

Feature extraction and working with tables

Robert Haase

With material from

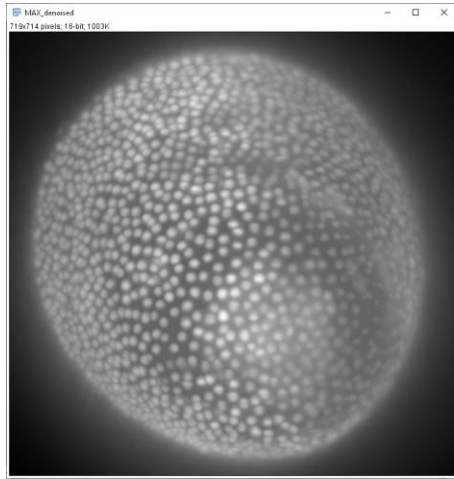
Johannes Müller, BiAPoL, PoL TU Dresden

Marcelo Zoccoler, BiAPoL, PoL, TU Dresden

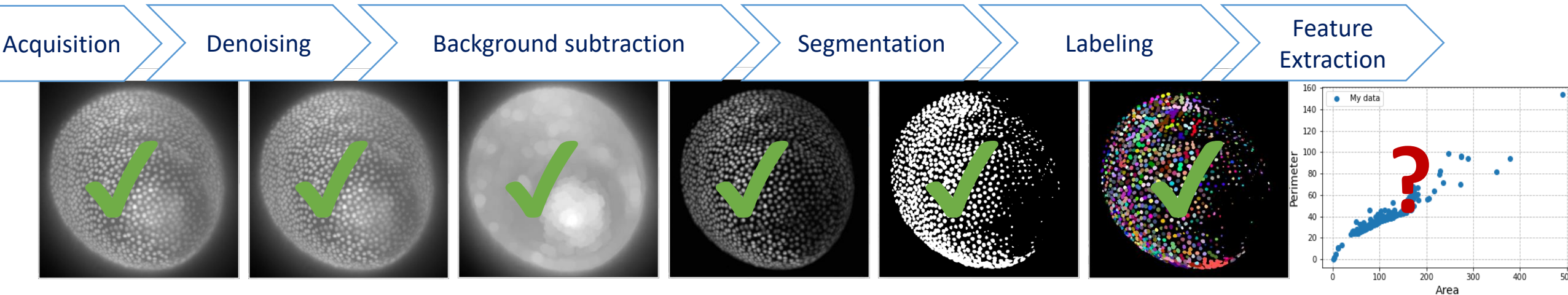
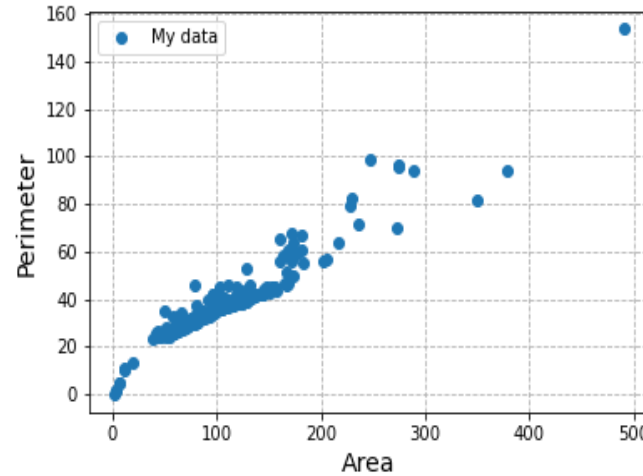
Benoit Lombardot, Scientific Computing Facility, MPI CBG

September 2022

- Feature extraction is a *late* processing step in image analysis.
- It can be used for images, or segmented/labelled images

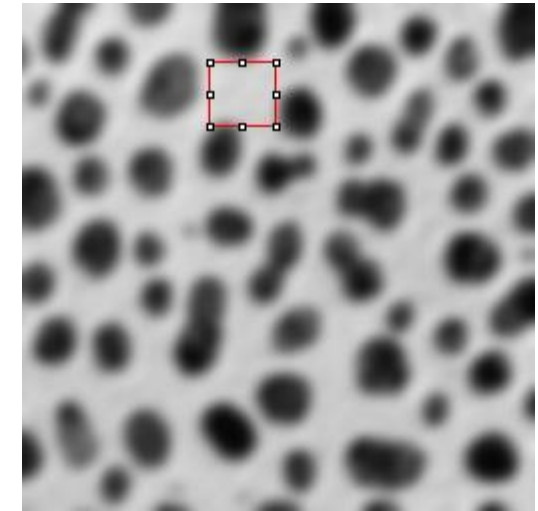
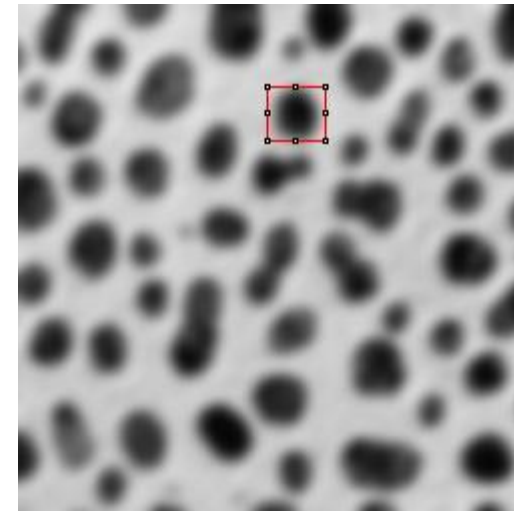
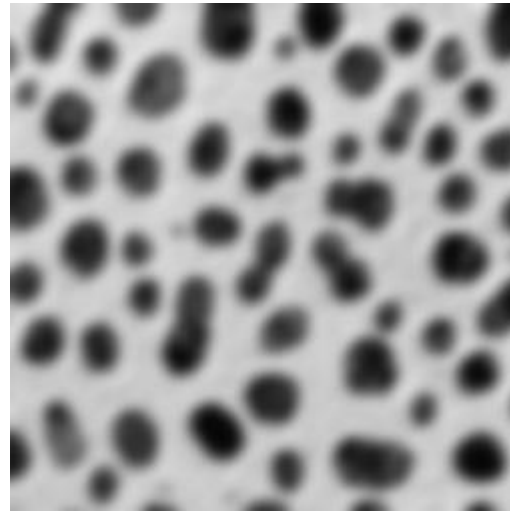


