ScaDS.AI
DRESDEN LEIPZIG
CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE

Data Visualizatinn

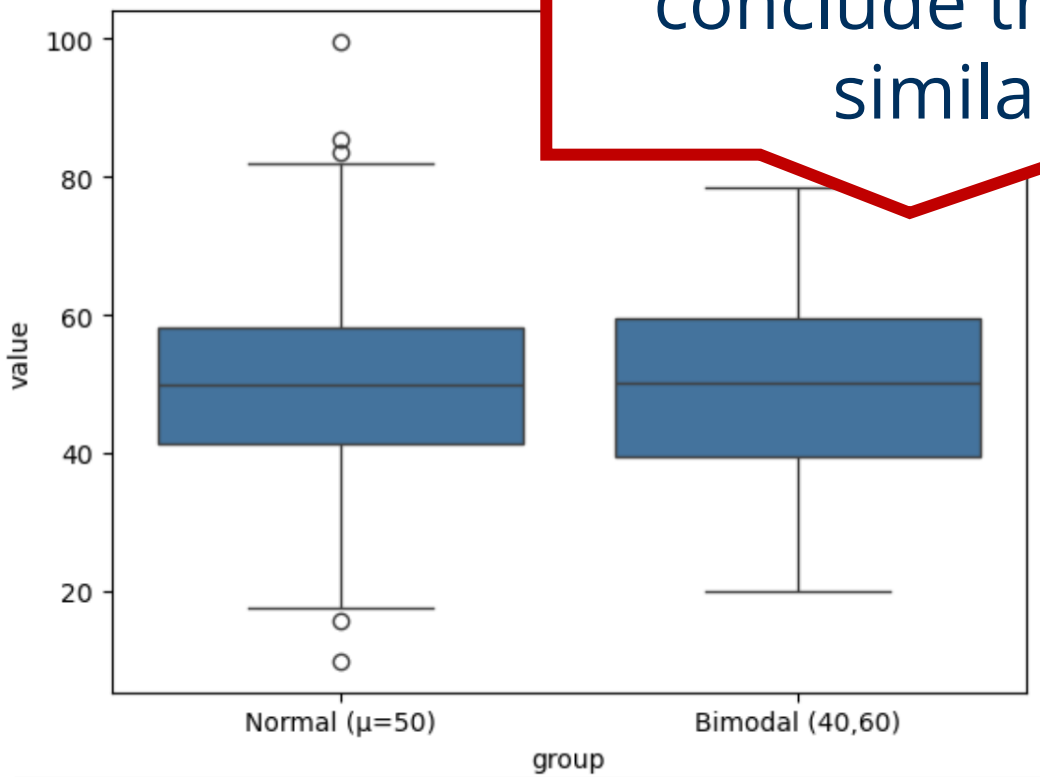
Robert Haase

Reusing Materials from Jan Ewald and Mara Lampert (ScaDS.AI, Uni Leipzig)

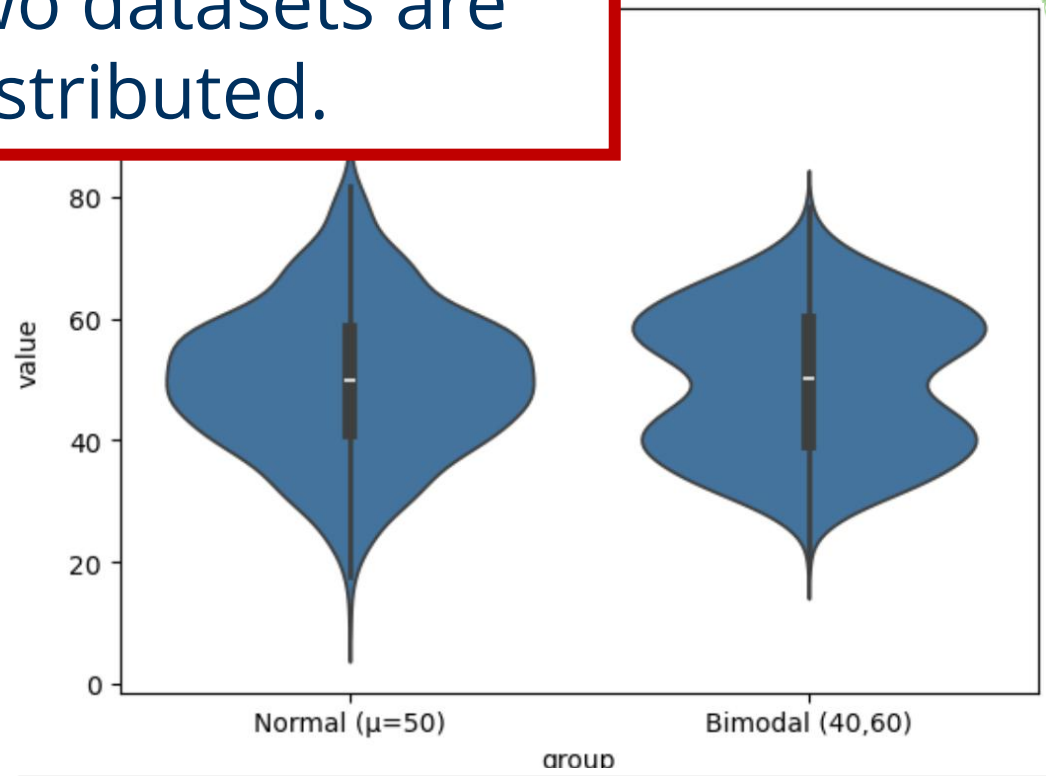
Funded by:
Bundesministerium für Forschung, Technologie und Raumfahrt
SACHSEN
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages. Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

Take home message: choose plots wisely

From such a visualization you may conclude that two datasets are similarly distributed.

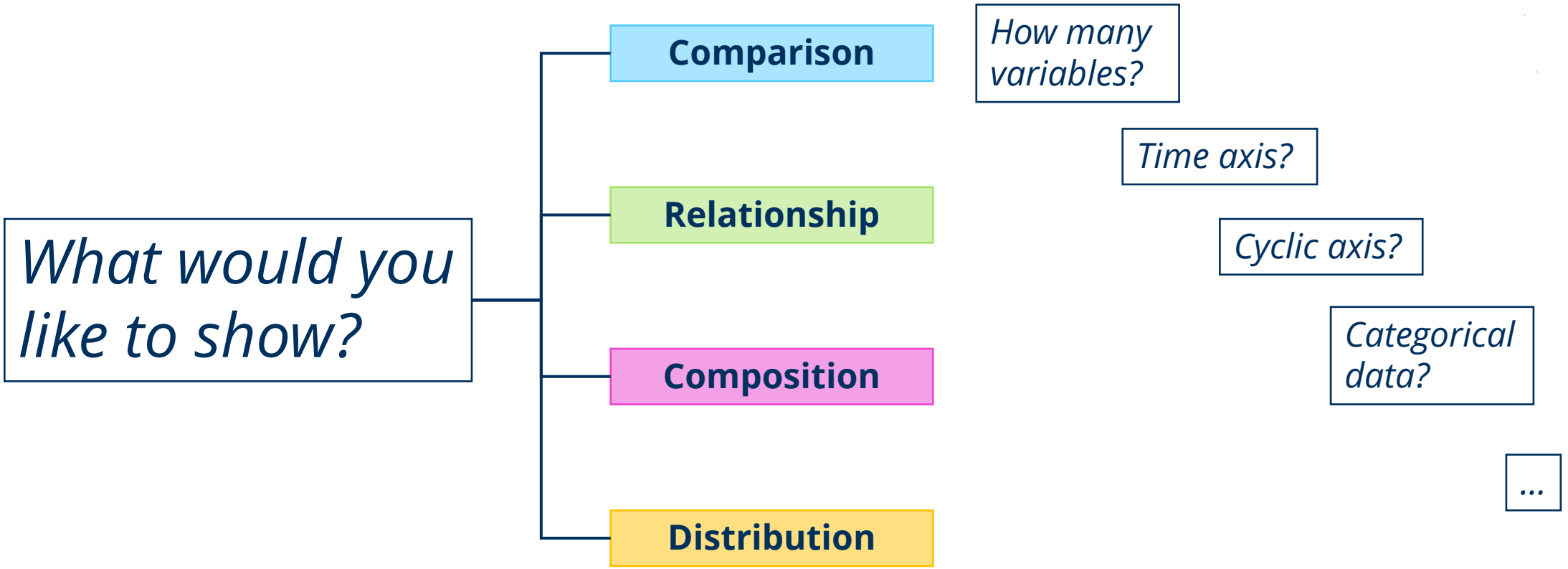



```
sns.boxplot(data=df, x="group", y="value")
```




```
sns.violinplot(data=df, x="group", y="value")
```

First things first.

