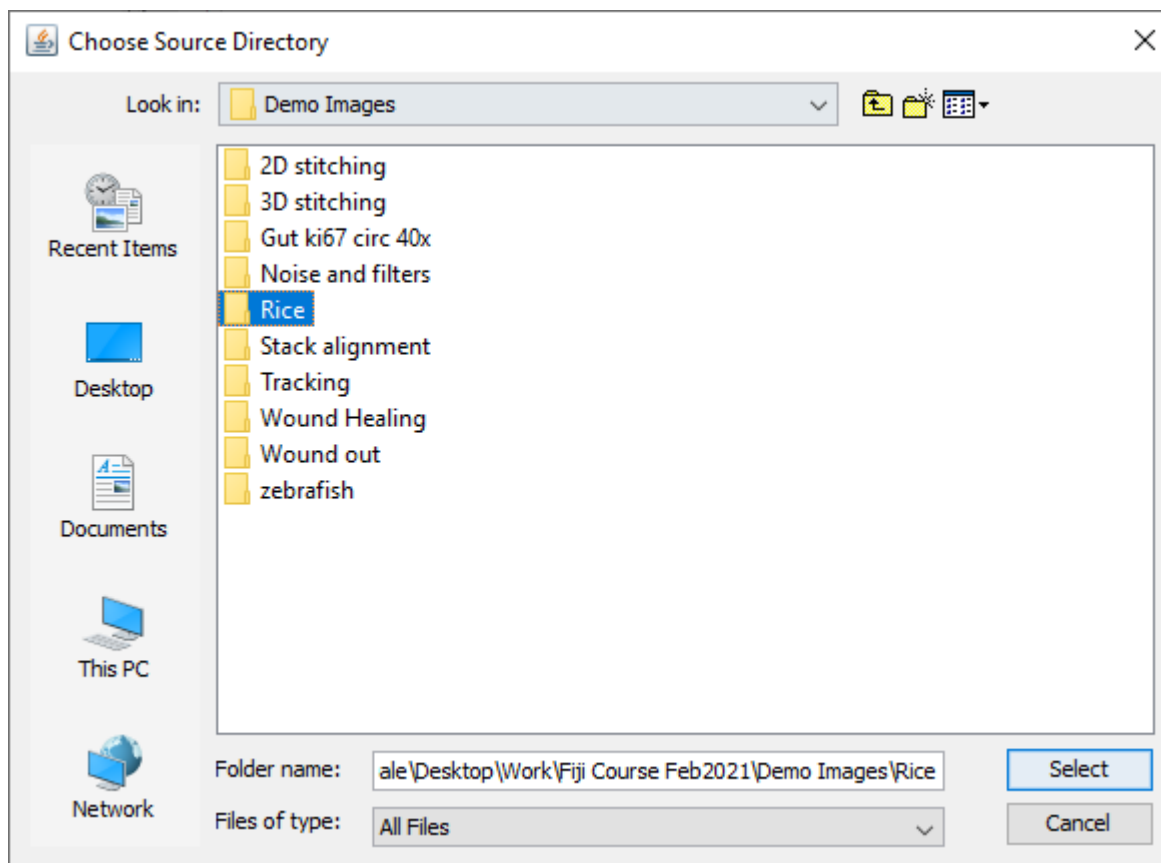
- Threshold
- Convert to mask (binary)
- Set measurements
- Analyze particles...
- Save the results...





## Macro to analyse a folder of files










Run the macro... it asks where the files are, then where to save the results...





## Macro to analyse a folder of files

Results...

Name	Date modified	Type	Size
 Rice_1a	13/05/2021 17:14	Microsoft Excel C...	3 KB
 Rice_1aEllipses	13/05/2021 17:14	JPG File	187 KB
 Rice_1aRoiSet	13/05/2021 17:14	Compressed (zipp...	20 KB
 Rice_2c	13/05/2021 17:14	Microsoft Excel C...	3 KB
 Rice_2cEllipses	13/05/2021 17:14	JPG File	229 KB
 Rice_2cRoiSet	13/05/2021 17:14	Compressed (zipp...	23 KB
 Rice_3a	13/05/2021 17:14	Microsoft Excel C...	2 KB
 Rice_3aEllipses	13/05/2021 17:14	JPG File	137 KB
 Rice_3aRoiSet	13/05/2021 17:14	Compressed (zipp...	14 KB



