


浙江大学爱丁堡大学联合学院
ZJU-UoE Institute

Quantitative analysis of biomedical images - an introduction

Nicola Romanò - nicola.romano@ed.ac.uk

Learning objectives

At the end of this lecture you should be able to:

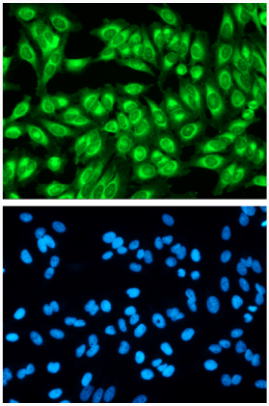
- Describe images as vectors and perform basic manipulation and visualization using Python
- Describe and explain how information is stored in images
- Explain and make use of basic concepts in image analysis such as SNR, histograms, LUT, …

What can (biomedical) image analysis tell us?

1. Images are a big source of information
2. Analysis can be **qualitative** or **quantitative**

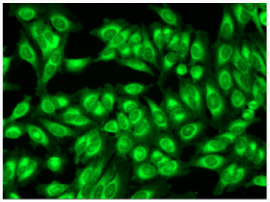
What can (biomedical) image analysis tell us?

1. Images are a big source of information
2. Analysis can be **qualitative** or **quantitative**



What can (biomedical) image analysis tell us?

1. Images are a big source of information
2. Analysis can be **qualitative** or **quantitative**

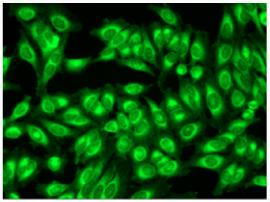


Qualitative:

- Spindle-shaped cells
- Similar in size

What can (biomedical) image analysis tell us?

1. Images are a big source of information
2. Analysis can be **qualitative** or **quantitative**



Qualitative:

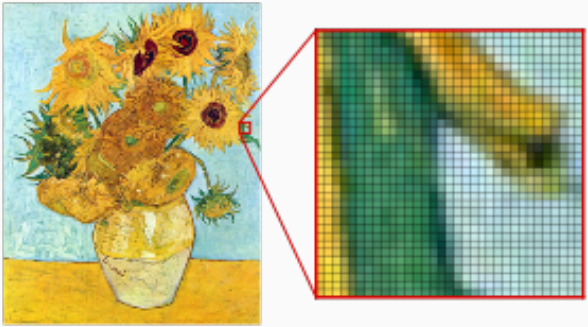
- Spindle-shaped cells
- Similar in size

Quantitative

- Number of cells
- Position
- Size
- Staining intensity
- ...

Images as matrices of pixels

In biomedical sciences, we deal with **raster images**, rectangular grids of pixels with varying intensity.



Sunflower, Vincent Van Gogh (not a biomedical image, yet a pretty one!)

Representing matrices in Python

If working with matrices in Python, an obvious choice is using the NumPy library.

NumPy is a Python open-source library for numerical computing, with support for n-dimensional arrays and many functionalities to work with matrices.

Our first NumPy matrix!

```
# This is the standard way of importing numpy
import numpy as np
a = np.array([1, 2, 3])
```

Multiple dimensions

Numpy supports n-dimensional arrays with any number of dimensions.

```
import numpy as np
b = np.array([[1, 2, 3][4, 5, 6]])
print(b.shape)
```

```
(2, 3)
```

The shape of an array is a vector showing the number of element in each of the dimensions of the array. In this case we have a matrix of 2 rows and 3 columns.

Not just 2D!

Biomedical images are not limited to 2 dimensions! Three and >3-dimensional imaging is becoming more common.

Not just 2D!