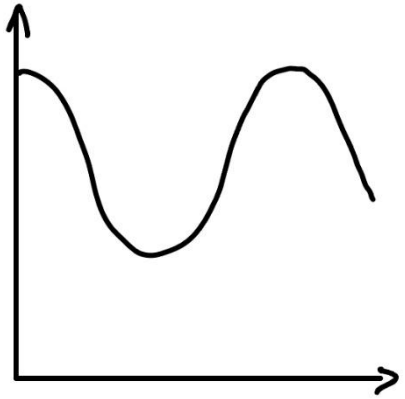


Distribution of data

Distribution

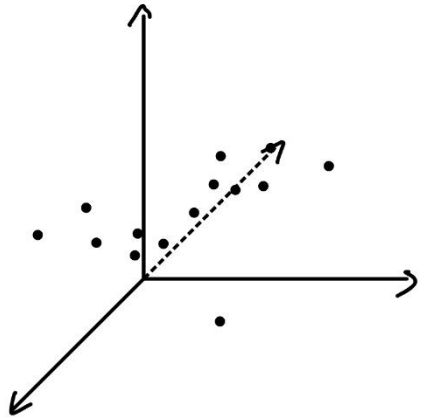
- **Single Variable**
  - *Few Data Points*
  - *Many Data Points*
- **Two Variables**
- **Three Variables**

Histogram



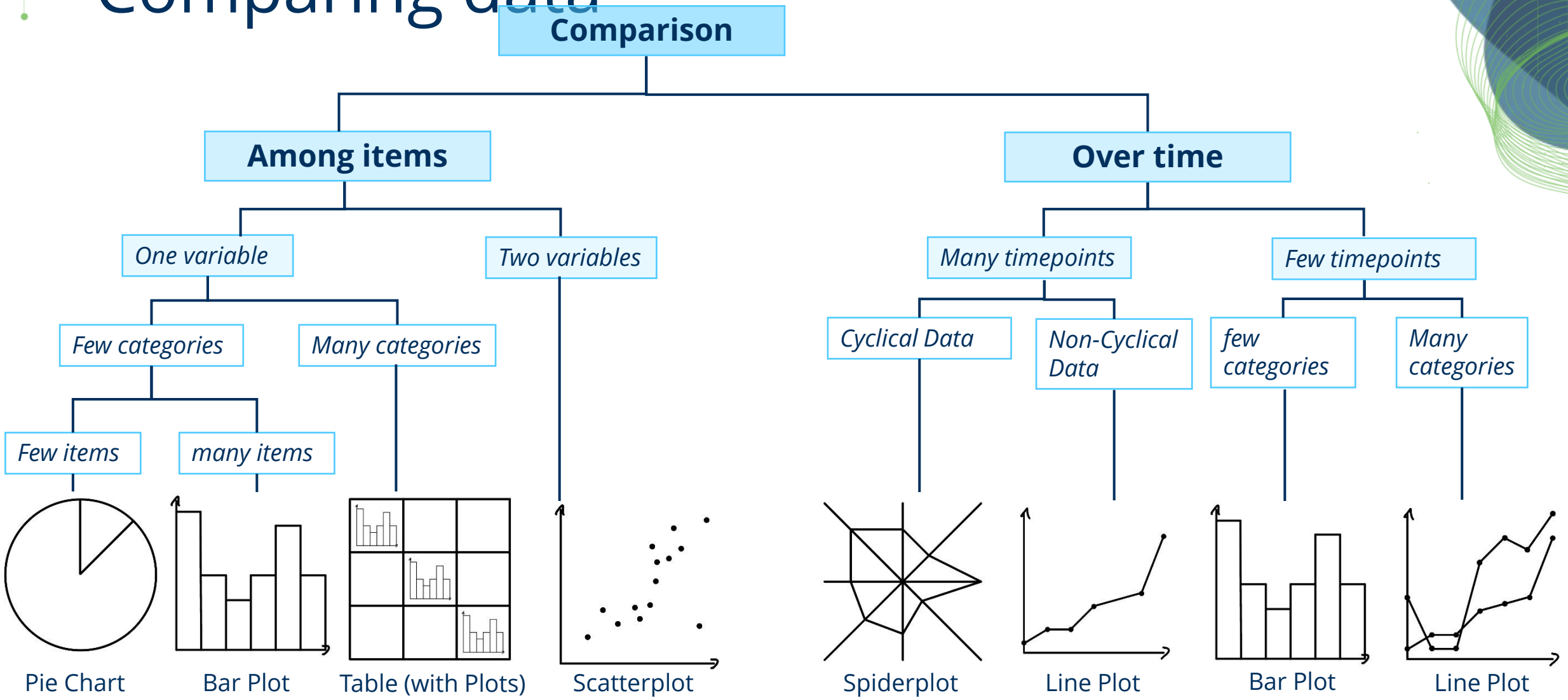
Line Plot

Scatterplot



3D Plot (e.g. 3D Scatterplot)

Comparing data



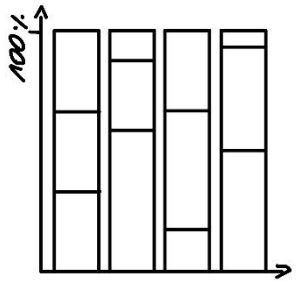
Composition of data

Composition

Changing Over Time

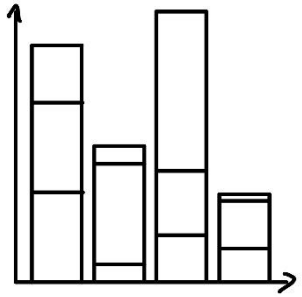
Few Periods

Only Relative Differences Matter



Stacked 100% Column Chart

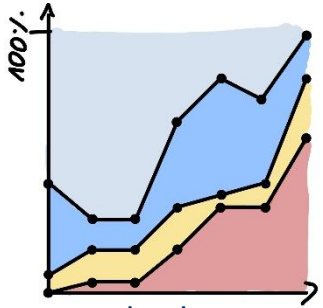
Relative and Absolute Differences Matter



Stacked Column Chart

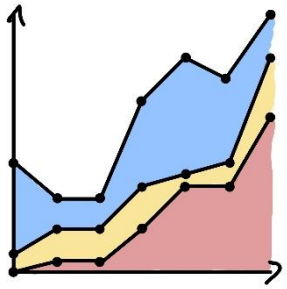
Many Periods

Only Relative Differences Matter



Stacked 100% Area Chart

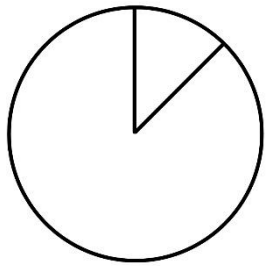
Relative and Absolute Differences Matter



Stacked Area Chart

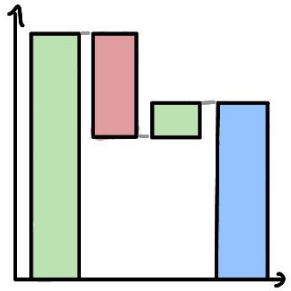
Static

Simple Share of Total



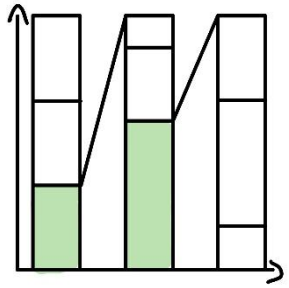
Pie Chart

Accumulation or Subtraction to Total



Waterflow Chart

Components of Components

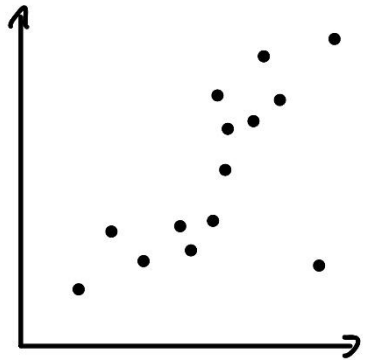


Stacked 100% Column Chart with Subcomponents

Relationships between aspects of data

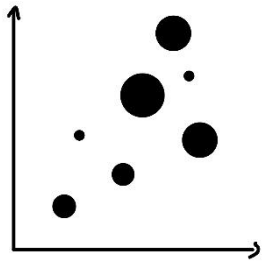
Relationship

Two Variables

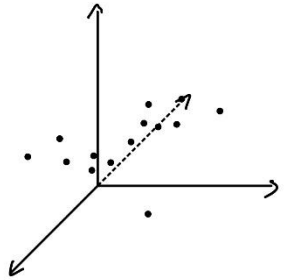


Scatterplot

Three Variables

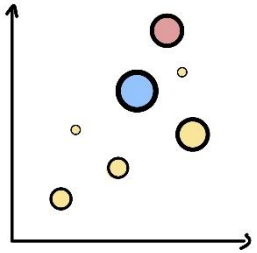


Scatterplot with 1 additional attribute (e.g. Bubble Plot)