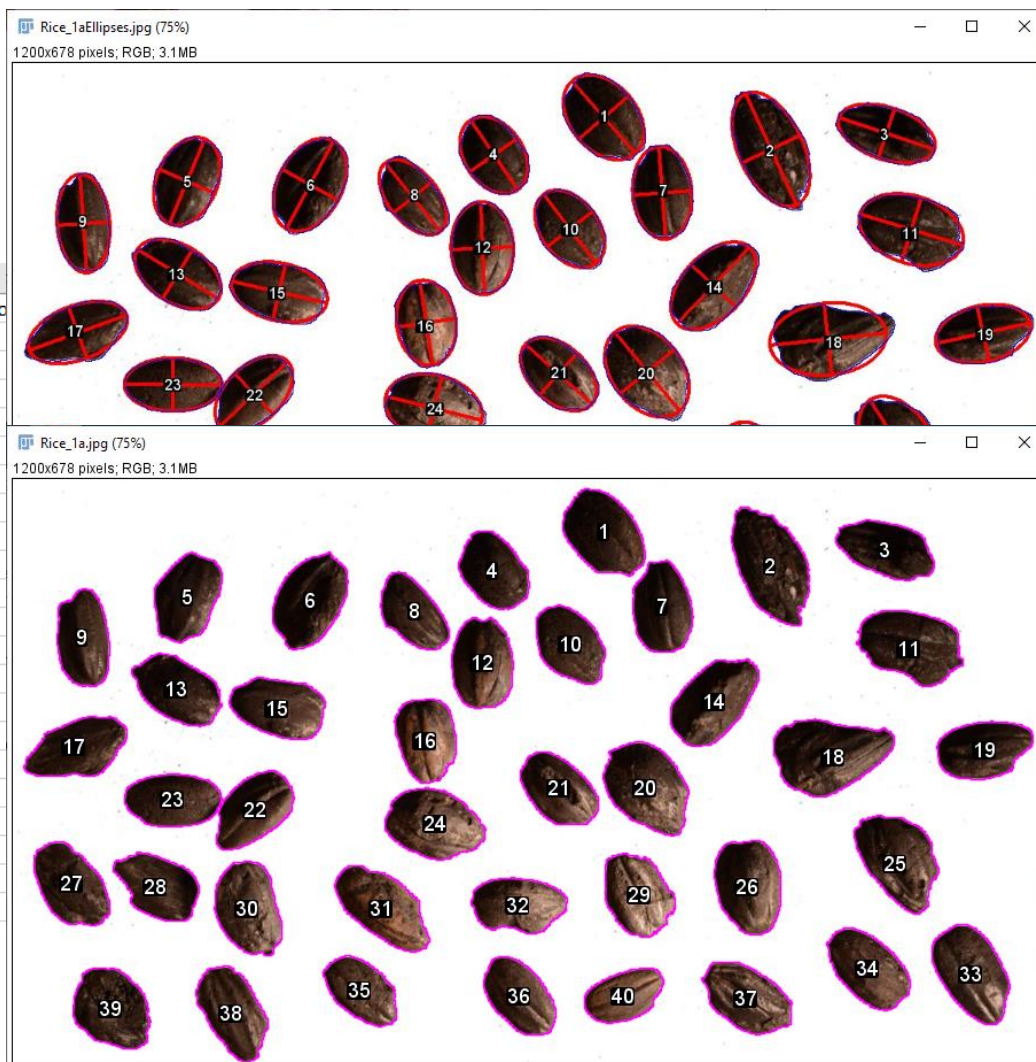


## Macro to analyse a folder of files

Results...

