- How an image is read in Python?
- What does masking do?

- Explaining data in images
- Understanding image masking

Image Handling

Reading and Processing Images

In biology, we often deal with images, for example from microscopy and different medical imaging modalities. In many cases, we wish to extract some quantitative information from these images. The focus of this session is to read and process images in Python. This includes:

- Working with 2-dimensional images
- Creating and applying binary image masks
- Working with 2-dimensional colour images, and interpreting colour channels

First, we want to read in an image. For this part of the lesson, we use a 2D image of brain part,Cerebellum, as an example. We use Matplotlib's `image` module, from which we import `imread` to store the image in a variable called image. The function `imread` can interpret many different image formats, including jpg, png and tif images.

```
from matplotlib.pyplot import subplots, show

from matplotlib.image import imread
```

```
image = imread('fig/Cerebellum.jpg')

fig, ax = subplots()

ax.imshow(image);

show()
```

```
image.shape
```

```
(154, 327, 3)
```

This tells us that our image is composed of 154 by 327 data units, or pixels as we are dealing with an image. It is equivalent to the image resolution. The array has three dimensions.

```
image[0, 0]
```

```
array([255, 255, 255], dtype=uint8)
```

The color intensities go up to 255. This is because RGB (red, green and blue) colours are defined within the range 0-255.

Let us now use matplotlib.pyplot's imshow function to plot the section of image to see what it looks like. The x and y coordinates are chosen to specify section of the image.

```
fig, ax = subplots()

ax.imshow(image[50:70, 60:100]);

ax.set_xticklabels(());
ax.set_yticklabels(());

show()
```

Each square is a pixel and it has one value. So how exactly are the pixel values assigned? By the numbers stored in the Numpy array, image.

```
fig, ax = subplots()

ax.imshow(image[:,:, 2], cmap='gray');

ax.set_xticklabels(());
ax.set_yticklabels(());

show()
```

Here we assigned a grey colour map.

Now that we know that the images are composed of a set of intensities that are just numbers in a Numpy array, we can start using these numbers to process our image.

As a first approach, we can plot a histogram of the original image intensities. We use the `.ravel()` method to turn the original 154 x 327 array into a one-dimensional array with 503,58 values. This rearrangement allows the inclusion of an image as a single column in a matrix or dataframe!

The histogram plot shows how many of the intensities are found in one layer of this image:

```
layer = 2

fig, ax = subplots()

ax.hist(image[:, :, layer][image[:, :, layer] < 255].ravel(), bins=500)
    ;

show()
```

```
image.size
```

```
151074
```

```
from PIL import Image

image = Image.open('fig/Cerebellum.jpg')

fig, ax = subplots()

ax.imshow(image);

show()
```

```
type(image)
```

```
<class 'PIL.JpegImagePlugin.JpegImageFile'>
```

```
image.show()
```

```
image.entropy()
```

```
5.594248793777234
```

```
image.rotate(-90, expand=True)
```

```
<PIL.Image.Image image mode=RGB size=154x327 at 0x7F5868598940>
```

Image Masking

```
from matplotlib.image import imread
from matplotlib.pyplot import subplots, show
```

```
image = imread('fig/rose.jpg')

fig, ax = subplots()

ax.imshow(image);

show()
```

Transpose Image

```
image.shape
```

```
(3648, 2736, 3)
```

```
image_t = image.transpose((1, 0, 2))

fig, ax = subplots()

ax.imshow(image_t)

show()
```

Only red Component

```
fig, ax = subplots()

ax.imshow(image_t[:, :, 0]);

show()
```

Histogram of Red Component

```
fig, ax, = subplots()

ax.hist(image_t[:, :, 0].ravel(), bins=500);

show()
```

Masking the Red Component

```
threshold = 90

mask = image_t[:, :, 0] < threshold

image_masked = image_t[:, :, 0] * mask

fig, ax = subplots(ncols=3)

ax[0].imshow(image_t[:, :, 0], cmap='gray')
ax[1].imshow(mask, cmap='gray')
ax[2].imshow(image_masked, cmap='gray');

fig.tight_layout()

```

```
15 show()
```

False Colour

```
1 fig, ax = subplots()
2
3 ax.imshow(image_masked);
4
5 show()
```

Apply mask to all layers

Choose grey value for background

```
1  image_new = image_t.copy()
2
3  grey_value = 100
4
5  image_new[mask, :] = grey_value
6
7
8  fig, ax = subplots()
9
10 ax.imshow(image_new);
11
12 show()
```

```
1 fig.savefig('fig/rose_masked.png', format='png')
```

Keypoints

- `imread` function is used to read images.
- `ravel` function flattens a multi-dimentional array into a single array.
- Image masking help in identifying objects based on colur intensities.