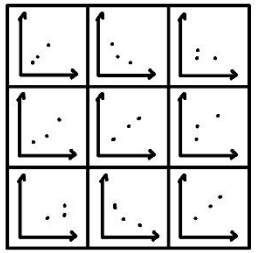


3D Scatterplot

More Variables



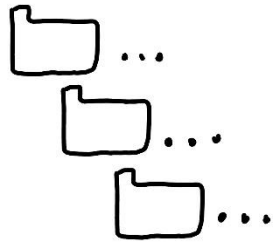
Scatterplot with additional attributes



Scatterplot Matrix

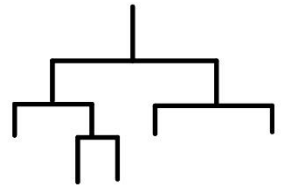
Tree Visualizations

Indentation



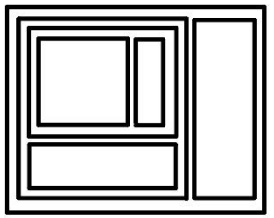
Folder Structure

Node-Link-Diagram



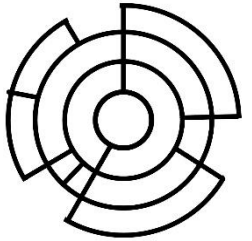
Reingold- and Tilford

Space-Filling Approaches



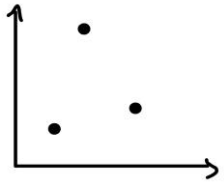
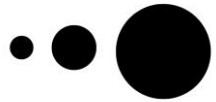
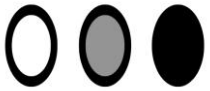
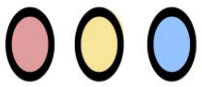
TreeMap

Layered Approaches



Sunburst

Basic elements of data visualization

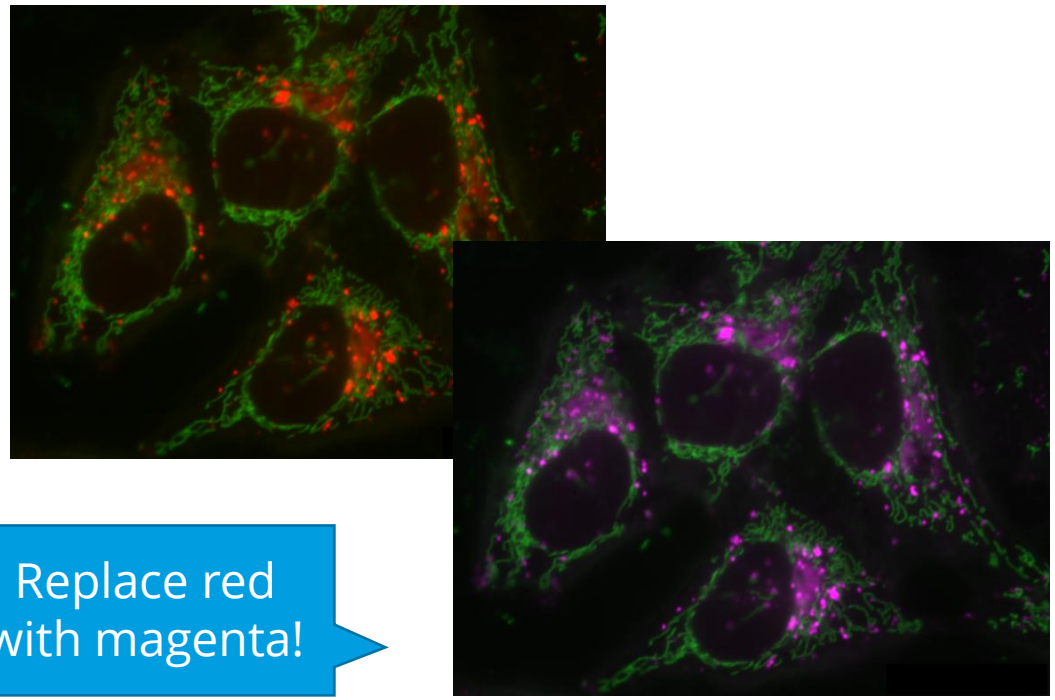
			Characteristics			
		Selective	Associative	Quantitative	Order	Length
Visual Variables	Position		yes	yes	yes	yes	infinite
	Size		yes	no	partially	yes	Selection: ~ 5 Distinction: ~ 20
	Shape		no	mostly	no	no	Infinite
	Value		yes	no	no	yes	Selection: < 7 Distinction: ~ 10
	Color		yes	yes	no	no	Selection: < 7 Distinction: ~ 10
	Orientation		yes	yes	no	no	infinite
	Texture		yes	yes	no	multi	infinite

Color maps / lookup tables

Choose visualization of your color tables wisely!

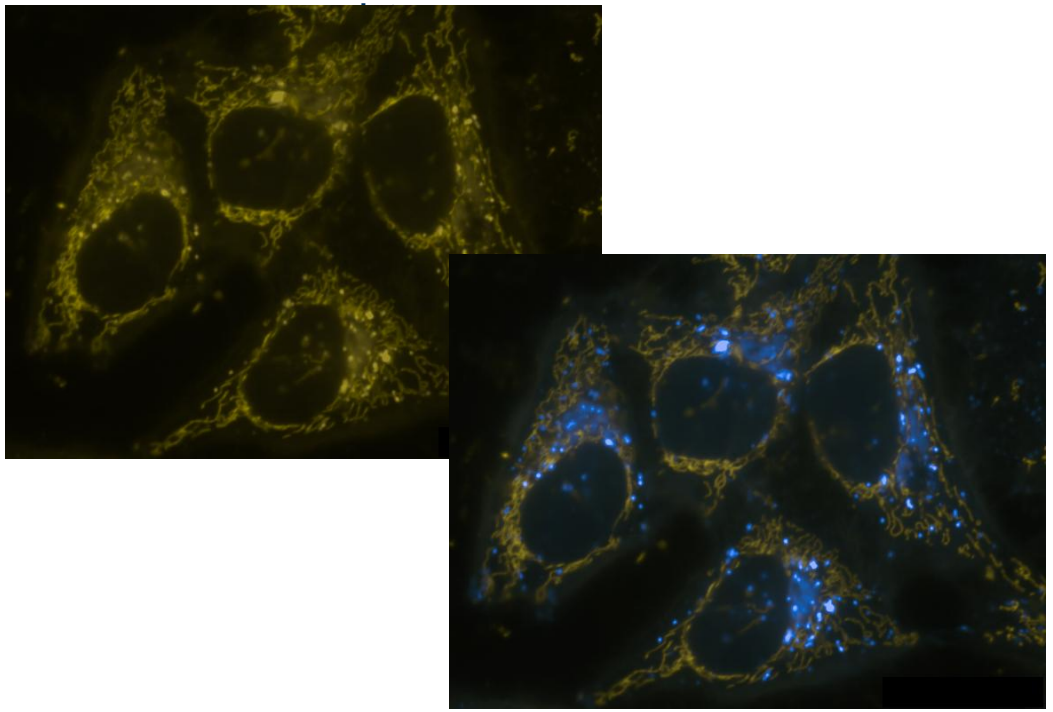
Think of people with red/green blindness!

Default view



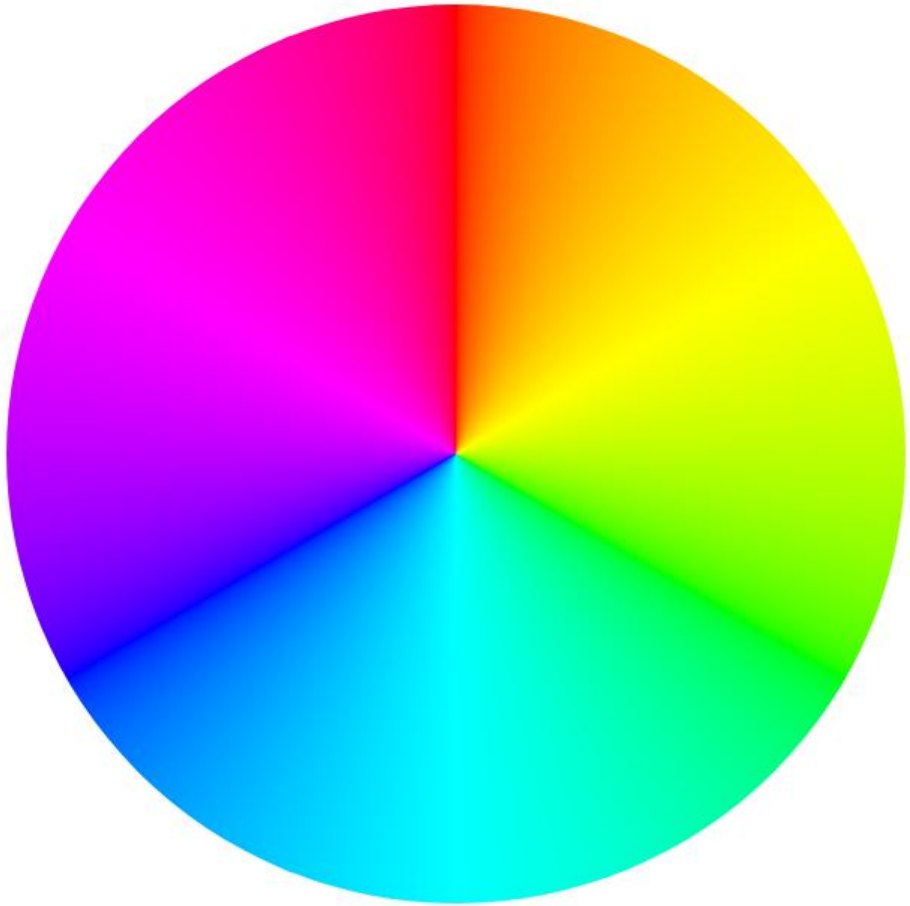
Replace red with magenta!