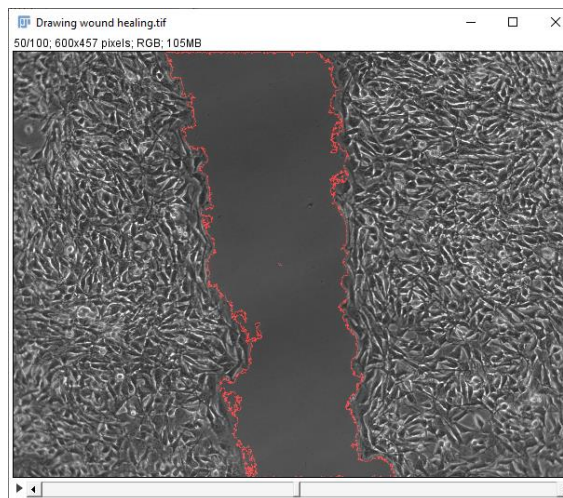
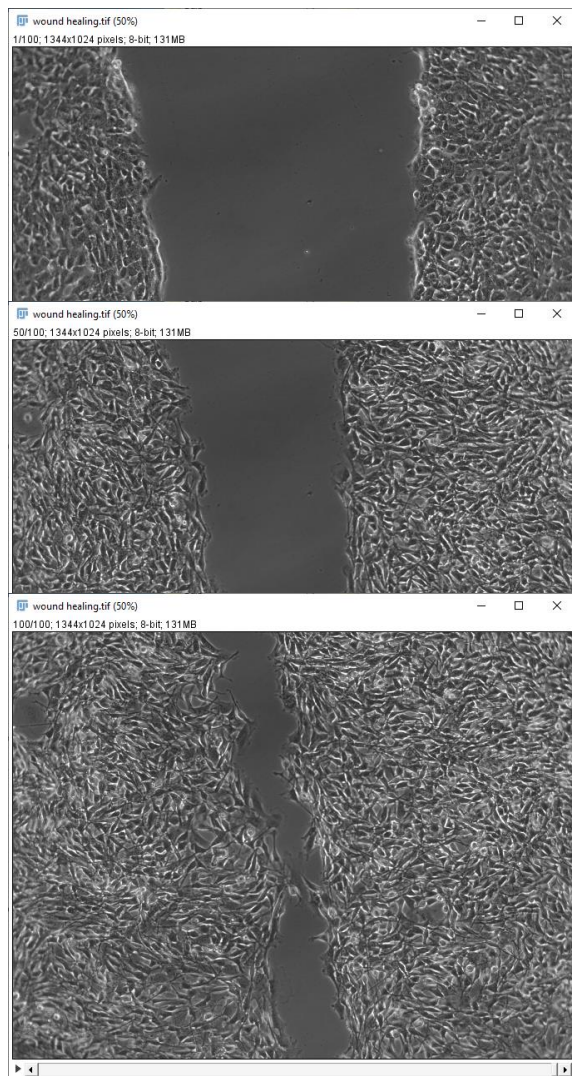
Always save images to show  
what was measured

	A	B	C	D	E	F	G	
1		Area	Major	Minor	Angle	Circ.	AR	Ro
2	1	6774	111.748	77.182	131.112	0.838	1.448	
3	2	8378	141.747	75.255	114.73	0.685	1.884	
4	3	5573	117.144	60.573	161.753	0.728	1.934	
5	4	5246	97.495	68.51	124.408	0.844	1.423	
6	5	5910	106.392	70.727	66.876	0.809	1.504	
7	6	6799	115.789	74.763	61.509	0.811	1.549	
8	7	5798	107.512	68.665	94.524	0.806	1.566	
9	8	4638	101.823	57.996	129.009	0.775	1.756	
10	9	5511	114.357	61.359	94.413	0.745	1.864	
11	10	5312	100.509	67.292	127.488	0.82	1.494	
12	11	7657	124.573	78.261	162.795	0.733	1.592	
13	12	6117	107.393	72.522	92.025	0.833	1.481	
14	13	5824	110.078	67.364	148.734	0.779	1.634	
15	14	6906	122.067	72.034	46.478	0.8	1.695	
16	15	5952	113.377	66.842	167.205	0.787	1.696	
17	16	5380	99.25	69.018	98.572	0.81	1.438	
18	17	5941	120.013	63.029	20.073	0.721	1.904	
19	18	8901	135.657	83.543	5.981	0.739	1.624	
20	19	5725	112.066	65.045	12.398	0.767	1.723	
21	20	7589	118.133	81.795	129.186	0.796	1.444	
22	21	5345	103.708	65.621	137.864	0.834	1.58	





## Example Macros: Measure wound closure



wound healing.txt				
File Edit Font				
Slice	Count	Total Area	Average Size	%Area
1	1	477594	477594	34.702
2	1	478947	478947	34.801
3	1	480627	480627	34.923
4	1	475940	475940	34.582
5	1	476331	476331	34.611
6	1	473175	473175	34.381
7	1	472244	472244	34.314
8	1	476942	476942	34.655
9	1	469363	469363	34.104
10	1	465667	465667	33.836
11	1	461460	461460	33.530
12	1	459464	459464	33.385
13	1	460596	460596	33.467
14	1	453267	453267	32.935
15	1	449490	449490	32.660
16	1	447224	447224	32.496
17	1	443745	443745	32.243
18	1	438302	438302	31.847
19	1	431619	431619	31.362
20	1	426238	426238	30.971
21	1	420313	420313	30.540
22	1	414973	414973	30.152
23	1	412564	412564	29.977
24	1	408126	408126	29.655
25	1	399864	399864	29.054
26	1	395633	395633	28.747
27	1	390360	390360	28.364
28	1	385178	385178	27.987
29	1	383062	383062	27.834
30	1	373095	373095	27.109
31	1	366453	366453	26.627
32	1	361802	361802	26.289
33	1	355362	355362	25.821
34	1	349969	349969	25.429
35	1	341672	341672	24.826
36	1	336270	336270	24.434
37	1	331954	331954	24.120
38	1	325643	325643	23.662



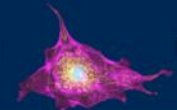
## Macro to measure wound closure

Lets run the Macro in ImageJ one line at a time...

```

1  dir1 = getDirectory("Choose Source Directory ");           // select input directory
2  dir2 = getDirectory("Choose Destination Directory");       // select ouput directory
3  list = getFileList(dir1);                                 // list of file in directory
4  setBatchMode(true);                                       // use this to save time by not displaying images
5
6  for (i=0; i<list.length; i++){                             // start of loop to process each file in the input directory
7      showProgress(i+1, list.length);
8      open(dir1+list[i]);                                    // open file in input directory
9      run("8-bit");                                          // convert to 8-bit
10     run("Duplicate...", "title=copy duplicate");          // duplicate the file
11     selectWindow(list[i]);                                 // select window corresponding to original file
12     run("Sharpen", "stack");                               // run sharpen filter this step really helps a lot for thin cells with thin lamellipodia
13     run("Find Edges", "stack");                           // highlights all edges in the image to increase contrast
14     setThreshold(0,20);                                    // very important to get an appropriate threshold. Threshold can be modifier here
15     run("Convert to Mask", " ");                          // create a mask out of the thresholded image
16     run("Set Measurements...", "area redirect=None decimal=3"); // just measure the area
17     run("Analyze Particles...", "size=12000-Infinity circularity=0.00-1.00 show=Outlines summarize stack"); // analyse the particle (wound), minimal size can be modified here
18     selectWindow("Summary of "+list[i]);                  // select the result window
19     saveAs("Text", dir2+list[i]);                          // save the result as a text file
20     selectWindow("Drawing of "+list[i]);                  // select the window that shows the measured region as an outline
21     run("Red");                                            // the ROI is shown as an outline. here we are making the outline Red.
22     run("Invert LUT");                                     // Make the line Red, and the background black
23     run("RGB Color");                                     // make the image an RGB image to save as a picture
24     selectWindow("copy");                                  // select the original image that was copied in line 10
25     run("RGB Color");                                     // make the image an RGB image to save as a picture
26     imageCalculator("Add stack", "copy", "Drawing of "+list[i]); // image calculator: merge the two pictures (outline and original image)
27     run("Size...", "width=600 constrain interpolate");    // rescale the image to make it smaller so the saved file is smaller
28     saveAs("Tiff", dir2+"Drawing "+list[i]);              // save the overlayed image as a tif
29     close();                                               // there are 3 windows still open, close them all
30     close();
31     close();
32 }
33

```



## Macro exercises

Use the macro recorder to make a macro.

Write a macro to process a folder of files

18) Count the drosophila nuclei in an image to record a macro.  
Use the macro to count the other files.

19) Use your new macro to measure a folder of files.  
This is difficult!

Try the other macros in the course macros folder.

- Western macro
- Montage macros