Tutorial 1 : Basics of Image Processing with ImageJ and Python

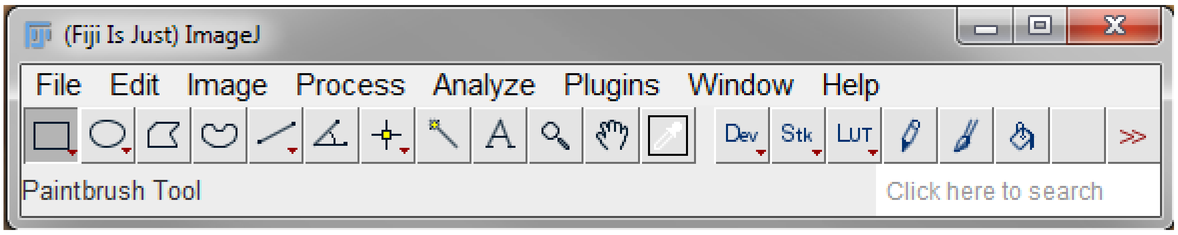
Course : Image Processing and Quantitative Data Analysis
Instructor : Marten Postma
Teaching assistants : Aaron Lin, Aoming Sun, Catherine Chia
Date : June 6th 2023

Table of content

1.0 ImageJ

In this tutorial we will get familiar imaging data using the free image analysis program ImageJ. We will use the extended ImageJ variant called FIJI, which is just the same as standard ImageJ but comes with many extra preinstalled tools, plugins and macros, including BioFormats! ImageJ is written in Java and can be installed on different operating systems as a stand-alone application. You can download FIJI for Windows, Mac and Linux from https://imagej.net/software/fiji/.

After starting FIJI you should see the program layout as shown below. The top row shows a number of menu items and the bottom row a number of icons for specific operations, which can be further customized by clicking the red arrow on the right hand side. At the bottom there is a status bar present with information and a search box for help.



For certain commands ImageJ can work with shortcut keys, in this tutorial we used the Windows notation, for Mac the command key needs to be used instead of the CTRL key. At the bottom right you see a search box that can be used to quickly search for commands without going through the menu.

In this tutorial we will get familiar with using ImageJ and also familiar with imaging data. In case help is needed then first try to find it online e.g. https://imagej.nih.gov/ij/docs/guide/index.html or via the Help menu. Of course the instructors are also present to help!

Using ImageJ

1.1. Open and close images with ImageJ

Digital images you can find on the internet include formats like, tiff, gif, png, bmp, jpeg etc, ImageJ can open these standard file types.

a. Open the image Diatoms.tif using the menu **File > Open** (or CTRL-O)**.** You will see a grayscale *phase contrast* image of diatoms. The title of the window shows the name of the file. The window name can be changed by using **Image -> Rename.**
b. You can close the window by clicking the cross, or by using **File->Close** or CTRL-W, which will close the *currently* active image window. Note: ImageJ commands work on the active window!
c. Using the menu for opening files requires many (mouse) clicks, now try to open the same image by dragging it from the folder onto the status bar at the bottom left. While dragging the status bar will display << Drag and Drop >>. We recommend using this drag-and-drop method.
d. Normally the image file will be completely loaded into memory, which is preferable for speed. However, if data files contain many images and are very large you can also open them as a virtual stack by dragging and dropping the file in the top right region, the status bar should show << Open as Virtual Stack >>. The title of the window now displays a (V) next to the file name after loading. ImageJ will read images from disk and cache when using this approach.

1.2. Zoom and pan:

By using the operations zooming and panning you can inspect the image in more detail.

a. Open the image Diatoms.tif and select the Zoom tool by clicking on the magnifying glass icon.

b. Now Zoom-in by using the left mouse button or use the '+' key. And Zoom-out using the right mouse button or use the '-' key. While holding the CTRL-key you can also zoom in and out with your mouse wheel.
c. Double-click the magnifying glass to reset the zoom factor to 1, which will yield the original size of the image.
d. If the image is larger than the window, i.e. when using a high zoom factor two blue squares will appear in the top left corner indicating which part of the image you are currently looking at. Now select the Pan tool by clicking on the hand icon. By holding the left-mouse button and dragging you can pan the image. While in zoom mode you can also pan by holding the spacebar and drag with the left mouse button.