Biomedical images are not limited to 2 dimensions! Three and >3-dimensional imaging is becoming more common. Examples of other dimensions may include:

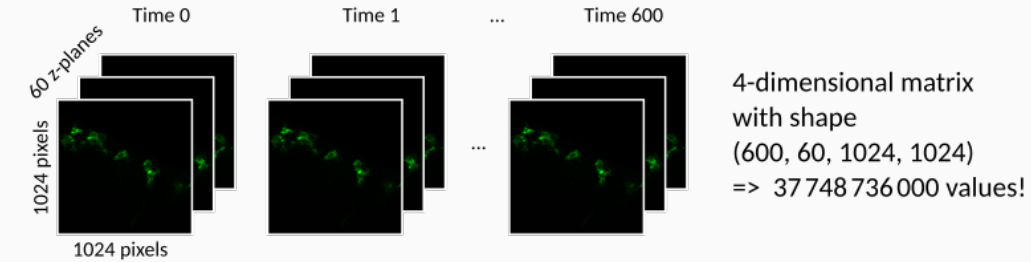
- z-axis
- time
- channels
- wells (for multiwell plate imaging)

Not just 2D!

Biomedical images are not limited to 2 dimensions! Three and >3-dimensional imaging is becoming more common. Examples of other dimensions may include:

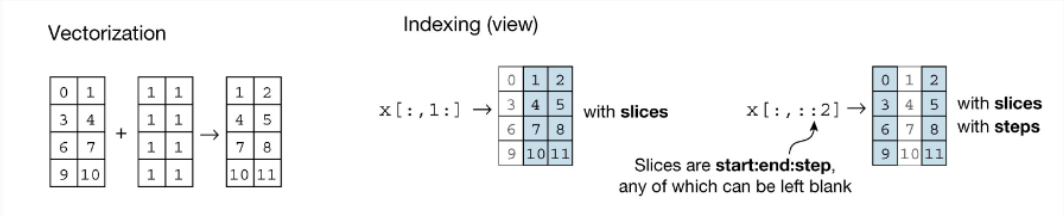
- z-axis
- time
- channels
- wells (for multiwell plate imaging)

These can also be stored as n-dimensional matrices! Note that the axes order in the file may vary depending on the software used to create images.



Array operations

Numpy allows matrix operations, selection of sub-matrices and much more, which can be used to manipulate our images.



More information can be found on the NumPy paper (Harris et al., Nature 2020) or on the NumPy website (www.numpy.org)

Using Scikit Image to read and show images

In order to display an image saved in a NumPy array, we can use the Scikit Image library.

```
from skimage.io import imread, imshow
# Read the image
img = imread("image.tif")
# Show the image
imshow(img)
```

Using Scikit Image to read and show images

In order to display an image saved in a NumPy array, we can use the Scikit Image library.

```
from skimage.io import imread, imshow
# Read the image
img = imread("image.tif")
# Show the image
imshow(img)
img_small = img[0:5, 0:5]
imshow(img_small)
```

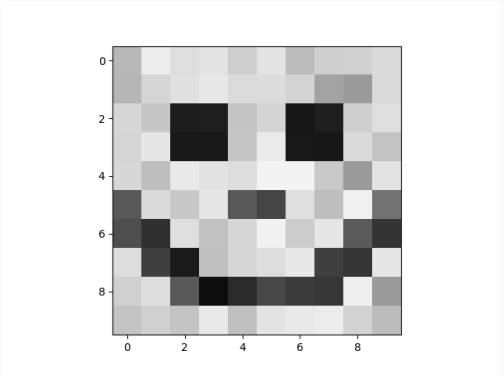
Using Scikit Image to read and show images

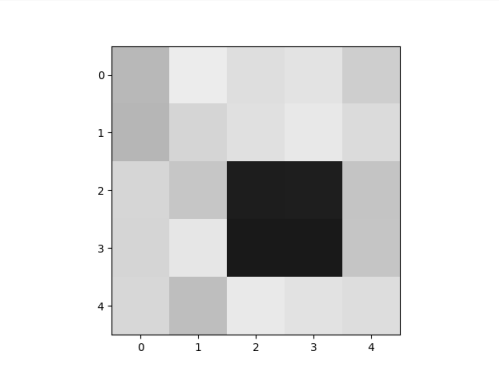
In order to display an image saved in a NumPy array, we can use the Scikit Image library.

```
from skimage.io import imread, imshow
# Read the image
img = imread("image.tif")
# Show the image
imshow(img)
img_small = img[0:5, 0:5]
imshow(img_small)
print(img.shape) # Prints (10, 10)
print(img_small.shape) # Prints (5, 5)
```

Using Scikit Image to read and show images

In order to display an image saved in a NumPy array, we can use the Scikit Image library.