Red/green blind people see it like



Colour theory

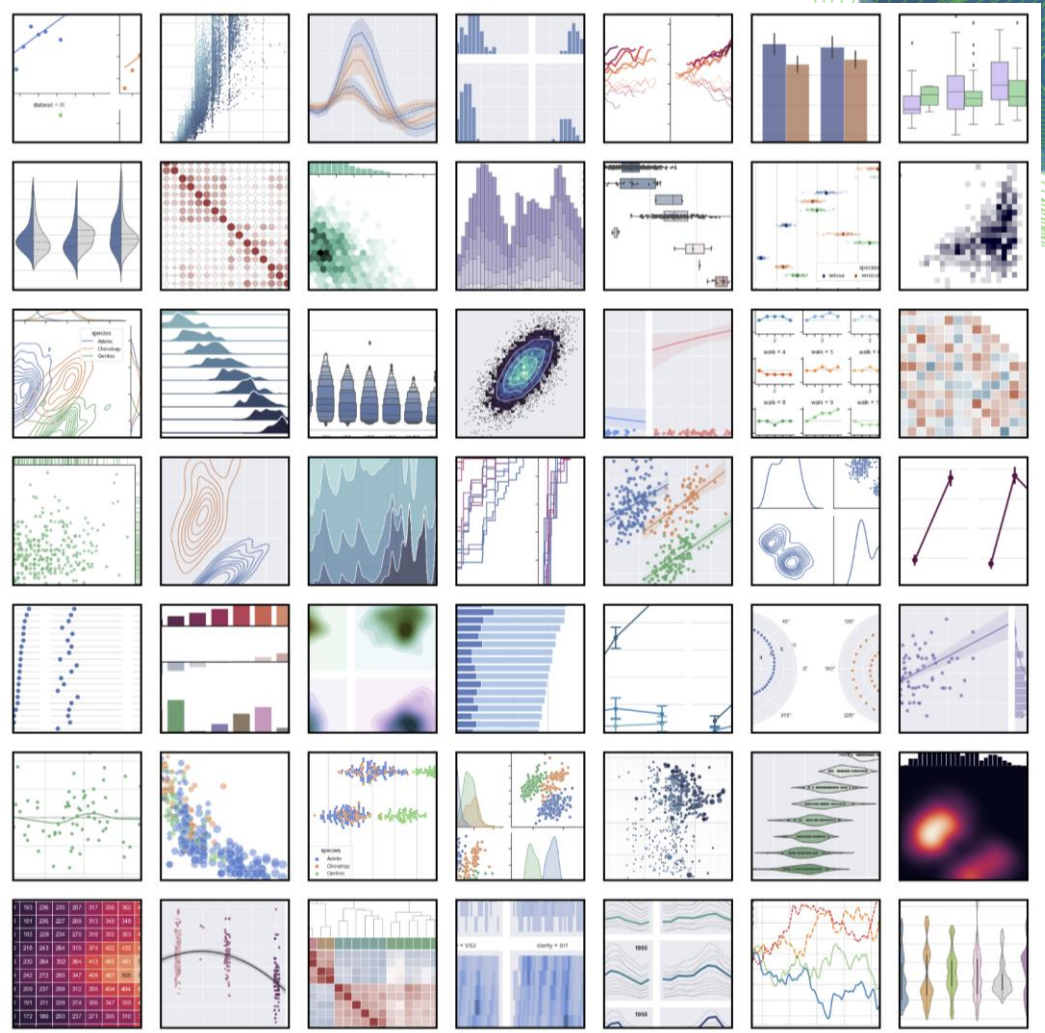
- Humans may not see the difference between adjacent colors properly
- Use opposite colours instead.



Data visualization using seaborn

Seaborn is a community standard library for advanced data visualization, based on matplotlib.

We strongly discourage data scientists from using plain matplotlib.



https://seaborn.pydata.org/tutorial/introduction.html

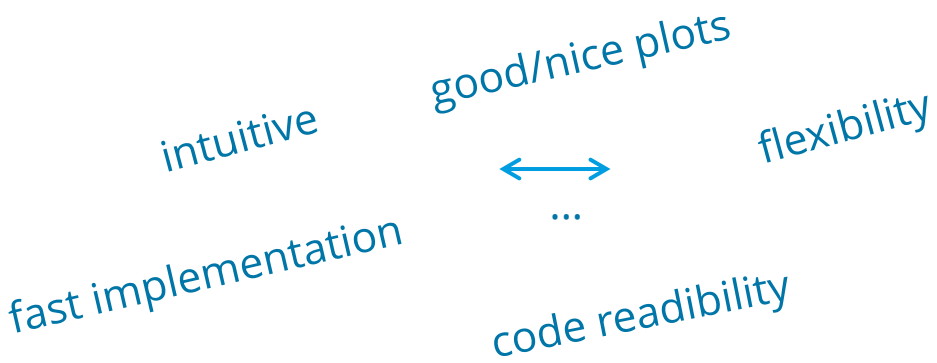
https://seaborn.pydata.org/examples/index.html

API comparison & philosophies

matplotlib

seaborn

seaborn.objects



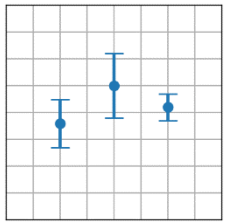
Plot type driven ⟷ **Data and geometry driven**

API comparison & philosophies

Data:

	a	*b*

Plot:



matplotlib

```
x=data['a']
y=data['b']
yerr = sd(y)

errorbar(x,y, yerr)
```

1) Choose plot type
2) Extract or calculate variables
3) Stuff into plot API

seaborn

```
pointplot(
        data,
        x='a', y='b',
        errorbar='sd'
        )
```

1) Choose basic type
2) Define data and variables

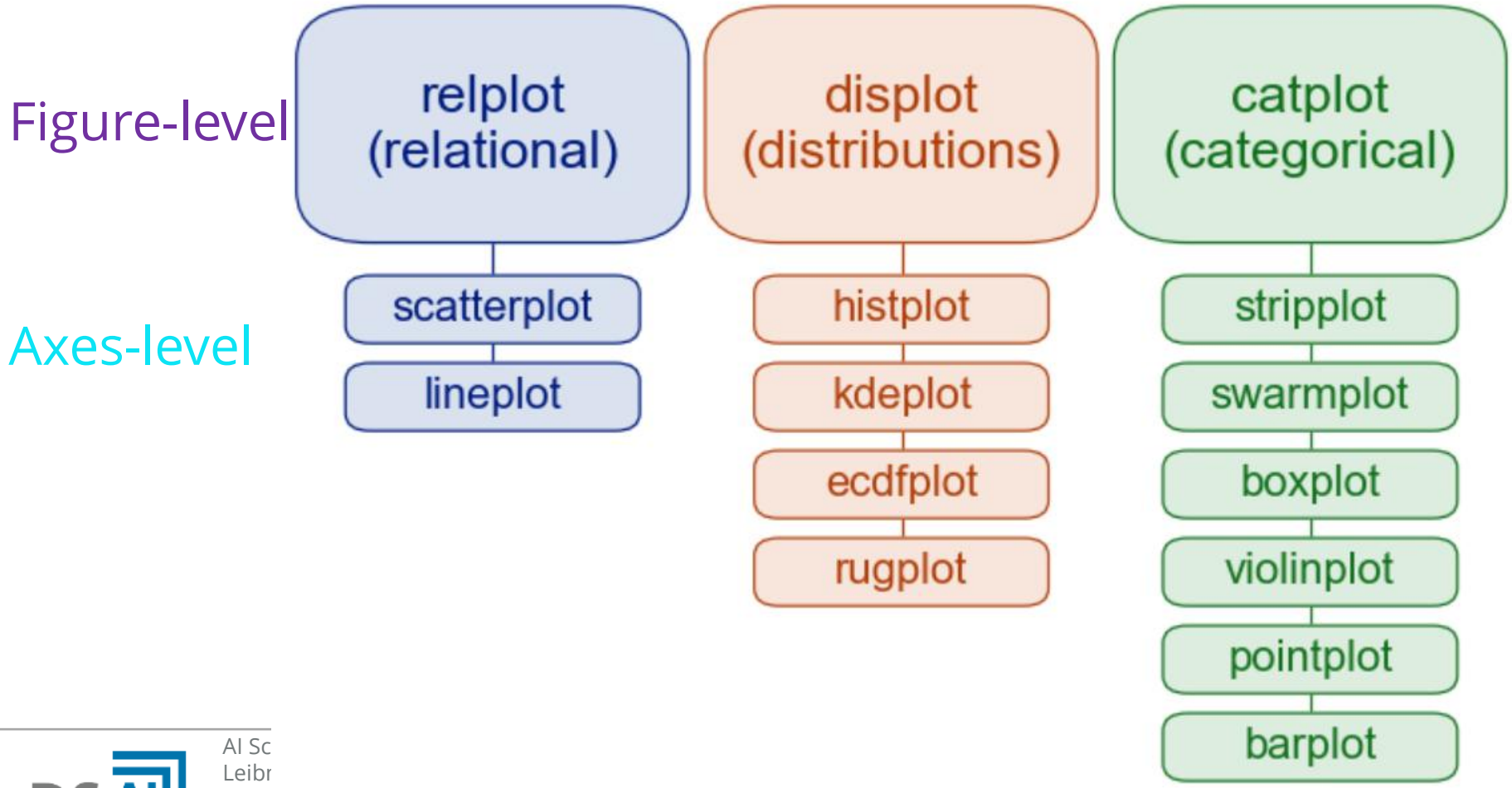
seaborn.objects

```
so.Plot(data,x='a', y='b')
 .add(  so.Dot(),
        so.Agg())
 .add(  so.Range(),
        so.Est(errorbar='sd'))
```