1.3. Duplicate and crop images or regions

a. It is often useful to duplicate an image before you start applying different operations to an image. Open the image Diatoms.tif and apply **Image -> Duplicate**, or use CTRL-SHIFT-D. You will see a dialogue that allows you to choose a title (name), which will help you to keep track of the different images (under **Window** all the currently open images are listed).
b. You can also select or duplicate a specific region in the image. Choose the rectangle tool by clicking the rectangle icon. Create a rectangle around the upper diatom by holding the left mouse button and dragging. After drawing you can change the size of the rectangle by dragging the white squares; you can position the rectangle by clicking inside the rectangle and dragging. You can use duplicate to make a duplication of the selected region.
c. Select another diatom in the original image and now apply **Image -> Crop**, or use CTRL-SHIFT-X. The original image will be cropped to the selected region.
d. Save the cropped image **File -> Save As**, choose for example Tiff for raw. You can also copy it to system **Edit –> Copy** to System (you can conveniently paste it into a powerpoint or word doc).
e. Also try other region shapes, and see what the effect is when you duplicate or crop.

1.4. Image transforms

a. Check the content of the folder named "IHC images", it contains four immunohistochemistry color images with nuclear staining and anti-CTNNB1. Select all images in the folder and drag and drop the folder onto the status bar.
b. Now use **Image -> Transform** operations to orient the image '20x_5min.tif', such that the outside region of the sample is at the bottom of the image.
c. Now drag the "IHC images" folder on the status bar and open the images. What happened? Go to **Image -> Stacks -> stack to images**, what is the result?

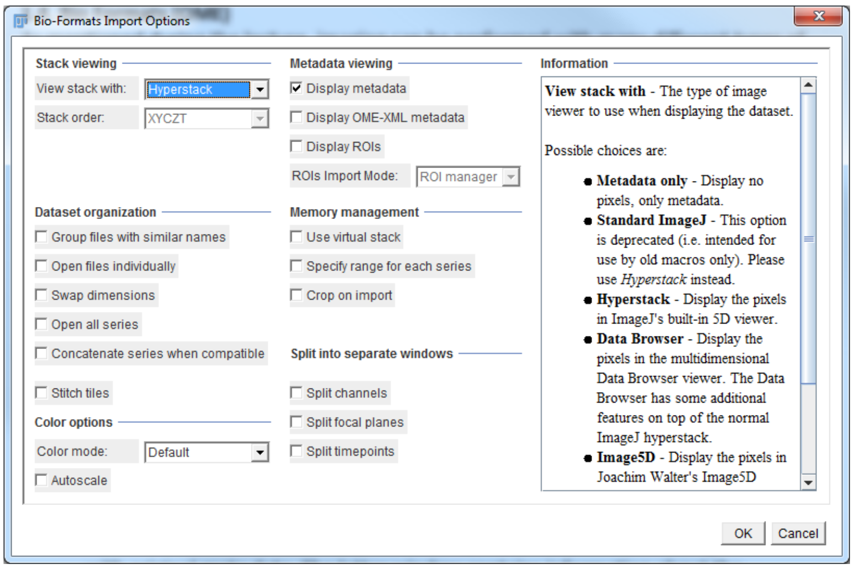
Note: By clicking outside the selection region you will automatically deselect the previous selected regions, you can draw multiple regions by shortly holding the SHIFT button while starting drawing, if you hold the shift key it will constrain the region to a square bounding box. Under the menu item **Edit –> Selections** more operations can be found that apply to selection regions. You can remove selections by using CTRL-SHIFT-A.

1.5. Bio Formats (OME)

As mentioned during the lecture, imaging can be performed with many different types of microscope systems (e.g. Leica, Zeiss, Nikon and Olympus microscopes etc.) or other biomedical imaging

techniques including, MRI, X-ray CT, PET scans, X-ray , 2D- and 3D-ultrasound imaging, fluorescence microscopy, etc.
Each technique and company or brand in general develops their own proprietary standard, and these files not only contain the raw image data, they also contain meta data collected during acquisition, the latter in most cases is not present in standard image formats.
Go to the website https://www.openmicroscopy.org/bio-formats/, check the supported formats at https://docs.openmicroscopy.org/bio-formats/6.10.0/supported-formats.html and also check the page https://docs.openmicroscopy.org/ome-model/6.0.0/ about the open microscopy data model.
How to use this in ImageJ:

a. If the file is not a standard image format you can open these by using the Bio-Formats importer. This is a plugin that allows opening many different file formats. In most cases dragging and dropping will start the importer automatically, if not you can find the importer at **Plugins -> Bio-Formats -> Bio-Formats Importer.** Open the Nikon file NvSnailA-mTq2.nd2 using the drag and drop method, the Bio-Formats importer should open automatically. You should get the following screen:


Bio-Formats Import Options

Stack viewing
View stack with: Hyperstack
Stack order: XYCZT

Dataset organization
[ ] Group files with similar names
[ ] Open files individually
[ ] Swap dimensions
[ ] Open all series
[ ] Concatenate series when compatible
[ ] Stitch tiles

Color options
Color mode: Default
[ ] Autoscale

Metadata viewing
[✓] Display metadata
[ ] Display OME-XML metadata
[ ] Display ROIs
ROIs Import Mode: ROI manager

Memory management
[ ] Use virtual stack
[ ] Specify range for each series
[ ] Crop on import

Split into separate windows
[ ] Split channels
[ ] Split focal planes
[ ] Split timepoints

Information
View stack with - The type of image viewer to use when displaying the dataset.

Possible choices are:

- **Metadata only** - Display no pixels, only metadata.
- **Standard ImageJ** - This option is deprecated (i.e. intended for use by old macros only). Please use *Hyperstack* instead.
- **Hyperstack** - Display the pixels in ImageJ's built-in 5D viewer.
- **Data Browser** - Display the pixels in the multidimensional Data Browser viewer. The Data Browser has some additional features on top of the normal ImageJ hyperstack.
- **Image5D** - Display the pixels in Joachim Walter's Image5D

OK Cancel

Before loading the file check the box for Display metadata and hit OK.

b. There should be two windows present now, one with the confocal image and one with original meta data. The latter window contains information about the microscope settings, and can also be opened by using **Image -> Show Info** or CTRL-I (make sure the image window is active). Check the meta data, can you make sense of this information? It actually contains all the microscope settings, including filters, zoom, pin hole size etc etc.

Note : It is also possible to open raw binary files with ImageJ, or a folder with a numbered sequence of files.

1.6. Pixel counts and image types

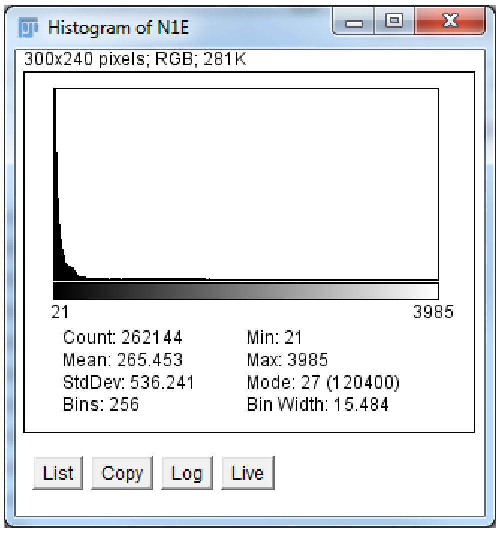
When an image is acquired using a microscope or another instrument it will be stored as bit maps. A bitmap consists of a matrix or array of pixels. Depending on the detector or camera the pixels represent a physical quantity. In the case of light microscopy, the light intensity is converted into a number of counts; the number of counts in each pixel is a quantitative measure of the amount of light or intensity that was collected and amplified by the detector at that point.