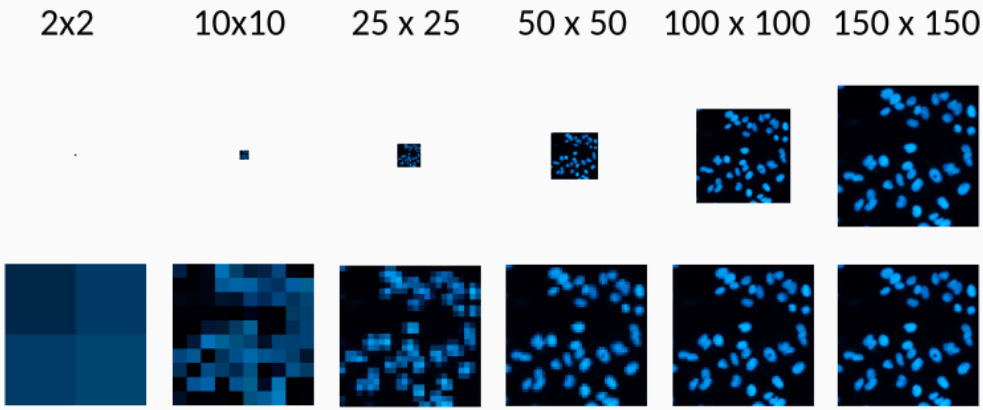


Information in images

Information in images - Resolution

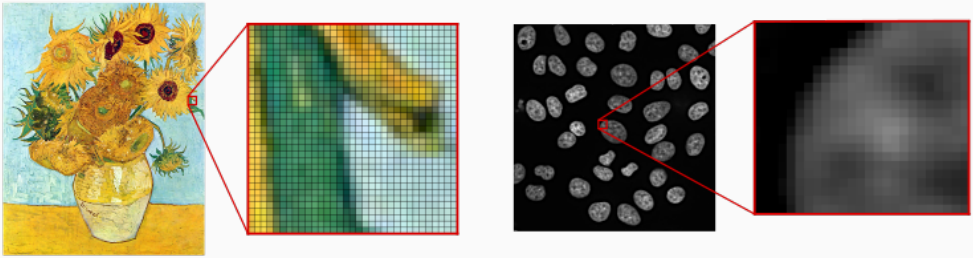
Several parameters influence the amount of quantitative information present in an image.

Resolution is the number of pixels in the image.



Information in images - Colour

Images can be either **colour** or **grayscale**.

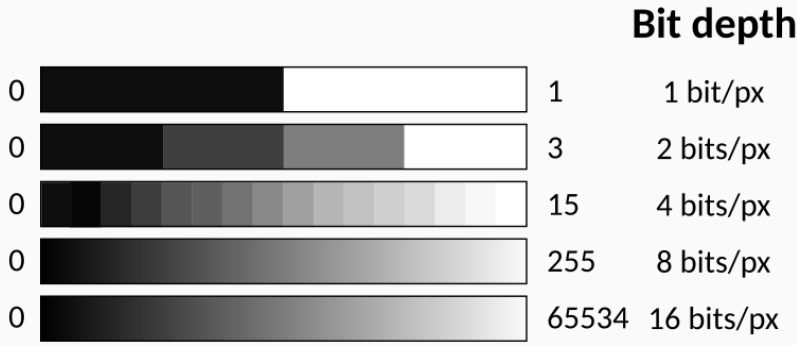


Colour > each pixel contains information for red, green and blue (RGB)
Grayscale > each pixel contains gray information (from black to white))

Most biomedical images are grayscale.

Information in images - Bit depth

Bit depth is the number of bits necessary to represent each pixel in the image.