1) Define data and variables
2) Choose plot geometries
3) Define statistics

Plot type driven ⟷ **Data and geometry driven**

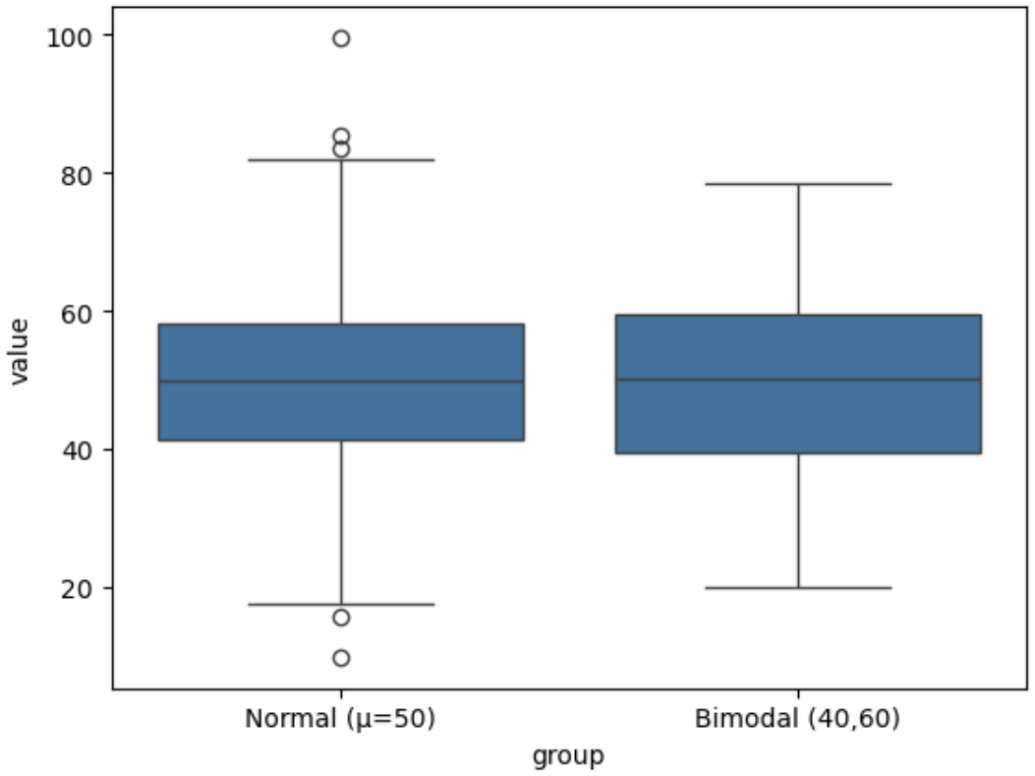
Concepts behind seaborn



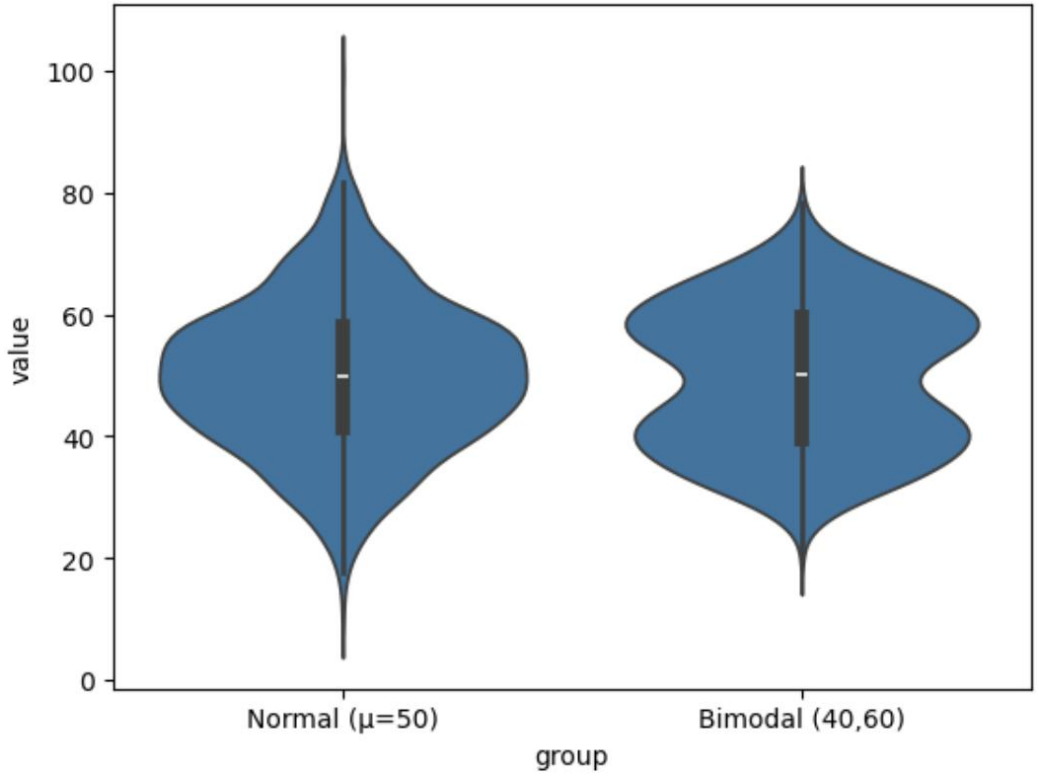
Comparison based on categories

Easy to switch plot type

```
sns.boxplot(data=df, x="group", y="value")
```




```
sns.violinplot(data=df, x="group", y="value")
```



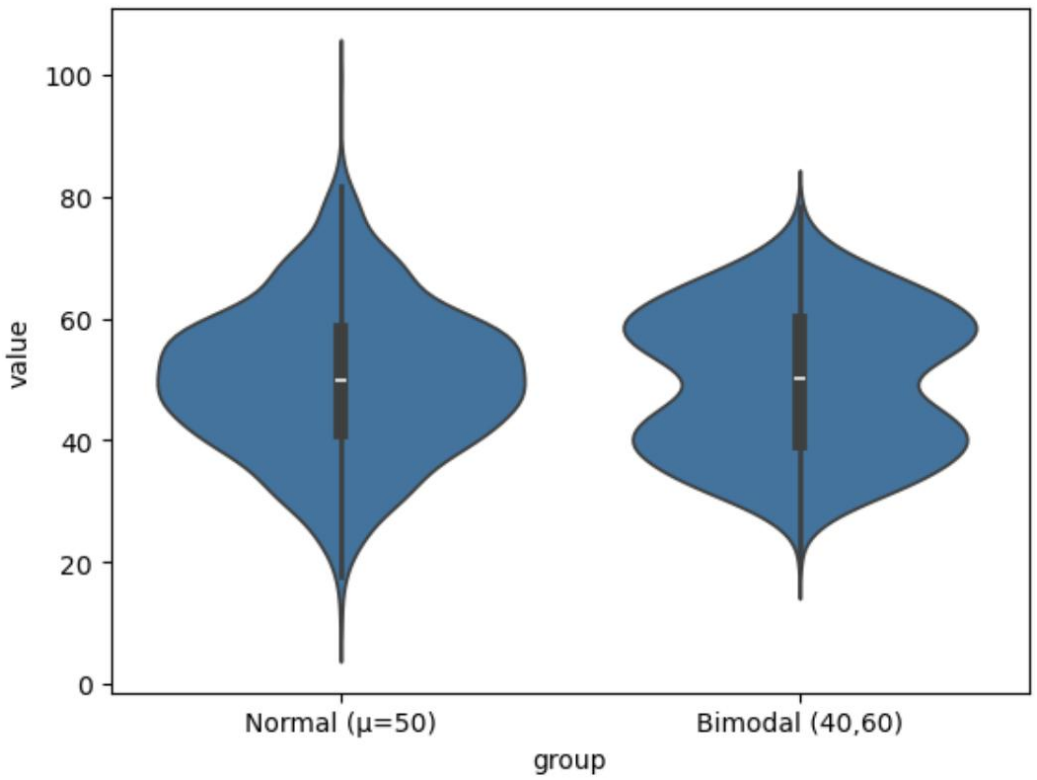
Comparison based on categories

Seaborn plotting is based on DataFrames (df)

Seaborn functions take a df and:

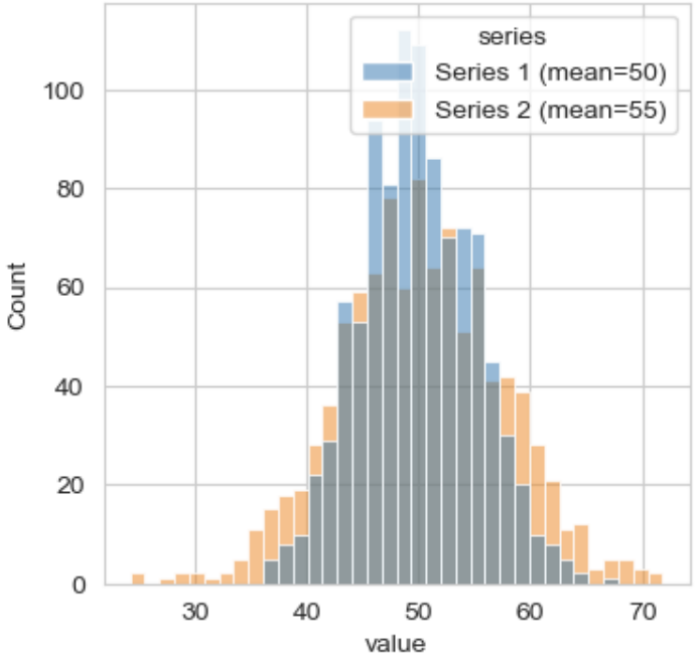
- X
- Y
- Hue
- Size
- …

```
sns.violinplot(data=df, x="group", y="value")
```

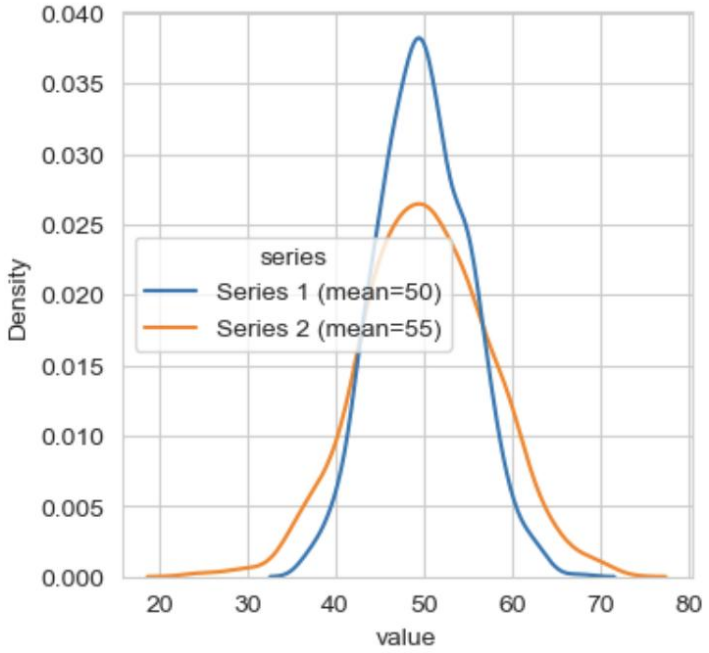


Alternative views for comparison

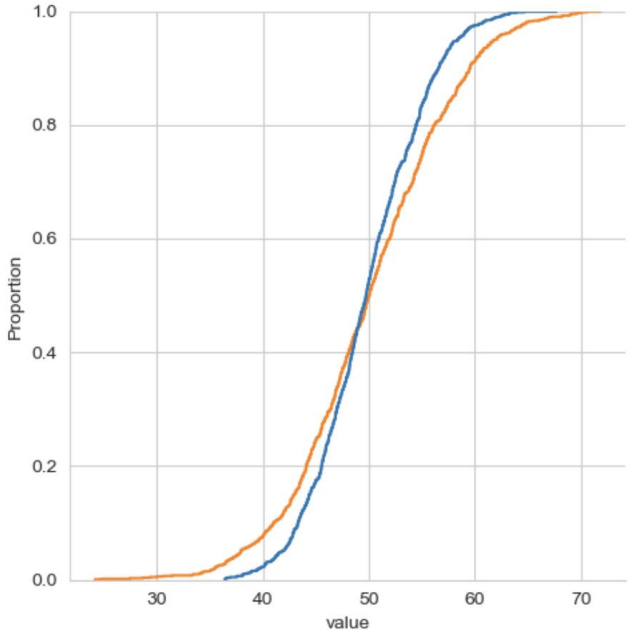
```
sns.histplot(data=data, x='value', hue='series')
```




```
sns.kdeplot(data=data, x='value', hue='series')
```

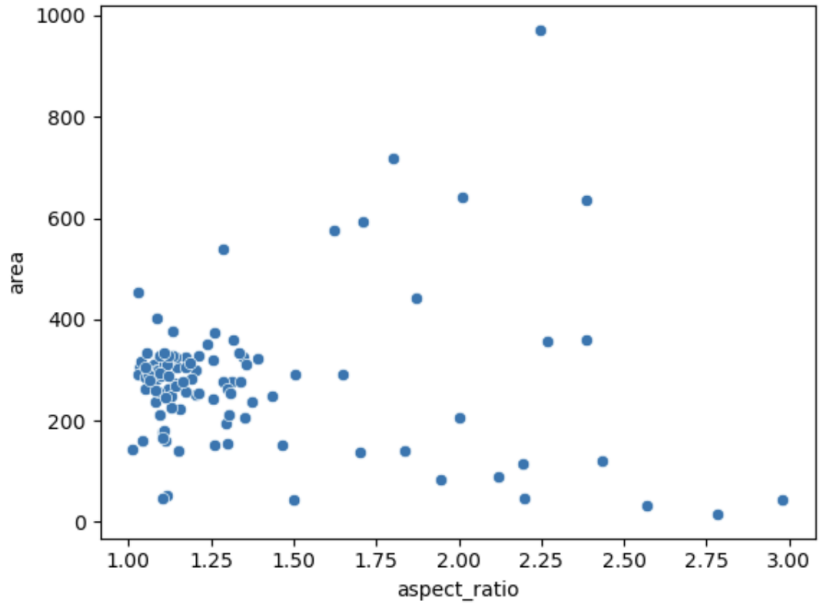



```
sns.displot(data=data, x='value',
            hue='series', kind="ecdf")
```

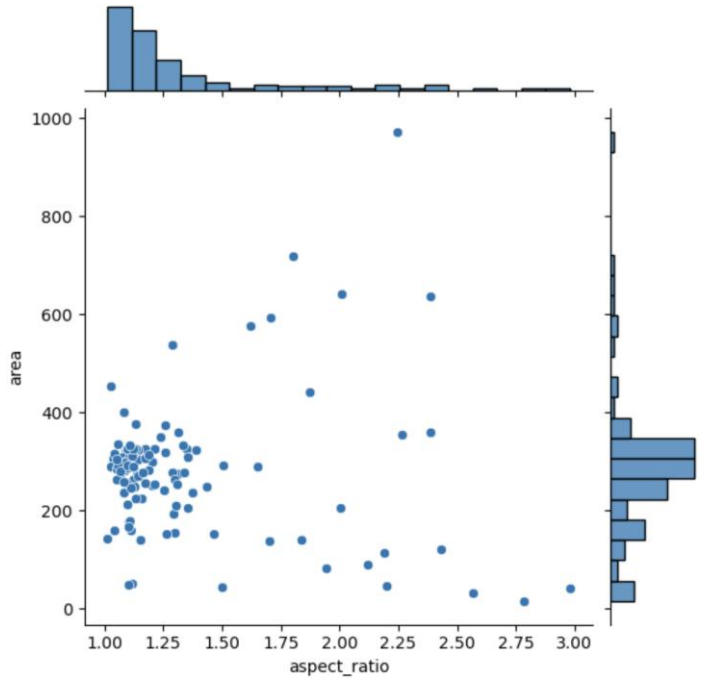


Inspection relationships

```
sns.scatterplot(data=df, x="aspect_ratio", y="area");
```




```
sns.jointplot(data=df, x="aspect_ratio", y="area");
```



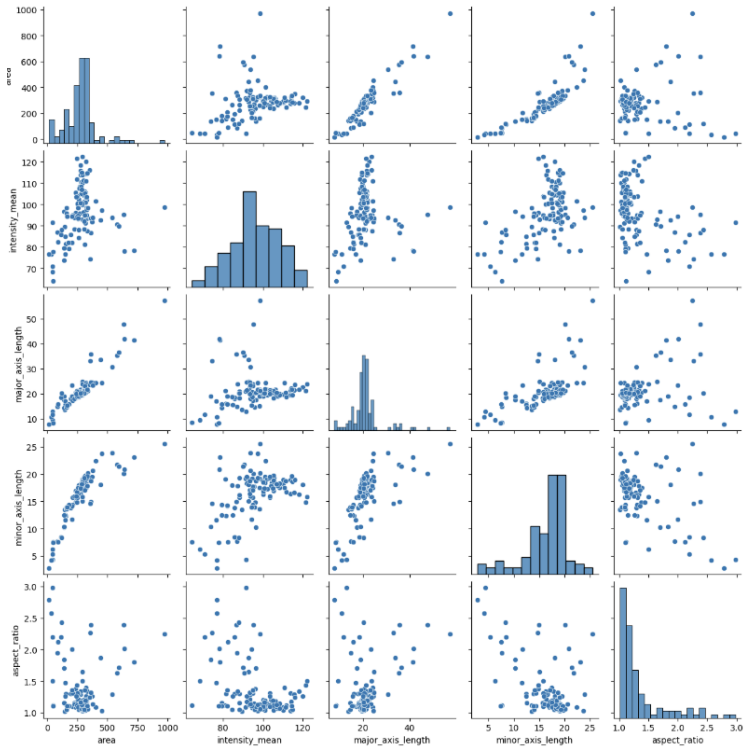
Inspection relationships

Pairplots are great for getting an overview

	area	intensity_mean	major_axis_length	minor_axis_length	aspect_ratio	file_name
0	139	96.546763	17.504104	10.292770	1.700621	20P1_POS0010_D_1UL
1	360	86.613889	35.746808	14.983124	2.385805	20P1_POS0010_D_1UL
2	43	91.488372	12.967884	4.351573	2.980045	20P1_POS0010_D_1UL
3	140	73.742857	18.940508	10.314404	1.836316	20P1_POS0010_D_1UL
4	144	89.375000	13.639308	13.458532	1.013432	20P1_POS0010_D_1UL
...	...	...	...	...	...	...
106	305	88.252459	20.226532	19.244210	1.051045	20P1_POS0007_D_1UL
107	593	89.905565	36.508370	21.365394	1.708762	20P1_POS0007_D_1UL
108	289	106.851211	20.427809	18.221452	1.121086	20P1_POS0007_D_1UL
109	277	100.664260	20.307965	17.432920	1.164920	20P1_POS0007_D_1UL
110	46	70.869565	11.648895	5.298003	2.198733	20P1_POS0007_D_1UL