a. Open the Diatoms.tif image again, you will notice that there is extra information printed just above the image (top-left). For this image you should see 515x486 pixels; 8-bit, 243K. The first two numbers correspond to the width and height of the image, the third number is the so-called bit-depth and the last number is the file size on disk.
b. Now move the mouse around in the image and check the status bar of FIJI. It will display and update the current position and number of counts/pixel value at that position. This is a quick way of inspecting pixel values in the image. If you zoom at the highest possible zoom, you will see that the image is displayed as squares, these are the image pixels.
c. The bit-depth of this image is 8, which means that we can only store a *fixed* number of *discrete* grey scale levels. Calculate the values of the minimum and maximum counts that can be stored for an 8-bit image. These values are called the dynamic range.
d. The distribution of grey scale values can be visualized using a histogram; open the image Coli-EM-512.tif and use **Analyze –> Histogram**, or CTRL-H to plot a histogram. A new window will open showing the histogram, what do you see and what kind of information is displayed here? Now activate the **live** button and select a rectangular or circular region in the image and move the rectangle around using the mouse and look at the histogram. Inspect the region inside the bacterium and outside the bacterium, what do you see? Find a way to save the histogram values and also the plot.
e. Now open the image Hela-YFP.tif and check the image information. This image has a different bit-depth, what would be the maximal possible value for this bit-depth?
f. Although the bit-depth of this image is 16, in general this is not the same as the real maximal bit-depth that can be reached by the camera. The real bit-depth depends on the detector and camera used on the microscope, this information is stored in the meta data. Check the values of BitsPerPixel and PixelType for NvSnailA-mTq2 and PiggyBac_mNG_ER_0016_Zstack, what is the maximal possible value according to these? Why are higher bit-depths useful in general?
g. In ImageJ you can easily convert the bit-depth of the image into another one. Open the image Hela-YFP.tif and convert the 16-bit image to 8-bit using **Image -> Type -> 8-bit**. Check the pixel values using the mouse and the histogram. What happened to the file size? If you would convert back to 16-bit, would you restore the original image? Explain your answer.
h. Another image type is 32-bit, this is a special image type that allows negative numbers and also fractional numbers (so-called floating point numbers). This is especially useful when you want to do image processing, without losing precision. Convert the Hela-YFP.tif image to 32-bit, now save the file and what happened to the file size? Plot a histogram; now you have to set a range and bin size for the histogram yourself.

1.7. Dynamic range and contrast

Images may contain a large dynamic range, this means that some areas are dim and other areas are bright giving rise to large contrast differences. When looking at an image some regions may not be visible using grey scale images.

a. Open the N1E.tif image, you will see a number of cells that express the fluorescent protein mTq2. How many cells do you see?
b. When ImageJ displays an image it will automatically map the intensity levels onto the screen ranging from the minimal number of counts to the maximal number of counts present in the pixels. Check the histogram:

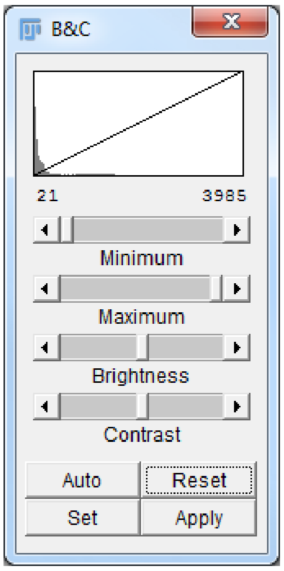

Histogram of N1E

300x240 pixels; RGB; 281K

21 3985

Count: 262144 Min: 21
Mean: 265.453 Max: 3985
StdDev: 536.241 Mode: 27 (120400)
Bins: 256 Bin Width: 15.484

We can however change this range by adjusting the contrast using **Image -> Adjust -> Brightness/Contrast** or CTRL-SHIFT-C, this will open a new dialogue (B&C)



Play with the options Min, Max, Brightness and Contrast to see what kind of effect these options have. How many cells are there really present? Beware! if you hit apply the image will be changed with the min-max values in the dialogue.

c. One of the major drawbacks of changing the contrast in an image is that the bright regions will be saturated if you want to see the dim regions as well. A commonly used solution to this problem is by applying 'false color'. By using a false color you can map each grey scale intensity level to a specific color, this achieved by defining a look up table (LUT). ImageJ has the option to use many different look up tables, to see all the available LUTs go to **Image -> Color -> Display LUTs.** The LUTs are located in the folder Fiji.app\luts, where you can add or remove LUTs, after restarting ImageJ the list will be updated. Also add the extra LUTs that have been provided to your LUTs folder.

d. Now choose the Fire LUT by using the LUT icon or go to **Image -> Lookup Tables**, what do you see? Also try some other LUTs like for example Royal and Red. While choosing different LUTs, are the underlying pixel values changed or retained? A drawback of false color LUTs is that the color intesnity no longer represents the intensity or pixel counts, it is therefore common to display a color bar that maps counts to colors. Go to **Analyze->Tools->Calibration Bar** and add a color bar to your image. Tick the overlay check box, as this will retain the pixel values allowing you to change the LUT and remove the color bar by using **Image->Overlay->Remove Overlay**. If you save the image as tiff the calibration bar will be saved as well.

e. Now open the EColiOverflow.tif image, this image of *E.coli* bacteria has been recorded incorrectly. The 8-bit image has pixel values ranging from 0 to 255, clipping at the bottom and top. For proper imaging one should adjust the detector-voltage and offset in such a way that no under saturated (value 0) or saturated (255 at 8-bit) pixels are present. Hover over the pixels, what do you see in the status bar? Make a histogram, what do you see at the lower and upper values? Now change the LUT to HiLo, this is a special grey scale LUT, for which the lowest pixel values are blue and the highest are red, what do you notice? Why is it important to calibrate the camera in such a way that there is no (under)saturation and you use the full dynamic range?