Feature
Extraction

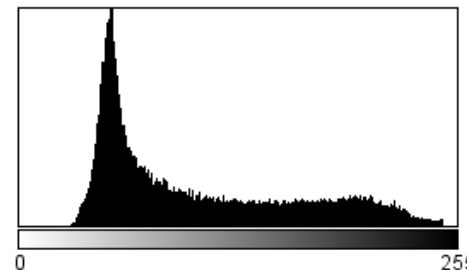


- A *feature* is a countable or measurable property of an image or object.
- Goal of feature extraction is finding a minimal set of features to describe an object well enough to differentiate it from other objects.
- **Intensity based**
 - Mean intensity
 - Standard deviation
 - Total intensity
 - Textures
- **Shape based /spatial**
 - Area / Volume
 - Roundness
 - Solidity
 - Circularity / Sphericity
 - Elongation
 - Centroid
 - Bounding box
- **Spatio-temporal**
 - Displacement,
 - Speed,
 - Acceleration
- **Others**
 - Overlap
 - Colocalization
 - Neighborhood
- **Mixed features**
 - Center of mass
 - Local minima / maxima

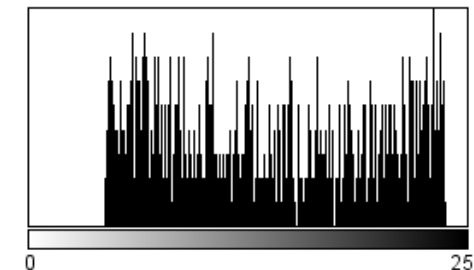
- Min / max
- Median
- Mean
- Mode
- Variance
- Standard deviation



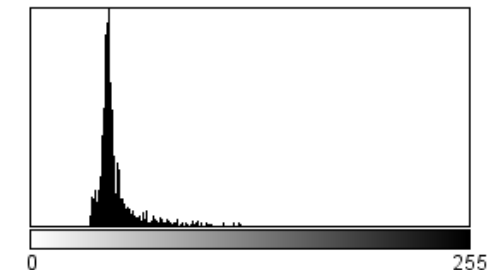
- Can be derived from pixel values
- Don't take spatial relationship of pixels into account
- See also:
 - descriptive statistics
 - histogram



Count: 65024
Mean: 103.301
StdDev: 57.991
Min: 29
Max: 248
Mode: 53 (1663)



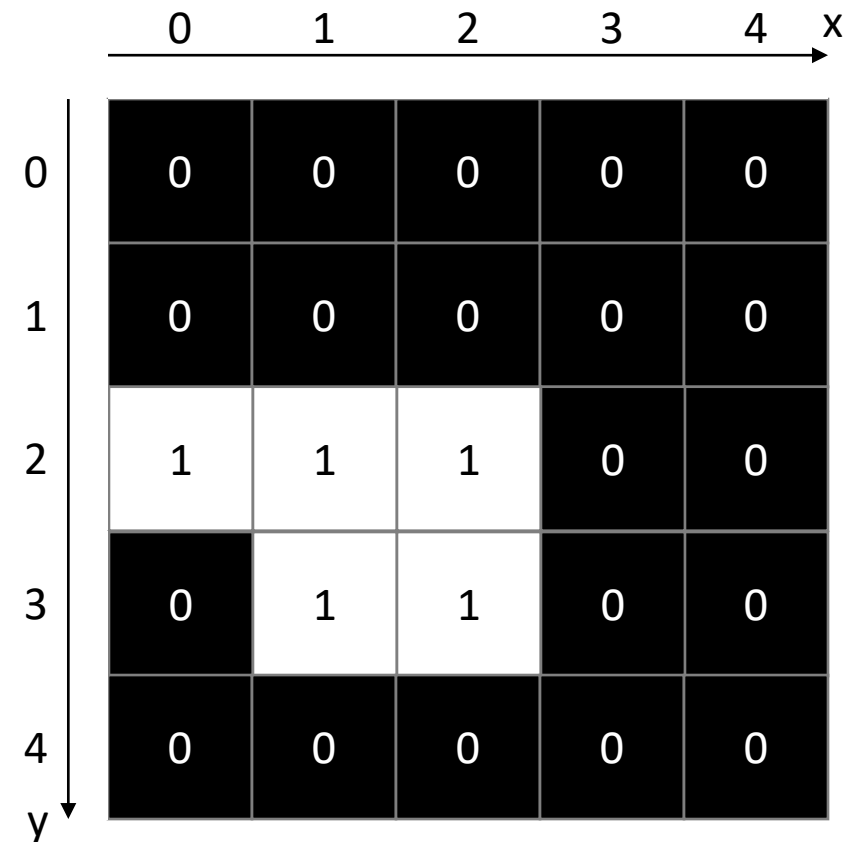
Count: 783
Mean: 141.308
StdDev: 61.876
Min: 44
Max: 243
Mode: 236 (9)



Count: 1056
Mean: 49.016
StdDev: 12.685
Min: 34
Max: 122
Mode: 45 (120)

- Position and size of the smallest rectangle containing all pixels of an object
 - x_b, y_b ... position of the bounding box
 - w_b ... width of the bounding box
 - h_b ... height of the bounding box

variable	value
x_b	0
y_b	2
w_b	3
h_b	2



- Relative position in an image weighted by pixel intensities

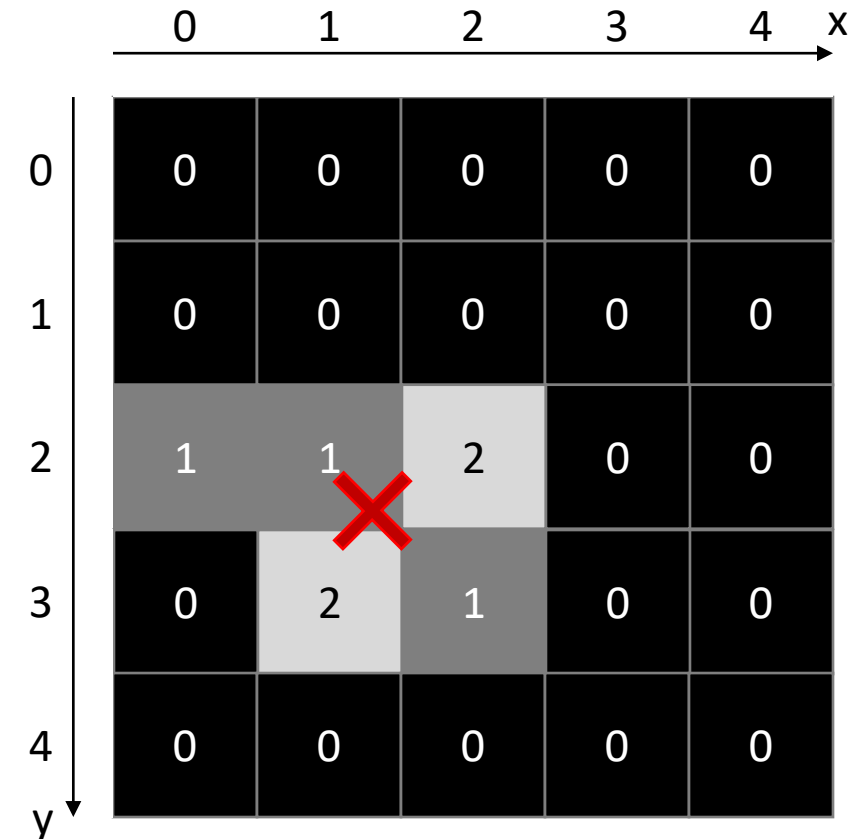
- x, y ... pixel coordinates
- w ... image width
- h ... image height
- μ ... mean intensity
- $g_{x,y}$... pixel grey value
- x_m, y_m ... center of mass coordinates

$$\mu = \frac{1}{wh} \sum_{y=0}^{h-1} \sum_{x=0}^{w-1} g_{x,y}$$

$$x_m = \frac{1}{wh\mu} \sum_{y=0}^{h-1} \sum_{x=0}^{w-1} x g_{x,y}$$

$$y_m = \frac{1}{wh\mu} \sum_{y=0}^{h-1} \sum_{x=0}^{w-1} y g_{x,y}$$

“sum intensity”
“total intensity”



$$x_m = 1/7 (1 \cdot 0 + 1 \cdot 1 + 2 \cdot 2 + 2 \cdot 1 + 1 \cdot 2) = 1.3$$

$$y_m = 1/7 (1 \cdot 2 + 1 \cdot 2 + 2 \cdot 3 + 2 \cdot 2 + 1 \cdot 3) = 2.4$$

- Relative position in an image weighted by pixel intensities
- Special case of center of mass for binary images
 - x, y ... pixel coordinates
 - w ... image width
 - h ... image height
 - μ ... mean intensity
 - $g_{x,y}$... pixel grey value, integer in range [0;1]
 - x_m, y_m ... center of mass coordinates

$$\mu = \frac{1}{wh} \sum_{y=0}^{h-1} \sum_{x=0}^{w-1} g_{x,y}$$

$$x_m = \frac{1}{wh\mu} \sum_{y=0}^{h-1} \sum_{x=0}^{w-1} x g_{x,y}$$

$$y_m = \frac{1}{wh\mu} \sum_{y=0}^{h-1} \sum_{x=0}^{w-1} y g_{x,y}$$

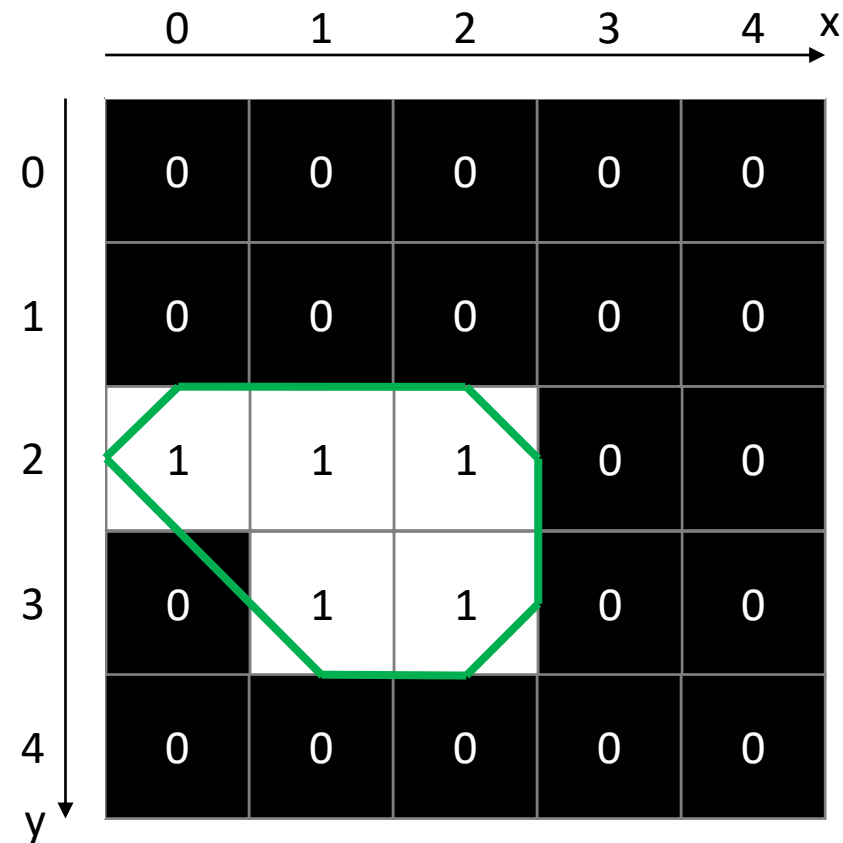
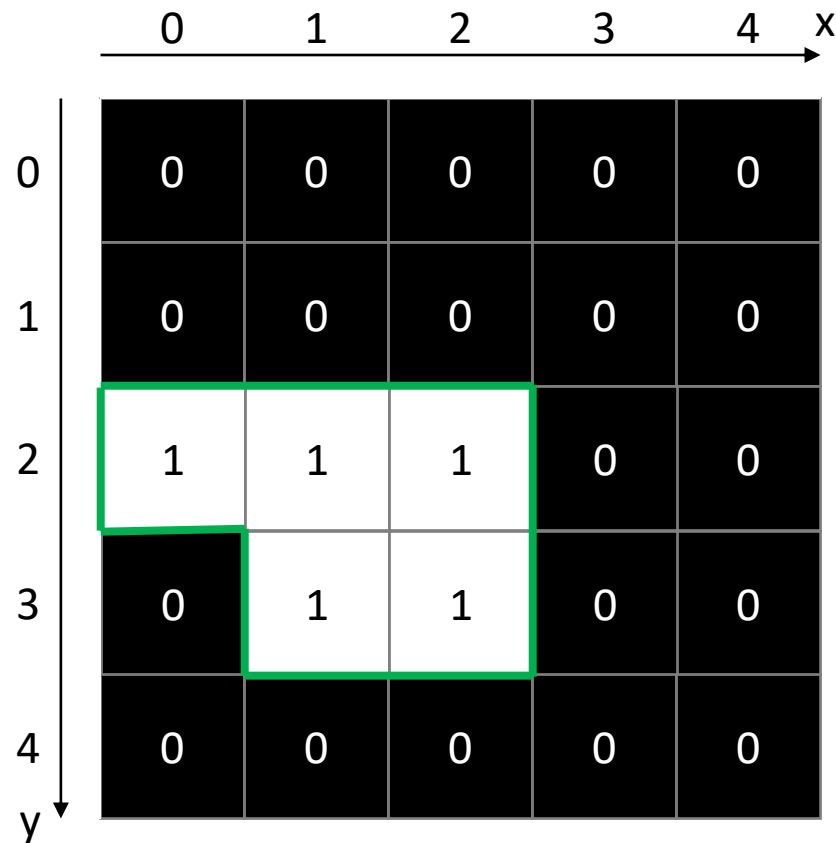
Number of white pixels

	0	1	2	3	4	x
0	0	0	0	0	0	
1	0	0	0	0	0	
2	1	1	1	0	0	
3	0	1	1	0	0	
4	0	0	0	0	0	
y						

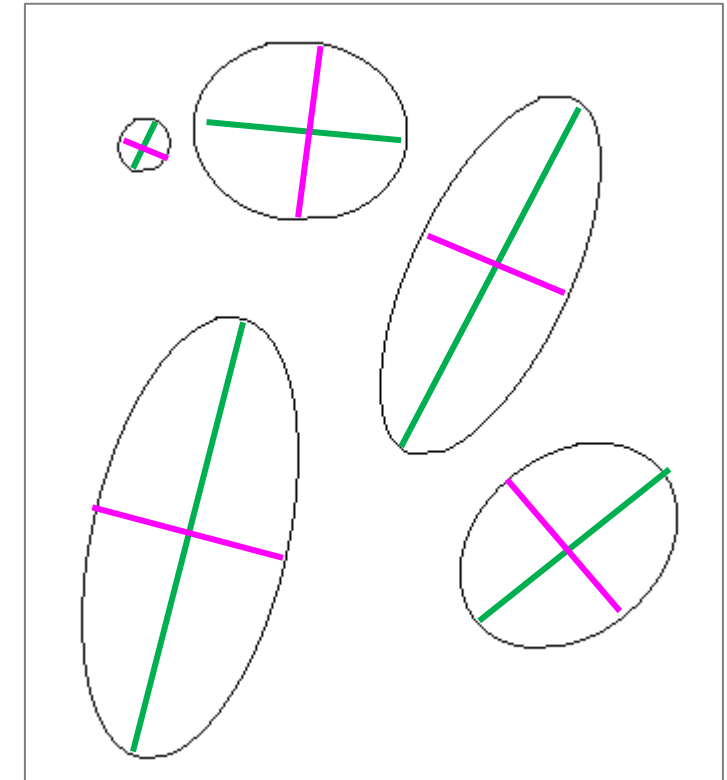
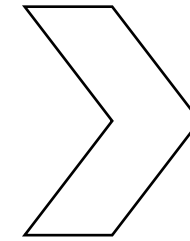
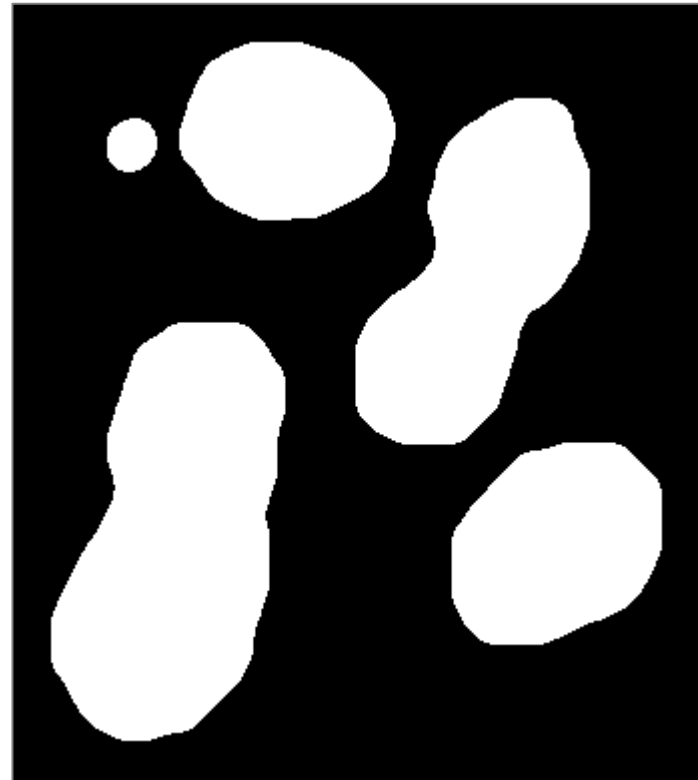
$$x_m = 1/5 (1 \cdot 0 + 1 \cdot 1 + 1 \cdot 2 + 1 \cdot 1 + 1 \cdot 2) = 1.2$$

$$y_m = 1/5 (1 \cdot 2 + 1 \cdot 2 + 1 \cdot 3 + 1 \cdot 2 + 1 \cdot 3) = 2.4$$

- Length of the outline around an object
- Depends on the actual implementation

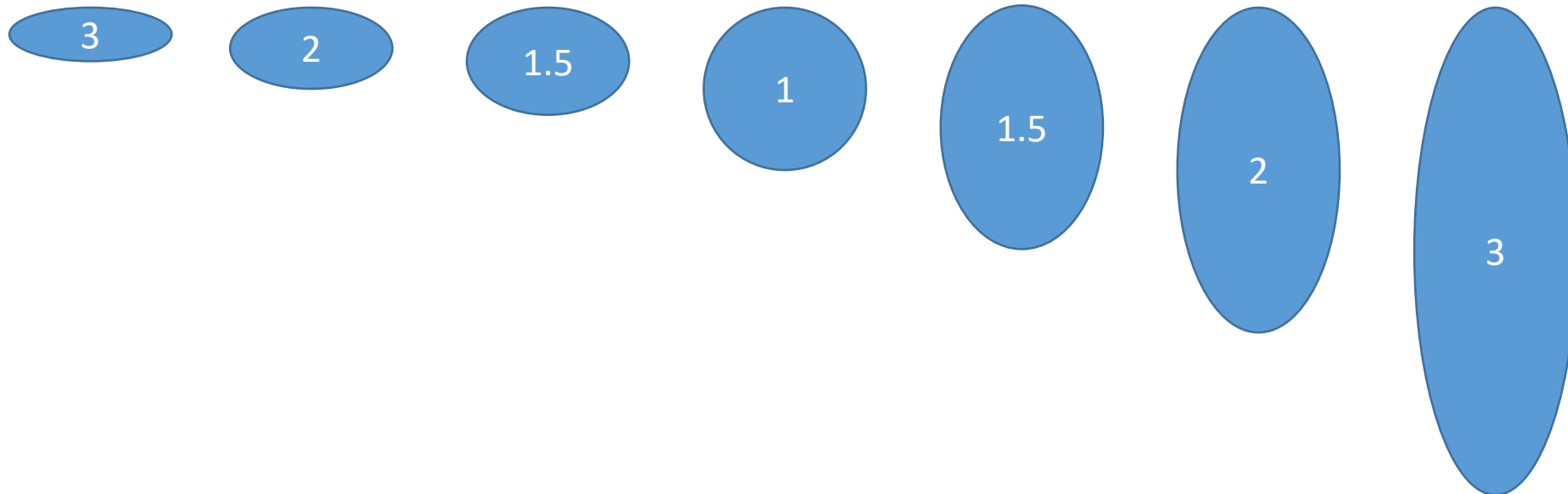


- For every object, find the optimal ellipse simplifying the object.
- Major axis ... long diameter
- Minor axis ... short diameter
- Major and minor axis are perpendicular to each other

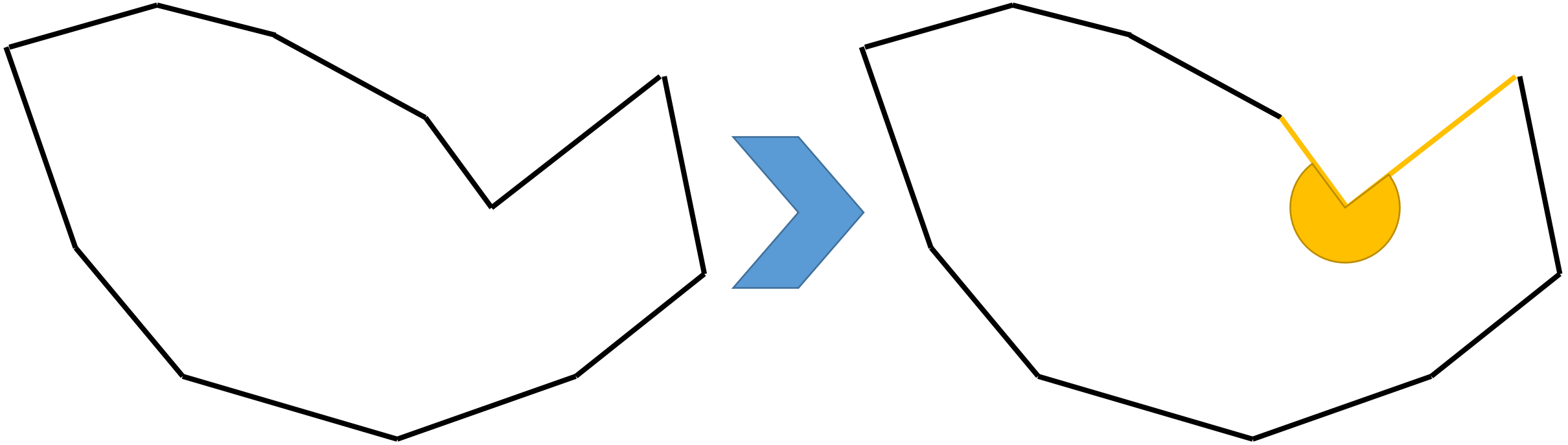


- The aspect ratio describes the elongation of an object.

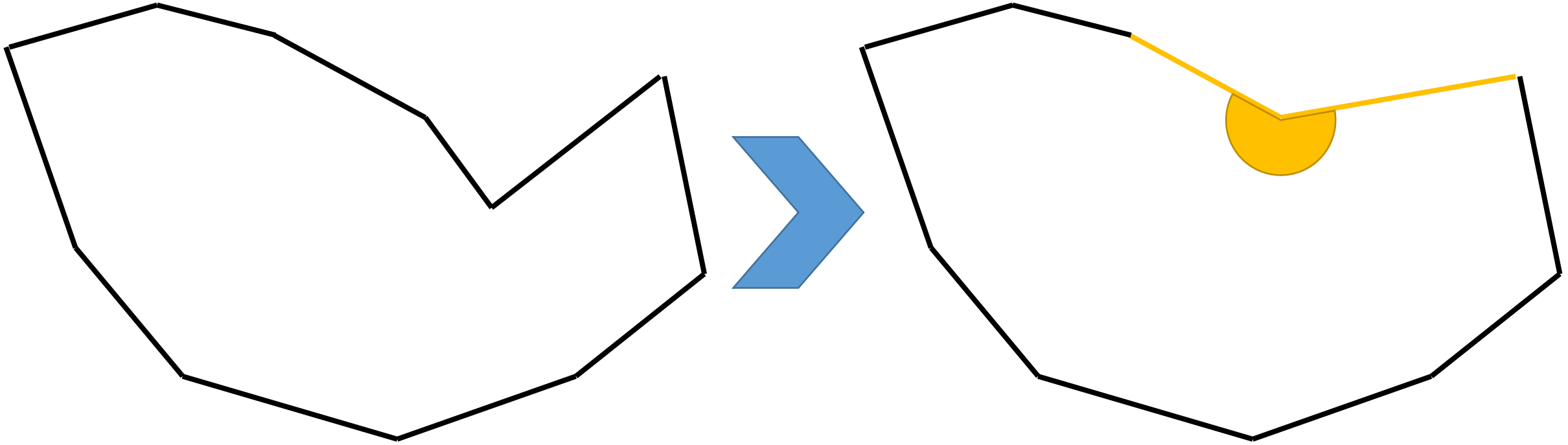
$$AR = \text{major} / \text{minor}$$



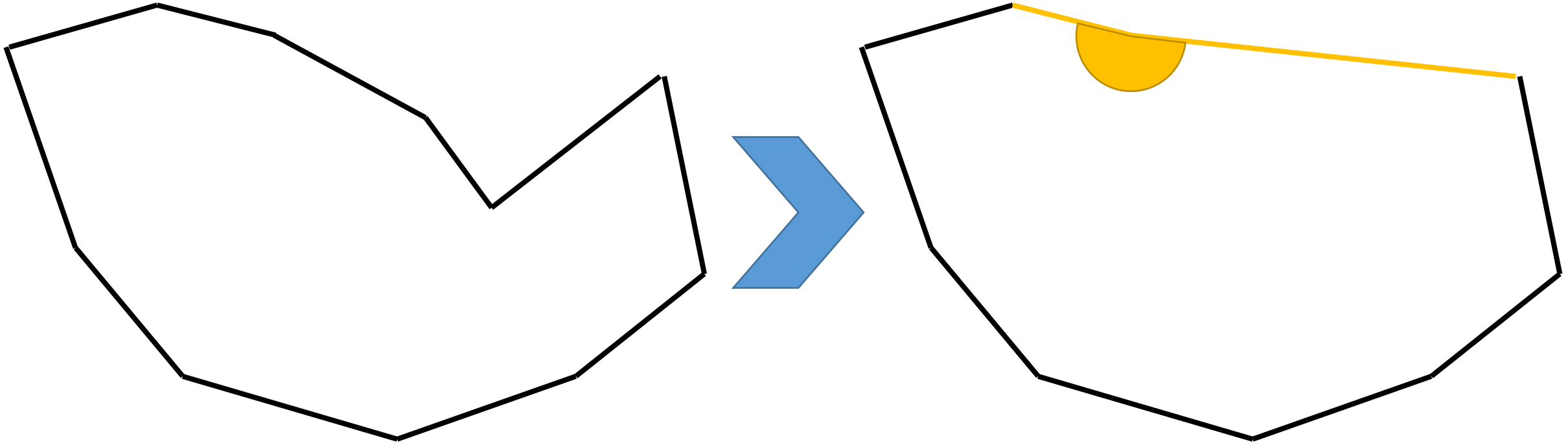
- By removing all concave corners of an object, we retrieve its **convex hull**.



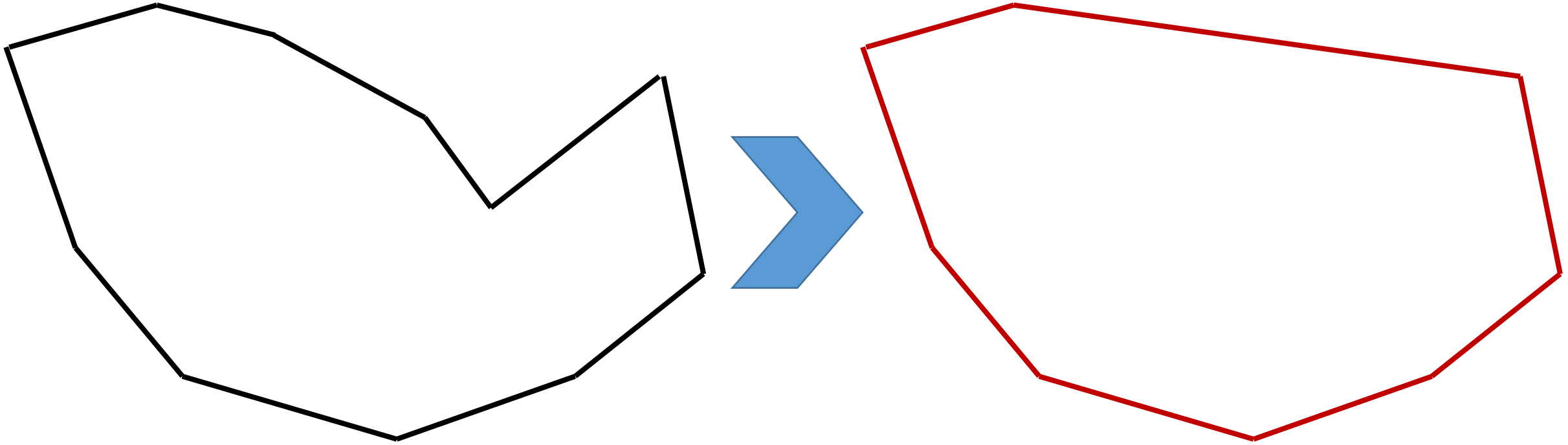
- By removing all concave corners of an object, we retrieve its **convex hull**.



- By removing all concave corners of an object, we retrieve its **convex hull**.



- By removing all concave corners of an object, we retrieve its **convex hull**.



$$solidity = \frac{A}{A_{convexHull}}$$

Roundness and circularity

- The definition of a circle leads us to measurements of circularity and roundness.
- In case you use these measures, define them correctly. They are not standardized!

Diameter

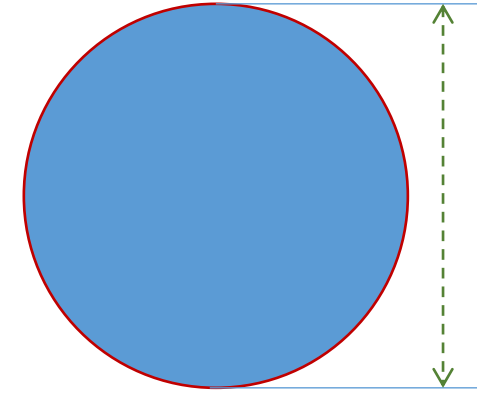
d

Circumference

$$C = \pi d$$

Area

$$A = \frac{\pi d^2}{4}$$



$$roundness = \frac{4 * A}{\pi major^2}$$

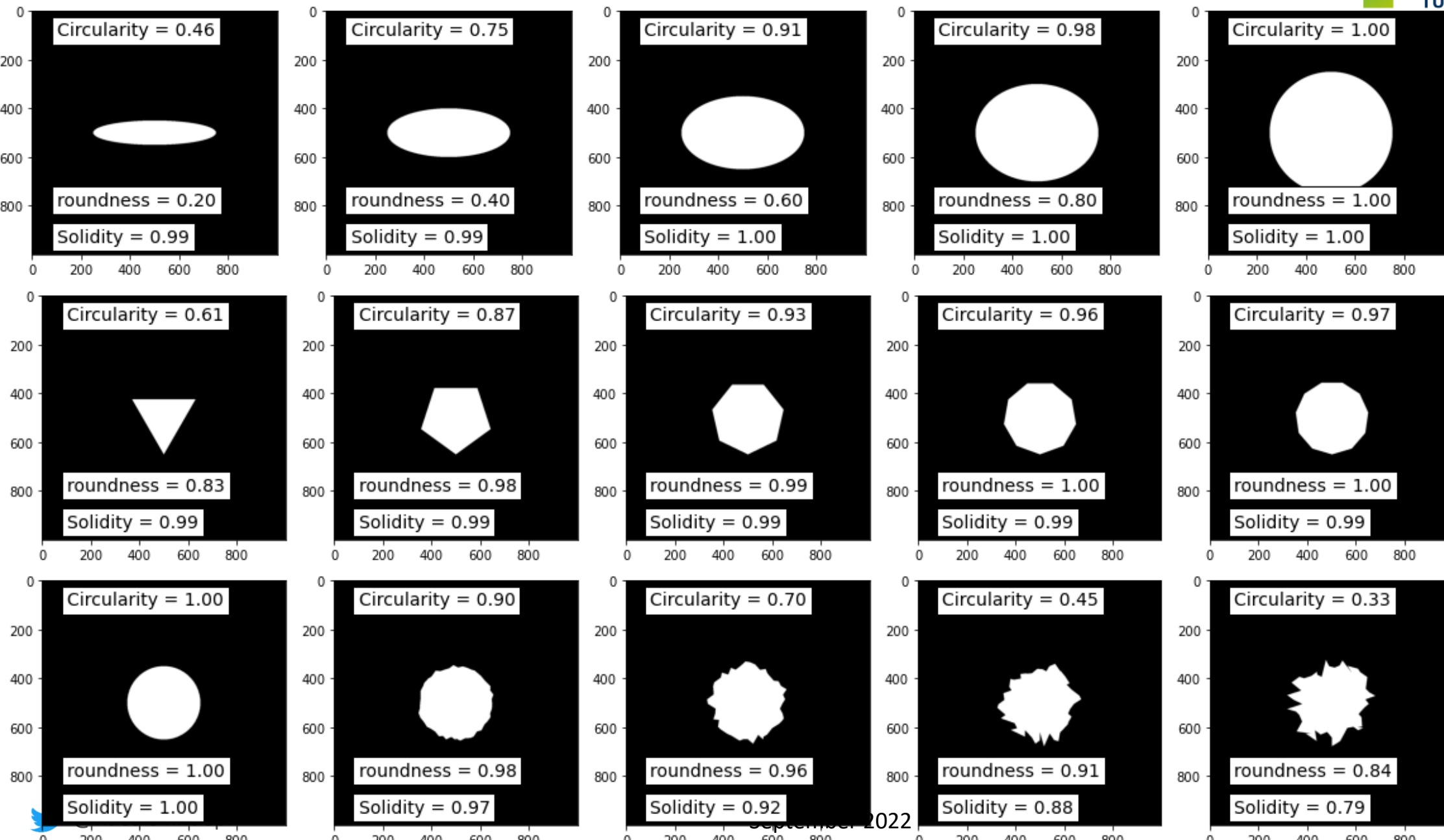
$$circularity = \frac{4\pi * A}{perimeter^2}$$

Roundness = 1
Circularity = 1

Roundness \approx 1
Circularity \approx 1

Roundness < 1
Circularity < 1

Roundness and circularity



$$roundness = \frac{4 * A}{\pi major^2}$$

$$circularity = \frac{4\pi * A}{perimeter^2}$$

$$solidity = \frac{A}{A_{convexHull}}$$

- In Fiji: *Analyze > Analyze Particles...*
- In Python: `from skimage import measure`

<https://scikit-image.org/docs/stable/api/skimimage.measure.html>

`skimage.measure.blur_effect` (image[, h_size, ...]) Compute a metric that indicates the strength of blur in an image (0 for no blur, 1 for maximal blur).

`skimage.measure.euler_number` (image[, ...]) Calculate the Euler characteristic in binary image.

`skimage.measure.find_contours` (image[, ...]) Find iso-valued contours in a 2D array for a given level value.

`skimage.measure.grid_points_in_poly` (shape, verts) Test whether points on a specified grid are inside a polygon.

`skimage.measure.inertia_tensor` (image[, mu]) Compute the inertia tensor of the input image.

`skimage.measure.inertia_tensor_eigvals` (image) Compute the eigenvalues of the inertia tensor of the image.

`skimage.measure.label` (label_image[, ...]) Label connected regions of an integer array.

`skimage.measure.regionprops` (label_image[, ...]) Measure properties of labeled image regions.

`skimage.measure.regionprops_table` (label_image) Compute image properties and return them as a pandas-compatible table.

area : int

Number of pixels of the region.

area_bbox : int

Number of pixels of bounding box.

area_convex : int

Number of pixels of convex hull image, which is the smallest convex polygon that encloses the region.

area_filled : int

Number of pixels of the region with all the holes filled in. Describes the area of the region.

axis_major_length : float

The length of the major axis of the ellipse that has the same normalized second central moments as the region.

axis_minor_length : float

The length of the minor axis of the ellipse that has the same normalized second central moments as the region.

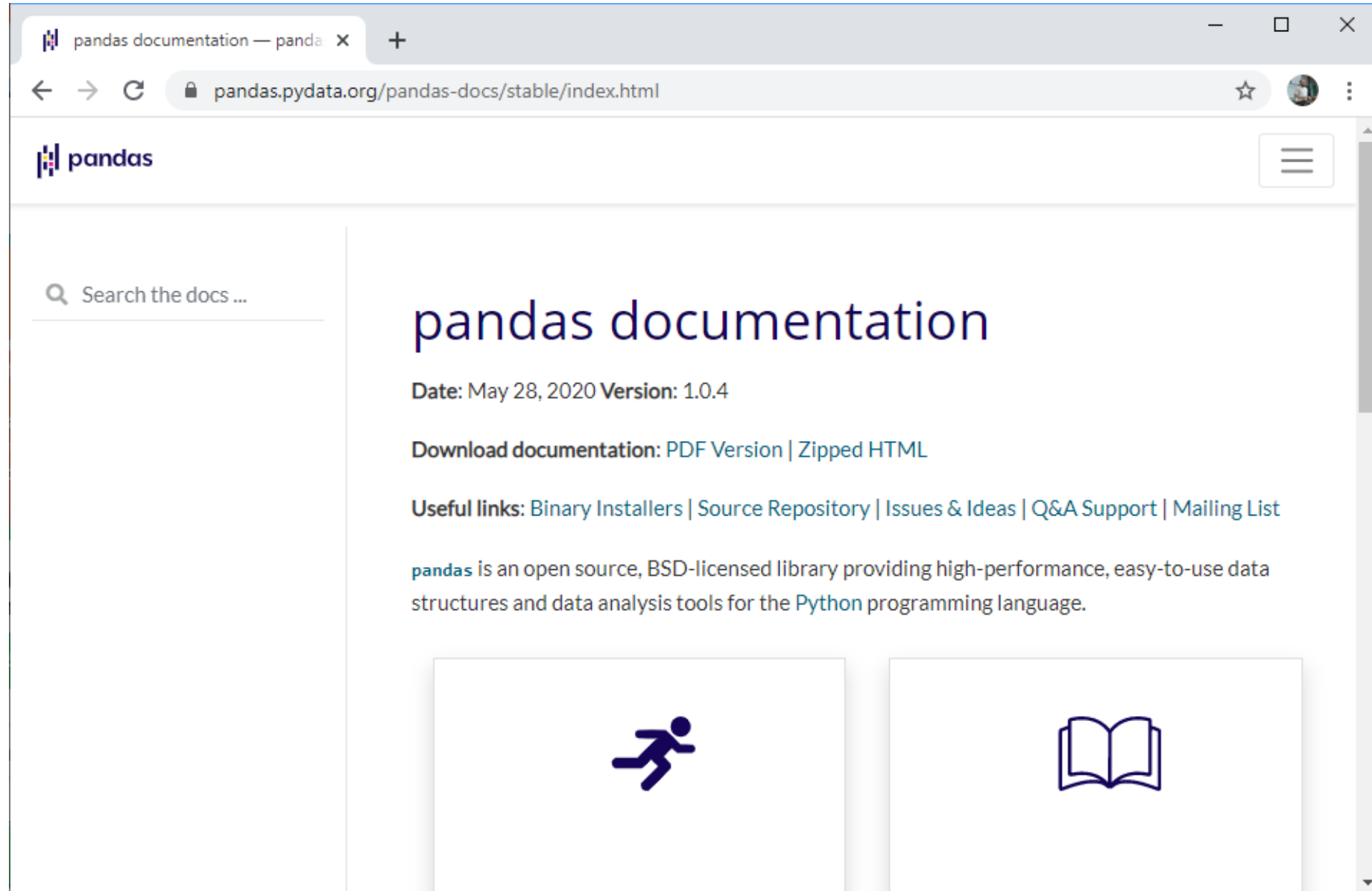
Processing tables with Python

Robert Haase

September 2022

- pandas is a library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

```
conda install pandas
```



- Typical use-case:
 - You get data from a colleague in form of a table.
 - Using pandas, you can analyze it in python.
- Loading a table in python using pandas:

```
▶ import pandas as pd  
  
data_frame = pd.read_csv("Measurements_ImageJ.csv", delimiter=',')  
data_frame
```

		Area	Mean	Circ.	AR	Round	Solidity
0	1	2610	96.920	0.773	1.289	0.776	1.0
1	2	2100	90.114	0.660	2.333	0.429	1.0
2	3	27	110.222	0.108	27.000	0.037	1.0

- Typical use-case:
 - You get data from a colleague in form of a table.
 - Using pandas, you can analyze it in python.

		Area	Mean	Circ.	AR	Round	Solidity
0	1	2610	96.920	0.773	1.289	0.776	1.0
1	2	2100	90.114	0.660	2.333	0.429	1.0
2	3	27	110.222	0.108	27.000	0.037	1.0

- Accessing a column

```
data_frame["Mean"]
```

```
0    96.920
1    90.114
2   110.222
Name: Mean, dtype: float64
```

- Determining mean of a column

```
import numpy as np
np.mean(data_frame["Mean"])

99.08533333333332
```

- Accessing an individual cell

```
data_frame["Mean"][0]

1.2890000000000001
```

- Creating tables with pandas

- Creating a new table

```
header = ['A', 'B', 'C']

data = [
    [1, 2, 3], # this will later be column A
    [4, 5, 6], #
    [7, 8, 9] #
]

# convert the data and header arrays in a pandas data frame
data_frame = pd.DataFrame(data, header)

# show it
data_frame
```

	0	1	2
A	1	2	3
B	4	5	6
C	7	8	9

- Rotate a table

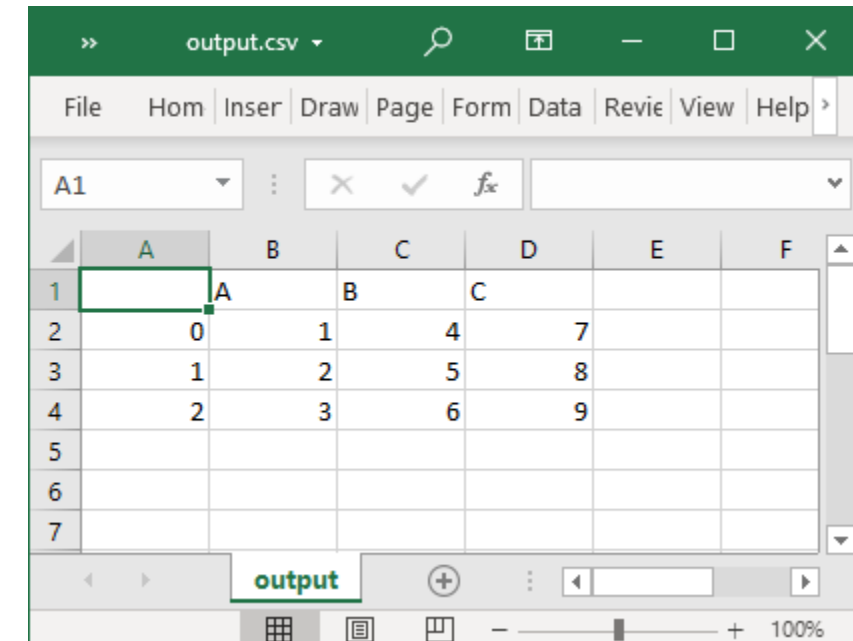
```
# rotate/flip it
data_frame = data_frame.transpose()

# show it
data_frame
```

	A	B	C
0	1	4	7
1	2	5	8
2	3	6	9

- Save it to disc

```
# save a dataframe to a CSV
data_frame.to_csv("output.csv")
```



	A	B	C	D	E	F
1						
2	0	1	4	7		
3	1	2	5	8		
4	2	3	6	9		
5						
6						
7						

Selecting rows and columns

	City	Country	Population	Area_km2
0	Tokyo	Japan	13515271	2191
1	Delhi	India	16753235	1484
2	Shanghai	China	24183000	6341
3	Sao Paulo	Brazil	12252023	1521
4	Mexico City	Mexico	9209944	1485



- Selecting columns

```
cities[['City', 'Country']]
```

	City	Country
0	Tokyo	Japan
1	Delhi	India
2	Shanghai	China
3	Sao Paulo	Brazil
4	Mexico City	Mexico



- Selecting rows

```
cities[cities['Area_km2'] > 2000]
```

	City	Country	Population	Area_km2
0	Tokyo	Japan	13515271	2191
2	Shanghai	China	24183000	6341

- The big art in data science is the ability of combining information from multiple sources to gain new insights.

	Country	Population
0	Japan	127202192
1	India	1352642280
2	China	1427647786
3	Brazil	209469323
4	Mexico	126190788

	City	Country	Population	Area_km2
0	Tokyo	Japan	13515271	2191
1	Delhi	India	16753235	1484
2	Shanghai	China	24183000	6341
3	Sao Paulo	Brazil	12252023	1521
4	Mexico City	Mexico	9209944	1485

```
combined = countries.merge(cities, on='Country', suffixes=['_country', '_city'])  
combined
```

	Country	Population_country	City	Population_city	Area_km2
0	Japan	127202192	Tokyo	13515271	2191
1	India	1352642280	Delhi	16753235	1484
2	China	1427647786	Shanghai	24183000	6341
3	Brazil	209469323	Sao Paulo	12252023	1521
4	Mexico	126190788	Mexico City	9209944	1485

```
# compute ratio  
combined['City_Country_population_ratio'] = combined['Population_city'] / combined['Population_country']  
  
# only show selected columns  
combined[['City', 'City_Country_population_ratio']]
```

	City	City_Country_population_ratio
0	Tokyo	0.106250
1	Delhi	0.012386
2	Shanghai	0.016939
3	Sao Paulo	0.058491
4	Mexico City	0.072984