111 rows × 6 columns

```
sns.pairplot(data=df);
```



Visualizing images

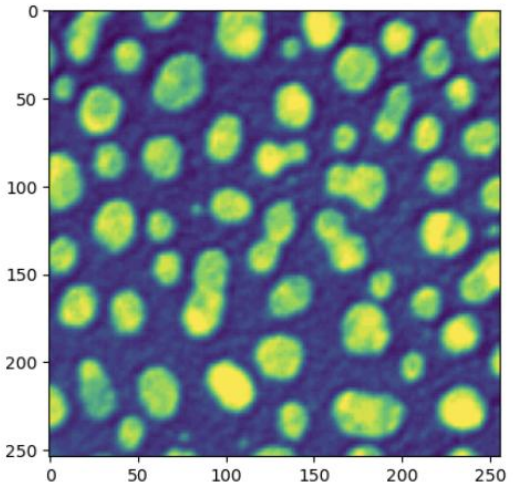
`imshow` originates from Matlab and has been implemented many times.

```
[1]: import matplotlib.pyplot as plt
     import stackview
     from skimage import io
```

matplotlib

```
[3]: plt.imshow(image)
```

```
[3]: <matplotlib.image.AxesImage at 0x1fa5b5145d0>
```

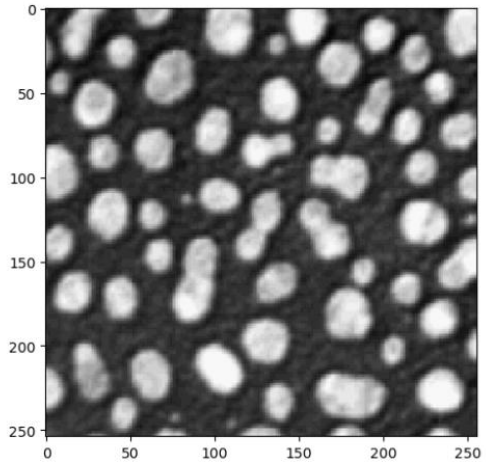


scikit-image

```
[4]: io.imshow(image)
```

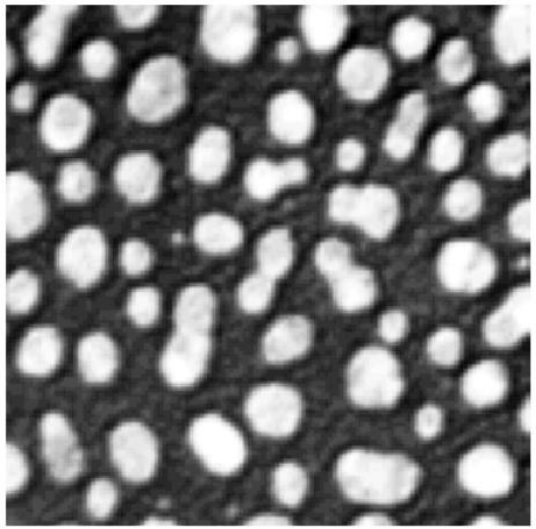
```
C:\Users\rober\AppData\Local\Temp\ipykernel_46732\2038164919.py:1: FutureWarning: `imshow` is deprecated since version 0.25 and will be removed in version 0.27. Please use `matplotlib`, `napari`, etc. to visualize images.
  io.imshow(image)
```

```
[4]: <matplotlib.image.AxesImage at 0x1fa580cbc50>
```



stackview

```
[5]: stackview.imshow(image)
```



Visualizing images

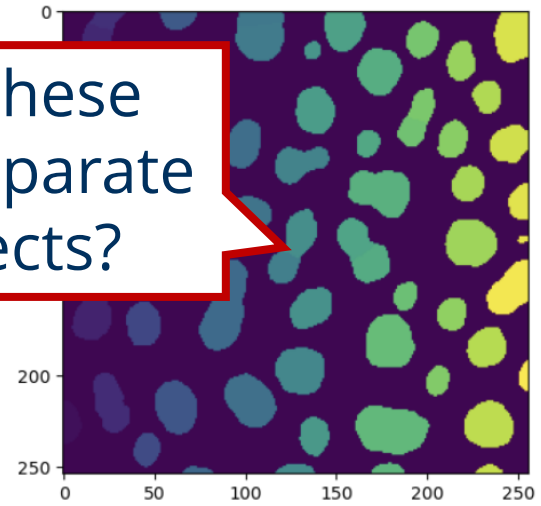
`imshow` originates from Matlab and has been implemented many times.

```
[1]: import matplotlib.pyplot as plt
     import stackview
     from skimage import io
```

matplotlib

```
[6]: plt.imshow(labels)
```

```
[6]: <matplotlib.image.AxesImage at 0x1fa5b6df210>
```



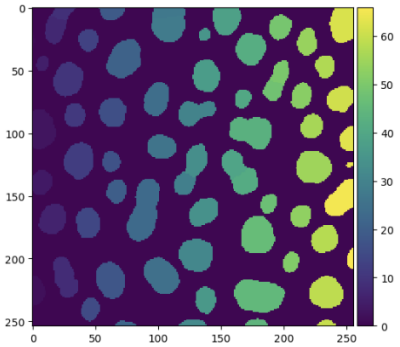
Are these two separate objects?

scikit-image

```
[7]: io.imshow(labels)
```

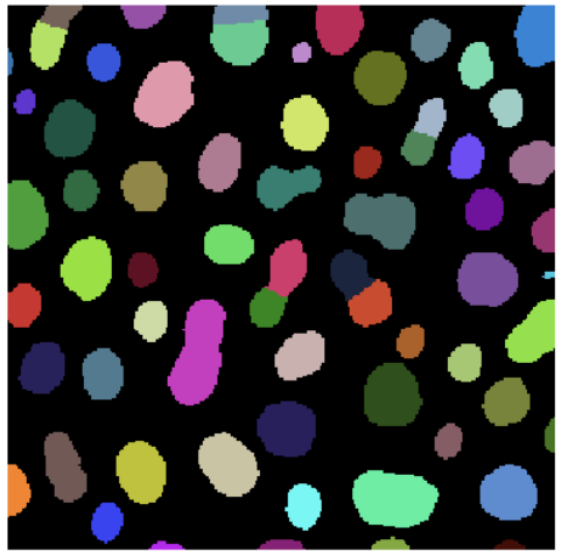
```
C:\Users\rober\AppData\Local\Temp\ipykernel_46732\28416559.py:1: FutureWarning: `imshow` is deprecated since version 0.25 and will be removed in version 0.27. Please use `matplotlib`, `napari`, etc. to visualize images.
  io.imshow(labels)
C:\Users\rober\miniforge3\envs\bob-env\Lib\site-packages\skimage\io\_plugins\matplotlib_plugin.py:158: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```