Note: Contrast can also be changed in a non-linear way by using the gamma transform **Process->Math->Gamma**. This will however change the raw pixel values and we do not advice to use this method as it will render processing incorrect and many journals do not approve of these methods even for displaying images in a publication.

1.8. Resolution: "Pixel size" and Sampling

When acquiring images using microscopes we use different magnifications for imaging samples, this means that the pixels we see in images have different spatial separation depending on the magnification used. The "pixel size" or sampling determines the resolution of the image and which details we can actually still discern.

a. Open the ColiA.tif image, this is a phase-contrast image of *E. coli* bacteria that was recorded at a resolution of 15.9 pixels/ µm. Normally the metadata contains the image resolution and ImageJ sets these parameters automatically, however in this case the information is missing. Go to **Analyze -> Set Scale** this will open a dialogue where you can set the resolution. Set the distance in pixels to 15.9, known distance to 1.0 and unit of length to micron and click OK. Now the size of the image and pixel positions are displayed in micrometers. If you save this image as a tiff by using **File -> Save As -> Tiff** the resolution will be saved in the meta data and when you reopen the file it will be set automatically, check this. Use Show Info to display the metadata or check the Set Scale dialogue.

b. Zoom in on a typical bacterium and select the straight line tool by clicking the straight line icon. Draw a line selection while holding the left mouse button across the length of the bacterium, the result is being displayed in the status bar. Repeat this for the diameter. What is the typical length and diameter of a bacterium?

c. Visible light has a wavelength of about 0.5 µm, how many waves fit within the diameter? The resolution for light microscopy is about half the wavelength, how does this compare to the resolution in the image (oversampling or under sampling)? How would you choose your pixel size when acquiring an image on a microscope?

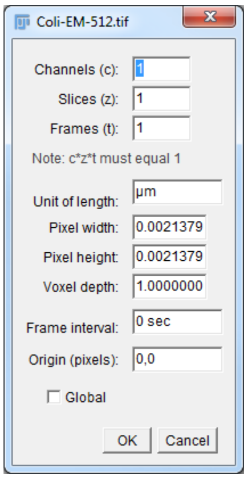
d. Once the resolution of the image is set it is also possible to display a scale bar in the image. Go to **Analyze -> Tools -> Scale Bar,** in the dialogue box you can set the length of your scale bar and other parameters. Make sure you tick the Overlay check box. If you save the image as tiff the scale bar will be saved as well.

e. Open the Coli-EM-512.tif, this is an image of a bacterium (512x512 pixels), acquired using an electron microscope (EM). One clearly observes the double membrane. What is the diameter of this cell in microns? Check the metadata, what is the resolution in pixels/µm and What is the “pixel size” in µm?

f. Make the image a factor of 4 smaller (number of pixels will be reduced by a factor of 16) by using binning (**Image –> Transform -> Bin**) or the scale command (**Image -> Scale)**. Now study the effect of downscaling by zooming in to 400%, such that that reduced image will be as large as the original image on screen, do you see a difference between the original and the reduced image? Does one of the images contain significantly more information than the other? If you would scale back to the original size do you obtain the same image as the original again? Give an explanation.

g. Now display this EM image at a resolution that is common for light microscopy (LM). Use the following numbers: EM resolution = ~450 pixels/µm and LM resolution = 15 pixels/µm, Reduce the image according to this factor, and judge the result after zooming in. Clearly, information is lost because the pixel size is too big. This is called under sampling, the image resolution is much lower than the microscope resolution.

