Quantitative Big Imaging - Statistics

Anders Kaestner

Apr 10, 2025

CONTENTS

This is the lecture notes fot the 8th lecture of the Quantitative big imaging class given during the spring semester 2021 at ETH Zurich, Switzerland.

0.1 Uncertainty, Statistics, and Reproducibility

0.1.1 Let's load some modules

```
%load_ext autoreload
%reload_ext autoreload
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
import skimage.filters as flt
import pandas as pd
import matplotlib.patches as patches
import tifffile as tiff
from lecture8_support import *
import confmap as cm
import sys
sys.path.append('../common/')
import plotsupport as ps
import pointcloud as pc
from IPython.display import Markdown, display

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

dcolors = plt.rcParams['axes.prop_cycle'].by_key()['color']
%matplotlib inline
%config InlineBackend.figure_format = 'retina' # This is get sharper plots on digh
 ↪density screens like the Retina screen on Macs
```

0.1.2 Literature / Useful References

Books

- Julien Claude, *Morphometry with R*, **Chapter 3**
- John C. Russ, *The Image Processing Handbook*,(Boca Raton, CRC Press)
- Gregory J. Privitera, *Statistics for the Behavioral Sciences* **Chapter 8**
- Drosg, 2009, "Dealing with uncertainties", Springer Verlag
- M. Grabe, 2014,"Measurement Uncertainties in Science and Technology", Springer Verlag
- Ch. Gillmann, 2018,"Image processing under uncertainty", PhD Thesis, Uni Kaiserslautern
- Leland and Wilkinson, *Grammar of Graphics*

Papers / Sites

- Databases Introduction
- Measurement errors - European Commission Glossary
- Detection limit - Wikipedia
- Error propagation
- Error bands - Stack exchange
- Visualizing Genomic Data (General Visualization Techniques)
- NIMRod Parameter Studies
- M.E. Wolak, D.J. Fairbairn, Y.R. Paulsen (2012) Guidelines for Estimating Repeatability. Methods in Ecology and Evolution 3(1):129-137.
- David J.C. MacKay, Bayesian Interpolation (1991)

Videos / Podcasts

- Google/Stanford Statistics Intro
- Last Week Tonight with John Oliver: Scientific Studies
- Credibility Crisis
- Veritasium: Is Most Published Research Wrong?
- Stand-up maths: The Minecraft boat-drop mystery

Further material

Slides

- Kieran Healy, Data Visualization - A practical introduction
- P-Values with Puppies

0.1.3 Previously on QBI ...

- Image Enhancement
- Highlighting the contrast of interest in images
- Minimizing Noise
- Understanding image histograms
- Automatic Methods
- Component Labeling
- Single Shape Analysis
- Complicated Shapes

0.1.4 Today's outline

- Motivation (Why and How?)
- Scientific Goals
- Reproducibility
- Predicting and Validating
- Statistical metrics and results
- Parameterization
  - Parameter sweep
  - Sensitivity analysis
- Data frames
- Visualization

0.1.5 Quantitative "Big" Imaging

The course has covered imaging enough and there have been a few quantitative metrics,

…but "big" has not really mentioned!

So, what does **big** mean?

- Not just / even large
- it means being ready for *big data*
  - The three V's: **V** olume, **V** elocity, **V** ariety
  - scalable, fast, easy to customize

So what is "big" imaging?

The three V's are one by themselves maybe not really producing big data. The data does, however, grow radically when you combine them. Tomography produce large images. Then we may want to study the sample over time to observe changes in the behaviour, the processes may even be rapid and require many sampling points to follow the process. This was only for a single sample, now we need variation to make a statistically solid conclusion from the measurement (today's lecture).

Still, filling hard drives with data is only one part of big. You also need a strategy to manage, process, and anlyze the data in a reproducible manner. The readiness to handle large amonts of data is what really matters.

0.1.6 Objectives

Scientific Studies all try to get to a single or few numbers

- Make sure this number is describing the structure well (earlier lectures)
- Making sure the number is meaningful (**today!**)

How do we:

1. Compare the number from different samples and groups?
   - Within a sample or same type of samples
   - Between samples

2. Compare different processing steps like filter choice, minimum volume, resolution, etc?
3. Evaluate our parameter selection?
4. *Ensure our techniques do what they are supposed to do?*
5. *Visualize so much data? Are there rules?*

0.1.7 What do we start with?

Going back to our original cell image

1. We have been able to
    - **get rid of the noise** in the image and
    - **find all the cells** (lecture 2-4)
2. We have **analyzed the shape** of the cells (lecture 5)
3. We even **separated cells joined together** using Watershed (lecture 6)
4. We have created even more **metrics characterizing the distribution** (lecture 7)

We have at least a few samples (or different regions),

- large number of metrics and
- and almost as large number of parameters to *tune*

How do we do something meaningful with it?

0.2 Correlation and Causation

One of the most repeated criticisms of scientific work is that correlation and causation are confused.

Correlation

- means a statistical relationship
- very easy to show (single calculation)

Causation

- implies there is a mechanism between A and B
- can be very difficult to show (impossible to prove)

0.2.1 Observational or Controlled

There are two broad classes of data and scientific studies.

- Observational
- Controlled

Each type appears, but it is more likely to perform observational studies in the early stages of a project to gain an overview of the working field. From this study it you will make observations for more detailed studies which then are controlled.

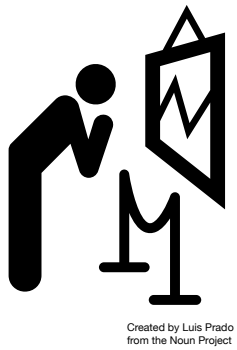


Fig. 1: In observational experiments you stand back and only observe what is happening.

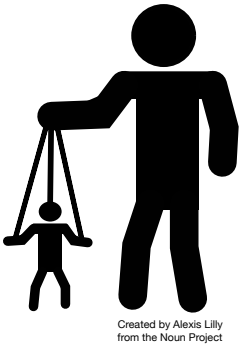


Fig. 2: In controlled experiments you prepare the samples for a specific purpose and control the environment.

Observational

In observational experiments you are not interfering with the observed phenomenon. You only make a selection of specimens or individuals that will be measured as they appear.

Exploring large datasets looking for trends

- Population is random
- Not always hypothesis driven
- Rarely leads to causation

Examples of observational experiments

- We examined 100 people
  - the ones with blue eyes were on average 10cm taller
- In 100 cake samples
  - we found a 0.9 correlation between baking time and bubble size

Controlled

The controlled experiments are designed to explore specific aspects of a population or phenomenon. To achieve this, you want to introduce differences between different groups and keep one as reference. The reference group can be the unmodified samples or samples prepared with an wellknown process.

Most scientific studies fall into this category

- Specifics of the groups are controlled
- Can lead to causation

Examples of controlled experiments

- We examined 50 mice with gene XYZ off and 50 gene XYZ on
  as the foot size increased by 10%
- We increased the temperature
  and the number of pores in the metal increased by 10%

0.3 Qualitative vs Quantitative

From lecture 1

0.3.1 Qualitative Assessment

- Evaluating metrics using visual feedback
- Compare with expectations from
  - other independent techniques
  - or approach
- Are there artifacts which are included in the output?
- Do the shapes look correct?
- Are they distributed as expected?
- Is their orientation meaningful?

0.3.2 Quantitative Metrics

With a quantitative approach, we can calculate

- the specific shape
- or distribution metrics on the sample

with each parameter and establish the relationship between

- parameter
- and metric.

This information will be used in our investigation to tell something about the sample condition allowing us to making conclusions in our investigation.

0.4 Handling uncertainties in image processing

0.4.1 Why do we need to talk about the uncertain?

> "Scientific knowledge is a body of statements of varying degree of certainty
> – some most unsure, some nearly sure, but none absolutely certain."
>
> As found in Feynman, RP (1997) Surely You Are Joking, Mr. Feynman, Norton, New York.

There is no measurement that doesn't have an uncertainty. The question how we use this information in our analysis.

0.4.2 What we want

Error bars beyond the confidence interval of the noise

- Each measurement has an uncertainty
- Uncertainty propagation in image processing is non-trivial

xkcd - 2110

Imaging and image processing are no exception to the fact that there uncertainties related the all measurement. The problem is that there is a chain of uncertainties to consider in the uncertainty analysis. Knowing each is a non-trivial task. In the following sections, you will see some examples of uncertainty analysis.

0.4.3 Error or uncertainty?

Error

The difference between measured value c'
and true value c^*

$$e = |c' - c^*|$$

You need a ground truth!

Uncertainty

The quantification of the doubt about the measurement result.

$$u_{range} = [c' - u, c' + u]$$

Ch. Gillmann, 2018

We often (sloppily) talk about measurement errors, but it is mostly the uncertainty we mean. The error needs the ground truth to be determined.

The uncertainty is an interval around the measured value, like the confidence interval, in which the true value is included.

0.4.4 Which uncertainties are we dealing with?

Technical uncertainties

- Metric
- SNR
- Unsharpness
- Effects from image processing
- Numerical errors

Field specific uncertainties

- Model simplifications
- Variations in population
- Noise

0.4.5 Uncertainty categories

Systematic

Reproducible - no statistical analysis

- Rounding errors
- Uncalibrated systems
- Algorithmic choices
- Etc.

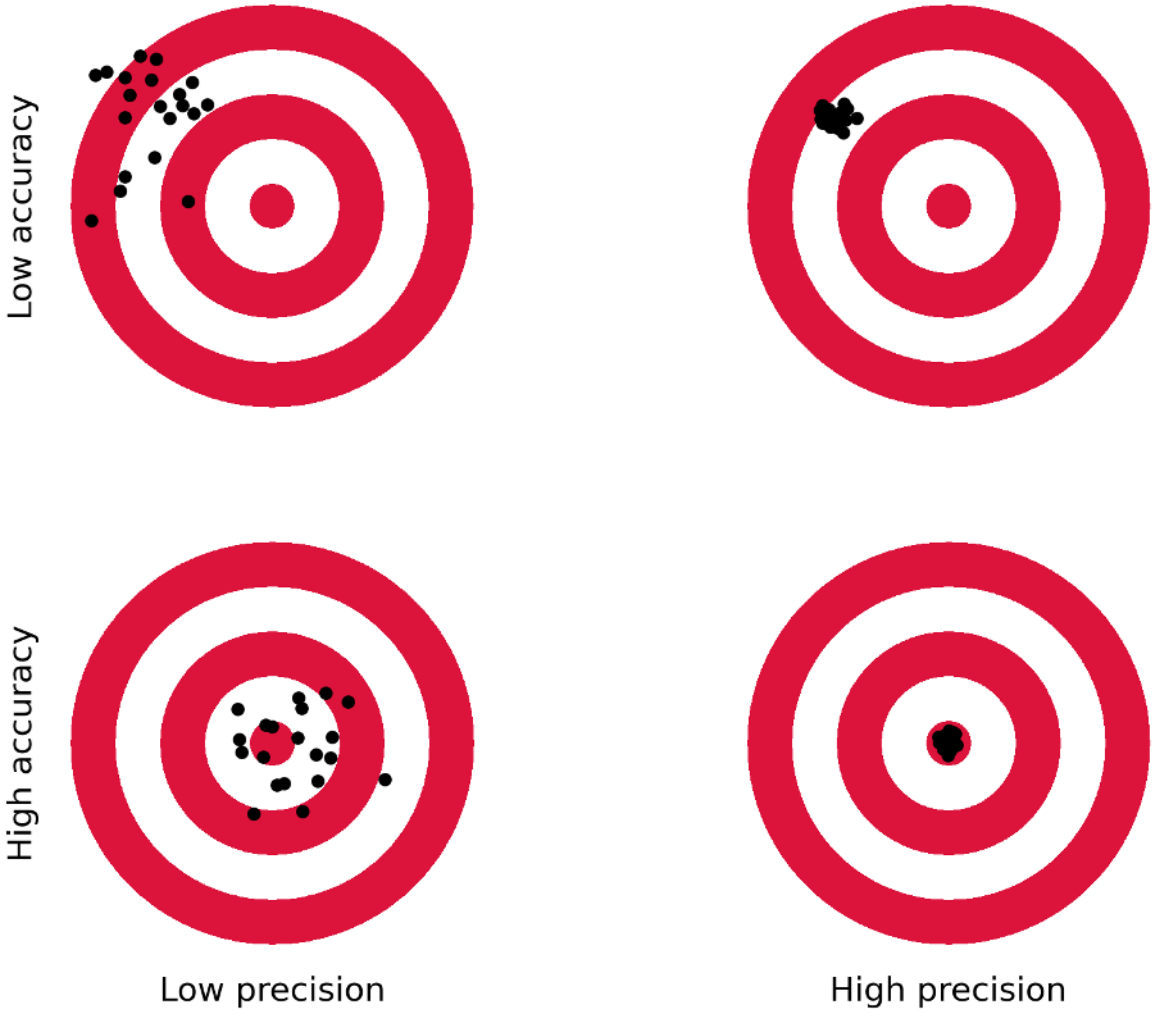
Random

Statistical fluctuations

- Natural variations in studied population
- Source fluctuations
- Detector noise

0.4.6 Target practicing

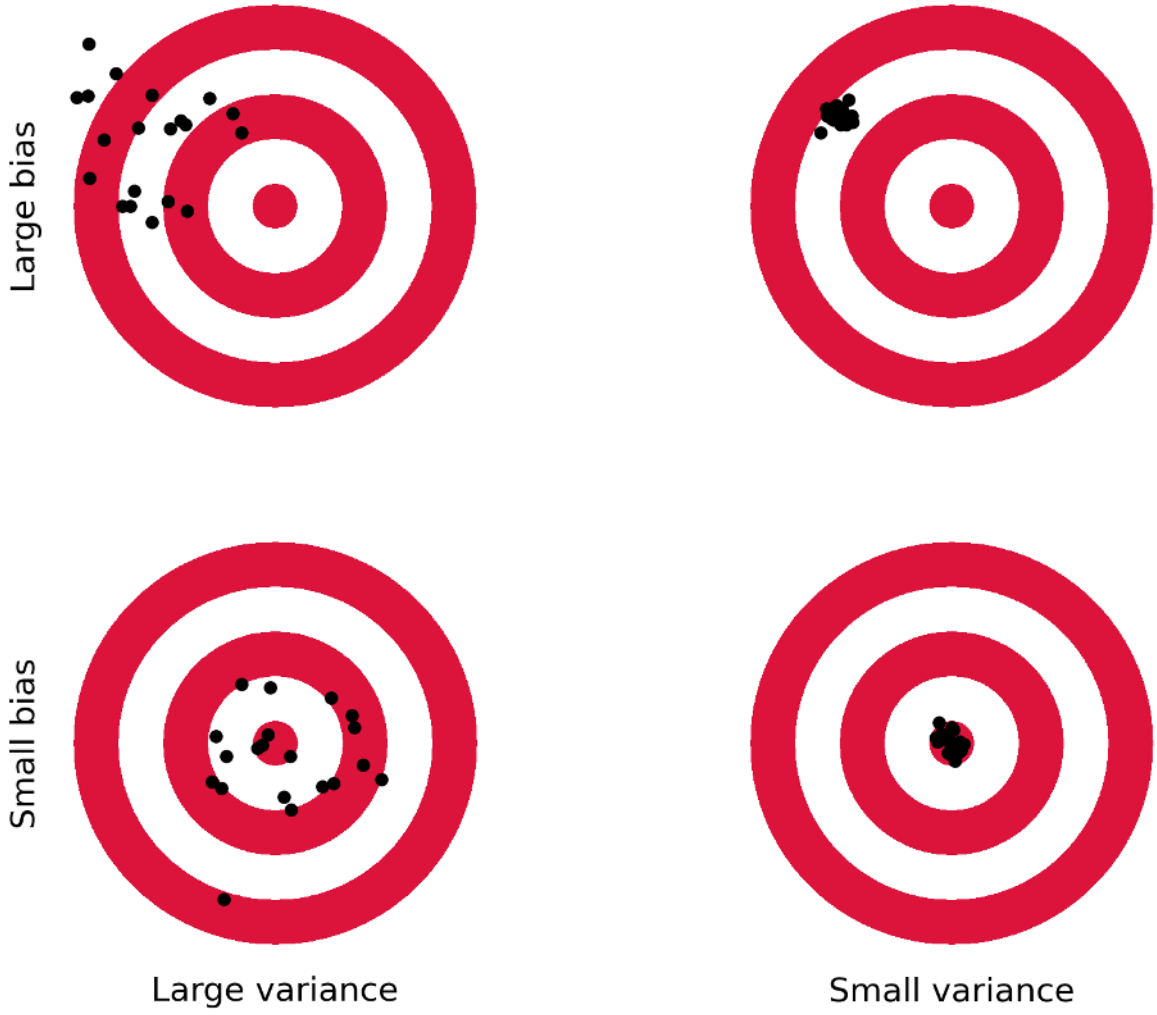
```
precacc(1000)
```



The uncertainty has two components precision and accuracy. The accuracy tells how far away our measurement is from the expected true value. The precision tell how widespread the measurements are.

Target practicing in statistical terms

```
precacc(1000,['Large bias','Small bias','Large variance','Small variance'])
```



Precision and accuracy can be translated in statistical terms such that

- The precicion corresponds to the variance of the measured values, i.e. high precision means that the variance if low.
- The accuracy is translated to a bias in the measurement, i.e. low accuracy means that the bias is great.

0.4.7 Counting or measuring

We always have uncertainties in our measurements. Some, more obvious than others. Here, we will look into the two cases counting and measuring.

Counting

- Discrete countable items
- Absolute values

Uncertainties: from preparation

Example: area of segmented region

The smallest countable item in an image is the pixel. The area is determined as the sum of the pixels in a specified region. Some perimeter metrics are based on counting the boundary pixels.

Counting seems like an absolute task, just count labelled regions and tell how many there are. The uncertainties originate from the image preparation leading to the regions:

1. Does image noise affect the number of detected items?
2. How well was the image segmented? How many misclassified pixels are there?

Measuring

- Physical quantities
- Values with uncertainties

Uncertainties: Noise, instrumentation

Example: mixing ratios from gray levels

In some experiments we observe changes in the gray levels. The gray level has several sources of uncertainty

- The conversion from measured radiation to a signal that can be measured by the detector
- The quantization. How many levels were used in the acquisition.
- Noise
  - counting noise from the radiation
  - detector noise
  - quantization noise
- Algorithmic uncertainty. The reconstruction may be biased.

0.4.8 Absolute vs relative uncertainties

Uncertainties must be compared from a neutral perspective

Absolute uncertainty

Assume we can measure a distance with an uncertainty of 1mm

Distance to the moon - 384400km

The diameter of a coin - 20mm

Relative uncertainty

Relates the error to the measured quantity

Distance to the moon - 0.000000002601457

Diameter of a coin - 0.05

0.4.9 Propagation of uncertainty

Uncertainty of f(x)

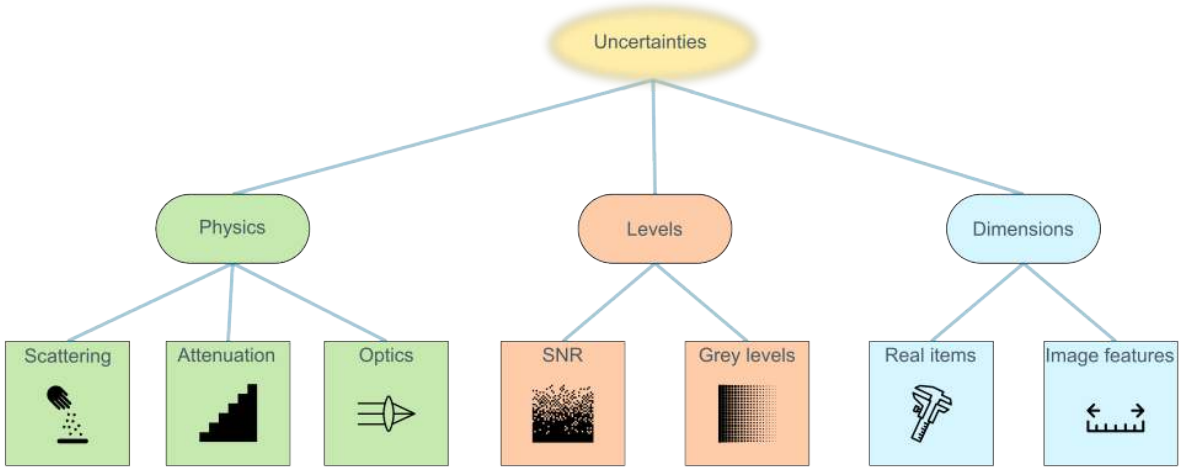
$$\sigma_f(x) = \frac{\partial f}{\partial x} \cdot \sigma_x$$

Uncertainty of f(x,y)

$$\sigma_f(x,y)^2 = \left(\frac{\partial f}{\partial x} \cdot \sigma_x\right)^2 + \left(\frac{\partial f}{\partial y} \cdot \sigma_y\right)^2 + \underbrace{\frac{\partial f}{\partial x} \cdot \frac{\partial f}{\partial y} \cdot \overline{\sigma_{xy}}}_{\text{x and y uncorrelated?}}$$

0.4.10 Uncertain quantities in imaging experiments

- Pixel size
- Segmentation
- Sampling time stamps in time series
- Intensity levels



Three examples

We want to measure:

- The pixel size
- The perimeter length
- Water volume behind a pixel

Example: measure the pixel size

To measure pixel size we need:

- An object with known length
- An image of the object

$$\text{pixel size} = \frac{\text{Object length}}{\text{Pixel distance between edges}}$$

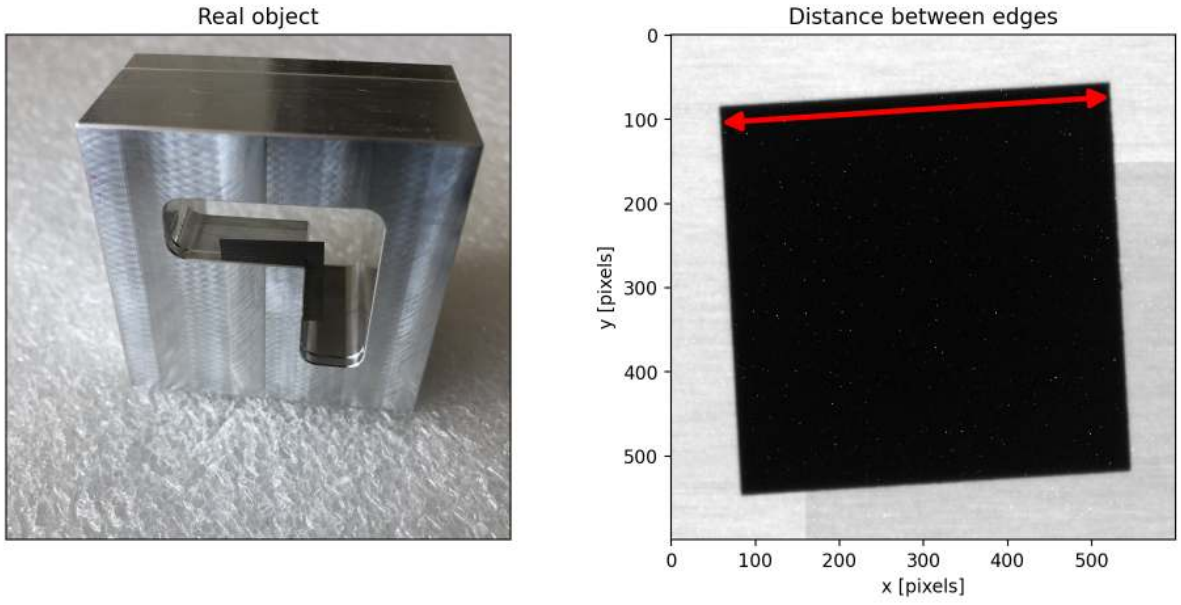
Uncertainty equation

We assume that the two measured values are uncorrelated $\$\frac{\sigma_{a/b}}{a/b} = \sqrt{\left(\frac{\sigma_a}{a}\right)^2 + \left(\frac{\sigma_b}{b}\right)^2}\$$

From this tutorial

Our measurements

```
img1 = tiff.imread('data/edge20mm_0000.tif')
pic = plt.imread('figures/edge_object.jpg')
fig,ax=plt.subplots(1,2,figsize=(12,5))

ax[0].imshow(pic)
ax[0].set_xticks([])
ax[0].set_yticks([])
ax[0].set_title('Real object')

ax[1].imshow(img1,vmin=300,vmax=30000,cmap='gray');
ax[1].set_xlabel('x [pixels]')
ax[1].set_ylabel('y [pixels]');
arrow = patches.FancyArrowPatch((55,104), (526, 73),
                                mutation_scale=20,
                                #ec='blue',fc='cornflowerblue',
                                color='red',
                                arrowstyle='<|-|>',linewidth=3
                               )
ax[1].add_patch(arrow);
#ax.annotate(text='', xy=(54,104), xytext=(527,71), arrowprops=dict(arrowstyle='<->',
 ↪color='yellow',lw=4))
d=np.sqrt((528-53)**2+(104-73)**2)
ax[1].set_title('Distance between edges');
```



Quantity	Measurement	Uncertainty	Unit
Caliper distance	20.0	0.05	mm
Pixel distance	464.9	0.32	pixels

A tutorial showing the detailed analysis

Pixel size with uncertainty

We know uncertainty equation for a/b.

Let's plug in the pixel meassurements:

$$\frac{\sigma_{Pixelsize}}{Pixelsize} = \sqrt{\left(\frac{\sigma_{pixels}}{pixels}\right)^2 + \left(\frac{\sigma_{length}}{length}\right)^2}$$

```
length       = 20.0 # mm
error_length = 0.05 # mm
pixels       = 464.9 # pixels
error_pixels = 0.32  # pixels

pixel_size = length/pixels

rel_uncertainty = np.sqrt((error_pixels/pixels)**2 + (error_length/length)**2)

display(Markdown('<span style="font-size:1.75em">Pixel size = {0:0.2f} $\mu$m +/-
 ↪{1:0.2f}$\mu$m</span>'.format(pixel_size*1000,pixel_size*rel_uncertainty*1000)))
```

Pixel size = 43.02 μm +/- 0.11μm

The uncertainties package

Uncertainties can be tricky to compute. The uncertainties package takes some of this burden from you

```
from uncertainties import ufloat

length = ufloat(20.0,0.05)
pixels = ufloat(464.9,0.32)

pixel_size = length/pixels

print(pixel_size)
```

```
0.04302+/-0.00011
```

Measurements in segmented images

In lecture 5 we learned to measure

- Area
- Perimeter
- Positions
- Distances

Considering them to be absolute values as we mostly just count pixels…

Let's see how the uncertainty of the pixel size among others changes this

Meassure the perimeter length

The perimeter length has three sources of uncertainty:

- How were the edge pixels identified? *Method choices can introduce biases*
- The pixel size…

and…

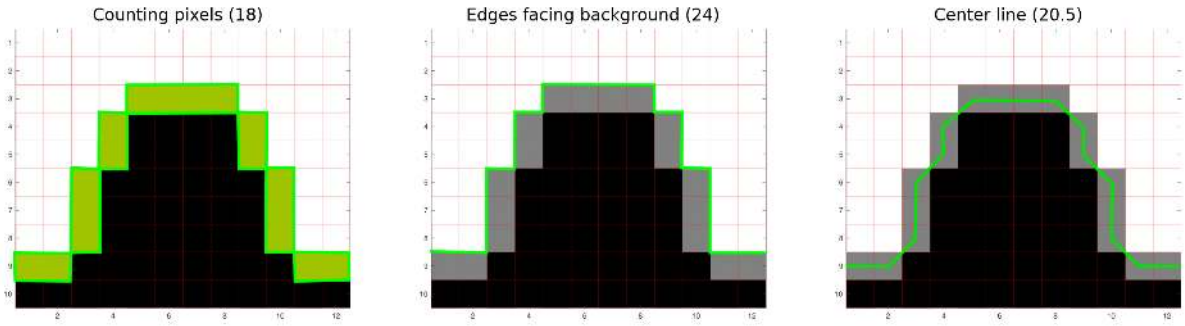
- How well was the image segmented?
  Edges are segmented with least confidence

Selecting edge pixels

A method to identify edge pixels is $$edge(f) = f - \varepsilon_{SE}(f)$$ which pixels depends on the used SE.

Counting perimeter elements

```
imgs=[plt.imread('figures/edge_detail_pixel_count.png'),
      plt.imread('figures/edge_detail_outline.png'),
      plt.imread('figures/edge_detail_centerline.png')]
lbls=['Counting pixels (18)', 'Edges facing background (24)','Center line (20.5)']

_,axs = plt.subplots(1,3, figsize=(15,6))
for ax,img,lbl in zip(axs,imgs,lbls) :
    ax.imshow(img)
    ax.set_title(lbl)
    ax.axis('off')
```



Perimeter length with uncertainty

Let's assume we can trust the segmentation:

- Edge is 18 pixels long
- The pixel size is 43.02 μm $\pm$ 0.11μm

Perimeter uncertainty equation

In this case we sum the edge pixels

Derivation of the equation

The uncertainty of a sum of measurements $\sum_{i=1}^{N} x_i$ each with all x_i uncorrelated and the same uncertainty σ_x.

Lets start with the derivative: $\frac{\partial f}{\partial x_i} = 1 \quad \forall i$

now the uncertainty is the sum of all x_i

$\sigma_{\sum x_i} = \sum_{i=1}^{N} \sigma_x = N\sigma_x$

which leads to

$$\sigma_{\sum x_i} = \sigma_x \cdot N$$

We have $\sigma_{Pixelsize} = 0.11\ \mu m$, which gives $\sigma_{Edgelength} = 0.11 \cdot 18 = 1.98 \mu m$

The edge length is $774 \pm 2.0\ \mu m$

Measure water volume from gray levels

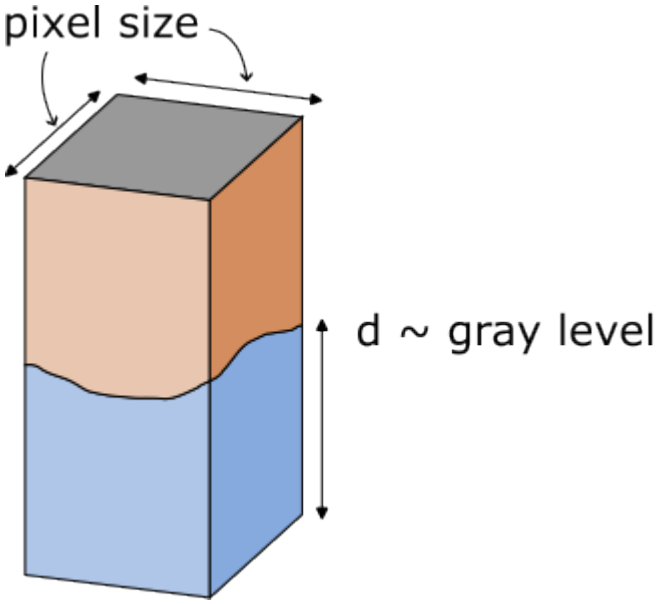
In neutron imaging it is common to quantify the water content from radiographs.

- The transmission (gray level) is $T = \frac{I}{I_0} = e^{-\mu \cdot d}$
- The metric pixel area (pixel size)

The water volume is $V_{water} = pixelsize^2 \cdot d$

Our uncertainties are now:

- Standard deviation of the transmission (confidence interval)
- The pixel size … again



0.5 Volume with uncertainty

Let's assume we can trust the segmentation:

- Thickness $d = 5 \pm 0.1\ mm$
- The pixel size is $p = 43 \pm 0.11\ \mu m$

0.5.1 Volume uncertainty equation

In this case, we know the uncertainty of $V = p^2 \cdot d$

Derivation of the equation

The uncertainty of a multi product $\prod_{i=1}^{N} x_i$ each with all x_i uncorrelated and the uncertainty σ_{x_i}, or more specific $x^2 \cdot y$ with σ_x σ_y.

Lets start with the derivative:

$\frac{\partial(x^2 \cdot y)}{\partial x} = 2xy$

and

$\frac{\partial(x^2 \cdot y)}{\partial y} = x^2$

Now the uncertainty is ${\sigma_{x^2 \cdot y}}^2 = \left(2xy \cdot \sigma_x\right)^2 + \left(x^2 \cdot \sigma_y\right)^2$

Dividing both sides by $x^2 \cdot y$ and taking the square root to get the relative uncertainty

$\frac{\sigma_{x^2 \cdot y}}{x^2 \cdot y} = \sqrt{\frac{\left(2xy \cdot \sigma_x\right)^2}{\left(x^2 \cdot y\right)^2} + \frac{\left(x^2 \cdot \sigma_y\right)^2}{\left(x^2 \cdot y\right)^2}} = \sqrt{4\left(\frac{\sigma_x}{x}\right)^2 + \left(\frac{\sigma_y}{y}\right)^2}$

Which gives the relative uncertainty $\$\frac{\sigma_V}{V} = \sqrt{4\left(\frac{\sigma_{Pixelsize}}{Pixelsize}\right)^2 + \left(\frac{\sigma_d}{d}\right)^2}\$$

0.5.2 Computing the volume with uncertainty

Plugging in the measurements in the equation:

$$\frac{\sigma_V}{V} = \sqrt{4\left(\frac{\sigma_{Pixelsize}}{Pixelsize}\right)^2 + \left(\frac{\sigma_d}{d}\right)^2}$$

```
d           = 5.0       # mm
d_uncertain = 0.1      # mm
p           = 0.043     # mm
p_uncertain = 0.00011   # mm

V = p**2 * d

rel_uncertainty = np.sqrt(4*(p_uncertain/p)**2 + (d_uncertain/d)**2)

display(Markdown('<span style="font-size:1.75em">Volume V = ({0:0.5f} +/- {1:0.5f}) mm
↪$^3$</span>'.format(V,rel_uncertainty*V)))
```

Volume V = (0.00924 +/- 0.00019) mm^3

0.5.3 Confidence of a segmentation

Freely from P. Moonen's talk at ICTMS2019

There are different ways to evaluate segmentation algorithm performance; *From lectures 2, 4, and 5*

- Confusion matrix
- ROC curve
- Hit map

… but they require a ground truth

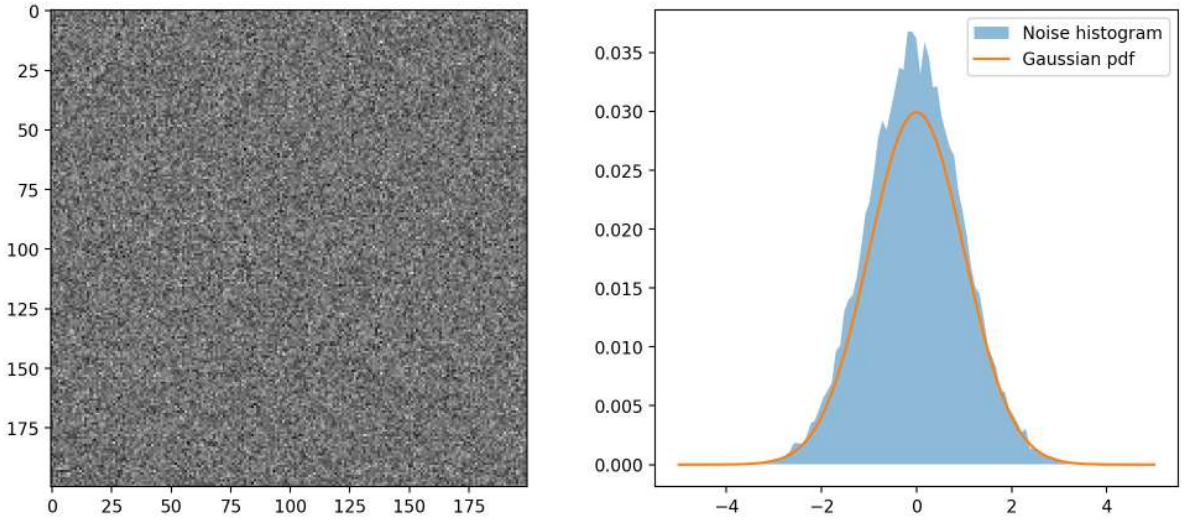
What if we want to know how well an image was segmented without ground truth?

```
import importlib
importlib.reload(cm);
```

0.5.4 Confidence map without ground truth

Assume Gaussian noise

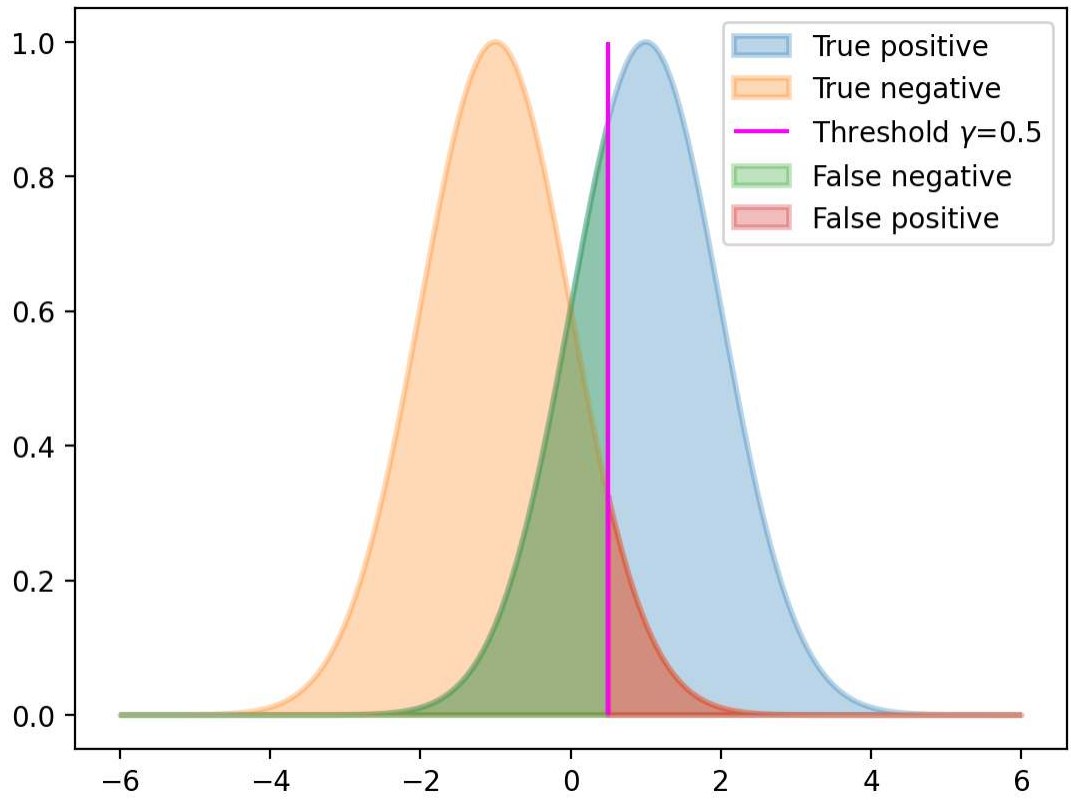
```
s0=1;
noise = np.random.normal(size=[200,200])
fig,ax = plt.subplots(1,2,figsize=(12,5))
ax[0].imshow(noise,cmap='gray')
h,bins = np.histogram(noise.ravel(),bins=100)
normh=h/h.sum()
ax[1].fill(bins[:-1],normh,alpha=0.5,label='Noise histogram')
x=np.linspace(-5,5,201)
pdf = cm.gaussian(x,0,1)
ax[1].plot(x,1.5*pdf/pdf.sum(),color=dcolors[1], label='Gaussian pdf')
ax[1].legend();
```



Threshold between two Gaussian classes

From lecture 4

```
x=np.linspace(-6,6,1000)
h0 = np.exp(-(x-1)**2/2)
h1 = np.exp(-(x+1)**2/2)

gamma=0.5

# Visualization
gidx = np.abs(x - gamma).argmin()
plt.fill(x,h0,label='True positive',alpha=0.3,ec=dcolors[0],lw=2)
plt.fill(x,h1,label='True negative',alpha=0.3,ec=dcolors[1],lw=2)

plt.vlines([x[gidx]],ymin=0,ymax=1,color='magenta',label='Threshold $\gamma$={0}'.
 ↪format(gamma))

plt.fill_between(x[:gidx],0,h0[:gidx],color=dcolors[2],label='False negative',alpha=0.
 ↪3,ec=dcolors[2],lw=2)
plt.fill_between(x[gidx:],0,h1[gidx:],color=dcolors[3],label='False positive',alpha=0.
 ↪3,ec=dcolors[3],lw=2)

plt.legend();
```



Compute the confidence based on the class pdf's

We want the probability mix of the classes for each gray level:

1. Find the pdf for each class using Gaussian Mixture Models (GMM)
2. Normalize
    - Peak to one
    - AOC is one
3. For each gray level x compute weights $w_i(x) = pdf_i(x) for Gaussians w_i(x)=e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}$
4. Then the confidence for class i is $c_i(x) = \frac{w_i(x)}{\sum_j w_j(x)}$
5. Create a confidence map using the segmented image to select c_i for each pixel

What are the weights?

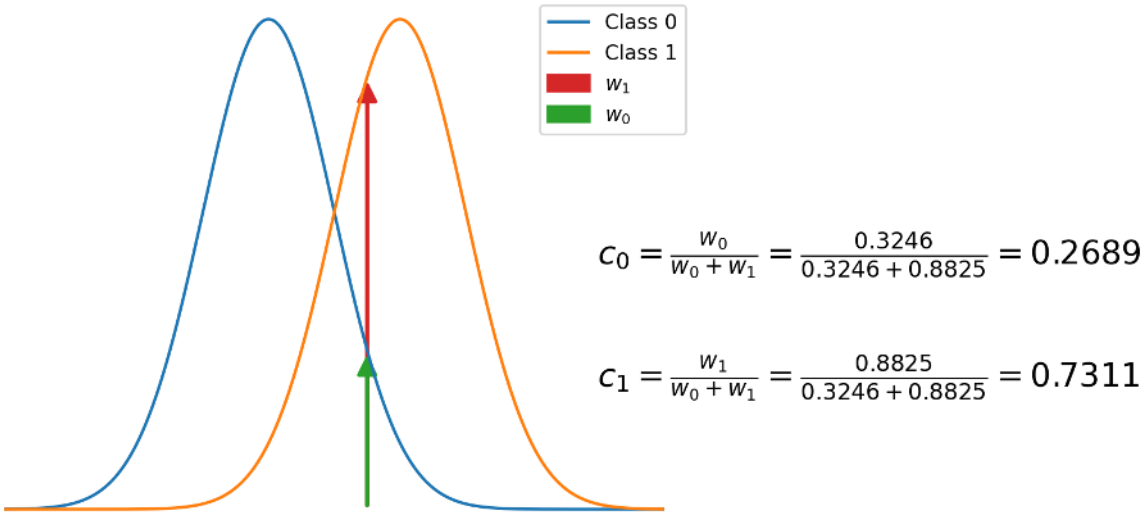
In our algorithm we used $w_i(x) = e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}$

```
x=np.linspace(-5,5,1000)
g0=cm.gaussian(x,-1,1)
g0m=g0.max()
g0=g0/g0.max()
g1=cm.gaussian(x,1,1)
g1m=g1.max()
g1=g1/g1.max()
_,ax=plt.subplots(1)
ax.plot(x,g0,label=r'Class 0')
ax.plot(x,g1,label=r'Class 1')
xx=0.5
arrow = patches.FancyArrowPatch((xx,0), (xx,cm.gaussian(xx,1,1)/g1m),
                                mutation_scale=20,
                                color=dcolors[3],
                                arrowstyle='-|>',linewidth=2,
                               label=r'$w_1$'
                               )
ax.add_patch(arrow);

arrow = patches.FancyArrowPatch((xx,0), (xx,cm.gaussian(xx,-1,1)/g0m),
                                mutation_scale=20,
                                color=dcolors[2],
                                arrowstyle='-|>',linewidth=2,
                               label=r'$w_0$'
                               )
ax.add_patch(arrow);

ax.text(4,0.5,r'$c_0=\frac{w_0}{w_0+w_1}=\frac{0.3246}{0.3246+0.8825}=0.2689$',
 ↪size=16)
ax.text(4,0.25,r'$c_1=\frac{w_1}{w_0+w_1}=\frac{0.8825}{0.3246+0.8825}=0.7311$',
 ↪size=16)

plt.legend()
plt.axis('off');
```

(continues on next page)

(continued from previous page)

```
# print(cm.gaussian(xx,-1,1)/g0m,cm.gaussian(xx,1,1)/g1m,cm.gaussian(xx,-1,1)/g0m/(cm.
↪gaussian(xx,-1,1)+cm.gaussian(xx,1,1)/g1m),cm.gaussian(xx,1,1)/g0m/(cm.gaussian(xx,-
↪1,1)+cm.gaussian(xx,1,1)/g1m))lass
```



Example: Three phase segmentation

Our test data

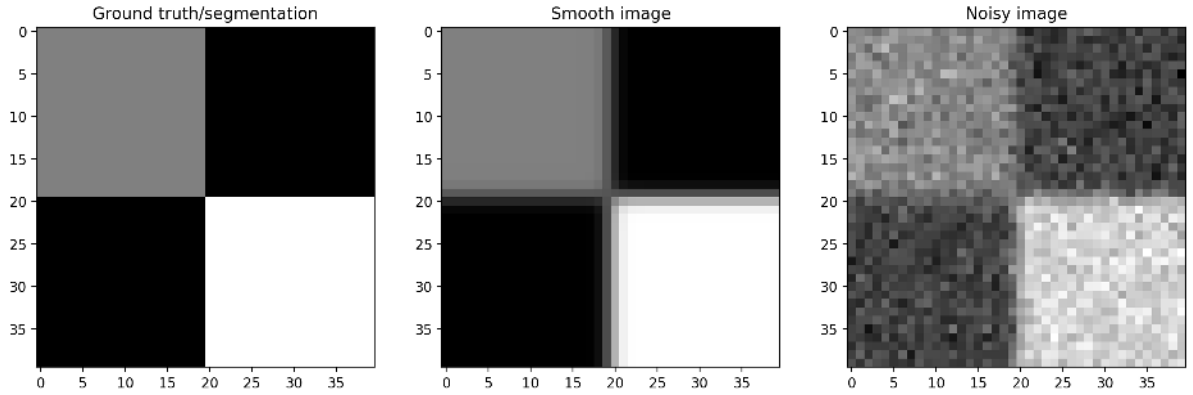
- Three classes with μ=[-3,1,5]
- Gaussian noise $\mathcal{N}(\mu, \sigma = 1.0)$
- Unsharpness - Gaussian filter (σ=1)

```
ss = 1
N = 20
m=np.array([-3,1,5])
s=np.array([ss,ss,ss])
xx=np.linspace(-10,10,1001)
res = np.array([cm.multi_gaussian(x,m=m,s=s) for x in xx])

img=np.repeat(np.repeat(np.array([[m[1],m[0]],[m[0],m[2]]]),N,axis=0),N,axis=1)
seg=np.zeros(img.shape)
fimg = flt.gaussian(img,sigma=1,preserve_range=True)
for idx in range(len(m)):
    seg[img==m[idx]]=idx

nm = fimg+np.random.normal(loc=0,scale=ss,size=img.shape)

cmap = cm.conf_map(nm,seg,m,s)

fig,ax=plt.subplots(1,3,figsize=(15,5))
ax=ax.ravel()
ax[0].imshow(seg,cmap='gray')
```

(continues on next page)

(continued from previous page)

```
ax[0].set_title('Ground truth/segmentation')
ax[1].imshow(fimg,cmap='gray')
ax[1].set_title('Smooth image')
ax[2].imshow(nm,cmap='gray')
ax[2].set_title('Noisy image');
```



The confidence maps for the segmenation

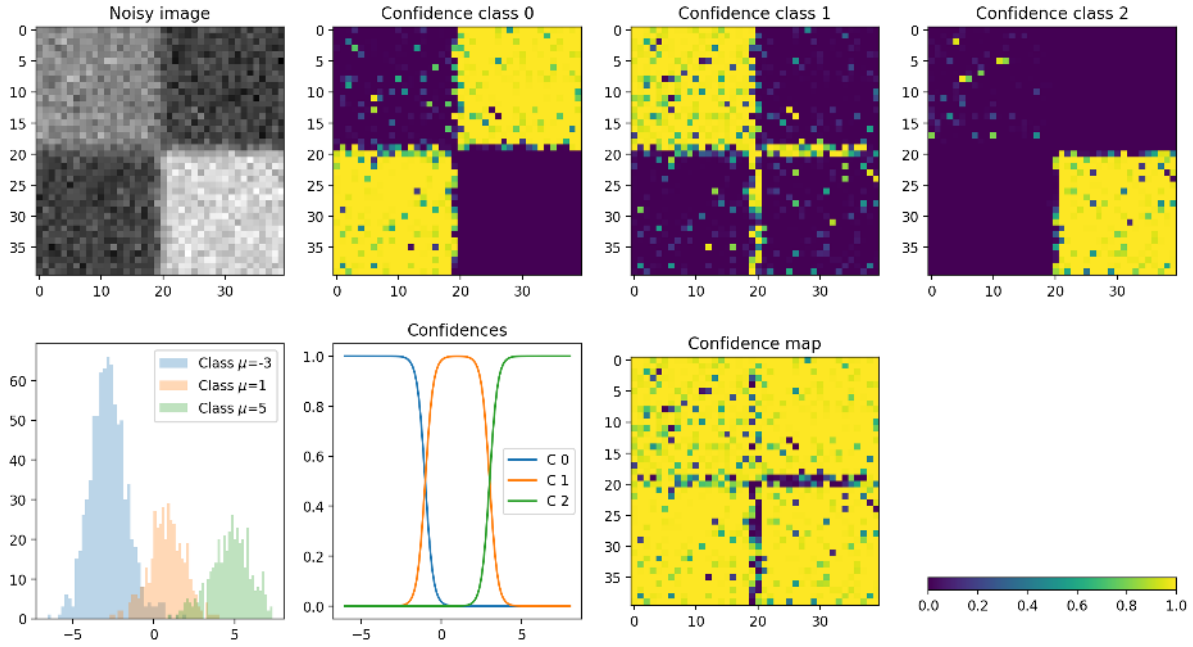
Here, we cheat by using the original class image as segmented image

```
seg=np.zeros(img.shape)
fimg = flt.gaussian(img,sigma=1,preserve_range=True)
for idx in range(len(m)):
    seg[img==m[idx]]=idx

fig,ax=plt.subplots(2,4,figsize=(15,8))

ax=ax.ravel()
ax[0].imshow(nm,cmap='gray')
ax[0].set_title('Noisy image')

for idx in range(3):
    cmap = cm.conf_map(nm,seg,m,s,c=idx)
    ax[idx+1].imshow(cmap,clim=[0,1])
    ax[idx+1].set_title('Confidence class {}'.format(idx))

ax[4].hist(nm[seg==0].ravel(),bins=40, label="Class $\mu$=-3", alpha=0.3);
ax[4].hist(nm[seg==1].ravel(),bins=40, label="Class $\mu$=1", alpha=0.3);
ax[4].hist(nm[seg==2].ravel(),bins=40, label="Class $\mu$=5", alpha=0.3);
ax[4].legend()

# ax[4].plot(xx,res);
# ax[4].set_title('Normalized class distributions')

cmap = cm.conf_map(nm,seg,m,s)
ax6=ax[6].imshow(cmap,clim=[0,1])
ax[6].set_title('Confidence map')
xx,w = cm.conf_plot(m,s)
for idx in range(len(m)) :
    ax[5].plot(xx,w[idx]/np.sum(w,axis=0), label="C {}".format(idx));
```

(continues on next page)

(continued from previous page)

```
ax[5].legend()
ax[5].set_title('Confidences');

ax[7].axis('off')
fig.colorbar(ax6,ax=ax[7],location='bottom');
```



0.5.5 When does it matter?

High SNR images

Edges have lowest confidence; most impact for

- Perimeter
- Turtosity
- Network connectivity
- Volumes of small items

Low SNR image

Low confidence on most pixels!

- All high-SNR issues

and

- Porosity
- Volumes
- Labeling

0.5.6 Concluding uncertainties

- Scientific analysis requires uncertainties
- Uncertainties in imaging is not straight forward
  - Physical uncertainties can be quantified
  - Noise can be measured
  - Image analysis precision depends on object/pixel size ratio
  - The effect of processing must be quantified through simulations

0.6 Statistical analysis of experiments

It often convenient to start with a simplified models for your experiments where most uncertainties are reduced. In particular here in this lecture we chose a simple model for the demonstration

We go for a simple model…

Since most of the experiments in science are usually

- Application specific,
- Noisy,
- and often very complicated

and are not usually good teaching examples.

0.6.1 A simple Model: Magic / Biased Coin

Our model is the task to flip a coin and determine if it is a fair or loaded. The coin has two outcomes

- head
- or tail

You buy a *magic* coin at a shop.

How many times do you need to flip it to *prove* it is not fair?

Next step is to describe an experiment strategy. Some examples are:

If I flip it 10 times and …

- another person flips it 10 times. Is that the same as 20 flips?
- then multiply the results by 10. Is that the same as 100 flips?

As you already may have guessed, these are not the best assumtions, in particular not the second one.

A different question is about collections of random variables:

What if

- I buy 10 coins and want to know which ones are fair, what do I do?

Fig. 1: Tossing a coin is a simple random process.

Experiment: Magic / Weighted Coin

1. Each coin represents a stochastic variable $\mathcal{X}$ and each flip represents an observation $\mathcal{X}_i$.
2. The act of performing a coin flip $\mathcal{F}$ is an observation $\mathcal{X}_i = \mathcal{F}(\mathcal{X})$

We normally assume:

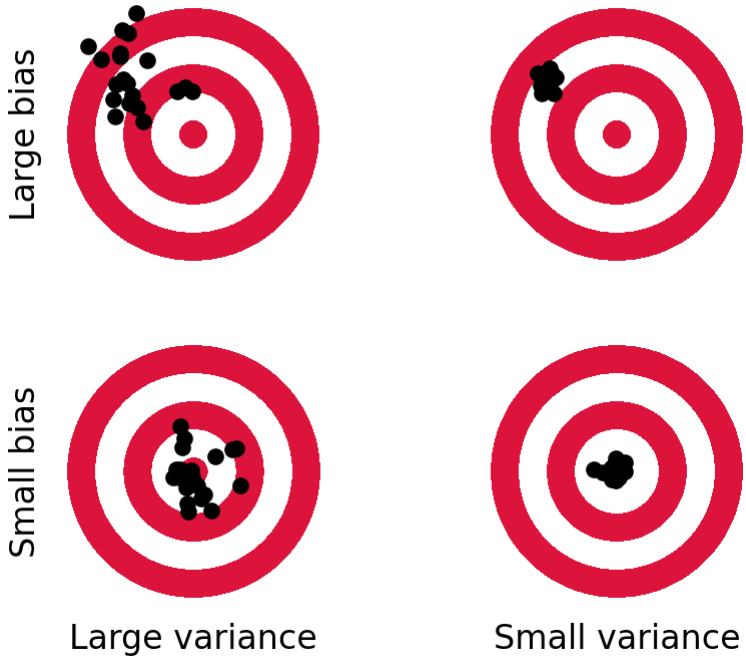
1. A *fair* coin has an expected value of $E(\mathcal{X}) = \frac{1}{2}$:
   - 50% Heads,
   - 50% Tails
2. An *unbiased* flip(er) means *each flip is independent of the others* $$P(\mathcal{F}_1(\mathcal{X}) \cdot \mathcal{F}_2(\mathcal{X})) = P(\mathcal{F}_1(\mathcal{X})) \cdot P(\mathcal{F}_2(\mathcal{X}))$$
   - the expected value of the flip is the same as that of the coin $$E\left(\prod_{i=0}^{\infty} \mathcal{F}_i(\mathcal{X})\right) = E(\mathcal{X})$$

0.6.2 Transfer the Simple Model to Reality

Coin Flip

1. Each flip gives us a small piece of information about
   - the coin
   - *and* the flipper
2. More flips provides more information
   - **Random / Stochastic variations** in coin and flipper **cancel out**
   - **Systematic variations accumulate**

```
precacc(1000,['Large bias','Small bias','Large variance','Small variance'],figsize=[5,
↪4],fontsize=12)
```



Real experiment

1. Each measurement tells us about:
    - our sample,
    - our instrument,
    - *and* our analysis
2. More measurements provide more information:
    - **Random / Stochastic** variations in *sample, instrument, and analysis* **cancel out**
    - *Normally*, the analysis has very little to no stochastic variation
    - **Systematic variations** accumulate

This is also the reason why we want many repeated observations in an experiment. Repetitions are however expensive, they require time to perform the experiment and more material for the specimens. Therefore, a pragmatic choice must be made that balances the cost versus a reasonable amount of observations.

0.7 A more complicated model

Coin flips are very simple and probably difficult to match to another experiment.

A very popular dataset for learning about such values beyond 'coin-flips' is called the Iris dataset.

It covers:

- a number of measurements
- from different plants
- and the corresponding species.

0.7.1 Let's load the Iris Dataset

Fisher, The Use of Multiple Measurements in Taxonomic Problems, 1936

The data set has information about dimensions of the flower anatomy for each of the three species. We load the data which is provided as a python dictionary and prepare a data frame for the table. You will get a more detailed introduction to pandas data frames later in this lecture.

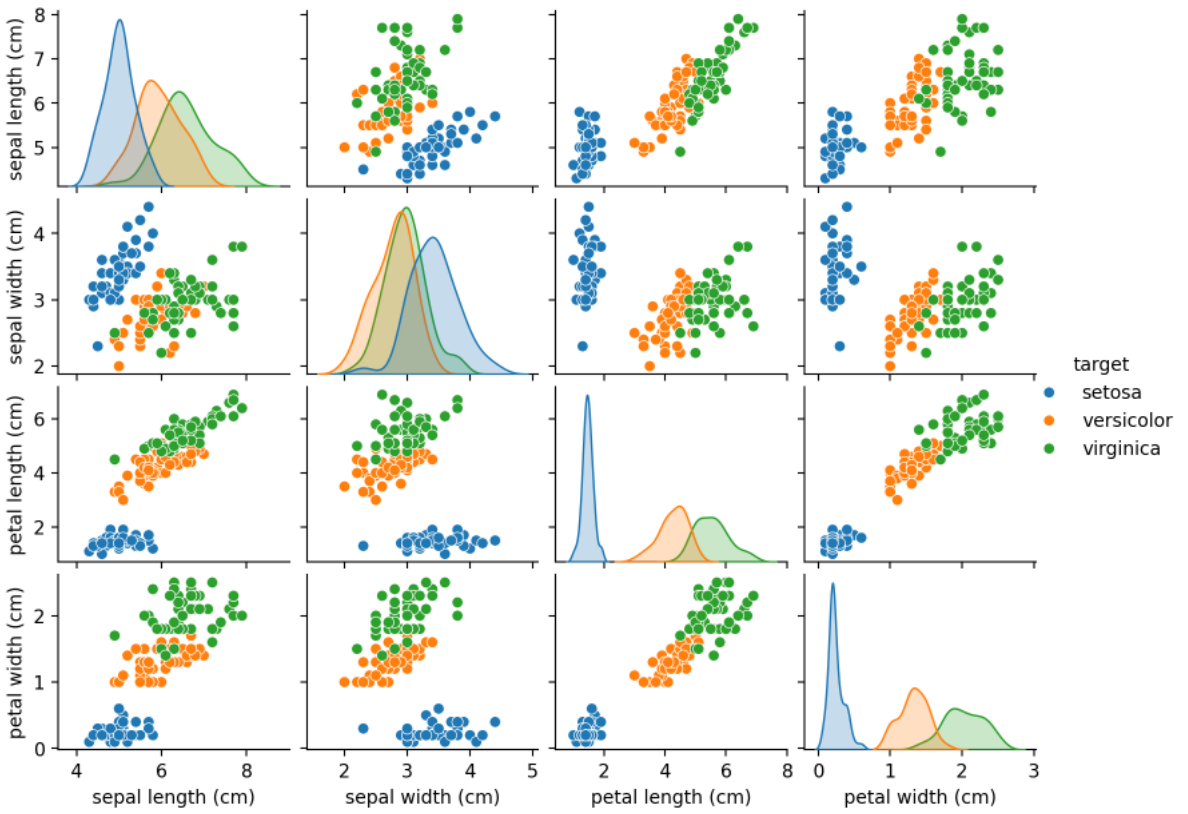
```
data              = load_iris()
iris_df           = pd.DataFrame(data['data'], columns=data['feature_names'])
iris_df['target'] = data['target_names'][data['target']]
iris_df.sample(5)
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
69                 5.6               2.5                3.9               1.1
82                 5.8               2.7                3.9               1.2
116                6.5               3.0                5.5               1.8
83                 6.0               2.7                5.1               1.6
138                6.0               3.0                4.8               1.8

         target
69   versicolor
82   versicolor
116   virginica
83   versicolor
138   virginica
```

0.7.2 A first inspection of the data

We use a pair plot to inspect the table. Each target species is assigned a color to allow conclusions regarding clusters.

```
p=sns.pairplot(iris_df, hue='target');
plt.gcf().set_size_inches(9, 6)
```



In the plot, we clearly see that one species (setosa) in general has other flower leaf dimensions than the other two.

0.7.3 Comparing Groups: Intraclass Correlation Coefficient

The intraclass correlation coefficient basically looking at

- how similar objects within a group are
- compared to the similarity between groups

Group similarity

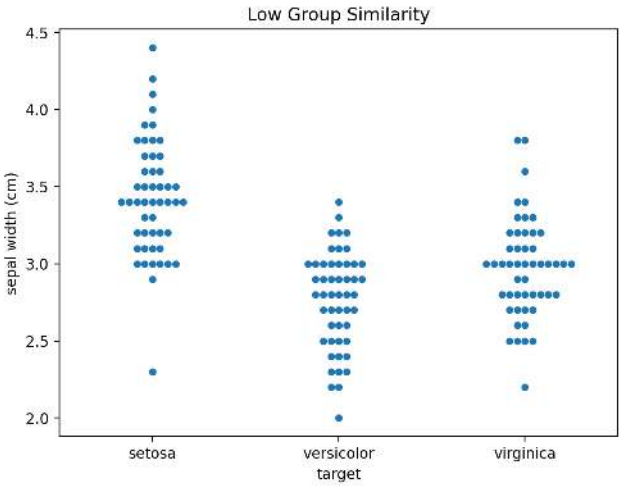
How well are groups separated in a study

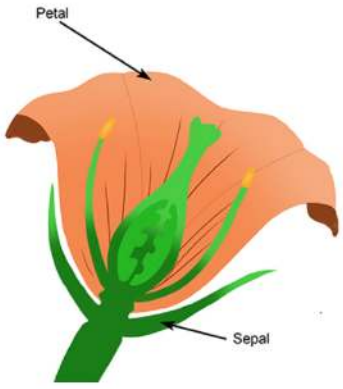
- Low group similarity - overlapping histograms, harder to separate
- High group similarity - separated histograms, easier to separate

Sepal width

Sepals are the green leaves of the flower bud. In this swarm plot we look at the width of the sepals and see that the variance of each class is about the same and also the the average width doesn't vary much. Under such conditions it is hard to separate the groups from each other and we are talking about a *low group similarity*.

```
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(15,5))
sns.swarmplot(data=iris_df, ax = ax1,
              x='target', y='sepal width (cm)');ax1.set_title('Low Group Similarity
 ↪');
ax2.imshow(plt.imread('figures/FlowerAnatomy.png')); ax2.axis('off');
```

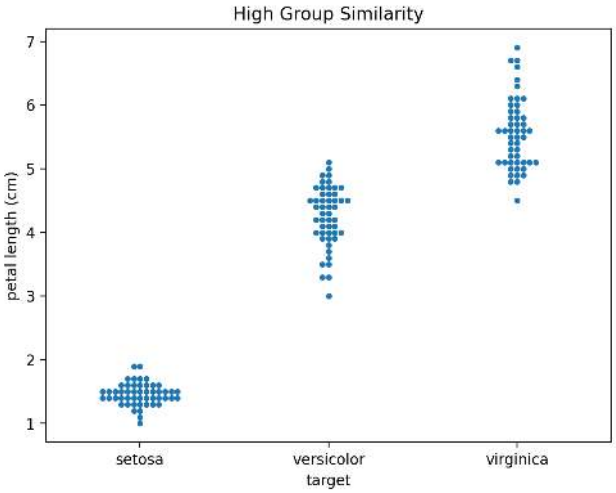


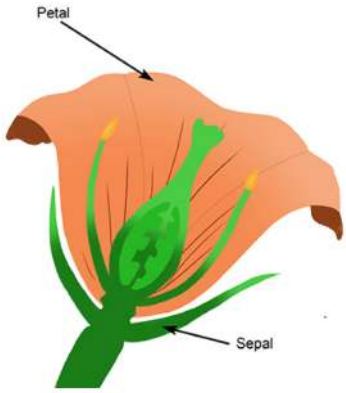


Petal length

Petals are the colourful and beautiful leaves of the flower. In this swarm plot of the petal length we see that the petals are more clustered and the averages are well separated from each other. This is a case we know is easy to separate the groups and we are talkning about data with a *high group similarity*.

```
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(15,5))
g = sns.swarmplot(data=iris_df, ax=ax1,
              x='target', y='petal length (cm)',size=4);g.set_title('High Group␣
 ↪Similarity');
ax2.imshow(plt.imread('figures/FlowerAnatomy.png')); ax2.axis('off');
```





0.7.4 Making quantitative statements

Intraclass Correlation Coefficient Definition

$$ICC = \frac{S_A^2}{S_A^2 + S_W^2}$$

where

- The variance among groups or classes
  Estimate with the standard deviations of the mean values for each group

$$S_A^2 = \mathrm{s}[\mathrm{E}[x_{group}]]^2$$

- The variance within groups or classes
  Estimate with the average of standard deviations for each group

$$S_W^2 = \mathrm{E}[\mathrm{s}[x_{group}]^2]$$

Interpretation of the ICC

$$ICC = \frac{S_A^2}{S_A^2 + S_W^2}$$

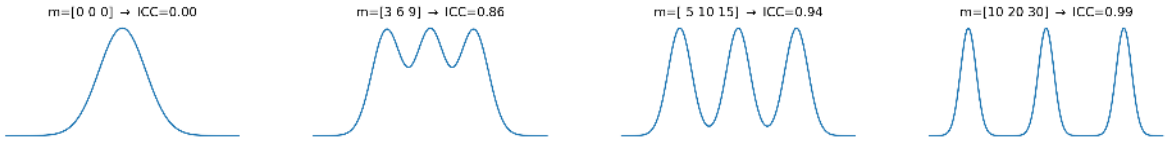
$$ICC = \begin{cases} 1 & \text{means 100 percent of the variance is between classes} \\ 0 & \text{means 0 percent of the variance is between classes} \end{cases}$$

```
def sum_gaussian(x,A,m,s) :
    g=np.zeros(x.shape[0])

    for aa,mm,ss in zip(A,m,s) :
        g=g+aa*np.exp(-(x-mm)**2/(2*ss**2))
    return g
```

(continues on next page)

(continued from previous page)

```
A=[1,1,1]
m=np.array([1,2,3])
s=np.array([1,1,1])

def icc(m,s) :
    return m.var()/(m.var()+s.mean()**2)

def plot_icc_examples(m,s,mscale) :
    N=len(mscale)
    fig,ax = plt.subplots(1,N,figsize=[5*N,2])

    for idx in range(N) :
        x=np.linspace(m[0]*mscale[idx]-5*s[0],m[-1]*mscale[idx]+5*s[-1],200)
        ax[idx].plot(x,sum_gaussian(x,A,m*mscale[idx],s))
        ax[idx].set(title=r"m={0} $\rightarrow$ ICC={1:0.2f}".format(m*mscale[idx],
 ↪icc(m*mscale[idx],s)),
                    xticks=[],yticks=[])
        ax[idx].axis('off')

plot_icc_examples(m,s,[0,3,5,10])
```



Intraclass Correlation Coefficient: Values

$$ICC = \frac{S_A^2}{S_A^2 + S_W^2}$$

When compute the ICC for sepal width and petal length, we see that the ICC confirms our first qualitative assessment about the group similiarity.

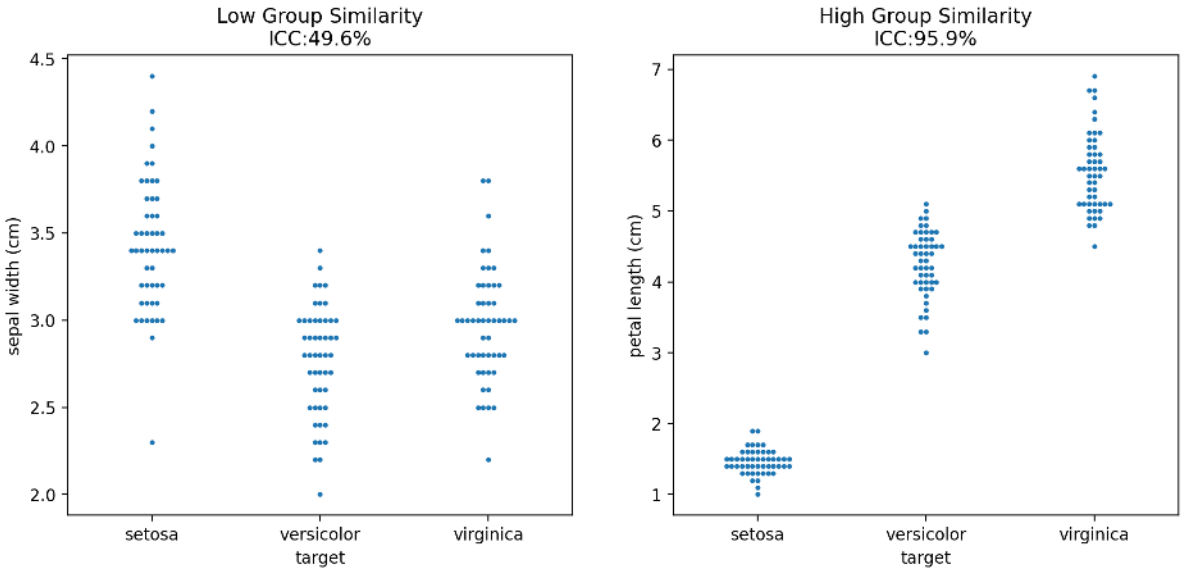
```
def icc_calc(value_name, group_name, data_df):
    data_agg = data_df.groupby(group_name).agg({value_name: ['mean', 'var']}).reset_
 ↪index()
    data_agg.columns = data_agg.columns.get_level_values(1)
    S_w = data_agg['var'].mean()
    S_a = data_agg['mean'].var()
    print('{0}: S_w={1:0.02f}, S_a={2:0.2f}'.format(value_name,S_w,S_a))
    return S_a/(S_a+S_w)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
sns.swarmplot(data=iris_df, ax=ax1,
              x='target', y='sepal width (cm)',size=3)
ax1.set_title('Low Group Similarity\nICC:{:2.1%}'.format(icc_calc('sepal width (cm)',
 ↪'target', iris_df)));

sns.swarmplot(data=iris_df,ax=ax2,
              x='target', y='petal length (cm)',size=3)
```

(continues on next page)

(continued from previous page)

```
ax2.set_title('High Group Similarity\nICC:{:2.1%}'.format(icc_calc('petal length (cm)
↪', 'target', iris_df)));
```

```
sepal width (cm): S_w=0.12, S_a=0.11
petal length (cm): S_w=0.19, S_a=4.37
```



0.8 Comparing Groups

Once the reproducibility has been measured, it is possible to compare groups.

The idea is to make a test to assess the likelihood that two groups are the same given the data

1. List assumptions
2. Establish a null hypothesis $\mathcal{H}_0$
   - Usually that both groups are the same
3. Calculate the probability of the observations given the truth of the null hypothesis
   - Requires knowledge of probability distribution of the data
   - Modeling can be exceptionally complicated

0.8.1 Outcomes for decision making

		Decision	
		Retain the null	**Reject the null**
Truth in the population	True	CORRECT $1 - \alpha$	TYPE I ERROR α
	False	TYPE II ERROR β	CORRECT $1 - \beta$ POWER

With error probabilites:

- α - probability of Type I errors / significance level
- β - probability of Type II errors

From Privitera 2017

0.8.2 Loaded Coin example

We have 1 coin from a magic shop. Our assumptions are:

- we flip and observe flips of coins accurately and independently
- the coin is invariant and always has the same expected value

Testing

- Our null hypothesis ($\mathcal{H}_0$): the coin is unbiased $E(\mathcal{X}) = 0.5$
- we can calculate the likelihood of a given observation given the number of flips (p-value)

How good is good enough?

0.8.3 Comparing Groups: Student's T Distribution

- Since we do not usually know our distribution very well
  or
- have enough samples to create a sufficient probability model

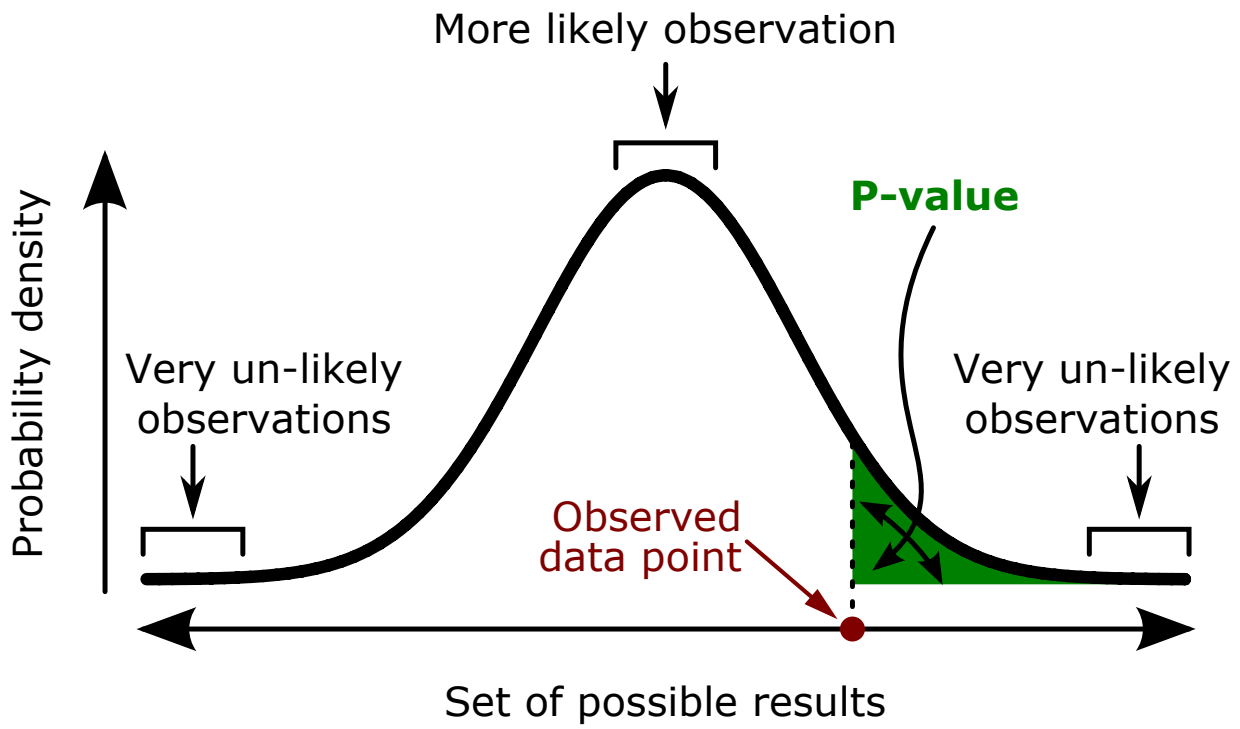


Fig. 1: Explaining p-value with the normal distribution.

Student T Distribution

We assume the distribution of our stochastic variable

- is normal (Gaussian)
- and the t-distribution provides an estimate for the mean of the underlying distribution based on few observations.

We estimate the likelihood of our observed values assuming they are coming from random observations of a normal process

Student T-Test

- Incorporates the Student T distribution
- and provides an easy method for assessing the likelihood that the two given sets of observations are coming from the same underlying process (null hypothesis, $\mathcal{H}_0$)

The test assume

- unbiased observations
- normal distribution

0.9 Multiple Testing Bias

Back to the magic coin, let's assume we are trying to publish a paper,

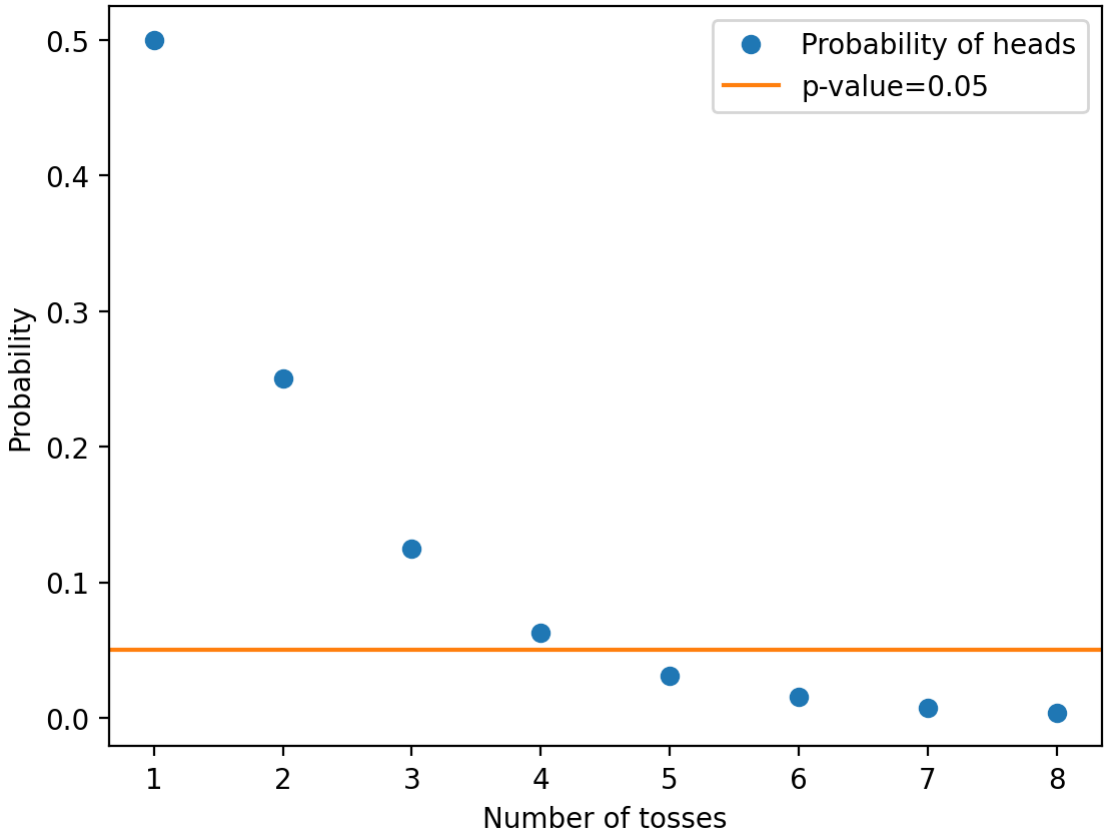
- Null-hypothesis ($\mathcal{H}_0$): the coin is fair
- we heard a p-value of < 0.05 (5%) was good enough.

That means if we get 5 heads ($P = 0.5^5 \approx 0.031$) we are good!

0.9.1 Probability with increasing number of tosses

$$P = \prod_i P(\mathcal{F}_i(\mathcal{X}))$$

```
# import pandas as pd
# from scipy.stats import ttest_ind
# from IPython.display import display
# all_heads_df = pd.DataFrame({'n_flips': [1, 4, 5]})
# all_heads_df['Probability of # Heads'] = all_heads_df['n_flips'].map(
#     lambda x: '{:2.1%}'.format(0.5**x))
# display(all_heads_df)

flips = np.linspace(1,8,8)
plt.plot(flips,np.power(0.5,flips),'o',label="Probability of heads")
pval=0.05
plt.axhline(pval,color=dcolors[1], label="p-value={0}".format(pval))
plt.gca().set(xlabel='Number of tosses',ylabel='Probability')
plt.legend();
```



0.9.2 Probability with many experiments

Let N friends make 5 tosses…

$$P = \overbrace{1 - (\underbrace{1 - 0.5^{N_{Tosses}}}_{\text{Not getting 5 heads}})^{N_{Friends}}}^{\text{Get 5 heads}}$$

```
friends_heads_df = pd.DataFrame({'n_friends': [1, 10, 20, 40, 80]})
friends_heads_df['Probability of 5 Heads'] = friends_heads_df['n_friends'].map(
    lambda n_friends: '{:2.1%}'.format((1-(1-0.5**5)**n_friends)))
display(friends_heads_df)
```

```
   n_friends Probability of 5 Heads
0          1                  3.1%
1         10                 27.2%
2         20                 47.0%
3         40                 71.9%
4         80                 92.1%
```

A flaw in the evaluation

Clearly this is not the case, otherwise we could keep flipping coins or ask all of our friends to flip until we got 5 heads and publish

The p-value is only meaningful when the experiment matches what we did.

- **We didn't say the chance of getting 5 heads ever was < 5%**
- **We said is if we have**
  - exactly 5 observations
  - and all of them are heads
  - the likelihood that a fair coin produced that result is <5%

There are many methods to correct.

Most just involve scaling p:

- The likelihood of a sequence of 5 heads in a row if you perform 10 flips is 5x higher.

0.9.3 Multiple Testing Bias: Experiments

This is very bad news for us. We have the ability to quantify all sorts of interesting metrics

- cell distance to other cells
- cell oblateness
- cell distribution oblateness

So, lets throw them all into a magical statistics algorithm and push the **publish** button

With our p value of less than 0.05 and a study with 10 samples in each group, how does increasing the number of variables affect our result

Let's look at multiple observations

We make five random variables with ten observations of a uniform distribution in the interval ± 1

$$var\_i \in \mathcal{U}(-1, 1)$$

and make two groups '1' and '2'

```
import pandas as pd
import numpy as np
pd.set_option('display.precision', 2)
np.random.seed(2017)

def random_data_maker(rows, cols):
    data_df = pd.DataFrame(
        np.random.uniform(-1, 1, size=(rows, cols)),
        columns=['Var_{:02d}'.format(c_col) for c_col in range(cols)])
    data_df['Group'] = [1]*(rows-rows//2)+[2]*(rows//2)
    return data_df

rand_df = random_data_maker(10, 5)

rand_df
```

```
   Var_00  Var_01  Var_02  Var_03  Var_04  Group
0   -0.96    0.53   -0.10   -0.76    0.86      1
1    0.30   -0.72   -0.54   -0.55   -0.48      1
2   -0.77    0.26   -0.23   -0.37    0.26      1
3   -0.41    0.89   -0.70   -0.85    0.41      1
4   -0.86   -0.39   -0.34   -0.38   -0.12      1
5    0.53   -0.05   -0.99    0.40    0.26      2
6   -0.94   -0.83    0.41   -0.09    0.41      2
7    0.86   -0.18   -0.92    0.24   -0.28      2
8    0.84    0.83   -0.46   -0.39   -0.97      2
9    0.08    0.34   -0.09    0.07    0.82      2
```

Compute p-values for the table

The Student-t test is computed using the python function

```
scipy.stats import ttest_ind

ttest_ind(var_i[Group==1],var_i[Group==2])
```

This is a two-sided test for the null hypothesis that two independent samples have identical average (expected) values. This test assumes that the populations have identical variances by default.

Here, we compute the p-values for the table we just created. The variables with p-values less than 0.05 are marked with yellow.

In the following example we compare the two groups of each random variable to determine if they are significantly different.

Essentially $ttest\_ind(var_i[Group == 1], var_i[Group == 2])$

We expect the two parts to be the same as all values are generated using the same random generator.

```
from scipy.stats import ttest_ind

def show_significant(in_df, cut_off=0.05):
    return in_df.sort_values('P-Value').style.apply(lambda x: ['background-color:␣
 ↪yellow' if v<cut_off else '' for v in x])

def all_ttest(in_df):
    return pd.DataFrame(
        {'P-Value': {c_col: ttest_ind(
            a=in_df[in_df['Group'] == 1][c_col],
            b=in_df[in_df['Group'] == 2][c_col]
        ).pvalue
            for c_col in
            in_df.columns if 'Group' not in c_col}})


show_significant(all_ttest(rand_df))
```

```
<pandas.io.formats.style.Styler at 0x163da11c0>
```

A larger table

Now, let's create a larger table with 150 rows and 20 independent variables.

```
np.random.seed(2019)
show_significant(all_ttest(random_data_maker(150, 20)))
```

```
<pandas.io.formats.style.Styler at 0x163d9d130>
```

Repeating the measurements

We saw with the coin tossing that the probability to detect the event we are looking for increased with the number of repeated independent measurements (friends tossing coins).

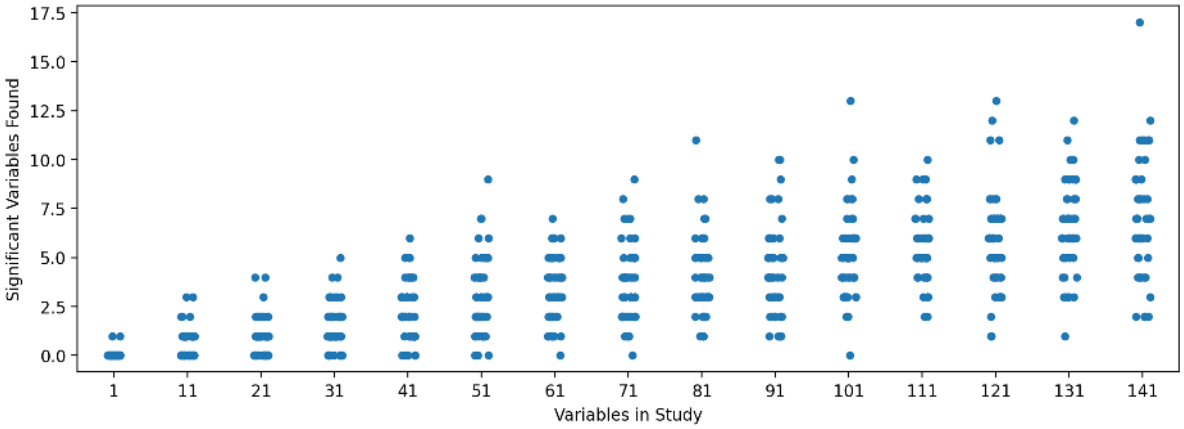
Let's see what happens when we do the the same with our table of experiments. First, we must generate the data.

- We will try using tables with 1 to 150 variables and 100 observations.
- Each measurement will be repeated 50 times.

```
import seaborn as sns
from tqdm import notebook # progressbar
out_list = []
for n_vars in notebook.tqdm(range(1, 150, 10)):
    for _ in range(50):
        p_values = all_ttest(random_data_maker(100, n_vars)).values
        out_list += [{'Variables in Study': n_vars,
                      'Significant Variables Found': np.sum(p_values < 0.05),
                     'raw_values': p_values}]
var_found_df = pd.DataFrame(out_list)
```

```
  0%|          | 0/15 [00:00<?, ?it/s]
```

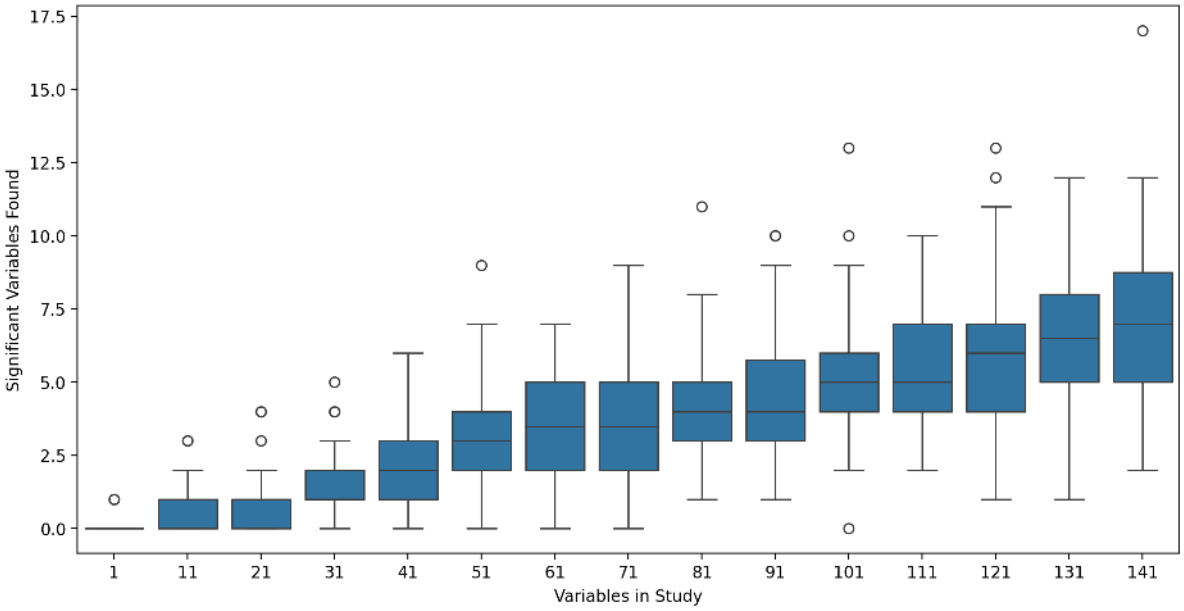
```
fig,ax = plt.subplots(1,1,figsize=(12,4))
sns.stripplot(data=var_found_df, x='Variables in Study', y='Significant Variables␣
↪Found');
```



Visualize the results differently

The strip plot we just used gets cluttered when we have too many observations. A different way to show the results is to use a boxplot.

```
plt.figure(figsize=(12,6))
sns.boxplot(data=var_found_df,
            x='Variables in Study', y='Significant Variables Found');
```



0.9.4 Multiple Testing Bias: Correction

We saw that increasing the number of tests also increases the probability of detection. This is missleading and needs to be corrected.

Using the simple correction factor (number of tests performed) as proposed by Bonferroni, we can make the significant findings constant again. $$p_{\text{cutoff}} = \frac{0.05}{\text{Number of Tests}}$$

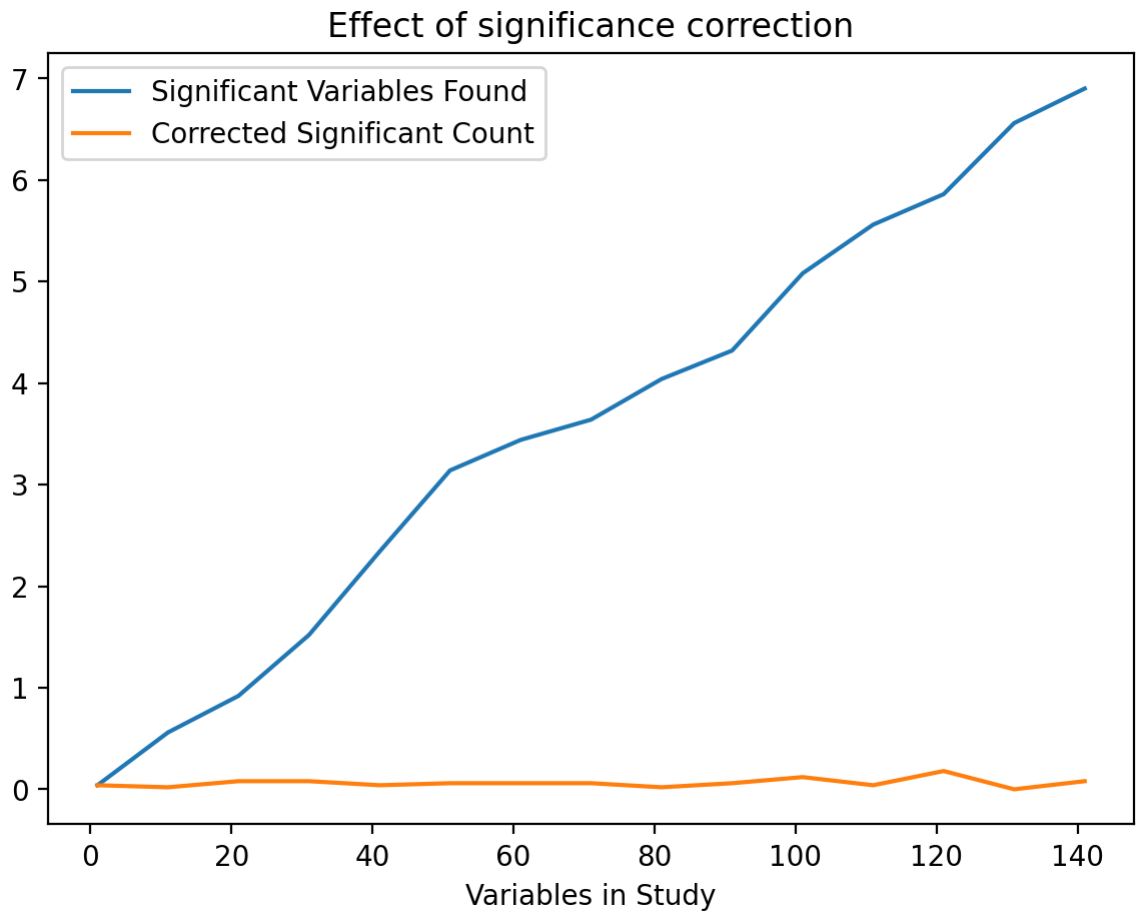
This comes from the familywise error $$\bar{\alpha} = 1 - \left(1 - \alpha_{\text{per comparison}}\right)^{m}$$

where m is the number of hypotheses tested. Then, with Boole's inequality we have

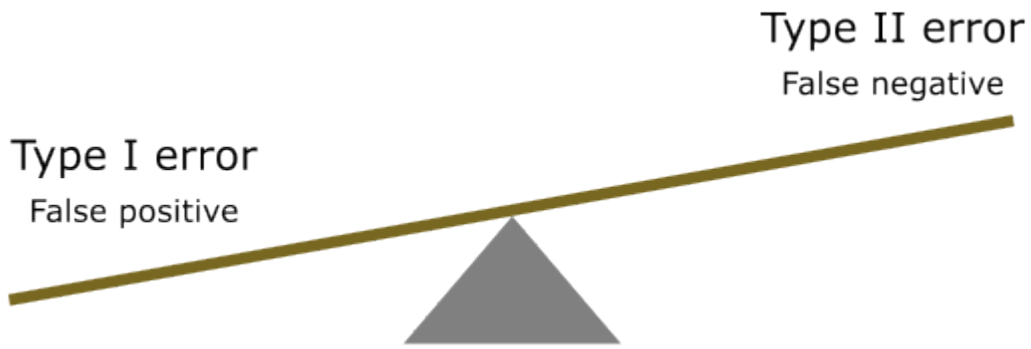
$$\bar{\alpha} \leq m \cdot \alpha_{\text{per comparison}}$$

Which leads to the Boniferroni correction.

```
fig,ax=plt.subplots(1,figsize=(7,5))
var_found_df['Corrected Significant Count'] = var_found_df['raw_values'].map(lambda p_
 ↪values:
                                                                              np.sum(p_
 ↪values<0.05/len(p_values)))
var_found_df.groupby('Variables in Study').agg({'Significant Variables Found':'mean',
 ↪'Corrected Significant Count':'mean'}).plot(ax=ax)
plt.title('Effect of significance correction');
```



No free lunch



There is no free lunch. If you adjust your statistical threshold to obtain less Type I errors you'll have to take the cost of increasting the number of Type II errors. This is the same as we have seen when thresholding images.

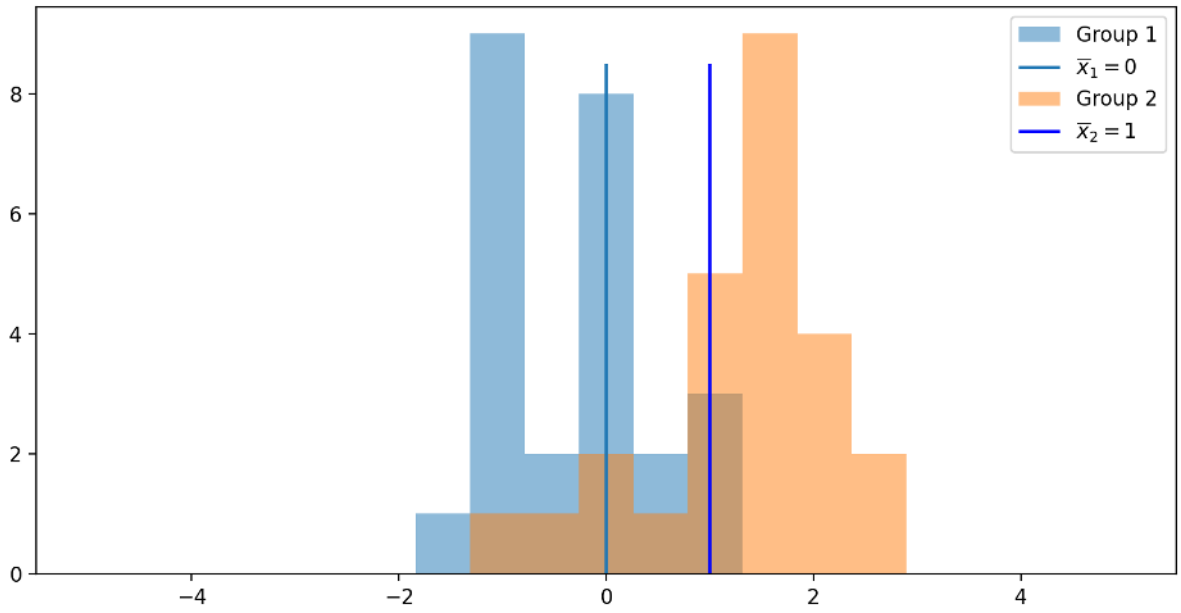
Is this correction factor sufficient?

So no harm done there we just add this correction factor right?

Well, what if we have exactly one variable with shift of 1.0 standard deviations from the other.

In a dataset where we check n variables?

```
table_df = random_data_maker(50, 10)
really_different_var = np.concatenate([
    np.random.normal(loc=0, scale=1.0, size=(table_df.shape[0]//2)),
    np.random.normal(loc=1, scale=1.0, size=(table_df.shape[0]//2))
])
table_df['Really Different Var'] = really_different_var
fig, ax1 = plt.subplots(1, 1, figsize=(10, 5))
ax1.hist(table_df.query('Group==1')['Really Different Var'], np.linspace(-5, 5, 20),␣
 ↪label='Group 1', alpha=0.5);
ax1.vlines(0,ymin=0,ymax=8.5,label='$\overline{x}_1=0$')
ax1.hist(table_df.query('Group==2')['Really Different Var'], np.linspace(-5, 5, 20),␣
 ↪label='Group 2', alpha=0.5);
ax1.vlines(1,ymin=0,ymax=8.5,color='blue',label='$\overline{x}_2=1$')
ax1.legend();
```



Run many tests

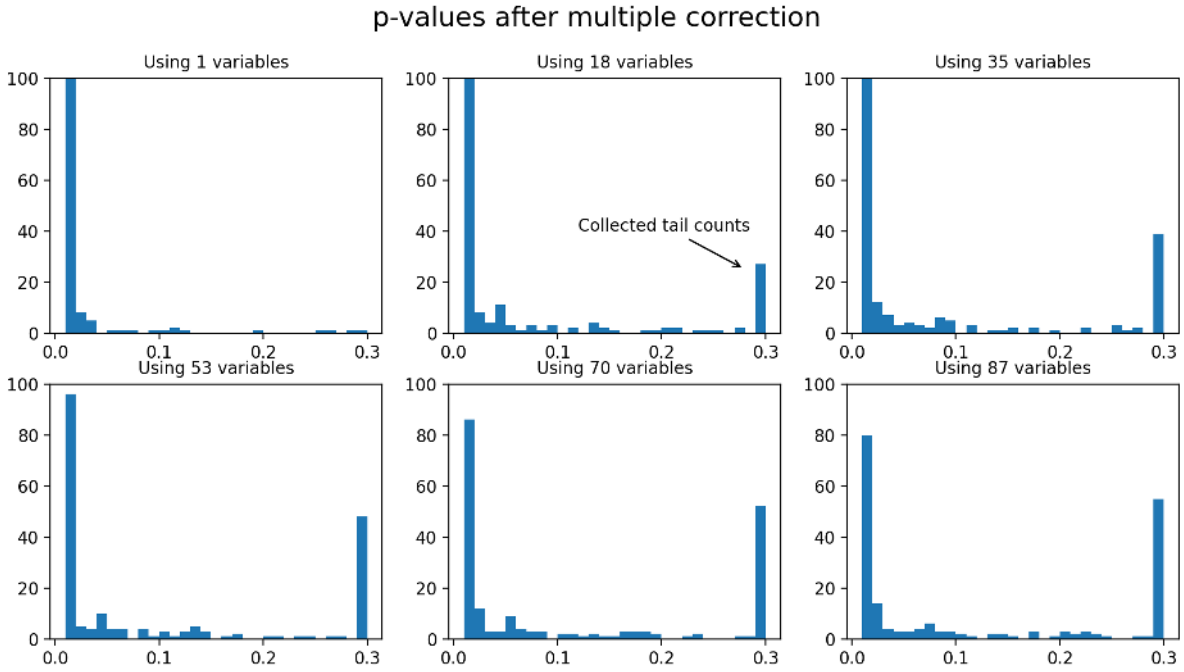
We run 200 tests with two variables with $x \in \mathcal{N}(0,1)$ and $y \in \mathcal{N}(1,1)$ and compute the p-values for each test.

Using $\mathcal{H}_0$: $x = y$

```
out_p_value = []
for _ in range(200):
    out_p_value += [ttest_ind(np.random.normal(loc=0, scale=1.0, size=(table_df.
 ↪shape[0]//2)),
          np.random.normal(loc=1, scale=1.0, size=(table_df.shape[0]//2))).pvalue]
```

When we look at the histograms of p-values scale by the number of variables in the test we see that there is a greater probability to accept the null-hypothesis.

```
fig, m_axs = plt.subplots(2, 3, figsize=(12, 6))

for c_ax, var_count in zip(m_axs.flatten(), np.linspace(1, 140, 9).astype(int)):
    c_ax.hist(np.clip(np.array(out_p_value)*var_count, 0.01, 0.3), np.linspace(0.01,␣
 ↪0.3, 30))
    c_ax.set_ylim(0, 100)
    c_ax.set_title('Using {} variables'.format(var_count),fontsize=10)

plt.suptitle('p-values after multiple correction',fontsize=16)
m_axs[0,1].annotate("Collected tail counts", xy=(0.28, 25), xytext=(0.12, 40),␣
 ↪arrowprops=dict(arrowstyle="->",color="black"),fontsize=10);
```



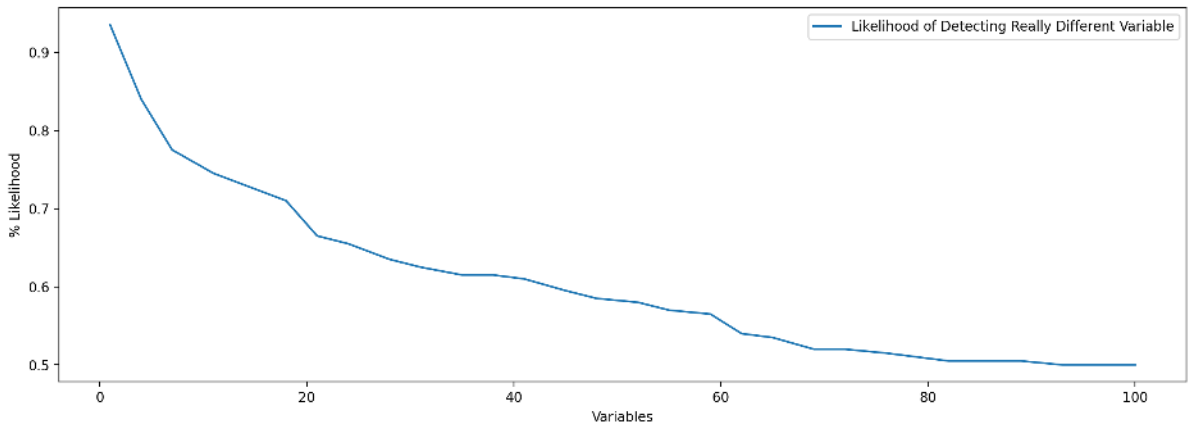
Note: Don't mind the peak on the right hand side of the histogram. It is there because we limited the interval of the histogram and all values beyond this limit are collected in this histgram bin.

The likelihood to find a different variable

The scaling by the number of variables means that we are less likely to reject the null hypothesis. So, what is the likelihood?

We count the number of the p-values less than 0.05 to compute the likelihood of detecting a really different variable.

```
var_find_df = pd.DataFrame({'Variables': np.linspace(1, 100, 30).astype(int)})
var_find_df['Likelihood of Detecting Really Different Variable'] = var_find_df[
 ↪'Variables'].map(
    lambda var_count: np.mean(np.array(out_p_value)*var_count<0.05)
)
fig, ax1 = plt.subplots(1, 1, figsize=(15, 5))
var_find_df.plot('Variables', 'Likelihood of Detecting Really Different Variable',␣
 ↪ax=ax1)
ax1.set_ylabel('% Likelihood');
```



Here, we see that the likelihood is very low for many varibles. A reason in that we are working on a limited number of samples and split these on an incresing number of variables.

0.9.5 A statistical study from images

Setting the stage

We have a new treatment and want to compare cell cultures

Volkov 2015

Are they different?

Recap from previous lectures

We saw that a cell culture can be reduced to a point cloud

- Segmentation
- Labelling
- Region properties
- Voronoi tesselation
- Area distribution

Point clouds to compare

In our test we generate two types of point clouds

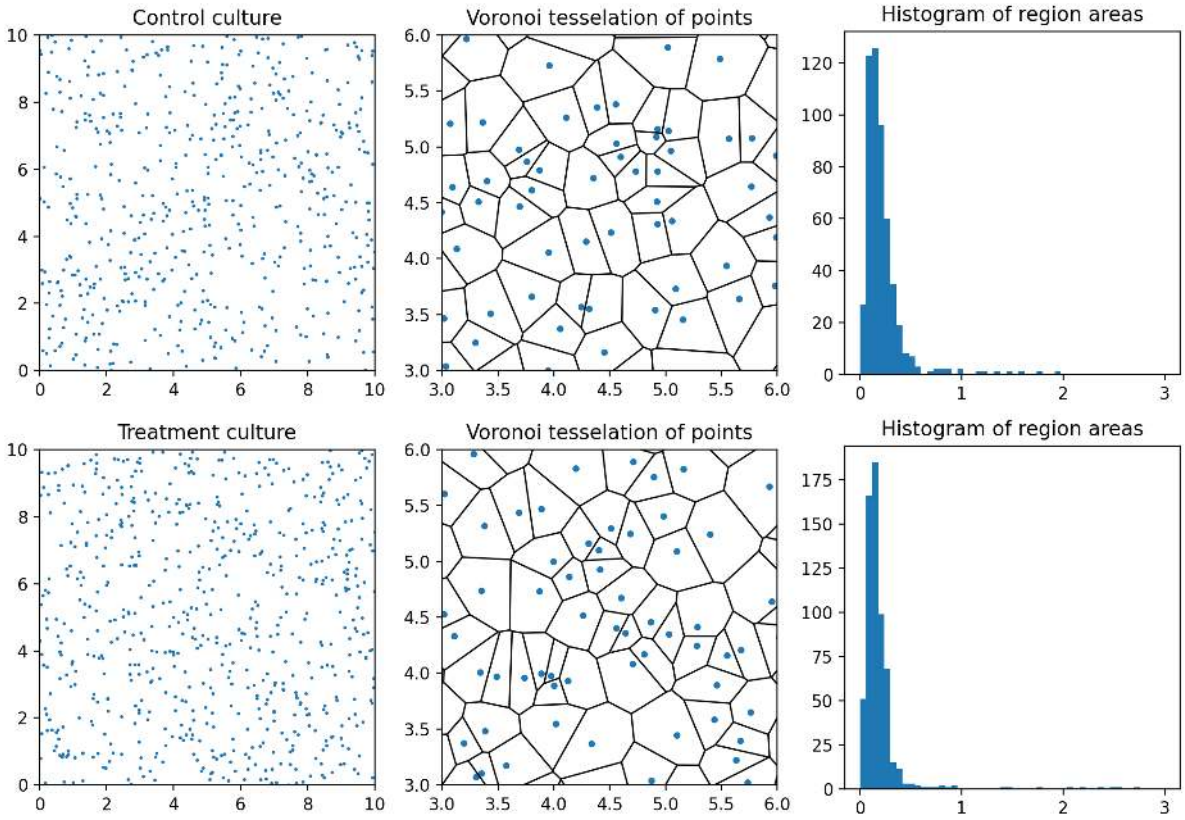
```
intensity_control = 5.5
intensity_treatment = 6.0
```

```
importlib.reload(pc)

fig,ax = plt.subplots(2,3,figsize=(10,7))
dpoints=pc.point_field(intensity=intensity_control,ax=ax[0,0],title='Control culture',
 ↪seed=100,plot=True)
```

(continues on next page)

(continued from previous page)

```
lpoints=pc.point_field(intensity=intensity_treatment,ax=ax[1,0], title='Treatment␣
↪culture',seed=200,plot=True)

from scipy.spatial import Voronoi, voronoi_plot_2d
dvor = Voronoi(dpoints)
dA=pc.compute_region_areas(dvor)
dA = dA[~np.isnan(dA)]
# dA=dA[dA<1.5]

lims = [3,6]
voronoi_plot_2d(dvor,show_points=True, show_vertices=False,ax=ax[0,1])
ax[0,1].set_aspect('equal')
ax[0,1].set_xlim(lims)
ax[0,1].set_ylim(lims)
ax[0,1].set_title('Voronoi tesselation of points')

ax[0,2].hist(dA,range=[0,3],bins=50);
ax[0,2].set_title('Histogram of region areas')

lvor = Voronoi(lpoints)
lA=pc.compute_region_areas(lvor)
lA = lA[~np.isnan(lA)]
# lA=lA[lA<1.5]
voronoi_plot_2d(lvor,show_points=True, show_vertices=False,ax=ax[1,1])
ax[1,1].set_aspect('equal')
ax[1,1].set_xlim(lims)
ax[1,1].set_ylim(lims)
ax[1,1].set_title('Voronoi tesselation of points')

ax[1,2].hist(lA,range=[0,3],bins=50);
ax[1,2].set_title('Histogram of region areas')
plt.tight_layout()
```



0.9.6 Comparing samples

Use the ICC to compare two samples

```
importlib.reload(pc)
m=np.array([lA.mean(),dA.mean()])
s=np.array([lA.std(),dA.std()])

pc.icc(m,s)
```

```
0.0032644209639845424
```

Not very separated

With a ICC of this magnitude we can conclude that the two cultures are not differing very much. Let's see what our hypothesis testing tells us. This can naturally also be confirmed by looking at the histograms; they are more or less overlapping.

Can the t-test help us?

Let's try the t-test on the region areas

$\mathcal{H}_0$: There is no difference between the cultures in the region area

```
from scipy.stats import ttest_ind

stat, p_value = ttest_ind(dA, lA, equal_var=False)

print(f"p-value: {p_value:.4f}")
```

```
p-value: 0.1635
```

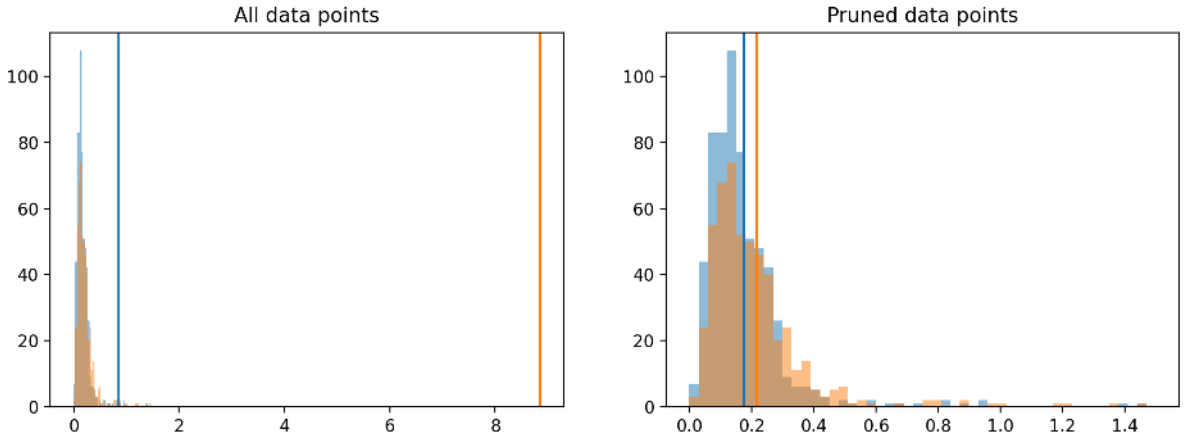
We confirm the null-hypothesis that there is no difference between the two samples.

Outliers biasing the hypothesis test

We observe that there are some extreme outliers in the data. Let's prune…

```
lAc=lA[lA<2]
dAc=dA[dA<2]
```

```
fig,ax=plt.subplots(1,2,figsize=(12,4))
ax[0].hist(lA,range=[0,1.5],bins=50,alpha=0.5);
ax[0].axvline(np.mean(lA),color=dcolors[0])
ax[0].hist(dA,range=[0,1.5],bins=50,alpha=0.5);
ax[0].axvline(np.mean(dA),color=dcolors[1]);
ax[0].set_title('All data points')

ax[1].hist(lAc,range=[0,1.5],bins=50,alpha=0.5);
ax[1].axvline(np.mean(lAc),color=dcolors[0])
ax[1].hist(dAc,range=[0,1.5],bins=50,alpha=0.5);
ax[1].axvline(np.mean(dAc),color=dcolors[1]);
ax[1].set_title('Pruned data points');
```



```
stat, p_value = ttest_ind(dAc, lAc, equal_var=False)

print(f"p-value: {p_value:.4f}")
```

```
p-value: 0.0001
```

What's Really Happening?

Outliers increase variance, and the t-test depends on the ratio:

$$t = \frac{\text{difference in means}}{\text{pooled standard error}}$$

→ Higher variance → larger denominator → smaller t → higher p-value

So outliers can dilute the test's power and hide real effects.

Key Takeaway

Outliers can obscure true group differences. But you shouldn't blindly remove them — it depends on context. In this particular case we know that voronoi regions at the boundary are infinite of very large. What we should have done here is to crop the regions at the image boundary. The cropping would of course also introduce an error. So, it is important to decide wether you can accept reducing the number of points or if the cropping is acceptable.

The Kolmogorov-Smirnov test

In statistics, the Kolmogorov–Smirnov test (also K–S test or KS test) is a nonparametric test of the equality of continuous one-dimensional probability distributions.

It can be used to test whether

- a sample came from a given reference probability distribution (one-sample K–S test), or
- two samples came from the same distribution (two-sample K–S test)

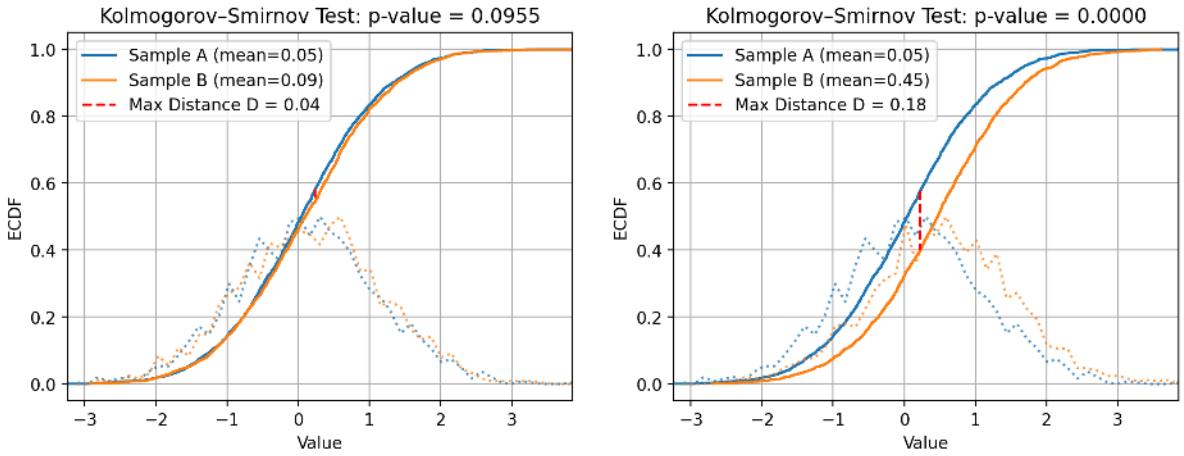
Intuitively, it provides a method to answer the question "How likely is it that we would see a collection of samples like this if they were drawn from that probability distribution?" or, in the second case, "How likely is it that we would see two sets of samples like this if they were drawn from the same (but unknown) probability distribution?"

```
importlib.reload(pc)
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import ks_2samp
dcolors = plt.rcParams['axes.prop_cycle'].by_key()['color']

# Generate two synthetic samples
np.random.seed(42)
N=2000
sample_A = np.random.normal(0, 1, N)
sample_B = np.random.normal(0.1, 1, N)  # Shifted mean
sample_C = np.random.normal(0.5, 1, N)  # Shifted mean

fig,ax = plt.subplots(1,2,figsize=(12,4))
```

(continues on next page)

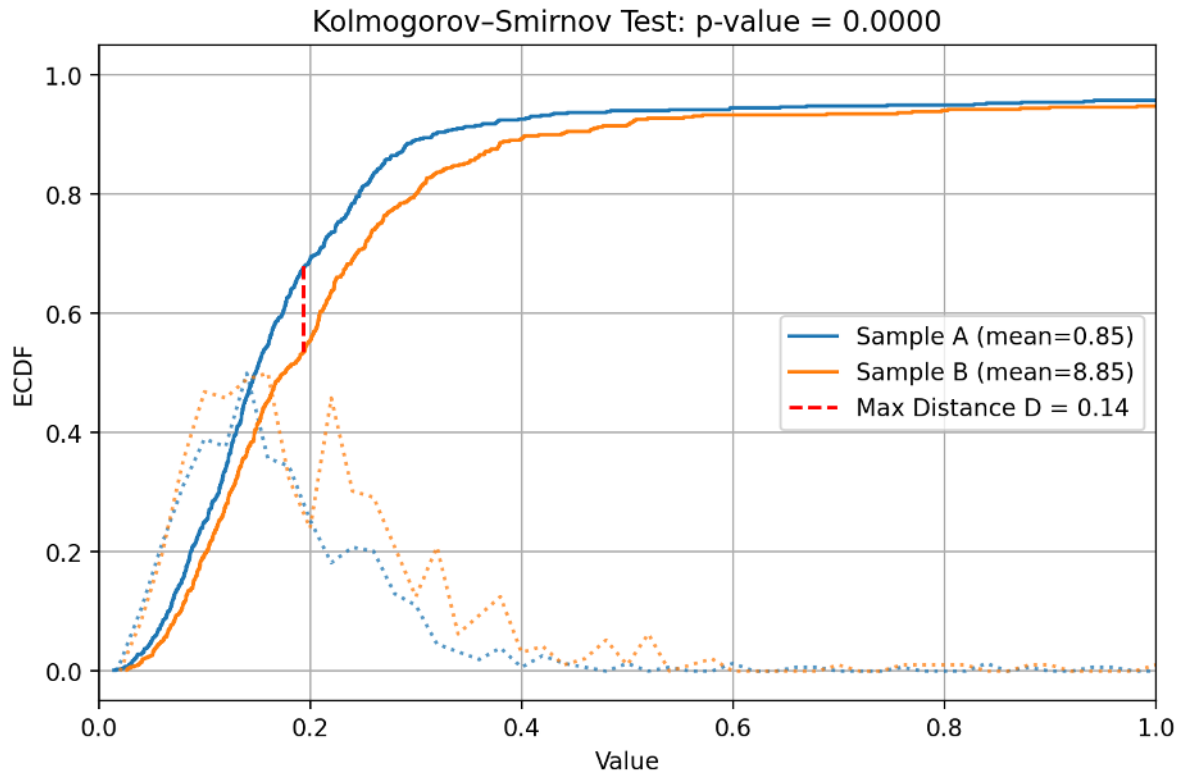
(continued from previous page)

```
pc.plot_ks_test(sample_A,sample_B,ax[0])
pc.plot_ks_test(sample_A,sample_C,ax[1])
```



Use the KS test to compare two cultures

```
pc.plot_ks_test(lA,dA,xlim=[0,1])
```



0.9.7 Comparing treatments

Comparing one sample from each treatment is not sufficient

- There are natural variations in each sample
- The test outcome relies on luck like tossing a coin

A scientific requires many samples from each treatment.

Revise our hypothesis: $\mathcal{H}_0$: Do treatment A and treatment B produce systematically different spatial patterns across replicates

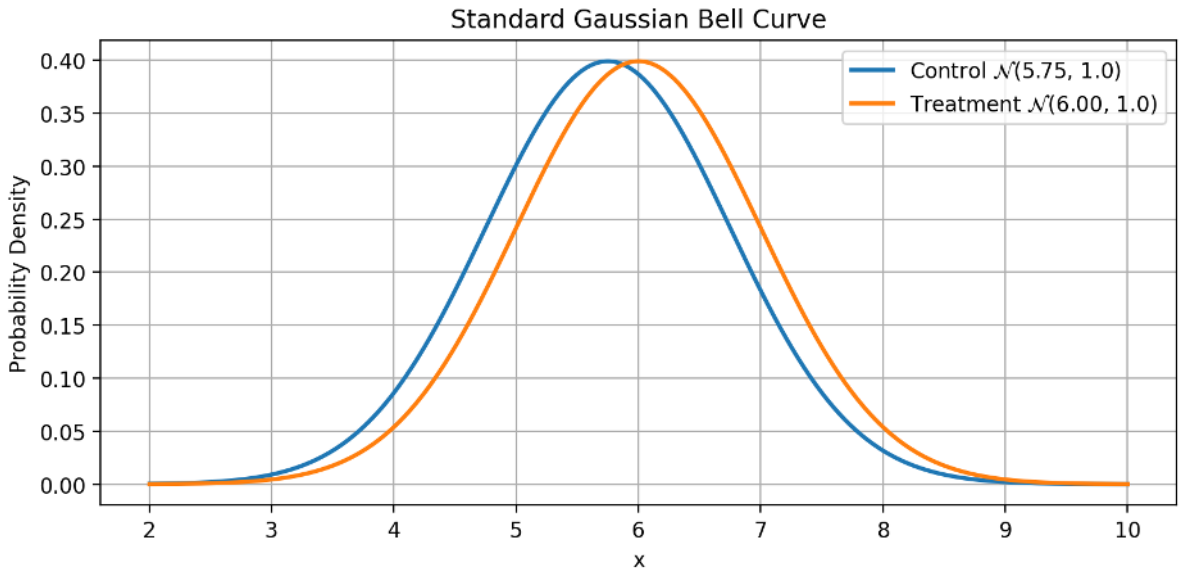
But this time increase the population

Produce data for the example

We generate two populations

- Gaussian distribution of the sample point intensity
- 10 samples in each population

```
from scipy.stats import norm

# Define range and parameters
x = np.linspace(2, 10, 500)
mu = [5.75, 6]       # Mean
sigma = [1,1]    # Standard deviation

# Compute Gaussian PDF
y0 = norm.pdf(x, mu[0], sigma[0])
y1 = norm.pdf(x, mu[1], sigma[1])

# Plot
plt.figure(figsize=(8, 4))
plt.plot(x, y0, label=r'Control $\mathcal{N}$('+'{0:0.2f}, {1:0.1f})'.format(mu[0],
 ↪sigma[0]), lw=2)
plt.plot(x, y1, label=r'Treatment $\mathcal{N}$'+'({0:0.2f}, {1:0.1f})'.format(mu[1],
 ↪sigma[1]), lw=2)
plt.title('Standard Gaussian Bell Curve')
plt.xlabel('x')
plt.ylabel('Probability Density')
plt.grid(True)
plt.legend()
plt.tight_layout()
```



Can we separate them using the area of the Voronoi regions?

Perform the experiment

We don't have experiment data here. So, we generate the points with random generators

1. Generate the point clouds
2. Compute Voronoi region areas

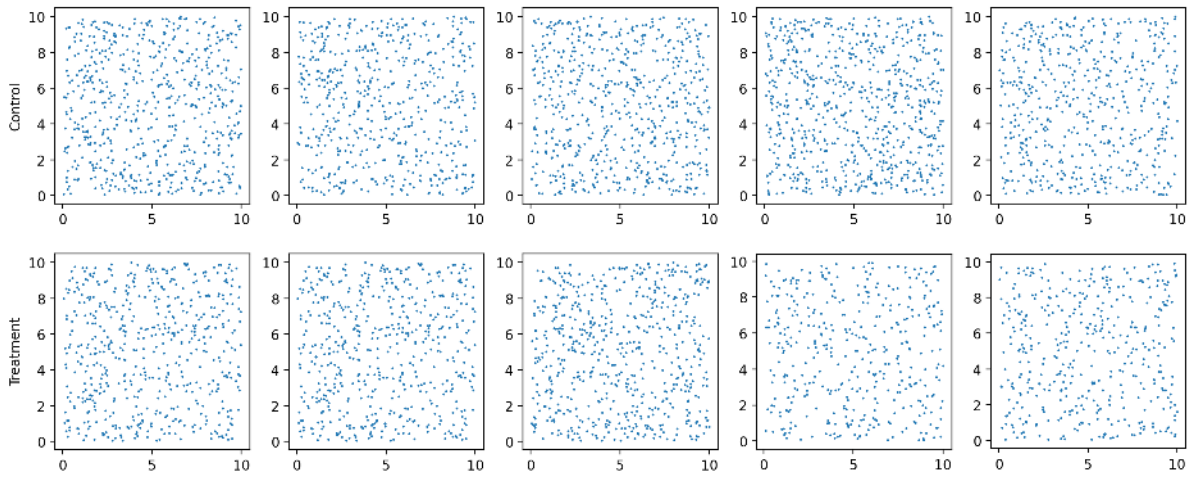
```
importlib.reload(pc)
nSamples = 10
np.random.seed(seed=42)
samples_control   = np.random.normal(5.75, 1, size=nSamples)
samples_treatment = np.random.normal(6.0, 1, size=nSamples)
width, height = 10,10
threshold=1.5

A_control,p_control     = pc.compute_area_in_samples(samples_control, width=width,
 ↪height=height,threshold=threshold)
A_treatment,p_treatment = pc.compute_area_in_samples(samples_treatment, width=width,
 ↪height=height,threshold=threshold)
```

```
nPanels = 5
fig,axes=plt.subplots(2,nPanels,figsize=(15,6))

for ax,p in zip(axes[0],p_control[:nPanels]) :
    ax.scatter(p[:,1],p[:,0],s=0.5)
    ax.set_aspect('equal')

axes[0,0].set_ylabel('Control')

for ax,p in zip(axes[1],p_treatment[:nPanels]) :
    ax.scatter(p[:,1],p[:,0],s=0.5)
    ax.set_aspect('equal')
```

(continues on next page)

(continued from previous page)

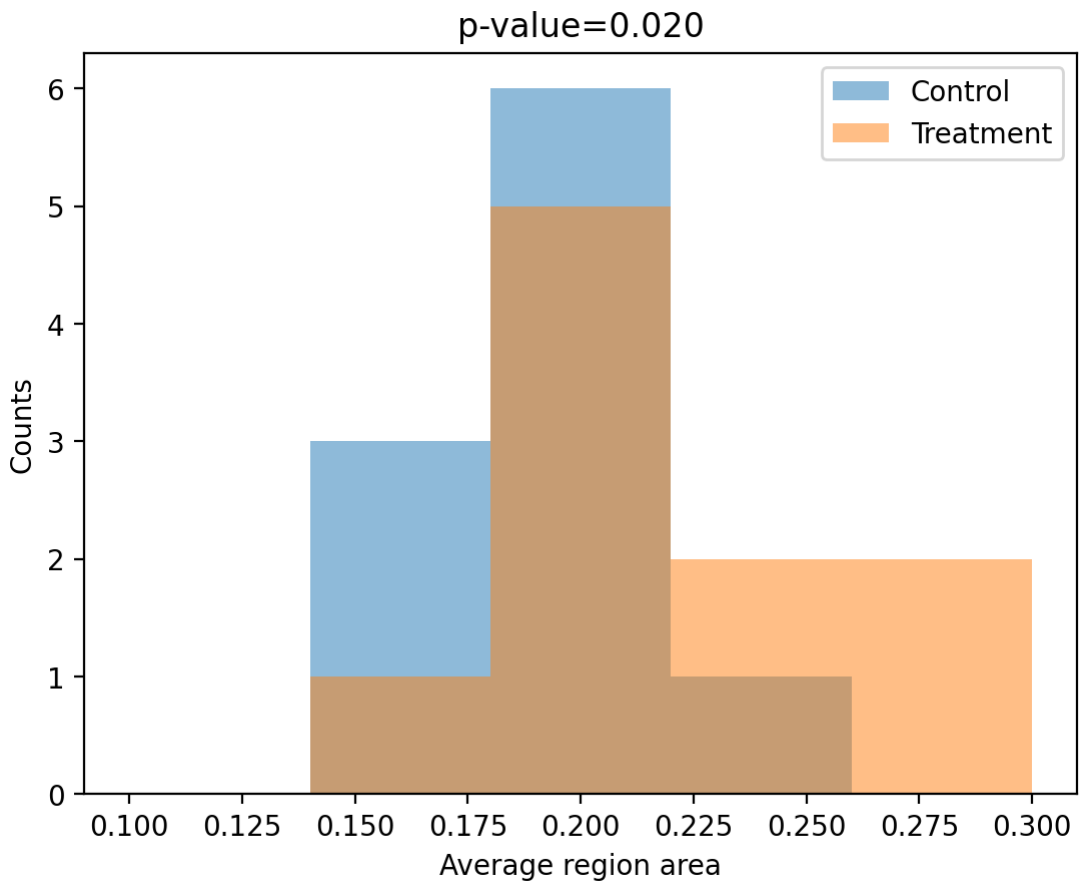
```
axes[1,0].set_ylabel('Treatment');
```



Compare the populations

```
stat_func = np.mean
avg_control   = np.array([stat_func(s) for s in A_control])
avg_treatment = np.array([stat_func(s) for s in A_treatment])

stat, p_value = ttest_ind(avg_control, avg_treatment, equal_var=False)
```

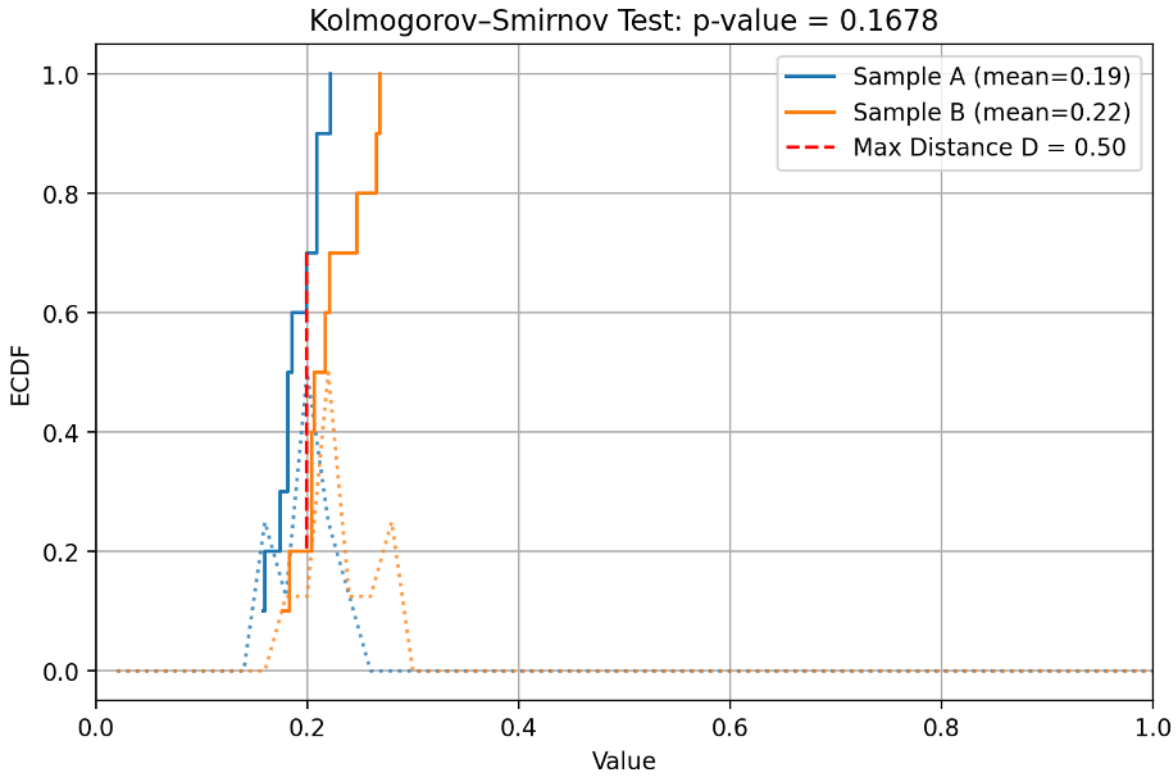
```
nBins = 5
plt.hist(avg_control,bins=nBins,alpha=0.5,range=[0.1,0.3], label='Control')
plt.hist(avg_treatment,bins=nBins,alpha=0.5,range=[0.1,0.3], label='Treatment');
plt.xlabel('Average region area')
plt.ylabel('Counts');
plt.legend()
plt.title('p-value={0:0.3f}'.format(p_value));
```



Conclusion: given the data our t-test tells us to reject the null-hypothesis since the p-value is <0.05.

Let's try the KS test

```
pc.plot_ks_test(avg_control,avg_treatment,xlim=[0,1])
```



This test confirms the null-hypothesis… why?

When is the KS-test reliable?

We saw in this example that the KS-test delivered a very different p-value compared with the t-test. The following table shows when it makes sense to use the KS-test. In our case we only had ten samples to compare.

Sample Size	Suitability for KS Test
< 20	Use with caution; very low power
20–50	Can work for large or obvious differences
≥ 50	Reasonable reliability for moderate differences
≥ 100	Good power and reliable p-values
≥ 1000	Very sensitive — may detect tiny, even irrelevant differences

How many samples are needed

The confidence in the test increases with the sample size…

… but

- Sample preparation
- Experiment
- Processing are expensive

For the current experiment, we get:

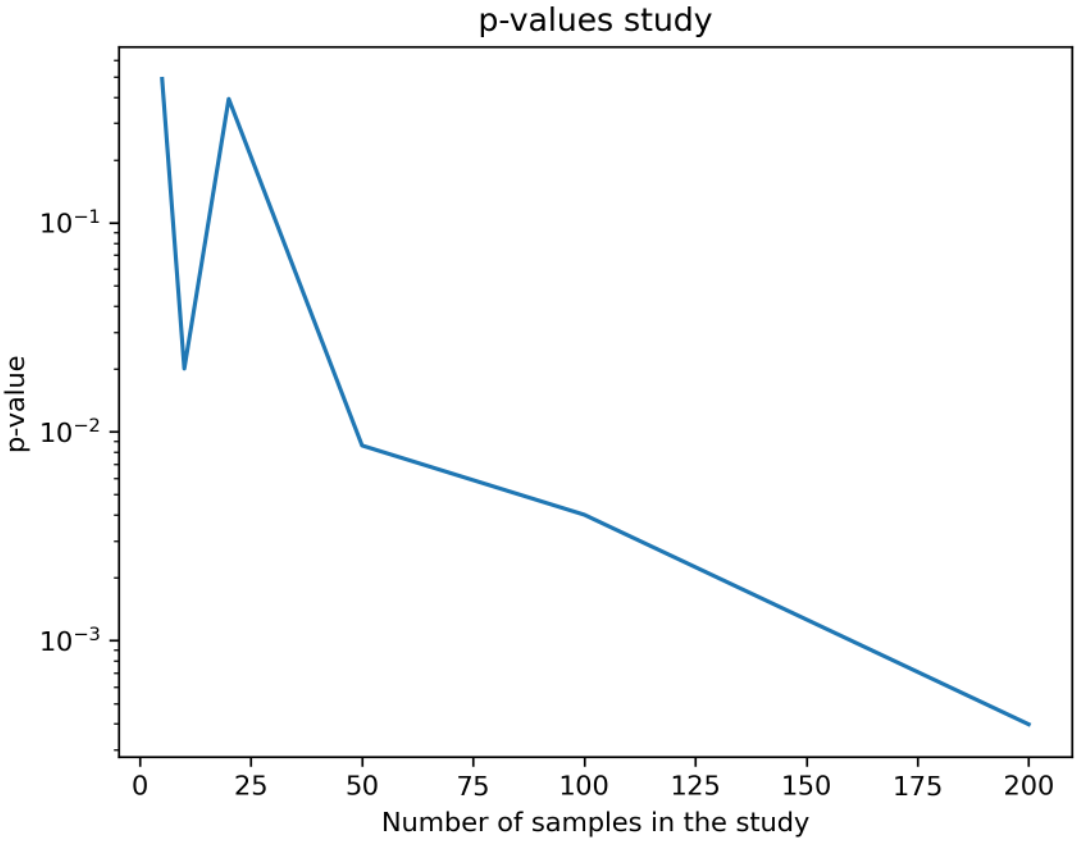


Fig. 1: p-values for different sample size.

The plot show that the p-value generally decreases with inceasing number of samples in the test. The peak at 20 samples is an unfortunate coincidence due to insufficient test runs with the random generator.

The number of required samples also depends of the distribution of the data. You need less data for well separated samples $|\mu_0 - \mu_1| >> \sigma$ than when there is great overlap.

0.9.8 Conclusions from the statistical analysis

- We often have too little information for perfect analysis
- Student T-tests help
- Reject Null hypothesis on a specified confidence level
- Repeated measurements should be used
- Correction is needed to avoid biasing in the conclusion

0.10 Sensitivity to analysis parameters

We have a workflow to analyze *shape* and *thickness* of items in an image:

```
from graphviz import Digraph

dot = Digraph()

dot.node('Raw images',color='limegreen'),        dot.node('Gaussian filter', color=
↪'lightblue')
dot.node('sigma=0.5', color='gray',shape='box'), dot.node('3x3 Neighbors', color='gray
↪',shape='box')
dot.node('Threshold', color='lightblue'),        dot.node('100', color='gray',shape=
↪'box')
dot.node('Thickness analysis',color='hotpink'),  dot.node('Shape analysis',color=
↪'hotpink')
dot.node('Input',color='limegreen'),        dot.node('Functions', color='lightblue')
dot.node('Parameters', color='gray',shape='box'),dot.node('Output',color='hotpink')

dot.edge('Raw images', 'Gaussian filter'),    dot.edge('sigma=0.5', 'Gaussian filter')
dot.edge('3x3 Neighbors', 'Gaussian filter'), dot.edge('Gaussian filter','Threshold')
dot.edge('Threshold', 'Thickness analysis'),  dot.edge('Threshold', 'Shape analysis')
dot.edge('100','Threshold')
dot
```

```
<graphviz.graphs.Digraph at 0x16c452fd0>
```

Three parameters can be controlled:

- Gaussian filter: σ and *neighborhood size*
- Threshold level

0.10.1 Parameter Sweep

The way we do this is usually a parameter sweep which means

- taking one (or more) parameters
- and varying them between the reasonable bounds (judged qualitatively).

The outcome of a parameter sweep can look like in the figure below.

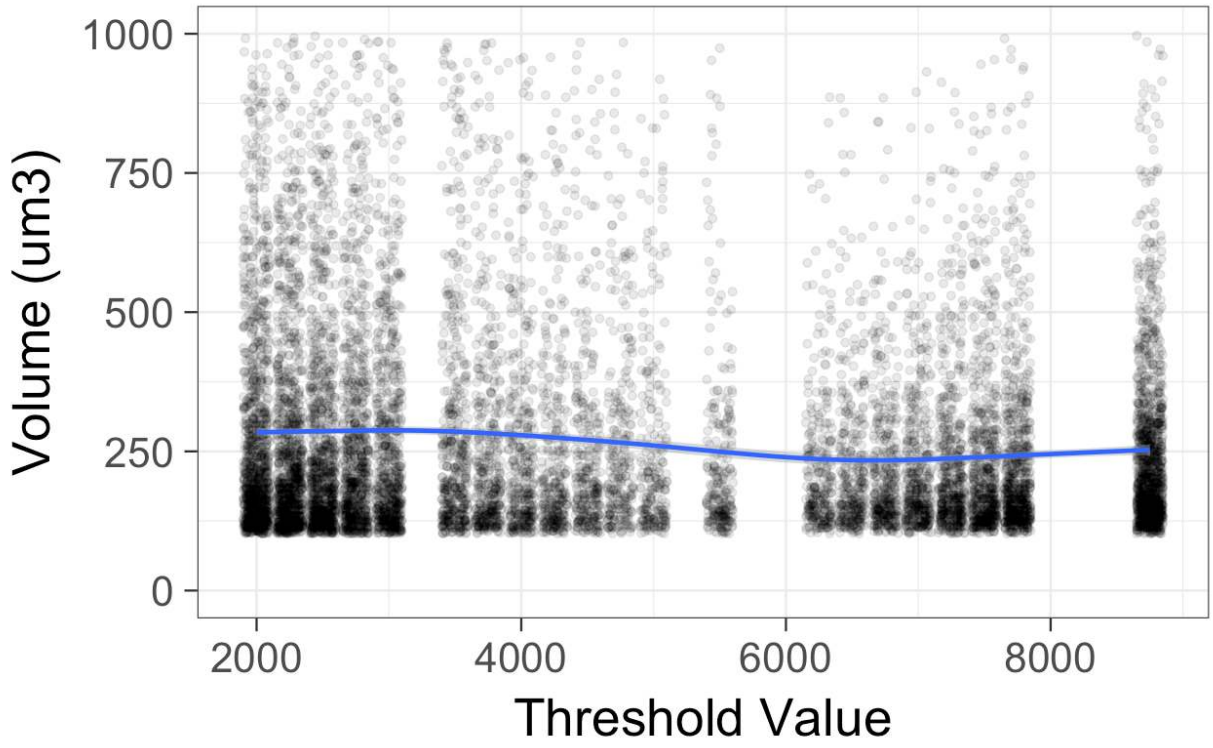


Fig. 1: Volume measurements for different thresholds

We can see that the volume is generally decreasing when the threshold increases. Still, it seems that the volume is not that sensitive to the choice of threshold. Variations in the order of about 50 voxels. That would correspond to a radius change from 6.3 to 6.7 for the equivalent spheres. On the other hand, this minor change could be the difference between separated or touching objects.

0.10.2 Is it always the same?

The jittered scatter plot in fig 1 makes it hard to see the distribution of the measurements. A violin plot as in the figure below is a histogram view of the data that allows stacking for different observations.

Now, what happens if we look at a different item metric like the orientation of the items.

Here, we see a similar trend as we saw with the volume.

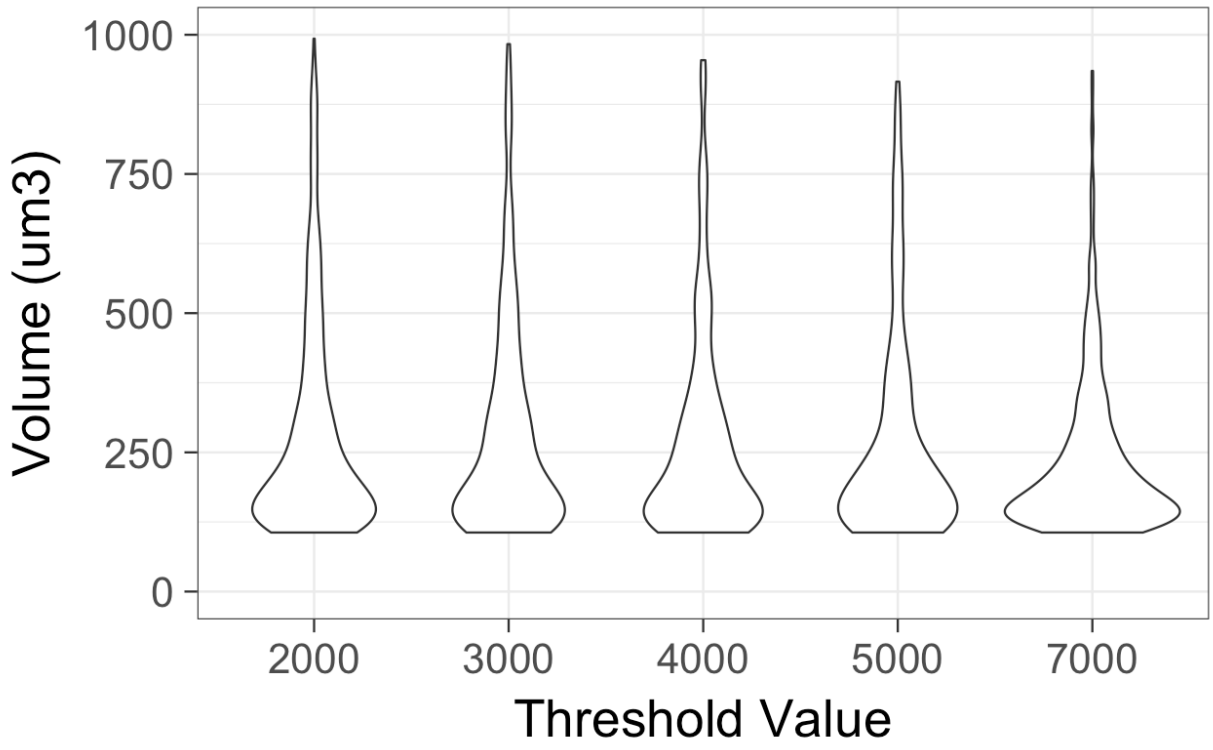


Fig. 2: A violin plot of the volume data.

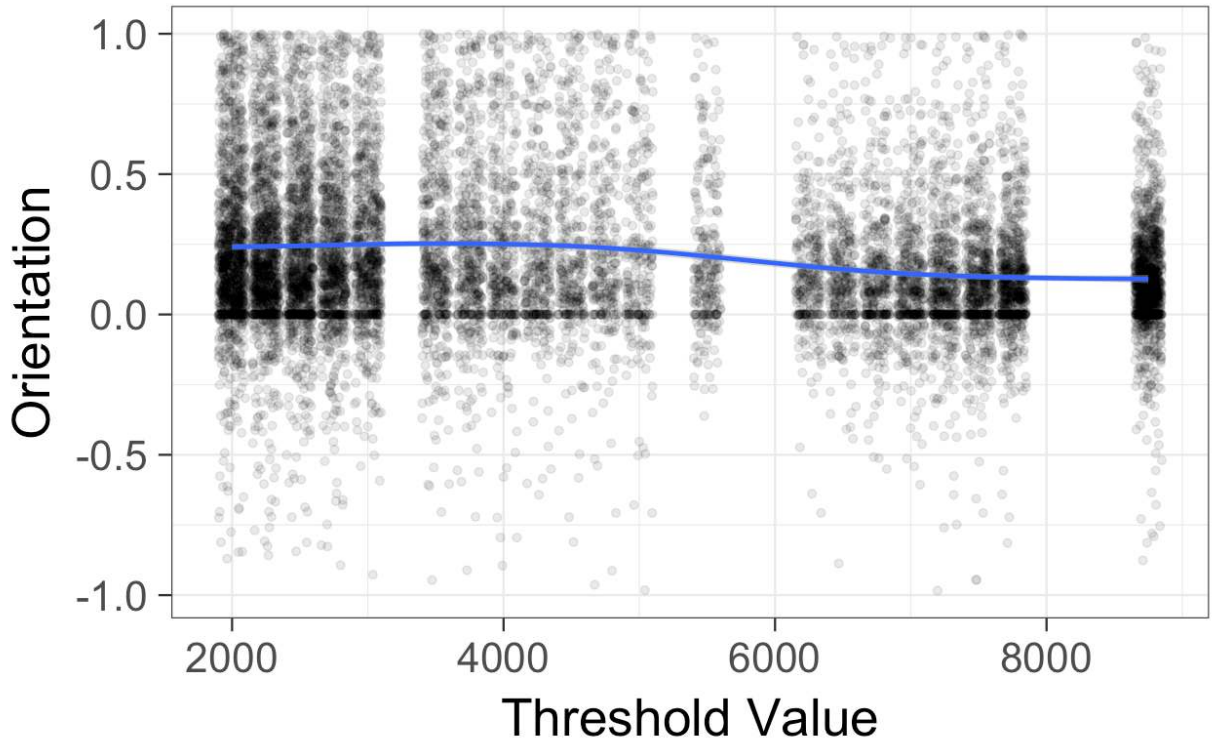


Fig. 3: Scatter plot of the orientation of the items.

0.10.3 Sensitivity

Control system theory

Sensitivity is defined as

- the change in the value of an output
- against the change in the input.

$$S = \frac{|\Delta \text{Metric}|}{|\Delta \text{Parameter}|}$$

Image processing

Such a strict definition is not particularly useful for image processing since

- a threshold has a unit of intensity and
- a metric might be volume which has m^3
  $\rightarrow$ the sensitivity becomes volume per intensity!

Practical Sensitivity

A more common approach is to estimate the variation in this parameter between images or within a single image (automatic threshold methods can be useful for this) and define the sensitivity based on this variation.

It is also common to normalize it with the mean value so the result is a percentage.

$$S = \frac{max(\text{Metric}) - min(\text{Metric})}{avg(\text{Metric})}$$

0.10.4 Sensitivity: Real Measurements

In this graph it is magnitude of the slope. The steeper the slope the more the metric changes given a small change in the parameter

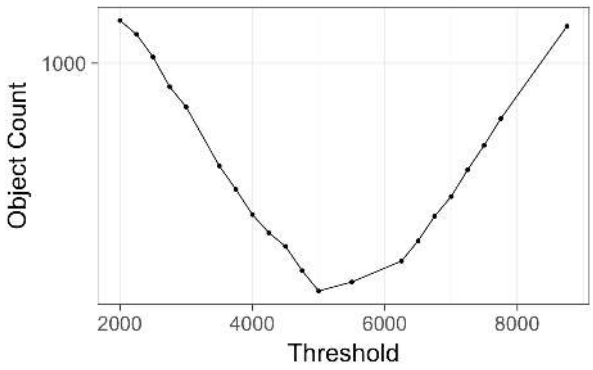


Fig. 4: Sensitivity measurement to measure how sensitive the object count is to the choice of the threshold.

0.10.5 Sensitivity: compare more than one variable

Comparing Different Variables we see that

- the best (lowest) value for the count sensitivity
- is the highest for the volume and anisotropy.

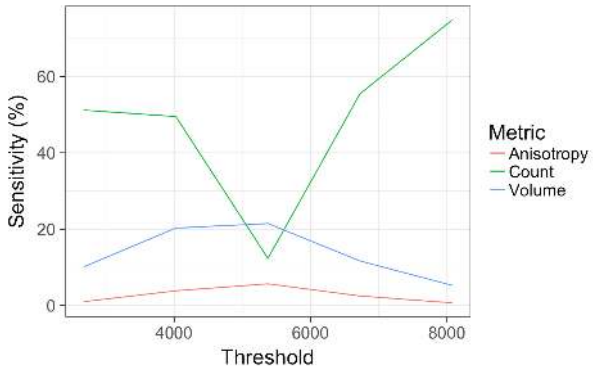


Fig. 5: Sensitivity comparison for different metrics (Anisotropy, Count, Volume) to the choice of the threshold.

A contradiction?

We see in `Fig <number> fig_comparesensitivity` that two parameters with relatively low sensitivity variations behave the same while the last one (*count*) fluctuates a lot with the threshold choice. Which one we use to guide our segmentation ultimately depends on the objective of the investigation.

0.10.6 Reproducibility

A very broad topic with plenty of sub-areas and deeper meanings. We mean two things by reproducibility

Measurement

Everything for analysis + taking a measurement several times (noise and exact alignment vary each time) does not change the statistics *significantly*

- No sensitivity to mounting or rotation
- No sensitivity to noise
- No dependence on exact illumination

Analysis

The process of going from images to numbers is detailed in a clear manner that **anyone**, **anywhere** could follow and get the exact (within some tolerance) same numbers from your samples

- No platform dependence
- No proprietary or "in house" algorithms
- No manual *clicking*, *tweaking*, or *copying*
- A single script to go from image to result

Doing analyses in a disciplined manner

- fixed, well-defined, steps
- easy to regenerate results
- no *magic*
- documentation

Advantages of a diciplined workflow

Having everything automated

- 100 samples is as easy as 1 sample
- Some intitial extra effort pays off

Being able to adapt and reuse analyses

- one really well working script
- modify parameters to address e.g.
  - different types of cells
  - different regions

xkcd 1319

0.10.7 Reproducible Analysis

Since we will need to perform the same analysis many times to understand how reproducible it is.

- Notebooks are good to develop and document analysis workflow.
- The basis for reproducible analysis are scripts and macros.

With python scripts

```
#!/$PYTHONPATH/python
import sys
from myAnalysis import analysisScript # some analysis script you implemented

imageFile = sys.argv[0] # File name from command line

threshold = 130
analysisScript(fname=imageFile, threshold = threshold)
```

0.11 Predicting and Validating - main categories

There are plenty machine-learning techniques available. Each one dedicated to a specific type of problem and data collection.

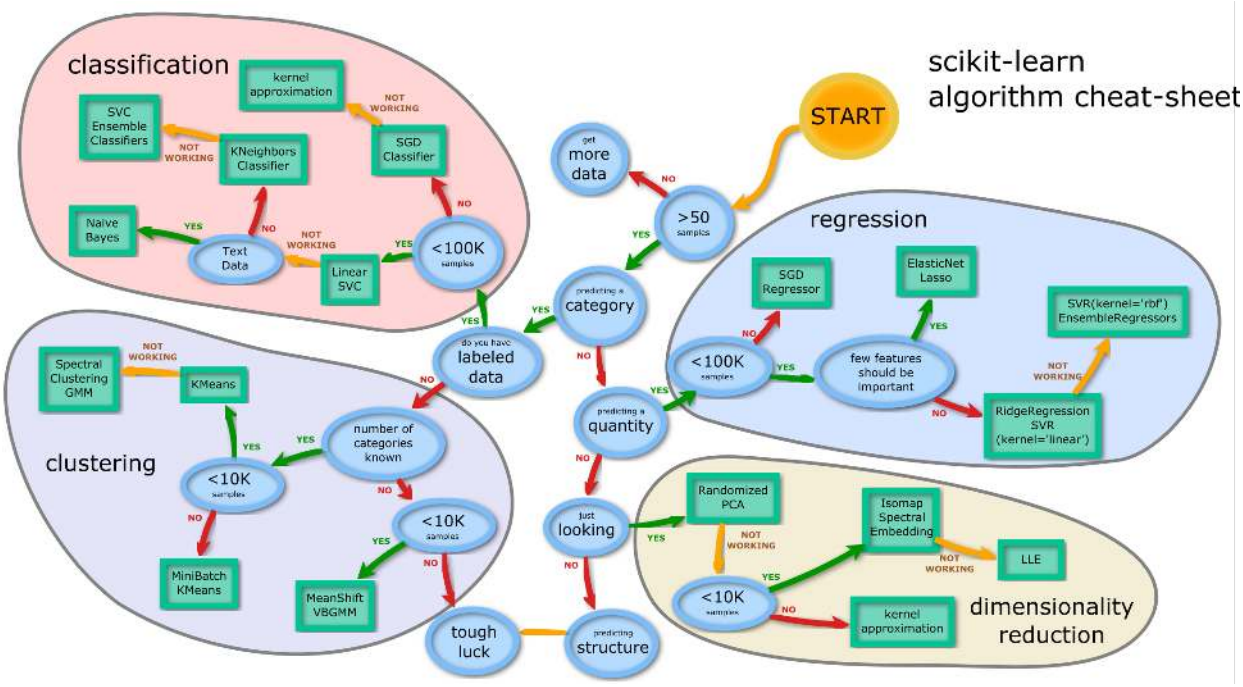


Fig. 1: A cheat sheet to identify the best machine learning technique for your problem.

`Fig <number> fig_mlcheatsheet` provides a guide to find the correct method for your problem.

A common task independent of which method you chose, it that you have to validate your processing workflow. This is important to be able to tell when and to what degree you can trust your workflow.

Borrowed from http://peekaboo-vision.blogspot.ch/2013/01/machine-learning-cheat-sheet-for-scikit.html

0.11.1 Overview

Basically all of these are ultimately functions which map inputs to outputs.

The input could be

- an image
- a point
- a feature vector
- or a multidimensional tensor

The output is

- a value (regression)
- a classification (classification)
- a group (clustering)
- a vector / matrix / tensor with *fewer* degrees of input / less noise as the original data (dimensionality reduction)

Overfitting

The most serious problem with machine learning and such approachs is overfitting your model to your data. Particularly as models get increasingly complex (random forest, neural networks, deep learning, …), it becomes more and more difficult to apply common sense or even understand exactly what a model is doing and why a given answer is produced.

Training a model like:

```
magic_classifier = {}
# training
magic_classifier['Dog']  = 'Animal'
magic_classifier['Bob']  = 'Person'
magic_classifier['Fish'] = 'Animal'
```

Now use this classifier, on the training data it works really well

```
magic_classifier['Dog']  == 'Animal' # true, 1/1 so far!
magic_classifier['Bob']  == 'Person' # true, 2/2 still perfect!
magic_classifier['Fish'] == 'Animal' # true, 3/3, wow!
```

On new data it doesn't work at all, it doesn't even execute.

```
magic_classifier['Octopus'] == 'Animal' # exception?! but it was working so well
magic_classifier['Dan']     == 'Person' # exception?!
```

This example appeared to be a perfect trainer for mapping names to animals or people, but it just memorized the inputs and reproduced them at the output and so didn't actually learn anything, it just copied.

0.11.2 Validation

Relevant for each of the categories, but applied in a slightly different way depending on the group.

The idea is to divide the dataset into groups called

- ideally training,
- validation,
- and testing.

The analysis is then

- developed on **training**
- iteratively validated on **validation**
- ultimately tested on **testing**

0.12 Presenting the results - bringing out the message

In the end you will want to present your results

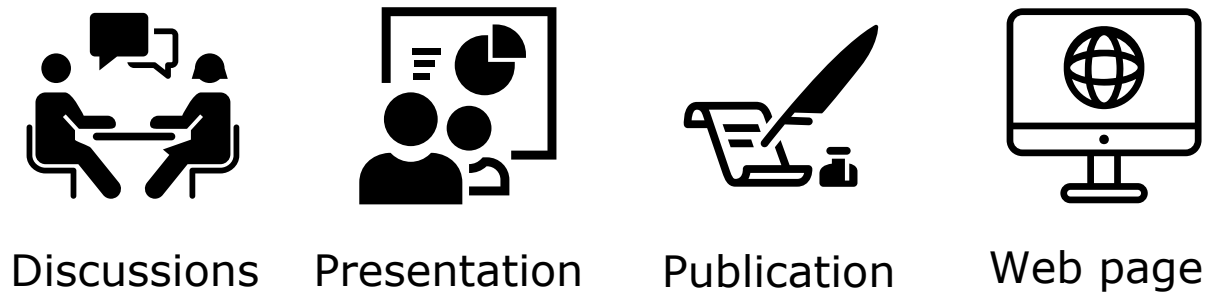


Fig. 1: Different ways to present your data.

0.12.1 Visualization

One of the biggest problems with *big* sciences is trying to visualize a lot of heterogeneous data.

- Tables are difficult to interpret
- 3D Visualizations are very difficult to compare visually
- Contradictory necessity of simple single value results and all of the data to look for trends and find problems

Purpose of the visualization

You visualize your data for different reasons:

1. **Understanding and exploration**
   - Small and known audience (you and colleagues)
   - High degree of understanding of specific topic.
2. **Presenting your results**

- Wider and sometimes unknown audience (reader of paper, person listening to presentation)
- At best general understanding of the topic.

from Knaflic 2015

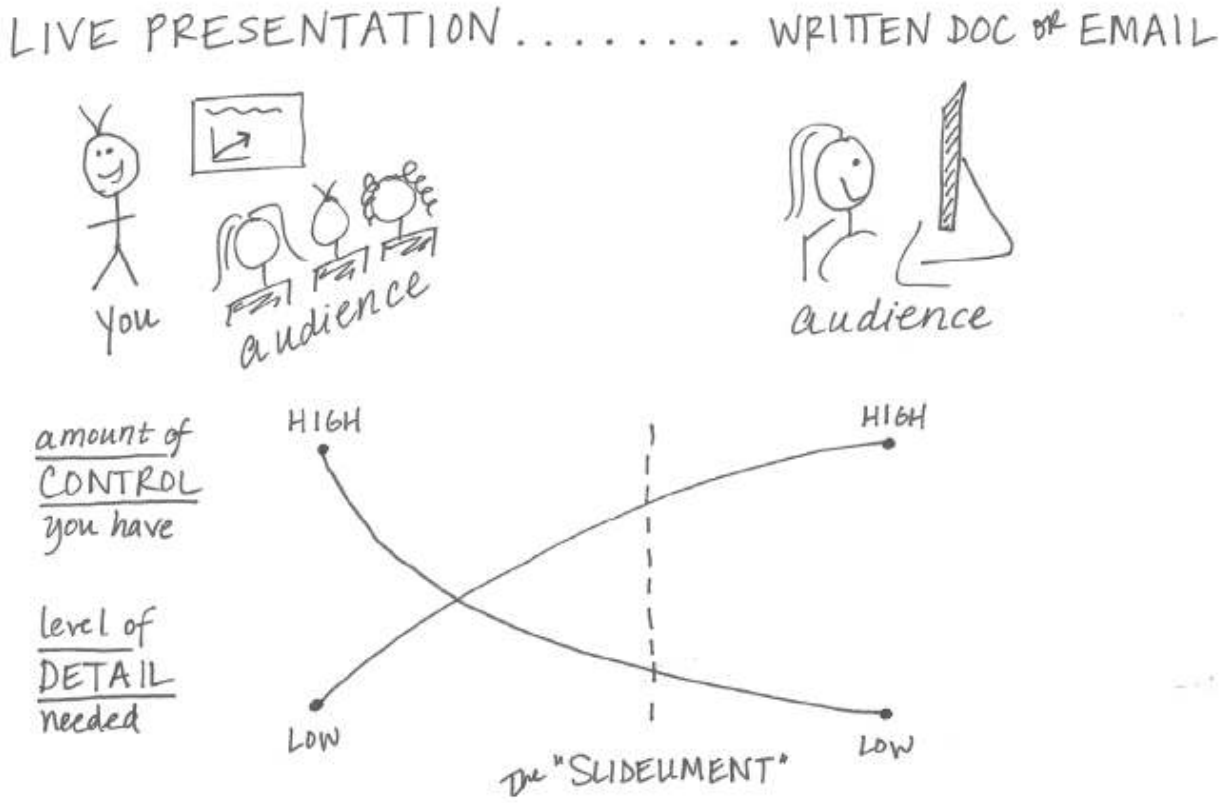


Fig. 2: The level of detail in a presentation depends on the medium it is presented Knaflic 2015.

0.12.2 Bad Graphs

There are too many graphs which say:

- *my data is very complicated*
- *I know how to use __ toolbox in Python/Matlab/R/Mathematica*
- Most programs by default make poor plots
- Good visualizations takes time to produce

xkcd

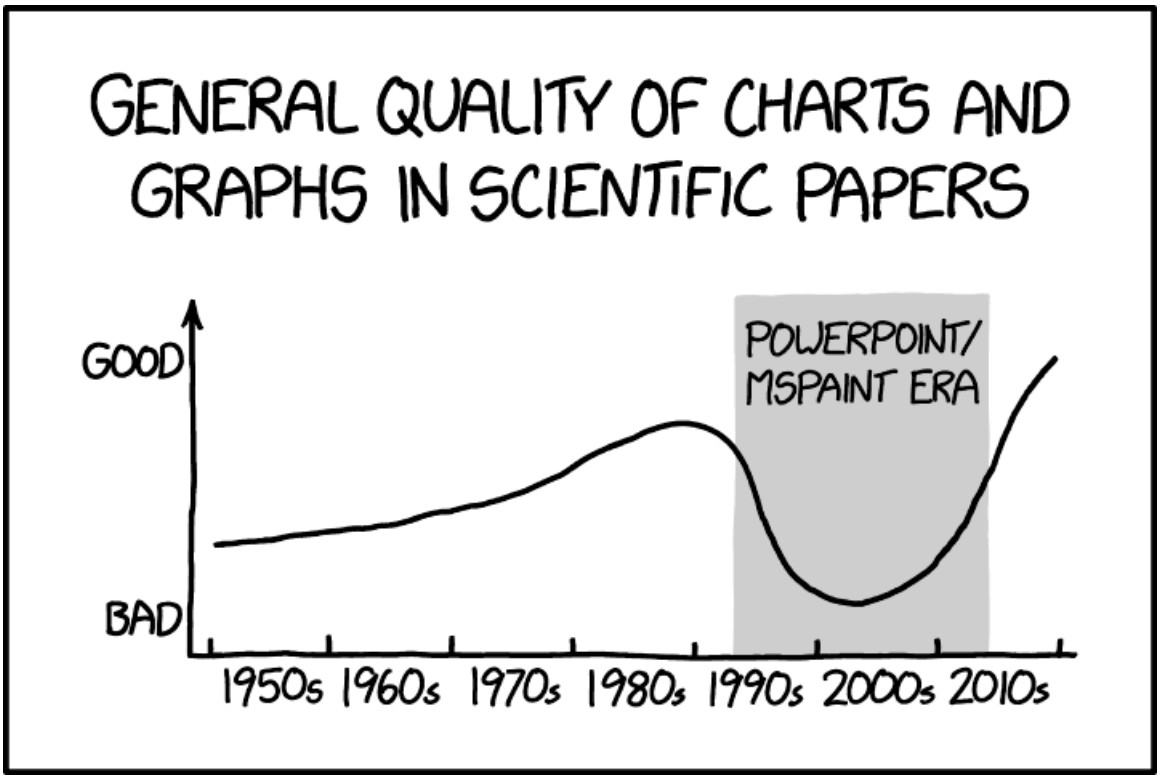


Fig. 3: This cartoon from XKCD highlights a problem with the access to software making it too easy to produce a graph or illustration. Unfortunately, without any thoughts about information content and artistic or illustration rules.

Some bad examples

There are plenty examples on how you shouldn't present your data. The problem is in general that there is way too much information that needs to be predigestend before it is ready to any audience.

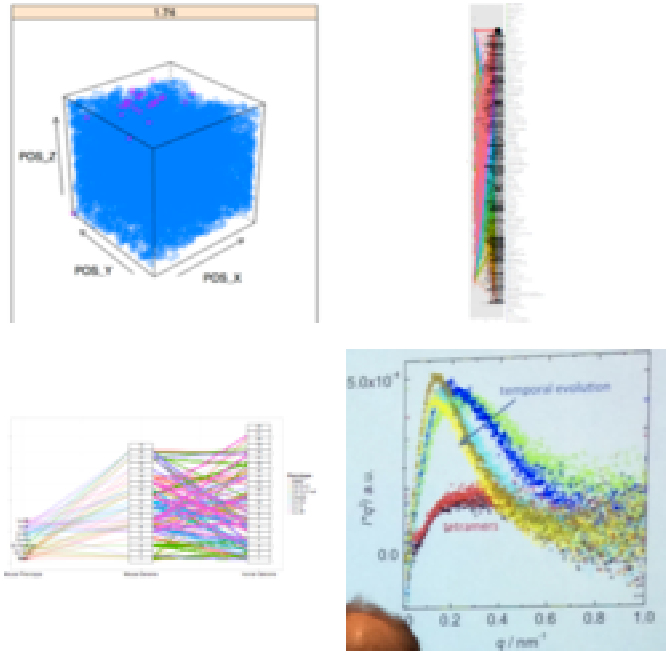


Fig. 4: Four examples of how *not* to present your data.

0.12.3 How to improve - Key Ideas

There is a need to consciously prepare your figures to bring your message to the audience in an understandable way. The first step is to ask yourself the following questions.

What is my message?

Is it clearly communicated?

Is it really necessary?

- Does every line / color serve a purpose?
- Pretend ink is very expensive

Keep this in mind every time you create a figure and you will notice that after while you will have a tool set that makes it easier and faster to produce well thought figures that clearly brings out you message to your audience.

Personally, I always write scripts to produce each plot of a publication. This makes it easier to revise the manuscript in a reproducible and efficient manner. The first implementation may take longer, but the revision is done in no time.

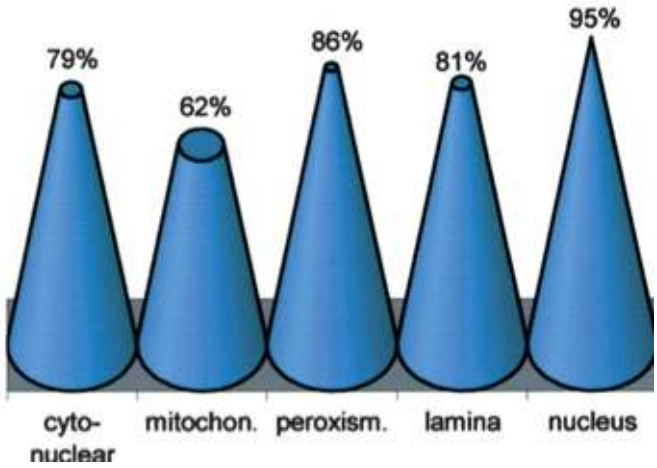
Some literature

If you want to read more about how to work with data visualization. I can recommend these:

- Knaflic, Storytelling with Data: A Data Visualization Guide for Business Professionals, 2015
- Few, Should data visualization always be beautiful?, 2012

Simple Rules

1. Never use 3D graphics when it can be avoided (unless you want to be deliberately misleading)



2. Pie charts can also be hard to interpret
3. Background color should almost always be white (not light gray)
4. Use color palettes adapted to human visual sensitivity
5. Use colors and transparency smart

0.12.4 Grammar of Graphics

What is a grammar?

- Set of rules for constructing and validating a sentence
- Specifies the relationship and order between the words constituting the sentence

How does grammar apply to graphics?

If we develop a consistent way of

- expressing graphics (sentences)
- in terms of elements (words) we can compose and decompose graphics easily

The most important modern work in graphical grammars is
"The Grammar of Graphics" by Wilkinson, Anand, and Grossman (2005).

This work built on earlier work by Bertin (1983) and proposed a grammar that can be used to describe and construct a wide range of statistical graphics.

Grammar Explained

Normally we think of plots in terms of some sort of data which is fed into a plot command that produces a picture

- In Excel you select a range and plot-type and click "Make"
- In Matlab you run `plot(xdata,ydata,color/shape)`

1. These produces entire graphics (sentences) or at least phrases in one go and thus abstract away from the idea of grammar.
2. If you spoke by finding entire sentences in a book it would be very ineffective, it is much better to build up word by word

Grammar

Separate the graph into its component parts

$$\begin{cases} var1 \rightarrow x \\ var2 \rightarrow y \end{cases}$$

Construct graphics by focusing on each portion independently.

Figure decorations

Besides the data you also need to provide annotating items to the visualization.

It may seem unnescessary to list these annotations, but it happens too often that they are missing. This leaves the observers wondering about what they see in the figure. It is true that it takes a little more time to add annotation to your figure. Sometimes, you may think that the plot is only for your own understanding and you don't need to waste the time on making it complete. Still, in the next moment it finds its way to the presentation and then all of a sudden it is offical…

Annotations are fundamental features of figures and available in any plotting library. In some cases you have to look a little longer to find them or write a little more code to use them, but they are there.

Plots

- Curve legend - telling what each curve represents.
- Axis labels - tellning what information you see on each axis.
- Figure title - if you use multiples plots in the same figure.

Images

- Color bar - to tell how the colors are mapped to the values.
- Scale bar - to tell the size of the object in the image.

Color maps revisited

Choosing the right color is a science.

Crameri, F., et al. (2020)

The choice depends on the type of data you want to present and how humans percieve different colors. Some combinations make it easier to highlight relevant features in the images. Still, you have to be cautious not put too much a priori information into the color map.

Visualization toolboxes provide a great collection of colormaps as we have seen several times already in this course. There are however cases when you have to define your own color map. An example is the colormap we created last week to be able to identify each item in a watershed segmented image.

0.12.5 What is my message?

Plots to "show the results" or "get a feeling" are usually not good

```
from plotnine import *
from plotnine.data import *
import pandas as pd
import numpy as np
# Some data
xd = np.random.rand(80)
yd = xd + np.random.rand(80)
zd = np.random.rand(80)

df = pd.DataFrame(dict(x=xd,y=yd,z=zd))
ggplot(df,aes(x='x',y='y')) + geom_point()
```

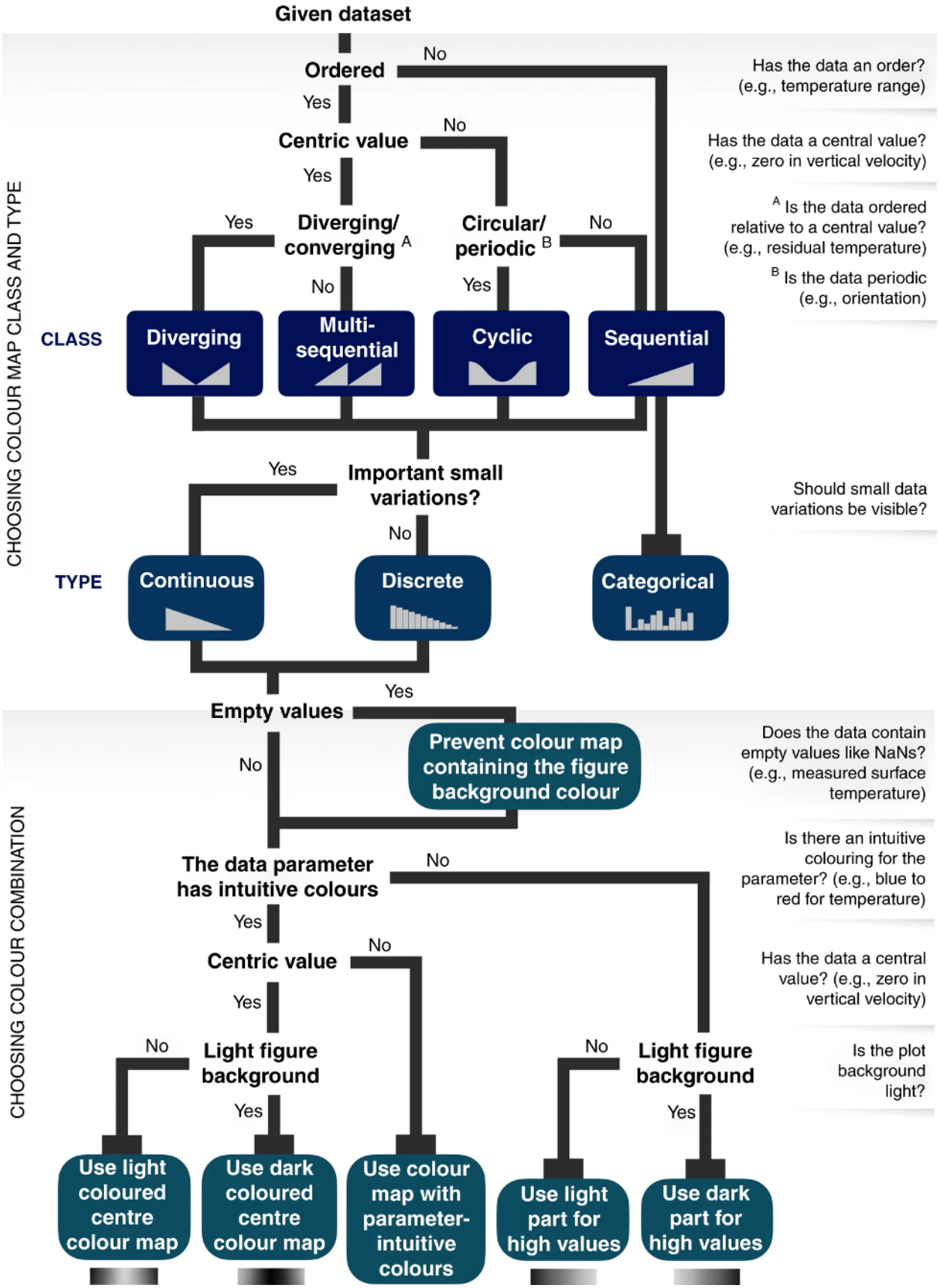
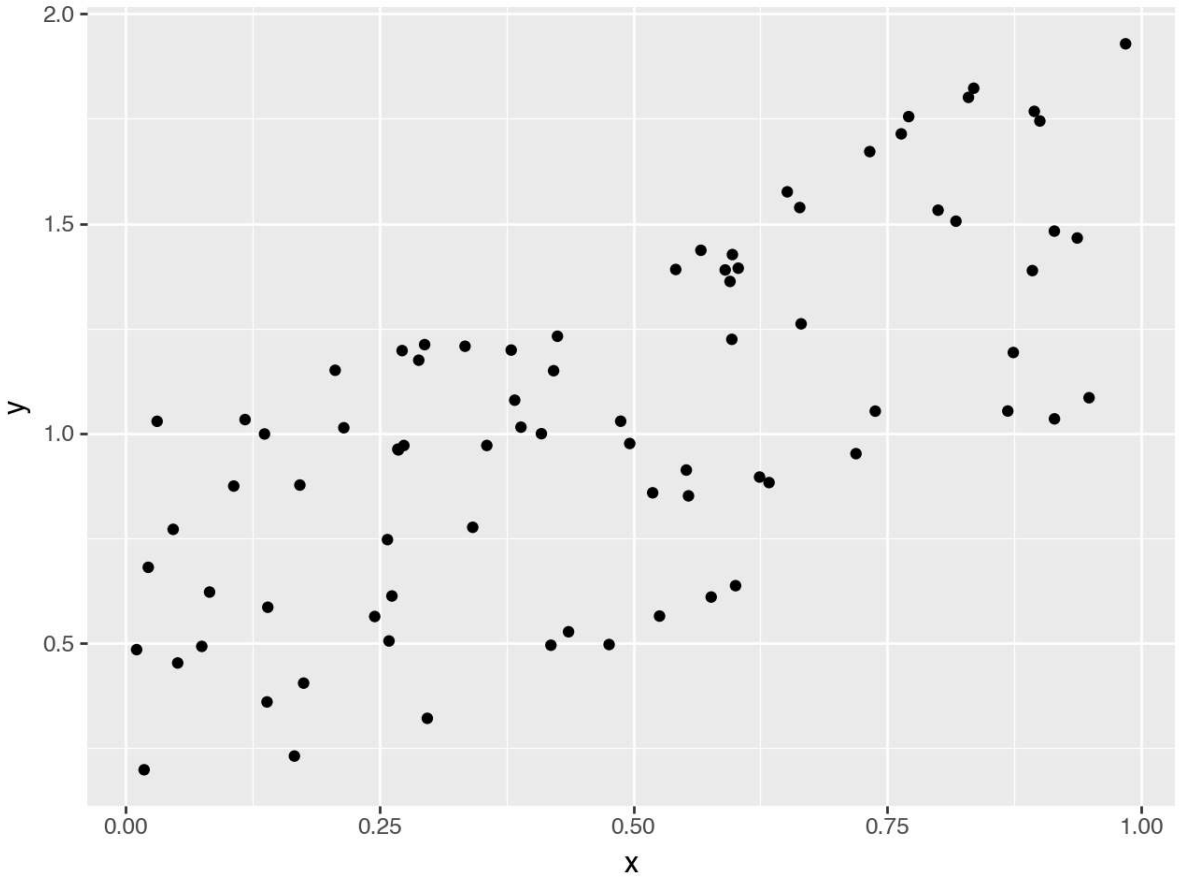


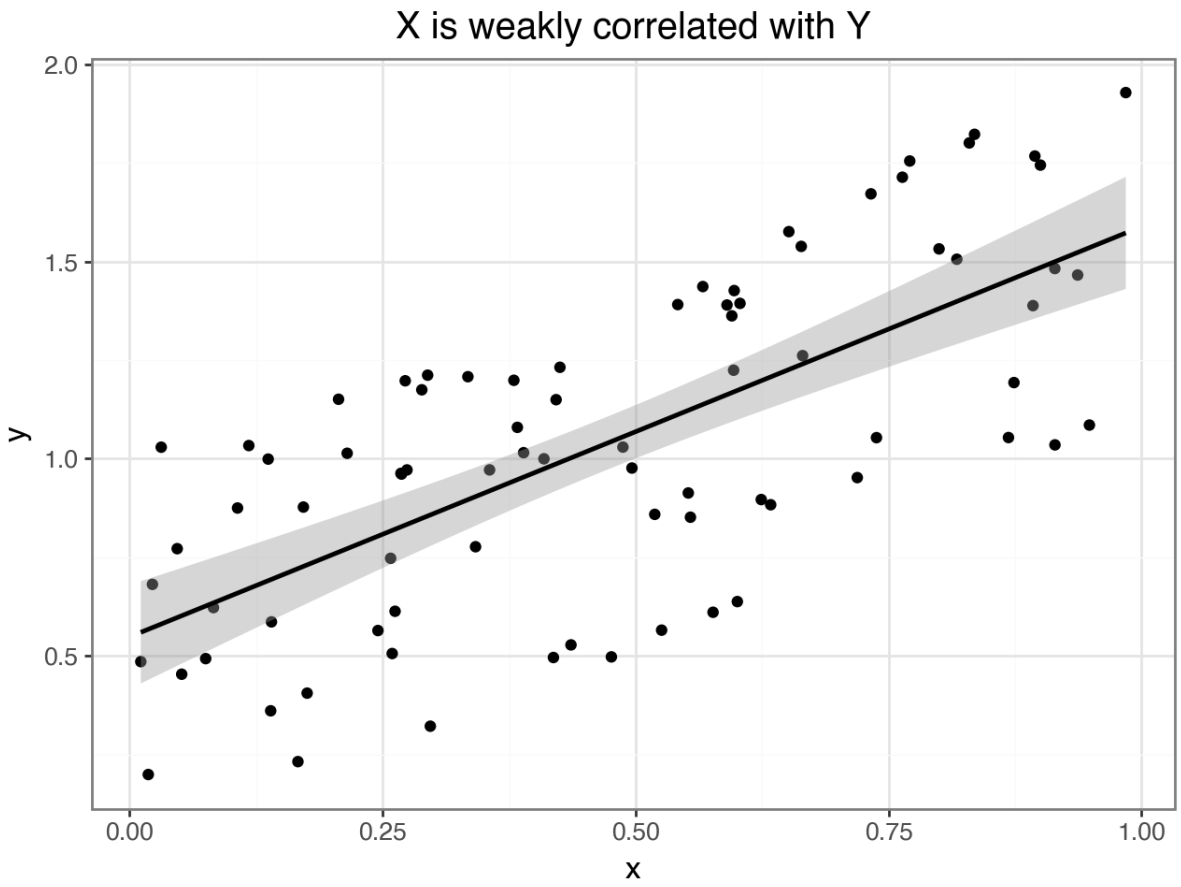
Fig. 5: Crameri et al. developed this decision flow chart to help you decide which type of color map is best suited for your data.



Focus on a single, simple message

"X is a little bit correlated with Y"

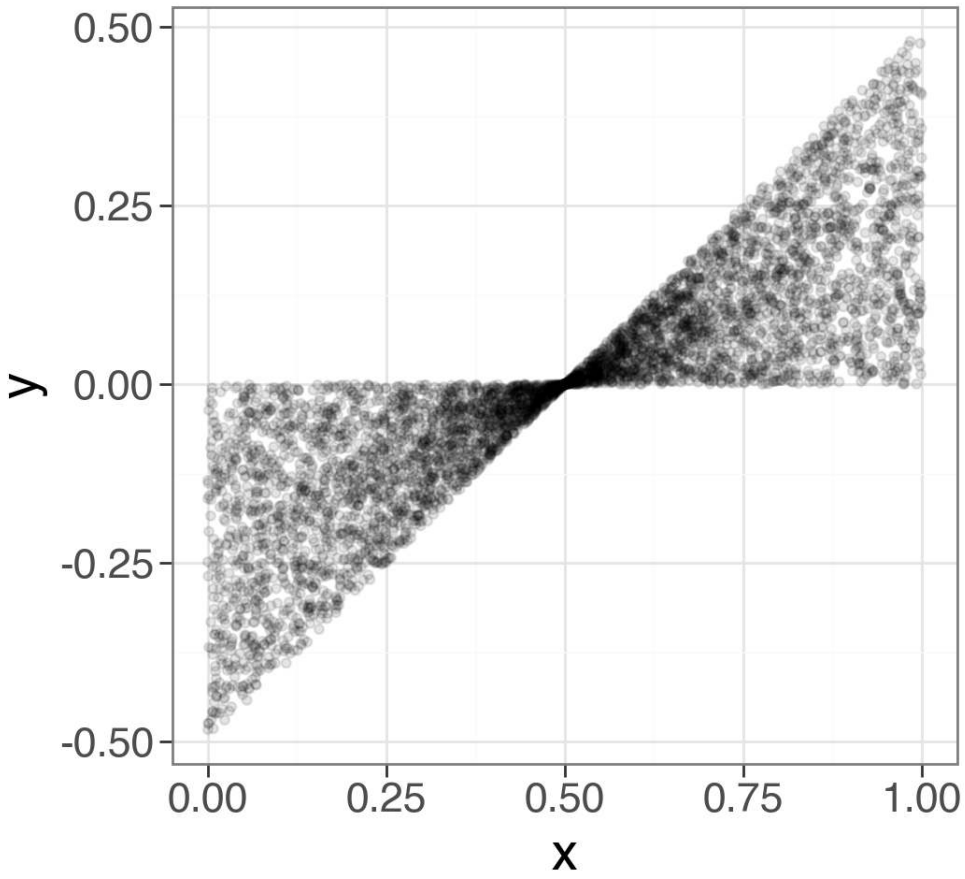
```
(ggplot(df,aes(x='x',y='y'))
 + geom_point()
 + geom_smooth(method="lm")
# + coord_equal()
 + labs(title="X is weakly correlated with Y")
 + theme_bw(12) )
```



Does my graphic communicate it clearly?

Too much data makes it very difficult to derive a clear message

```
xd = np.random.rand(5000)
yd = (xd-0.5)*np.random.rand(5000)

df = pd.DataFrame(dict(x=xd,y=yd))
(ggplot(df,aes(x='x',y='y'))
# + geom_point()
 + geom_point( alpha = 0.1 )
 + coord_equal()
 + theme_bw(20))
```



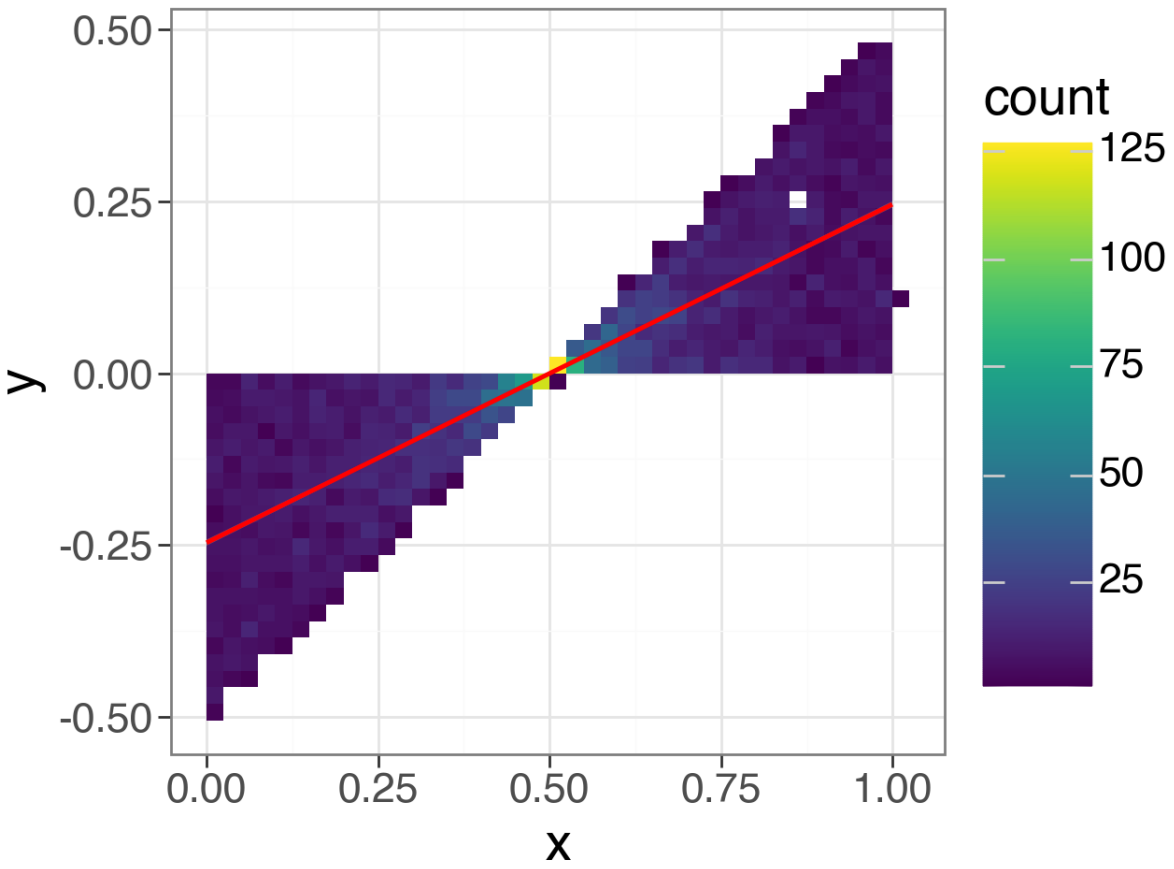
We have earlier used transparency to better visualize dense scatter plots. You can see the effect by setting the alpha parameter to geom_point. Using transparency is a qualitative way of showing higher density in the data.

Reduce the data

Filter and reduce information until it is extremely simple

In this plot we create a density count view of the data by downsampling the grid and count the amount of points in each bin. It is related to a histogram but it count in space instead of in the intensity levels.

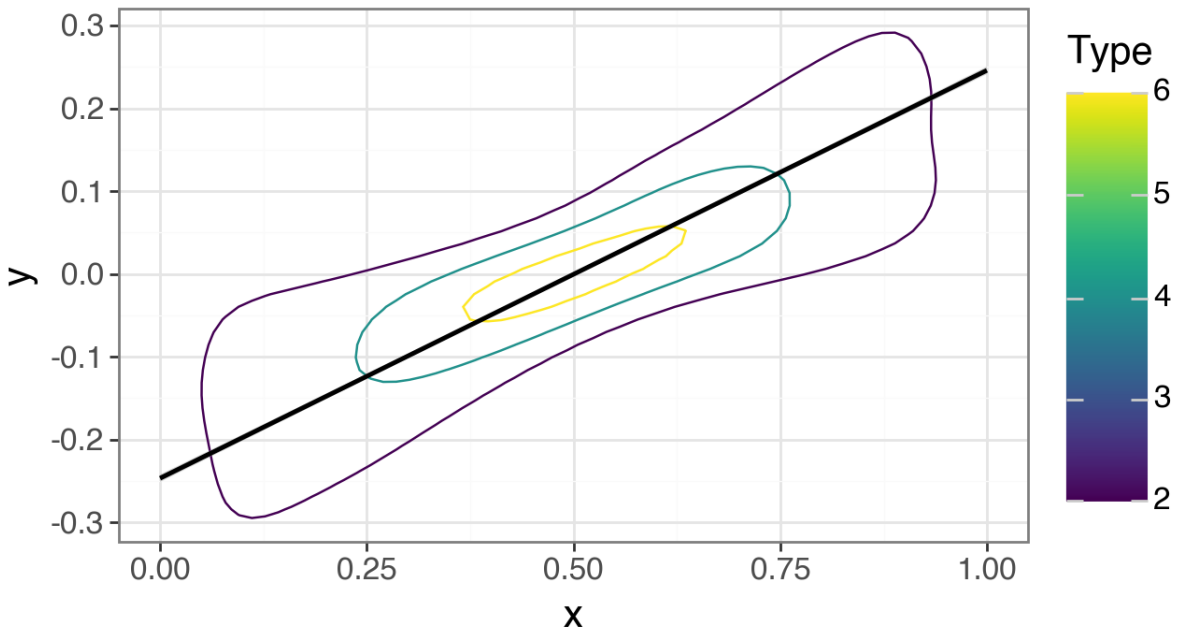
```
(ggplot(df,aes(x='x',y='y'))
 + stat_bin_2d(bins=40)
 + geom_smooth(method="lm",color='red')
 + coord_equal()
 + theme_bw(20)
 + guides(color='F')
)
```



Using this kind of plot allows us to measure how many points there are in each bin and thus we are now going towards a quantitative plot. The colorbar on the side helps us to interpret the colors.

Reduce even further

```
(ggplot(df,aes(x='x',y='y'))
  + geom_density_2d(aes(x='x', y='y', color='..level..'))
  + geom_smooth(method="lm")
  + coord_equal()
  + labs(color="Type")
  + theme_bw(15)
)
```



0.12.6 Common visualization packages for python

- Matplotlib Matplotlib 3.0 Cookbook or ETHZ lib, code examples
- Plotly
- Seaborn
- ggplot R using the ggplot2 library, which is ported to python.

A short summary of these packages can be found here.

0.13 Summary

0.13.1 Uncertainties

- Every measurement has an uncertainty
- Reporting these is important for good science.

0.13.2 Statistics

- Try and understand the tests you are performing
- Simulations (even simple ones) are very helpful
- If you have questions or concerns
  - Both ETHZ and Uni Zurich offer **free** consultation with real statisticians
  - They are rarely bearers of good news - you allways need more data…

0.13.3 Visualization

- Visualization is the crowning piece of your investigation - make it count!
- Many toolboxes can be used, choose the one that fits your needs.