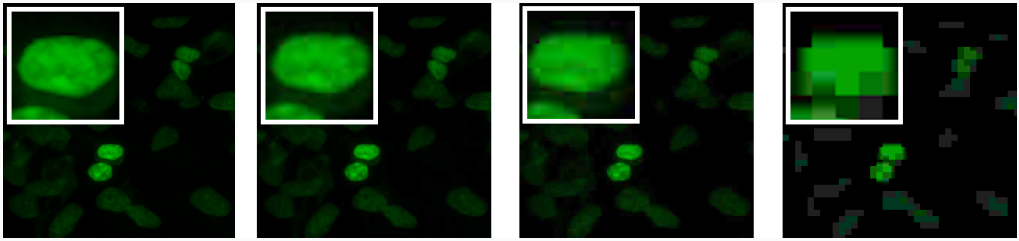
(Increasing bit depth > Increasing information)

Most often you will deal with 8- and 16-bits/pixel images (what most microscopes and cameras use) or 1 bit/pixel images, which are used as masks.

Information in images - Formats and compression

There are a lot of file formats used for images.

Images can be compressed or uncompressed. Compression can be lossless (e.g. LZW) or lossy (e.g. JPEG). Lossy compressed images should not be used as the base for quantitative imaging.

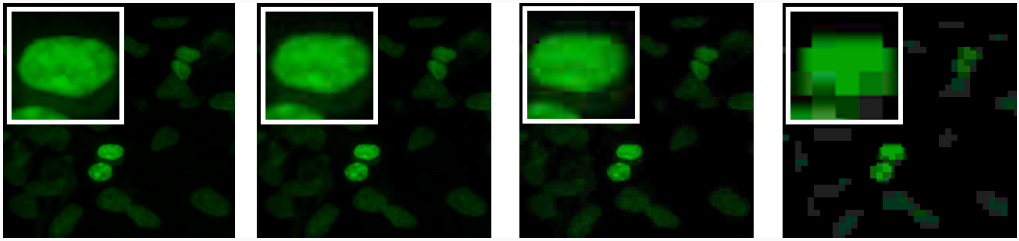


Increasing JPEG compression generates artefacts and loses information.

Information in images - Formats and compression

There are a lot of file formats used for images.

Images can be compressed or uncompressed. Compression can be lossless (e.g. LZW) or lossy (e.g. JPEG). Lossy compressed images should not be used as the base for quantitative imaging.

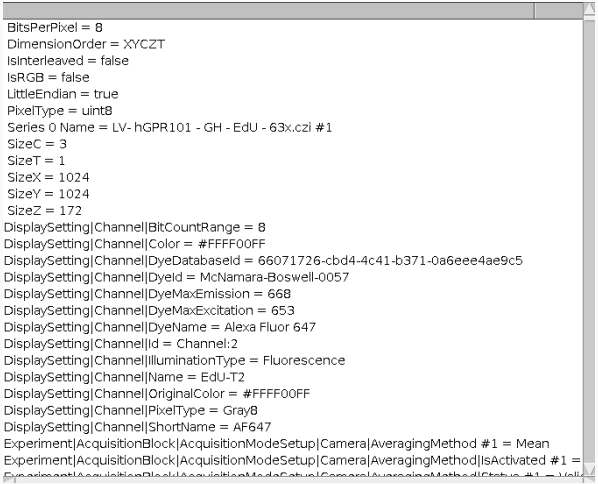


Increasing JPEG compression generates artefacts and loses information.

TIFF is one of the most common image files used in biomedical sciences. Many microscopes save images in proprietary formats (e.g. Zeiss > CZI or Nikon > ND2). Newer formats such as **HDF5** and **Zarr** allow fast access to big data by allowing chunked access to data.

Information in images - Metadata

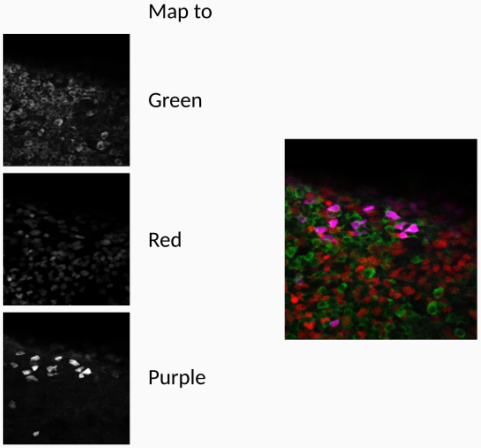
Metadata is all non-image data contained in the file. For example when the image was taken, the instrument used, the wavelength and microscope filters used, a description of the experiment etc.



Colouring

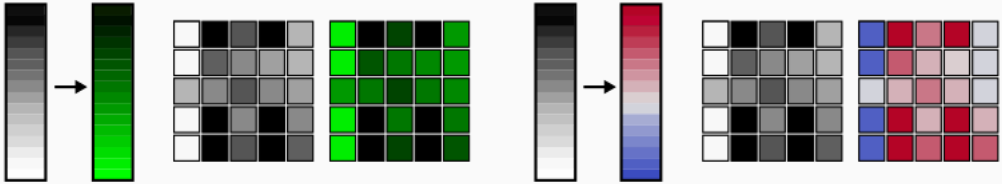
Pseudocoloring

Often colour is applied to microscopic images **after** acquisition to mark different parts of the image. This is called **pseudocolouring** or **false colouring**.



Look-up tables (LUTs)

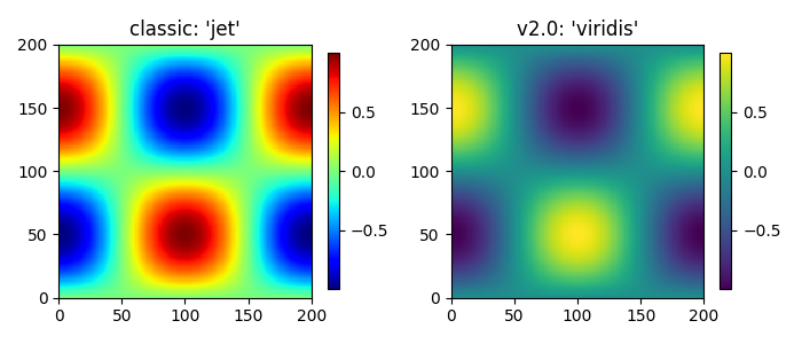
Pseudocoloring is obtained through look-up tables (sometimes referred to as "palettes" or "colourmaps")



Colourmaps can be sequential/linear or non sequential.
Divergent and non linear colour maps can be good to enhance differences in the image,
but should be used with caution!

The Jet palette - when colourmaps go bad...

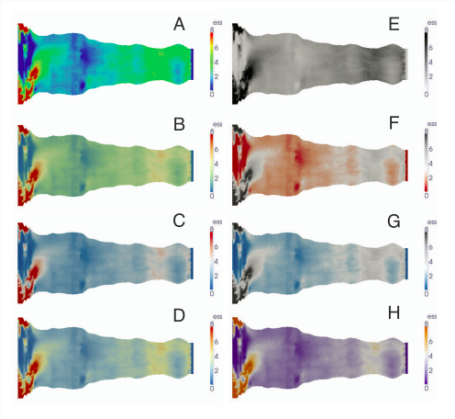
The Jet colourmap has for a long time been the standard palette in Matlab, is used in many pieces of software and is seen in many published articles. Jet is problematic because it can create artefacts in your data.



Jet can create artefacts in the data. [Source: Matplotlib website]
See https://www.youtube.com/watch?v=xAoljeRJ3lU

The Jet palette - when colourmaps go bad...

The Jet colourmap has for a long time been the standard palette in Matlab, is used in many pieces of software and is seen in many published articles. Jet is problematic because it can create artefacts in your data.



Rainbow palette resulted in significantly more diagnostic errors - Borkin 2011

Colourmapping in Matplotlib

We can easily colourmap an image loaded in Matplotlib

```
# We will use the Matplotlib library this time
import matplotlib.pyplot as plt
from skimage.io import imread

img = imread("smiley.tif")
# plt.subplots allows plotting more than one image
# side by side.  It creates and returns a figure and
# a list of axes (the subplots)
fig, ax = plt.subplots(ncols=3, nrows=1, figsize=(10, 5))
```

Colourmapping in Matplotlib

We can easily colourmap an image loaded in Matplotlib

```
# We will use the Matplotlib library this time
import matplotlib.pyplot as plt
from skimage.io import imread

img = imread("smiley.tif")
# plt.subplots allows plotting more than one image
# side by side.  It creates and returns a figure and
# a list of axes (the subplots)
fig, ax = plt.subplots(ncols=3, nrows=1, figsize=(10, 5))
ax[0].imshow(img, cmap="viridis")
ax[1].imshow(img, cmap="terrain")
ax[2].imshow(img, cmap="spring")
plt.show()
```

Colourmapping in Matplotlib

We can easily colourmap an image loaded in Matplotlib

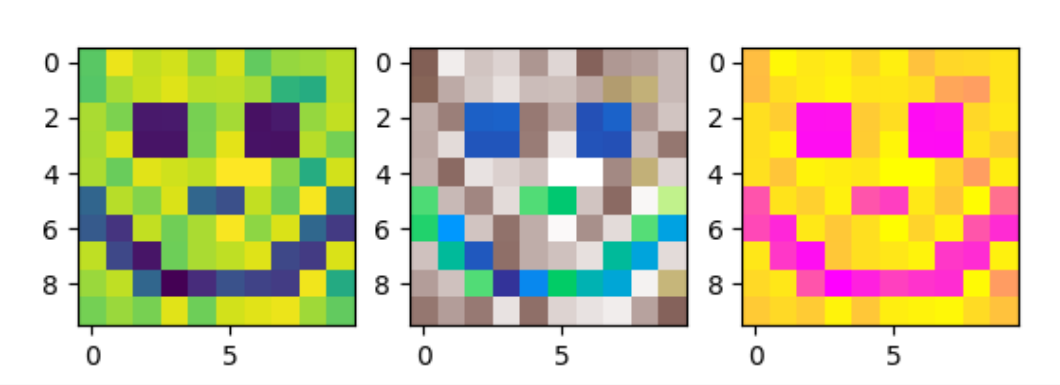


Image histograms

An easy way to describe an image is to use a histogram. For each intensity level, we count the number of pixels that have that intensity.

14	0	4	0	9
14	5	7	8	9
9	7	4	7	8
14	0	7	0	7
14	0	4	0	5

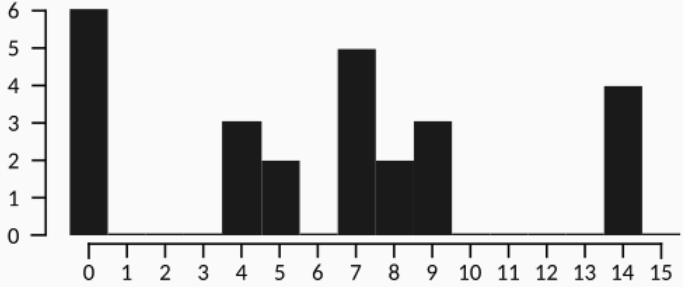
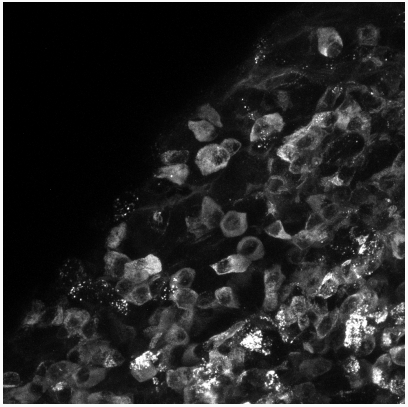
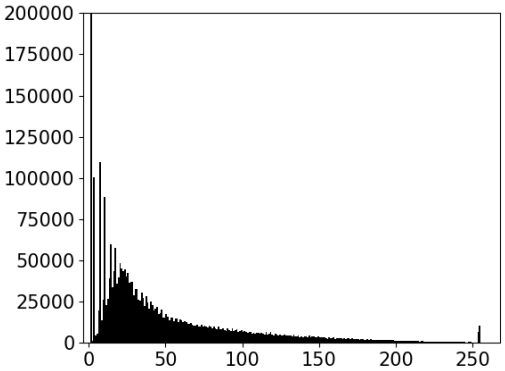


Image histograms

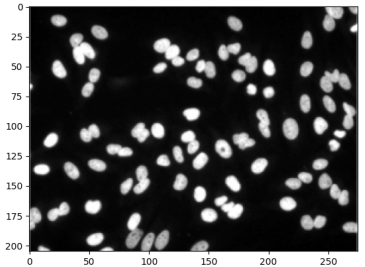
An easy way to describe an image is to use a histogram. For each intensity level, we count the number of pixels that have that intensity.

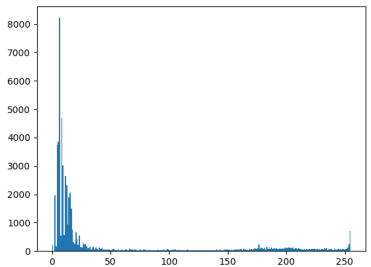




Plotting a histogram using Matplotlib

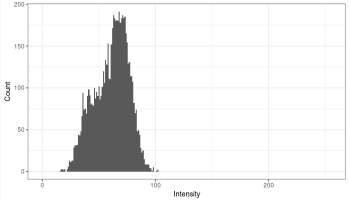
```
nuclei = imread("nuclei.tif")
imshow(nuclei, cmap="gray")
plt.hist(nuclei.ravel(), bins=range(255))
plt.show()
```

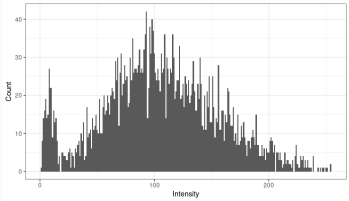


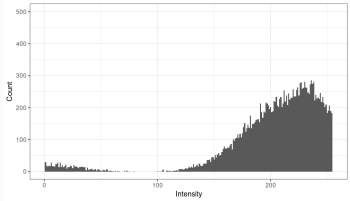


Over- and under-exposure

Histograms are a great tool to check the quality of your image. When adjusting the acquisition parameters on a microscope, care should be taken to use as much of the dynamic range as possible to avoid losing information. Although there are techniques to improve this after the fact… remember that *garbage in, garbage out!*





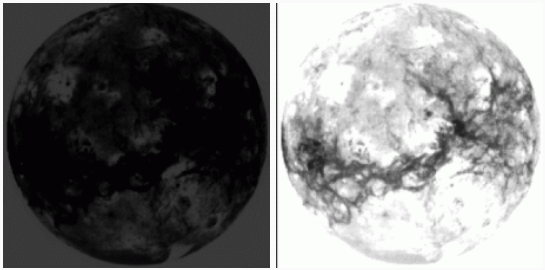


Brightness and contrast

- **Brightness** is the overall lightness or darkness of an image.
- **Contrast** is the difference in brightness between different objects or regions of the image. Often we consider the difference in brightness between the experimental sample and the background.

Brightness and contrast

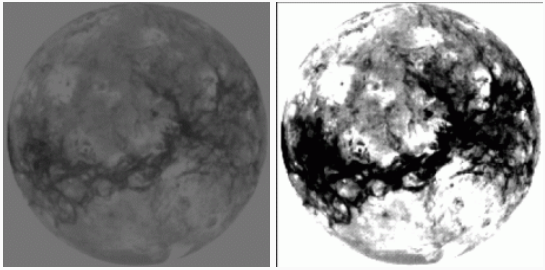
- **Brightness** is the overall lightness or darkness of an image.
- **Contrast** is the difference in brightness between different objects or regions of the image. Often we consider the difference in brightness between the experimental sample and the background.



Low and high brightness versions of the same image. The left image is underexposed, while the right image is overexposed. [Source: Digital Signal Processing - S. W. Smith]

Brightness and contrast

- **Brightness** is the overall lightness or darkness of an image.
- **Contrast** is the difference in brightness between different objects or regions of the image. Often we consider the difference in brightness between the experimental sample and the background.



Low and high contrast versions of the same image. The left image has low contrast, so it is more difficult to distinguish the object from the background. [Source: Digital Signal Processing - S. W. Smith]

Noise

Noise is an aspect that **decreases** the amount of information contained in an image. There are several sources of noise:

- Low amount of light
  - In microscopy because of low intensity staining or because of low power light or other microscope settings.
  - In images taken with a camera because of low-ambient light.

Noise

Noise is an aspect that **decreases** the amount of information contained in an image. There are several sources of noise:

- Low amount of light
  - In microscopy because of low intensity staining or because of low power light or other microscope settings.
  - In images taken with a camera because of low-ambient light.
- Shot noise. This is due to fluctuations of the number of photons detected by the system due to the dual wave/particle nature of light.
- Detector noise. In digital images, this is due to the inherent noise of the sensor, because of electronic circuit noise, temperature, etc

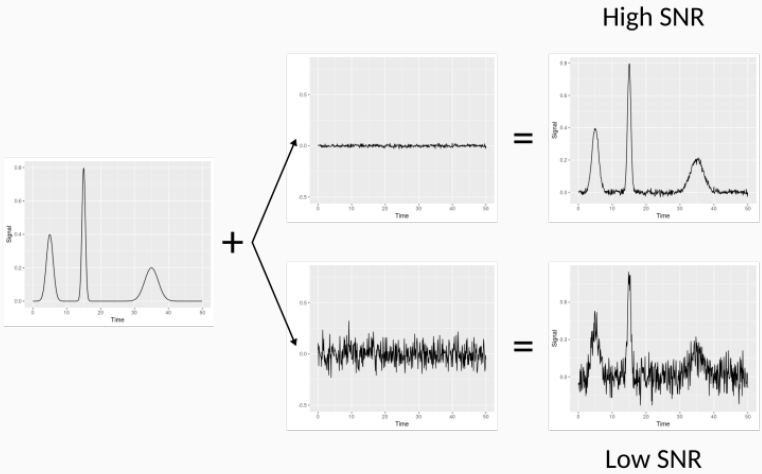
Noise

Noise is an aspect that **decreases** the amount of information contained in an image. There are several sources of noise:

- Low amount of light
  - In microscopy because of low intensity staining or because of low power light or other microscope settings.
  - In images taken with a camera because of low-ambient light.
- Shot noise. This is due to fluctuations of the number of photons detected by the system due to the dual wave/particle nature of light.
- Detector noise. In digital images, this is due to the inherent noise of the sensor, because of electronic circuit noise, temperature, etc
- Autofluorescence of sample. Particularly problematic for green and red wavelengths; non-biological debris often fluoresce strongly in the UV. Far-red and infrared less affected.
- Ambient light interference.

Noise and loss of information

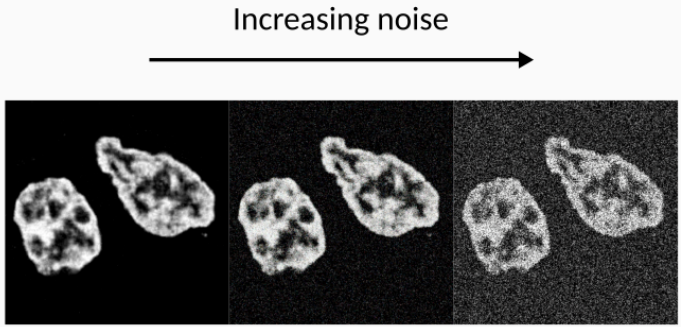
Noise can mask important information in a signal. **Signal-to-noise ratio (SNR)** is a measure of the intensity of the signal we are trying to measure compared to the noise.



SNR in images

There are different, domain-specific ways of calculating SNR; for biomedical images it is often useful to consider the area containing the sample as signal and the background as "noise".

$$SNR = 10 * log_{10} \frac{\mu_{signal}}{\sigma_{noise}}$$



Summary

Images are used as an important source of information in biomedical sciences.

Quantitative imaging relies on the representation of images as n-dimensional matrices of pixels.

Python provides tools for dealing with matrices (e.g. numpy) and for image analysis in general.

Information depends on many aspects of the image and

Next lecture will deal with some basic, but important preprocessing steps that you will use in most image analysis workflows.