Note: In ImageJ you can also set the pixel/voxel size by using **Image -> Properties (CTRL-SHIFT-P)**


Coli-EM-512.tif

Channels (c): 1
Slices (z): 1
Frames (t): 1
Note: c*z*t must equal 1
Unit of length: µm
Pixel width: 0.0021379
Pixel height: 0.0021379
Voxel depth: 1.0000000
Frame interval: 0 sec
Origin (pixels): 0,0
☐ Global

1.9. Creating RGB images

Until now we have been working with grayscale or so-called monochrome images, in these images to each pixel a gray scale value is attributed. By using lookup tables it is possible to map a specific color to these pixel values, which yields a false color image on your screen, while retaining the pixel values. With ImageJ it is also possible to convert these images to so-called RGB images and save the image as color images.

a. Open the HeLa-Actin.tif image, this image shows HeLa cell labelled with an F-actin marker. In order to increase the contrast in dim regions choose the Orange Hot LUT. Add a 10 µm scale bar to the left bottom corner and a calibration bar to the top right corner. In both cases tick the overlay check box. You can repeat the process, which will replace the existing scale bar or calibration bar.

b. Now save the image as a separate tiff image (HeLa-Actin-Overlay.tif) and reopen the tiff image. By saving as a tiff image, the LUT, scale bar and calibration bar should be there and the image should still have the original pixel counts. Now place the image in a word or PowerPoint document, what do you notice?

c. Save the HeLa-Actin-Overlay.tif as a png image and reopen the image in ImageJ and also in word or PowerPoint. What do you see now? Try to change the LUT in the png image.

d. When you tried to change the LUT ImageJ gave a message saying that LUTs cannot be assigned to RGB images. By saving the image as a png it was automatically converted to an RGB image. Hover over the pixels and check the pixel values in the status bar.

e. Each pixel now has three values attributed to it these represent the amount of red, green and blue present in that pixel. Each of the three colors can have a value in the range of 0-255 and by mixing you can create any color in the spectrum. Reopen the HeLa-Actin-Overlay.tif image and apply **Image -> Type -> RGB Color** or CTRL-SHIFT-F (flatten). What happened?

Note: The use of RGB images is suitable for presentation and publication purposes. When saving the RGB image it is best to use the lossless PNG compression and not the lossy JPEG compression. The latter will yield a smaller file size but will also introduce artefacts in the image.

1.10 Image data layout

ImageJ has different build in image data layout types, which will be automatically created when loading a data file or when using the importer you can set the layout. The simplest type is a single image as we have used so far in the tutorial.

a. When multiple fluorophores are imaged ImageJ will load this as a hyperstack, it will contain a number of channels (C), and a slider will appear below the image with a ‘c’ next to it. For each channel different LUTs and B&C settings can be applied independently. Open the NLS-NES-HeLa.tif image, this data set has two channels, the first one labels the cytoplasm of the cells and the second one labels the nucleus. Change the LUT for the first channel to Cyan and the second channel to Yellow and check if LUTs are retained. Save the file as tiff and reload to see if the information is stored. You can flip between the channels by using the slider however we can also make an overlay with the channels tool. Go to **Image –> Color -> Channels tool** (CTRL-SHIFT-Z) and select composite. By ticking both check boxes you will see an overlay of both channels.

b. The Phalloidin_POI_DAPI.tif image contains an example of three channels, it will become increasingly harder to find good color combinations with more channels with overlapping structures, and you can try to optimize color combinations (switch channels and choose another LUT) and/or brightness and contrast for each channel.

c. Another very common data layout type is a time series, this is called a stack, which contains multiple images measured at different time points. Within this context the image is called a frame, and a slider will appear below the image with a triangle next to it (play button). Open the Alpha tub HeLa.tif stack, and go to **Image -> Properties (CTRL-SHIFT-P)**, which will show you the layout of the stack and also the frame interval. Normally the frame interval is saved in the meta data, set the frame interval to 5 sec. Now go to **Image -> Stacks -> Animation -> Animation options** and set the speed to 10 frames per second. You can play the movie by clicking the play button.

d. Now save the movie as an animated gif and open it in a browser or place it in a PowerPoint presentation.

e. Open the Nikon data set PiggyBac_mNG_ER_0016_Zstack, this file is a z-stack with frames (z-slices), check the image properties, what is the voxel size? Now make a montage using **Stacks -> Montage,** choose an optimal starting and end slice**.**

f. Another type is the z-stack with multiple channels, in this case the images in the stack are called slices, now a slider will appear below the image with a ‘z’ next to it. Open the confocal-series.tif image and check the properties of the image. This data set contains two channels and 25 slices and is called a hyperstack. Set the voxel-depth to 0.2 in **Image ->**

Properties. Go to **Image -> Color -> Channel tool** (CTRL-SHIFT-Z) and create a composite. FIJI comes with a simple 3D-viewer for 8-bit data, go to **Plugins -> 3D Viewer**, a dialogue will the settings should be correct, click OK. Now by using the left mouse button and dragging you can rotate the 3D object.

Note: Time series can also contain multiple channels, in this case also two sliders will appear below the image. The most complex layout contains frames (time), slices (z-position) and channels and will have three sliders below the image. You can change the lay-out of a stack by using **Hyperstack -> Stack to Hyperstack** and set the channels, slices and frames**.**

1.11 Create basic macros

a. When analyzing microscopy data, conclusions can be made only if there is a significant amount of data available. This will require analysis of many datasets and thus one has to repeat the analysis multiple times. This is tedious and time-consuming, but luckily ImageJ provides the option to automate image processing using macros and plugins.

b. A macro is a simple program that automates a series of ImageJ commands. A macro is saved as a text file. There are more than 300 example macros on the ImageJ website. To try one, open it in a browser window, copy the text to the clipboard (crtl-a, ctrl-c), switch to ImageJ, open an editor window **Plugins -> New -> Macro**, paste (ctrl-v). Then run it using the editor's **Macros>Run Macro command (ctrl-r)**. Most of the example macros are also available in the macros folder, inside the ImageJ/FIJI folder. A detailed description and examples on how to use macros can be found at http://ImageJ.nih.gov/ij/developer/macro/macros.html.

c. A typical layout of a macros is as follows:

```
macro "Close All Windows [F2]" {
        while (nImages>0) {
                selectImage(nImages);
                close();
        }
}
```

This macro will close all images that are open if you hit the F2 key. You can also define multiple macros using different keys in one text file and install it Macros -> Install, after which the macros will become available.

d. Study the example provided to correct the white balance of an immunohistochemistry image from Q1.4 Correct_whitebalance_IHC.txt and run the macro on an IHC image.

```
// normalizes channels to intensity in background region
// this sets the background to white
// drag macro on Fiji/ImageJ
// install macro via Macros>Install
// open 3-channel tif image and hit "c" key
```

e. Check the studyGroup example on ImagJ scripting macros to see a step-by-step example : https://github.com/ScienceParkStudyGroup/studyGroup/tree/gh-pages/lessons/20190409_ImageJ-Macro_Franka

f. The easiest way to create a macro yourself is to record a series of commands using the command recorder and copy your command lines from this recorder into your macro text file. Open the **macro recorder -> menu Plugins -> Macros -> Record.**

g. Watch how the commands in the recorder window when you perform any operation via the ImageJ menu. Place these in your editor. a. More information on commands and functions can be found at : https://imagej.nih.gov/ij/developer/macro/functions.html

2.0 Image processing with Python

There are many ways to solve a specific problem, especially with Python. In this document, we will give you some starting points for your image processing adventures with Python without stepwise instructions, and reference links for you to further explore the tools. There might be additional methods to get to the same results not mentioned in this document. So, be creative!

2.1 Suggested packages

- scikit-image - https://scikit-image.org/docs/stable/
- PyImageJ - https://pyimagej.readthedocs.io/en/latest/
- Napari - https://napari.org/stable/
- Pandas - https://pandas.pydata.org/docs/
- Numpy - https://numpy.org/doc/stable/
- Matplotlib - https://matplotlib.org/stable/users/index

2.2 Importing images (e.g. tiff, Bio-formats) or tabular data (e.g. CSV) exported from ImageJ

- **Image data types**

  https://scikit-image.org/docs/stable/user_guide/data_types.html

  In skimage, images are simply numpy arrays, which support a variety of data types, i.e. “dtypes”. To avoid distorting image intensities (see Rescaling intensity values), we assume that images use the following dtype ranges:

Data type	**Range**
uint8	0 to 255
uint16	0 to 65535
uint32	0 to 2^{32} - 1
float	-1 to 1 or 0 to 1
int8	-128 to 127
int16	-32768 to 32767
int32	-2^{31} to 2^{31} - 1

- **General importing image using scikit-image**

  https://scikit-image.org/docs/stable/api/skimage.io.html#skimage.io.imread

  Code

```
from skimage import io
image = io.imread('path/to/file_name')
io.imsave('path/to/file', image)
```

  Return
  ndarray (Numpy array) that can be treated like a matrix. The returned image objects have different dimensions, based on the format of the dataset. This allows you to access different channels, stacks and time points. Dimensions of the indexing to access elements in an image object. For example, if you want to access the first time-point of the first channel of an image, you use: image[0,0,:,:].

Table 1 Input image types and their expected ndarray attributes.

Input Image Type	**Shape of imread() output**
2D grayscale	(row, col)
2D color scale (e.g. RGB)	(row, col, num_colors)
2D multichannel	(channel, row, col)
Z-stack/time-series (3D) grayscale	(stack/time, row, col)
Z-stack/time-series (3D) color scale	(stack/time, row, col, num_colors)
Z-stack/time-series (3D) multichannel	(stack/time, channel, row, col, num_colors)

- **General importing of bio-formats using PyImageJ**

  https://docs.openmicroscopy.org/bio-formats/6.8.0/developers/python-dev.html

  Code

```
import imagej
ij = imagej.init('sc.fiji:fiji')
jcells = ij.io().open('/path/to/my/cells.ome.tif')
cells = ij.py.from_java(jcells)
```

- **General importing of CSV using pandas**

  https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

  Code

```
import pandas as pd
data = pd.read_csv('/path/to/my/csv')
```

- **Using the default image data provided by skimage.data**

  https://scikit-image.org/docs/stable/api/skimage.data.html

  Code

```
from skimage import data
data_2Dcell_8bit = data.cell()   #(660, 550) uint8 array, image of a cell
data_3Dcells_16bit = data.cells3d()   #(60, 2, 256, 256) uint16 ndarray, image of cells
```

2.3 Visualization for image data inspection

- **Quick visualizing of a 2D grayscale image using scikit-image and matplotlib**

  Code

```
from skimage import io
from matplotlib import pyplot as plt
image = io.imread('path/to/file_name')
io.imshow(image)
plt.show()
```

- **Visualizing image using Napari**

  https://napari.org/stable/tutorials/fundamentals/viewer.html
  https://napari.org/stable/howtos/layers/image.html

  Code

```
import napari
from skimage import data
cells = data.cells3d()[30, 1]  #ZCYX image data

# load multichannel image in one line
viewer = napari.view_image(cells, channel_axis=1)

# load multichannel image in one line, with additional options
```

```
viewer = napari.view_image(
        cells,
        channel_axis=1,
        name=["membrane", "nuclei"],
        colormap=["green", "magenta"],
        contrast_limits=[[1000, 20000], [1000, 50000]],
        )
```

2.4 Basic image analysis

- **Inspect image data and plot a histogram to check the distribution of pixel intensities**

  https://scikit-image.org/docs/stable/auto_examples/applications/plot_3d_image_processing.html#load-and-display-3d-images
  https://scikit-image.org/docs/stable/auto_examples/applications/plot_3d_image_processing.html#adjust-exposure

  Code

```
from skimage.data import cells3d
import matplotlib.pyplot as plt
from skimage import util

#Import image data
data = util.img_as_float(cells3d()[:, 1, :, :])  # grab just the nuclei

print(f'shape: {data.shape}')
print(f'dtype: {data.dtype}')
print(f'range: ({data.min()}, {data.max()})')

# Report spacing from microscope
original_spacing = np.array([0.2900000, 0.0650000, 0.0650000])

# Account for downsampling of slices by 4
rescaled_spacing = original_spacing * [1, 4, 4]

# Normalize spacing so that pixels are a distance of 1 apart
spacing = rescaled_spacing / rescaled_spacing[2]

print(f'microscope spacing: {original_spacing}\n')
print(f'rescaled spacing: {rescaled_spacing} (after downsampling)\n')
print(f'normalized spacing: {spacing}\n')

#plot a histogram to check the distribution of pixel intensities

#Create a function to plot histogram
```

```
def plot_hist(ax, data, title=None):
    # Helper function for plotting histograms
    ax.hist(data.ravel(), bins=256)
    ax.ticklabel_format(axis="y", style="scientific", scilimits=(0, 0))

    if title:
        ax.set_title(title)


_, d= plt.subplots()
plot_hist(d, data)
```

- **Look up table (LUT) - change the way it looks**

  There are multiple ways to change the color map, depending on which visualization tool you are using. In napari, for example, you can change the colormap parameter in the view_image() function to the specific color map of desire. A list of all color maps can be printed out using:

  Code

  ```
  list(napari.utils.colormaps.AVAILABLE_COLORMAPS)
  ```