```
[7]: <matplotlib.image.AxesImage at 0x1fa5b67b410>
```



stackview

```
[8]: stackview.imshow(labels)
```

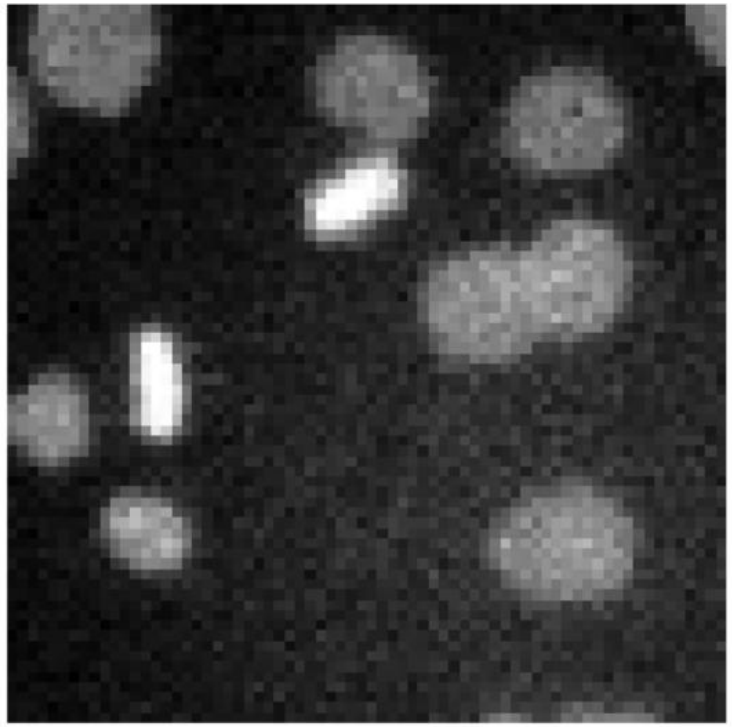


Visualizing images

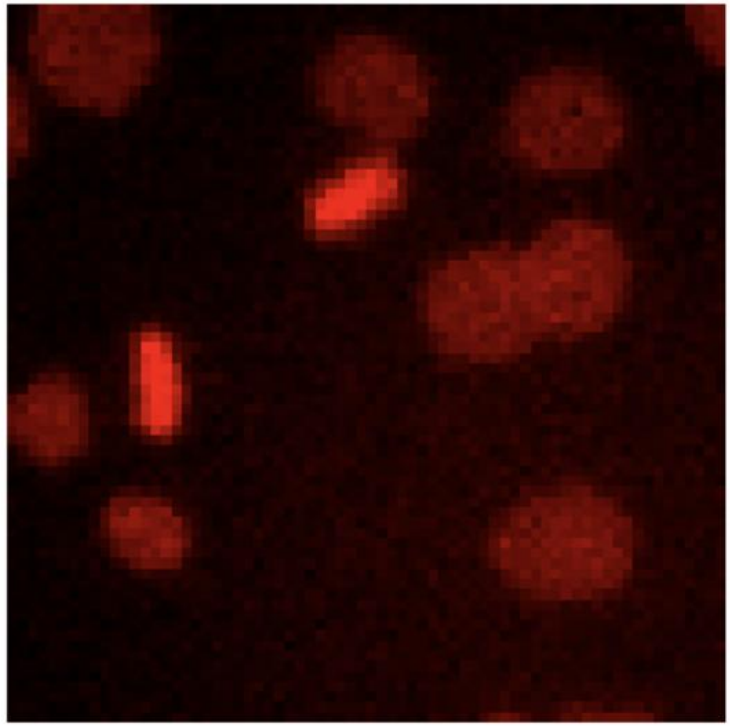
… using *stackview*

Disclosure: I maintain this

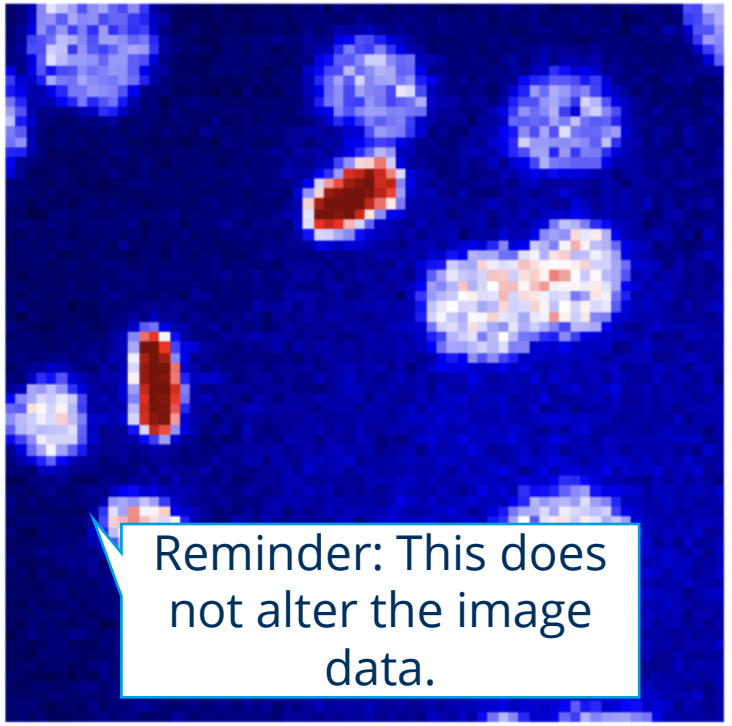
```
[3]: import stackview
     stackview.imshow(image)
```



```
[5]: stackview.imshow(image, colormap="pure_red")
```



```
[6]: stackview.imshow(image, colormap="seismic")
```



Reminder: This does not alter the image data.

https://github.com/haesleinhuepf/stackview
https://github.com/ScaDS/BIDS-lecture-2025/blob/main/02a_image_processing/02_opening_visualizing_images.ipynb

Visualizing images

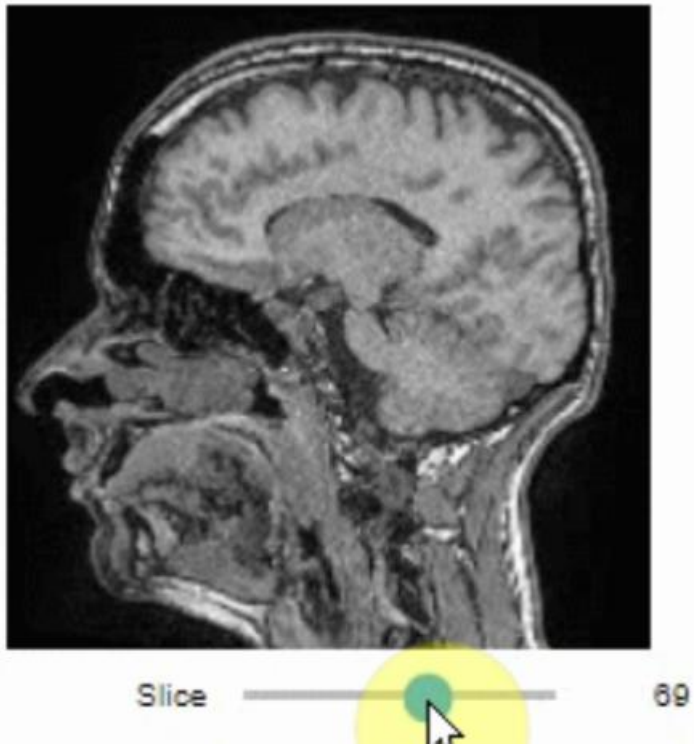
… using *stackview*

```
[6]: stackview.slice(mri_image)
```

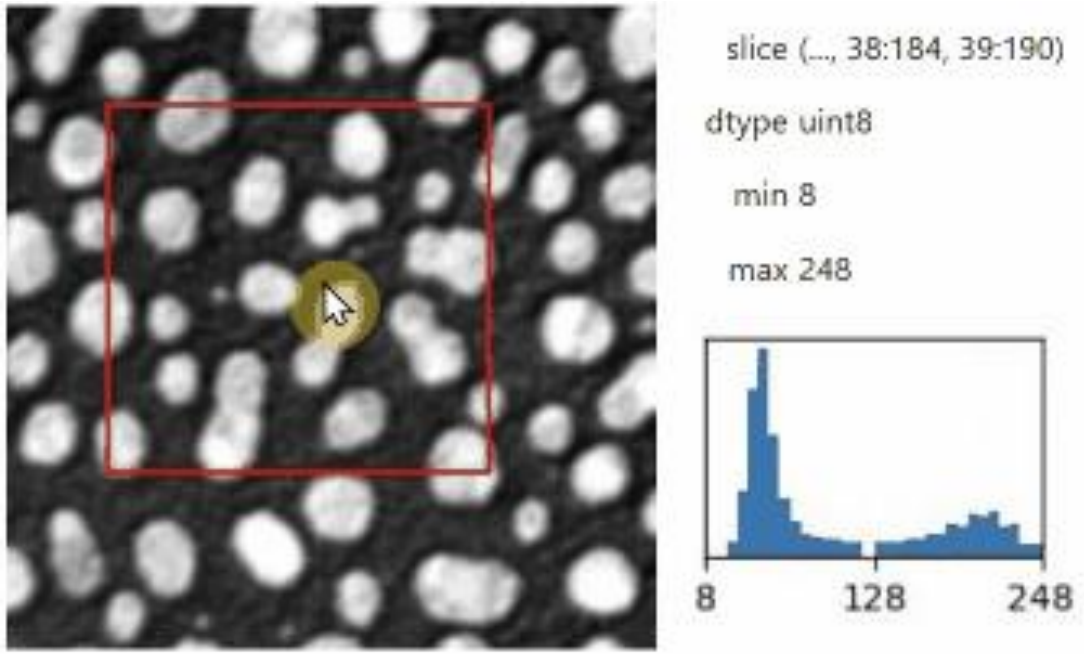
Visualizing images

… using *stackview*

```
[6]: stackview.slice(mri_image)
```




```
[8]: stackview.histogram(image)
```



Selecting objects according to their properties

Understanding what certain measurements *mean* may require interactive user interfaces

```
[6]: stackview.clusterplot(image=image,
                           labels=labeled_image,
                           df=df,
                           column_x="area",
                           column_y="aspect_ratio",
                           zoom_factor=1.6,
                           alpha=0.7)
```

[6]: