PoL
Physics of Life
TU Dresden

CENTER FOR
SYSTEMS BIOLOGY
DRESDEN

Plotting Data

Till Korten

With material from

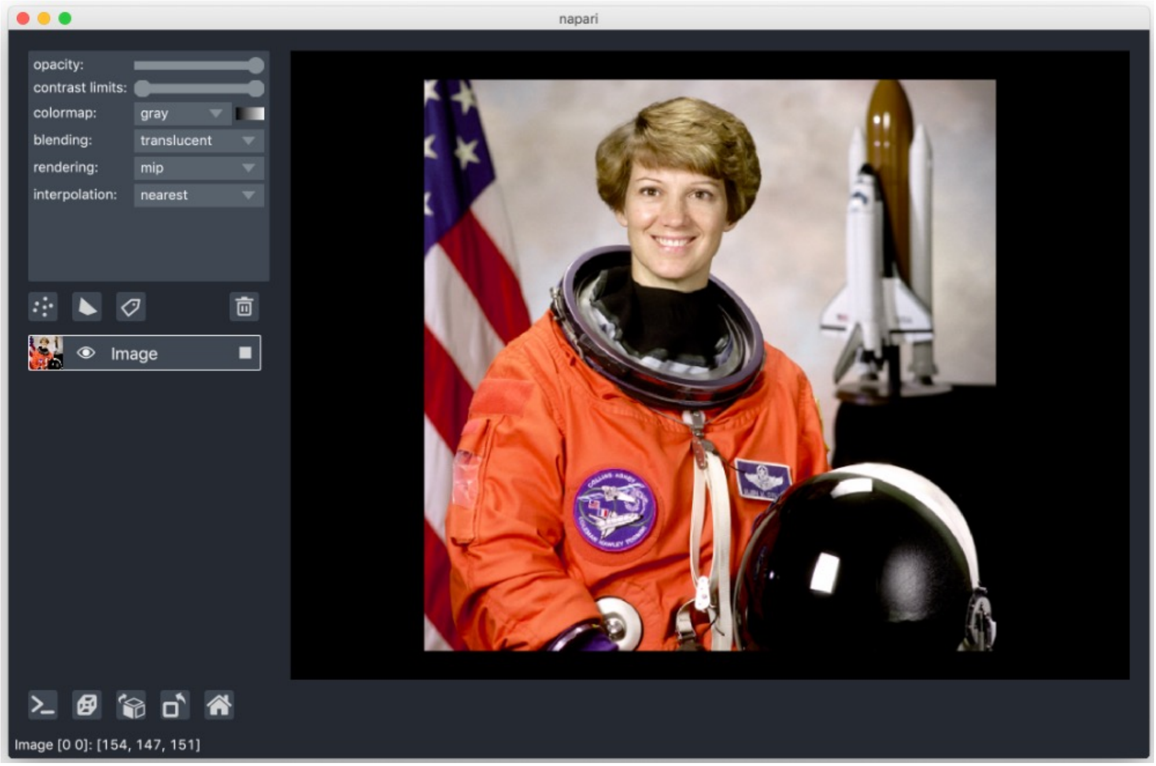
Marcelo Leomil Zoccoler Robert Haase, BiAPoL, PoL TU Dresden

@TillKorten

December 2022

Displaying images

napari

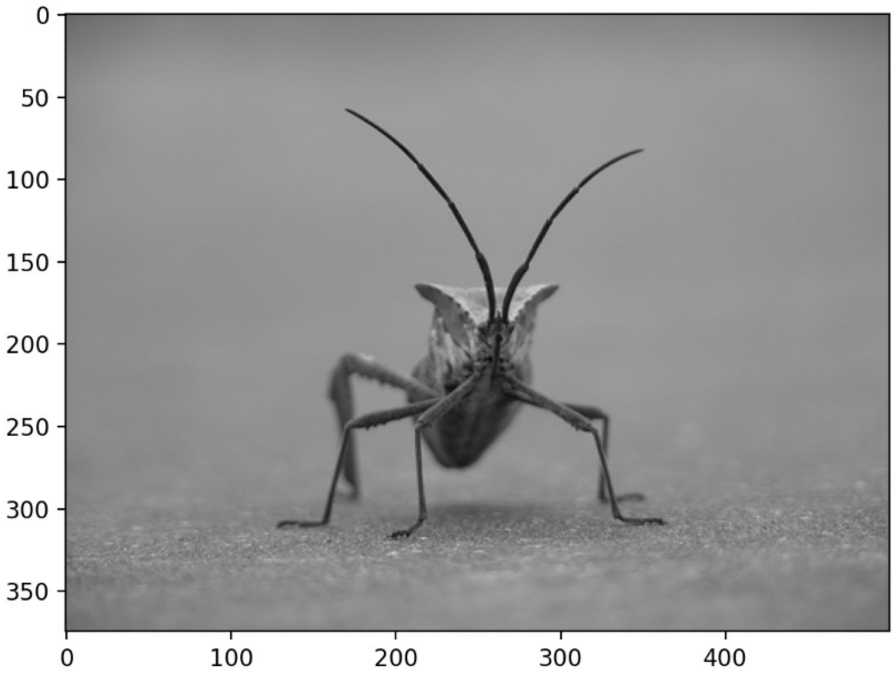


opacity:
contrast limits:
colormap: gray
blending: translucent
rendering: mip
interpolation: nearest

Image [0 0]: [154, 147, 151]

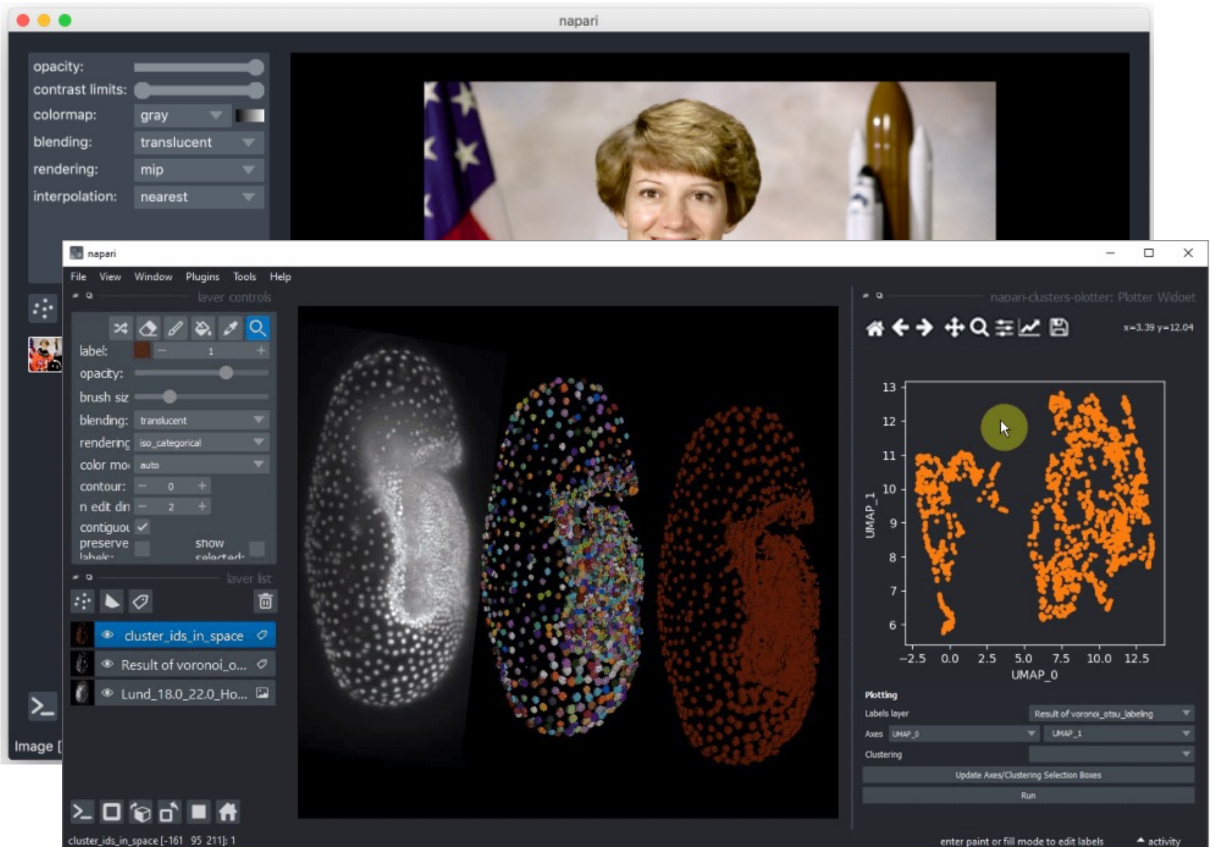
matplotlib

```
imgplot = plt.imshow(img)
```



Displaying graphs

napari



napari

opacity:

contrast limits:

colormap: gray

blending: translucent

rendering: mip

interpolation: nearest

label: 1

blending: translucent

rendering: iso_categorical

color mode: auto

contour: 0

n edit dim: 2

contiguous ✓

cluster_ids_in_space

Result of voronoi_o...

Lund_18.0_22.0_Ho...

Plotting

Labels layer: Result of voronoi_otsu_labeling

Axes: UMAP_0 UMAP_1

Clustering

Update Axes/Clustering Selection Boxes

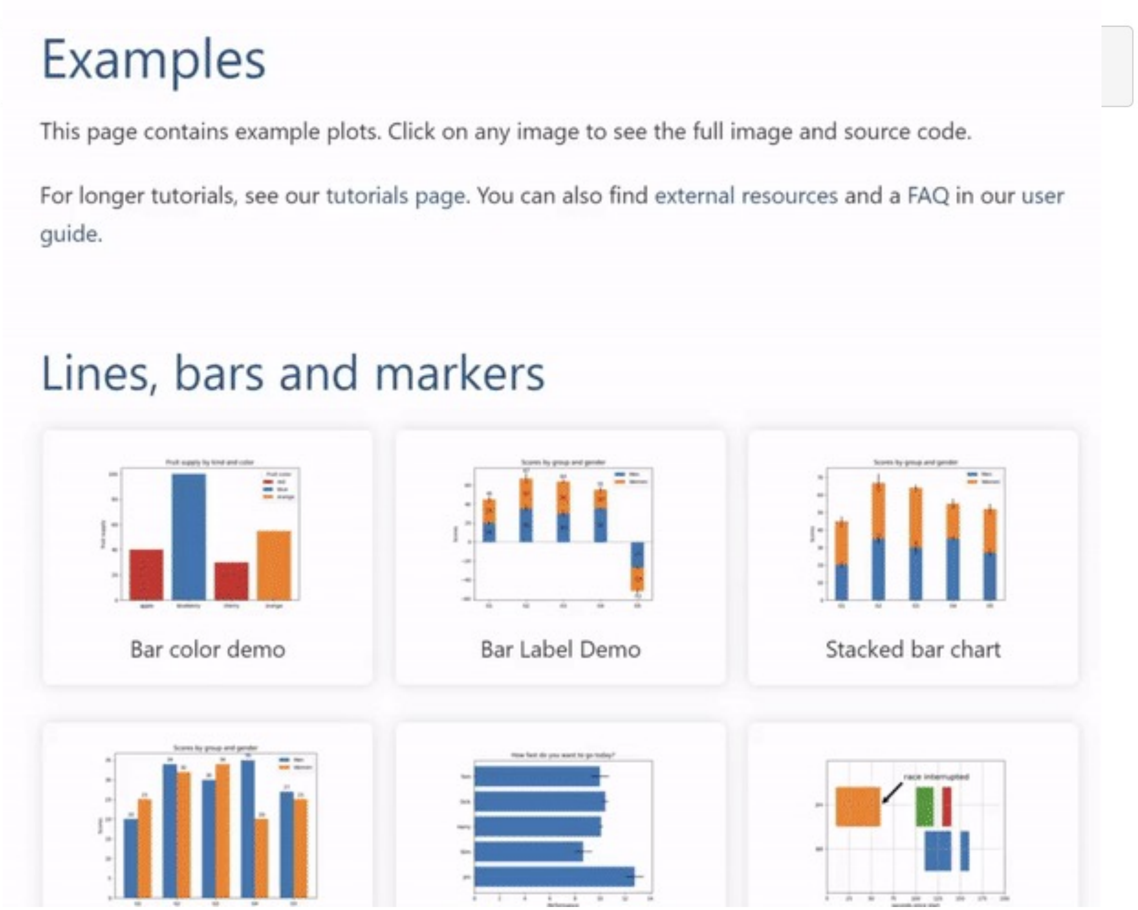
Run

cluster_ids_in_space [-161 95 211]: 1

enter paint or fill mode to edit labels

https://github.com/BiAPoL/napari-clusters-plotter

matplotlib



Examples

This page contains example plots. Click on any image to see the full image and source code.

For longer tutorials, see our tutorials page. You can also find external resources and a FAQ in our user guide.

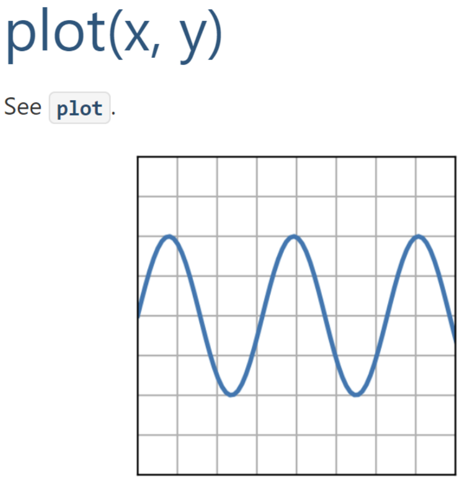
Lines, bars and markers

Bar color demo

Bar Label Demo

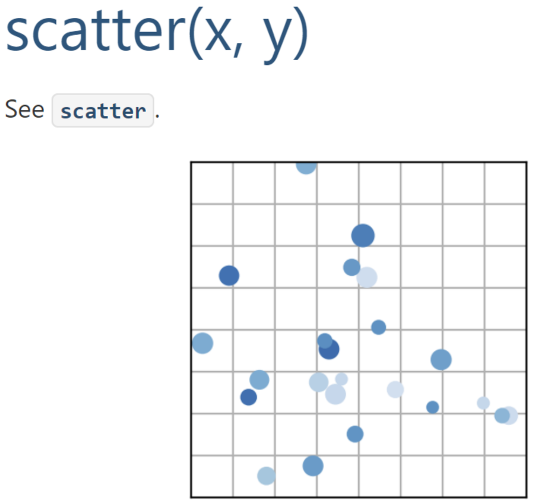
Stacked bar chart

Matplotlib: Basic Plot Types



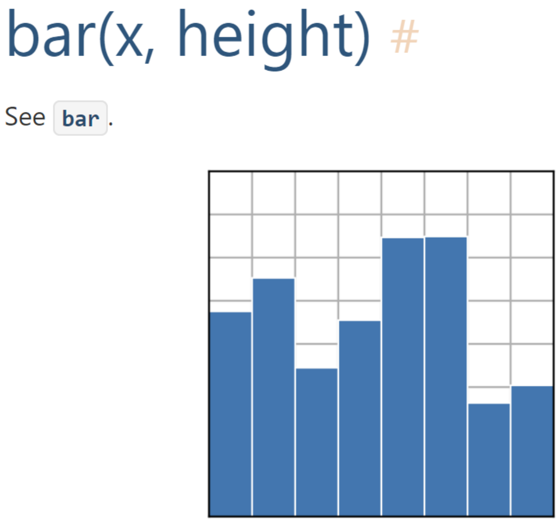
plot(x, y)

See `plot`.



scatter(x, y)

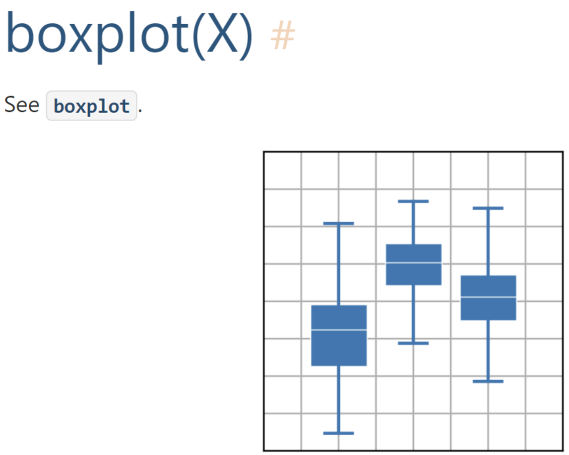
See `scatter`.



bar(x, height) #

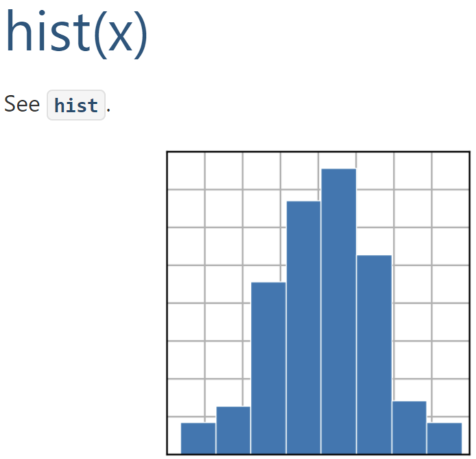
See `bar`.

Statistical Plots



boxplot(X) #

See `boxplot`.

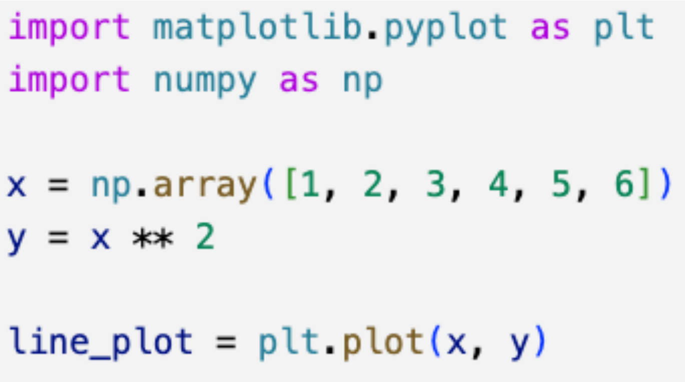


hist(x)

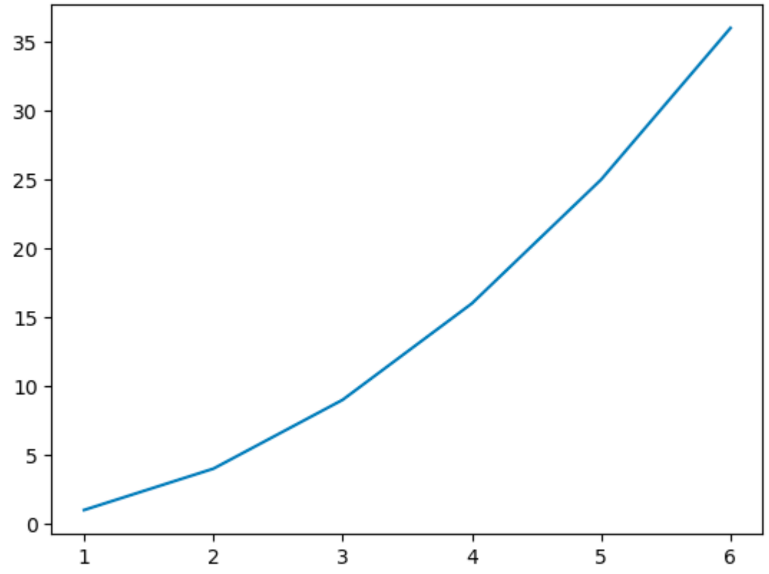
See `hist`.

Plotting Data with Matplotlib

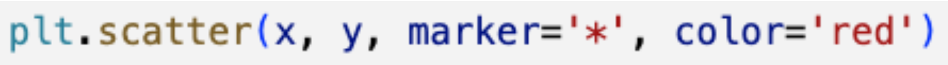
Line plot



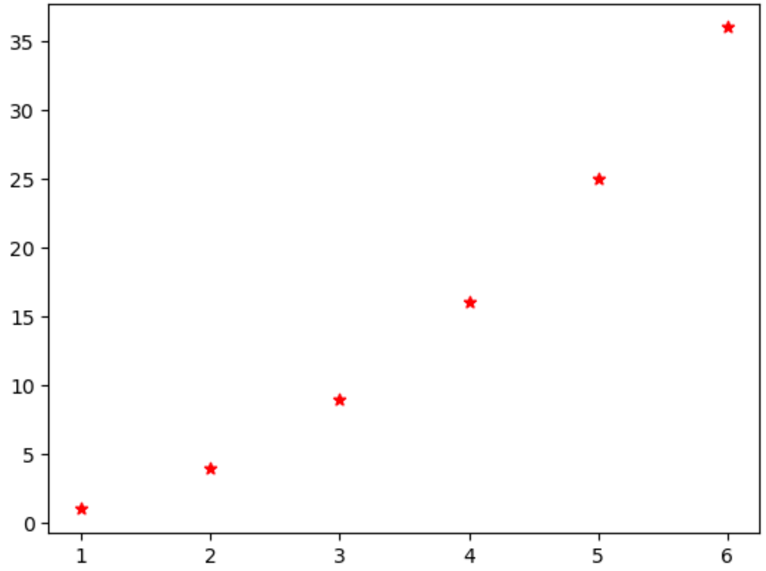
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([1, 2, 3, 4, 5, 6])
y = x ** 2

line_plot = plt.plot(x, y)
```



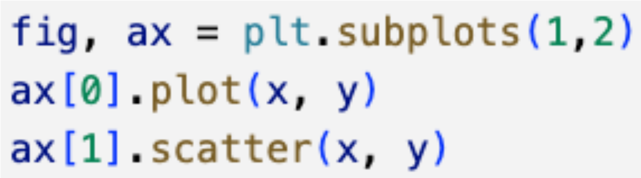
Scatter plot



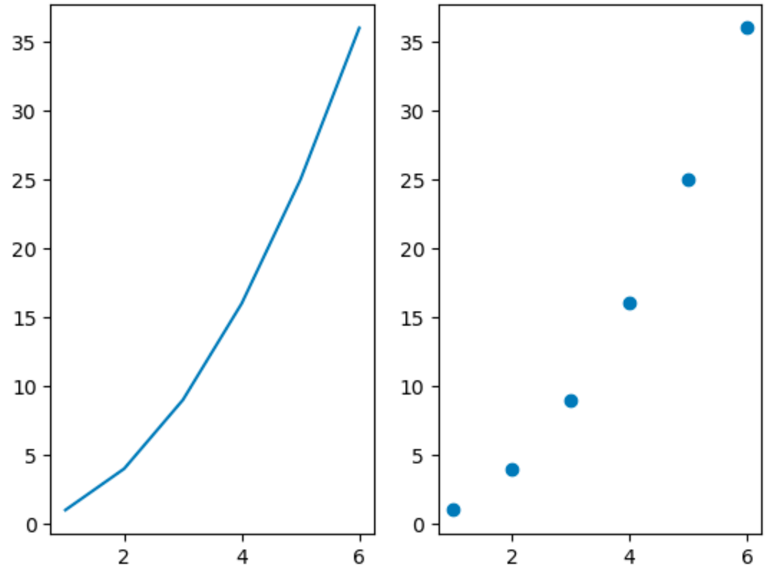
```
plt.scatter(x, y, marker='*', color='red')
```



Multiple plots in one figure



```
fig, ax = plt.subplots(1,2)
ax[0].plot(x, y)
ax[1].scatter(x, y)
```



subplots: Create multi-panel figures

```
fig, axis = plt.subplots(1,3)
```

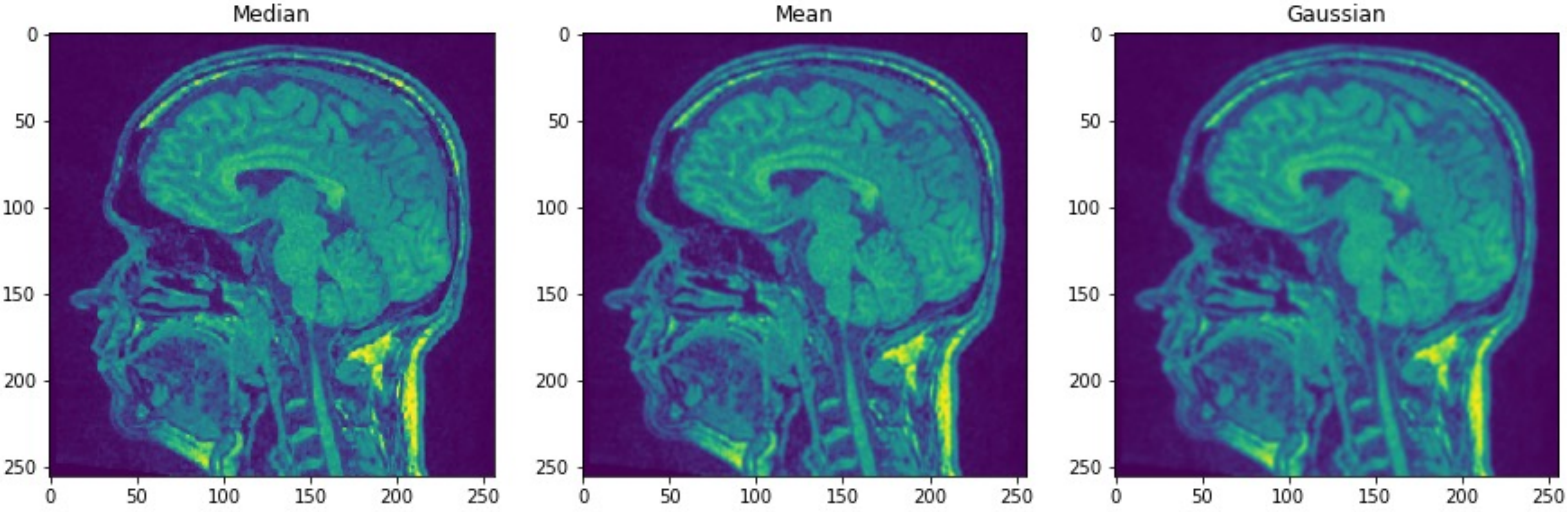
rows

columns

```
axis[0, 0].imshow(median)
axis[0, 1].imshow(mean)
axis[0, 2].imshow(gaussian)
```

row

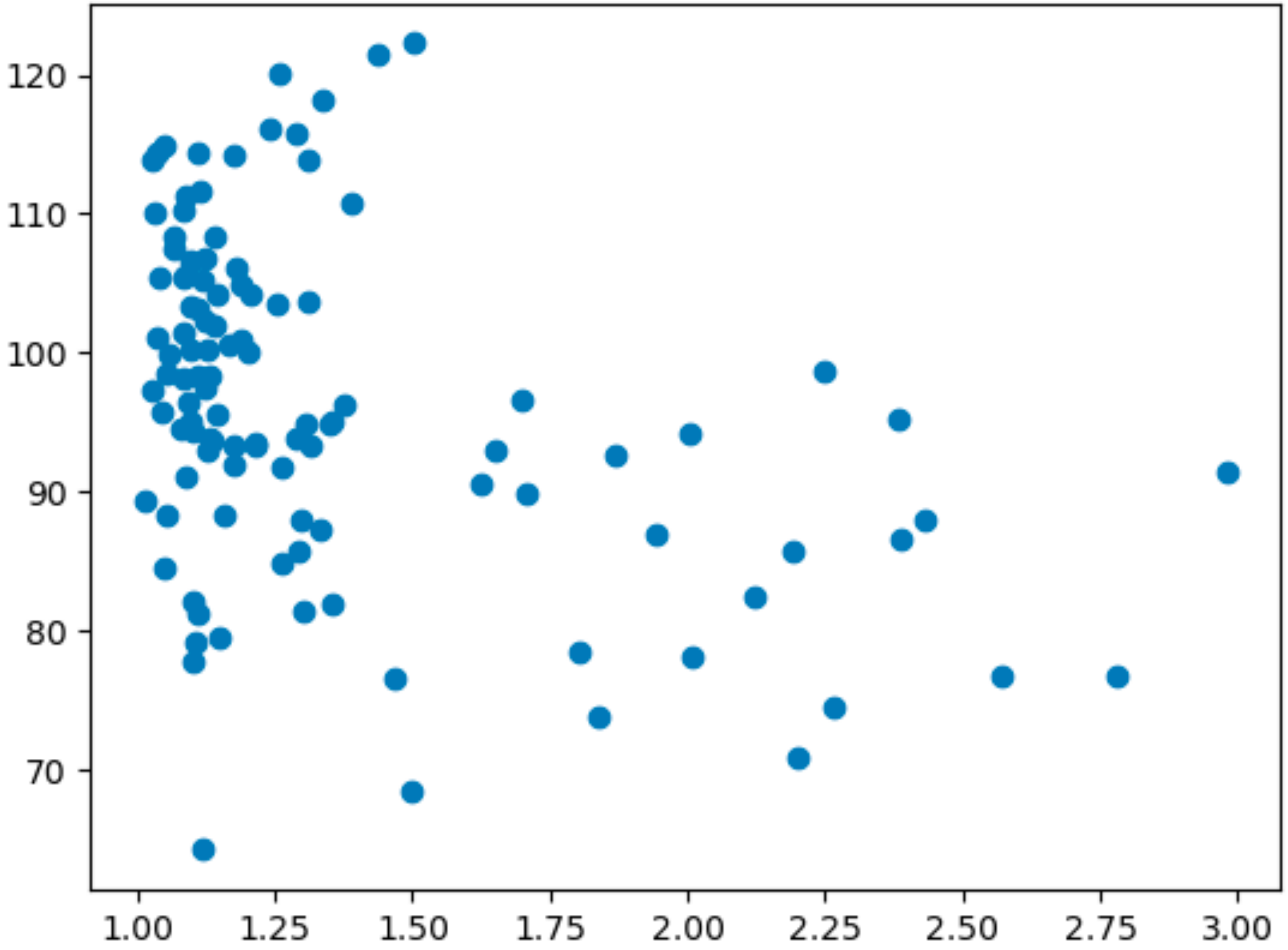
column



https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html

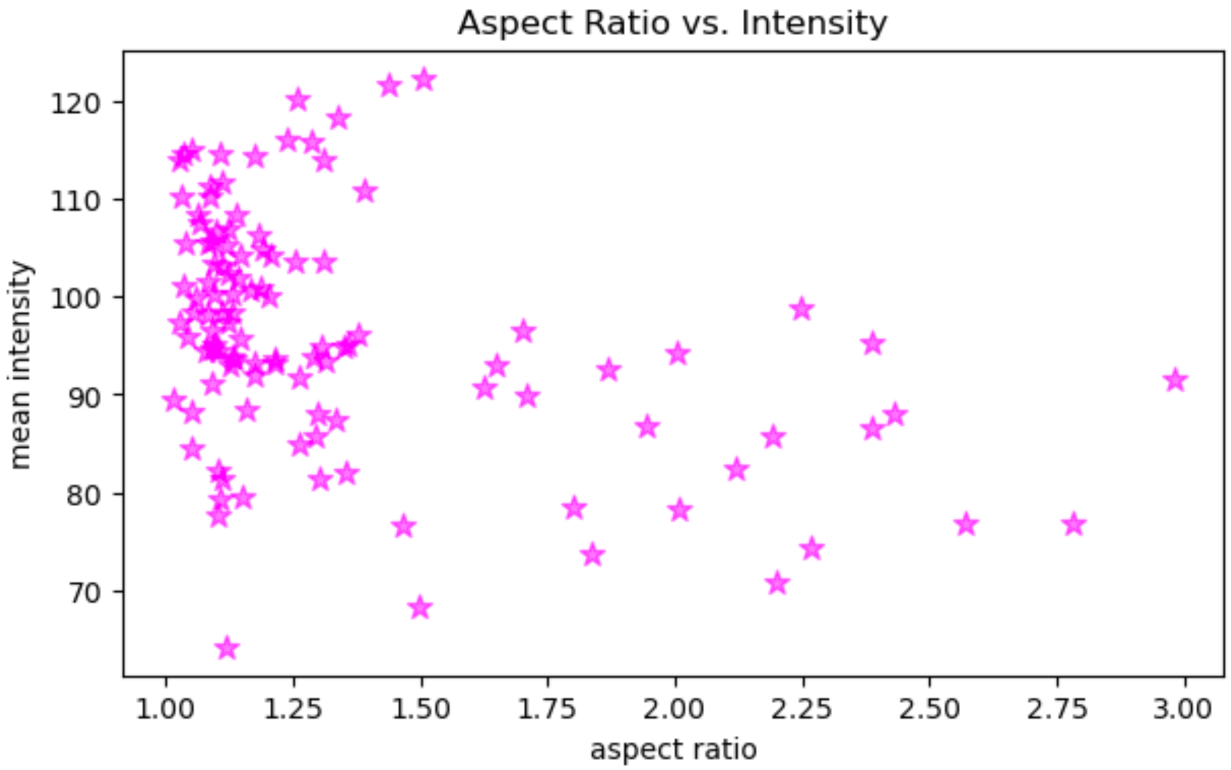
Plotting Tabular Data

```
x = df['aspect_ratio']
y = df['intensity_mean']

plt.scatter(x, y)
```



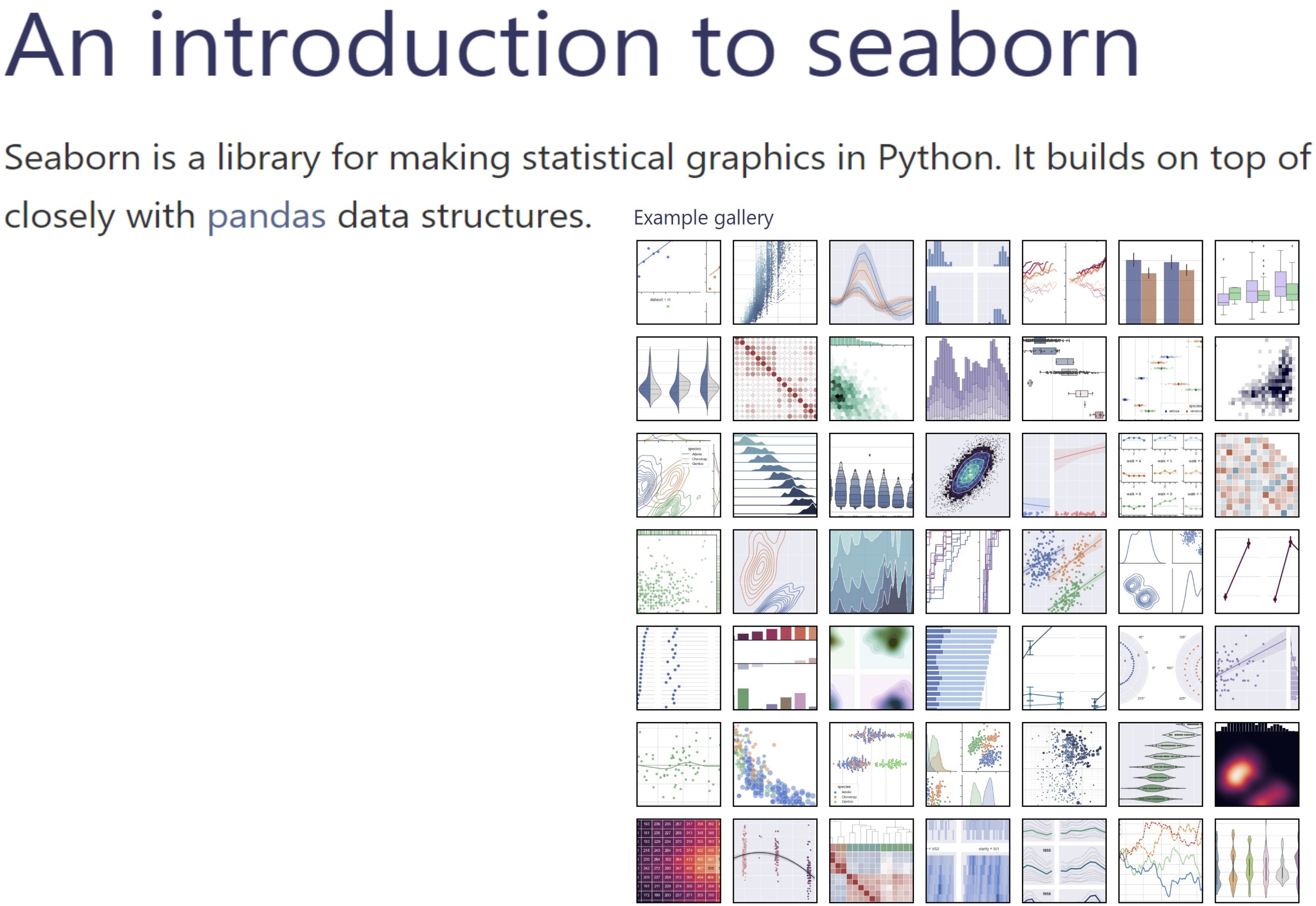
Customize your plot via the axis object

```
fig, axis = plt.subplots(figsize=(7,4))
axis.scatter(x, y, color='magenta',
marker='*', s=80, alpha=0.5)
axis.set_xlabel('aspect ratio')
axis.set_ylabel('mean intensity')
axis.set_title('Aspect Ratio vs.
Intensity')
```



Plotting Data with Seaborn

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



An introduction to seaborn

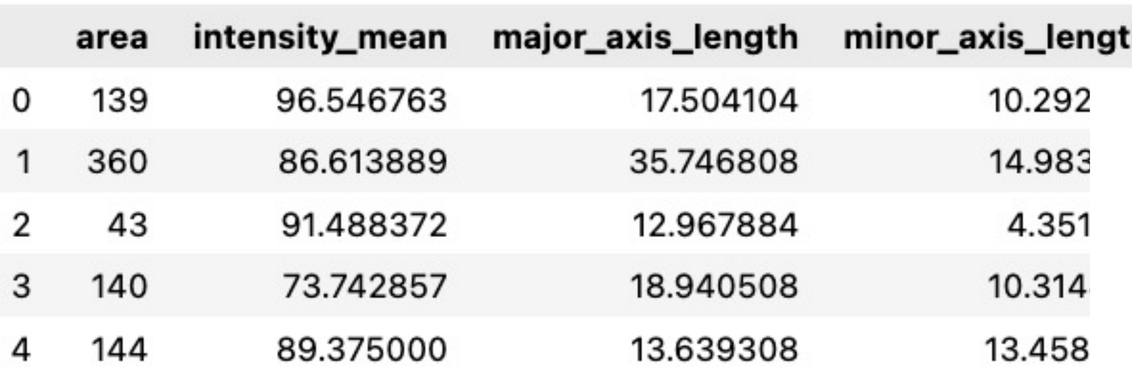
Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Example gallery

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

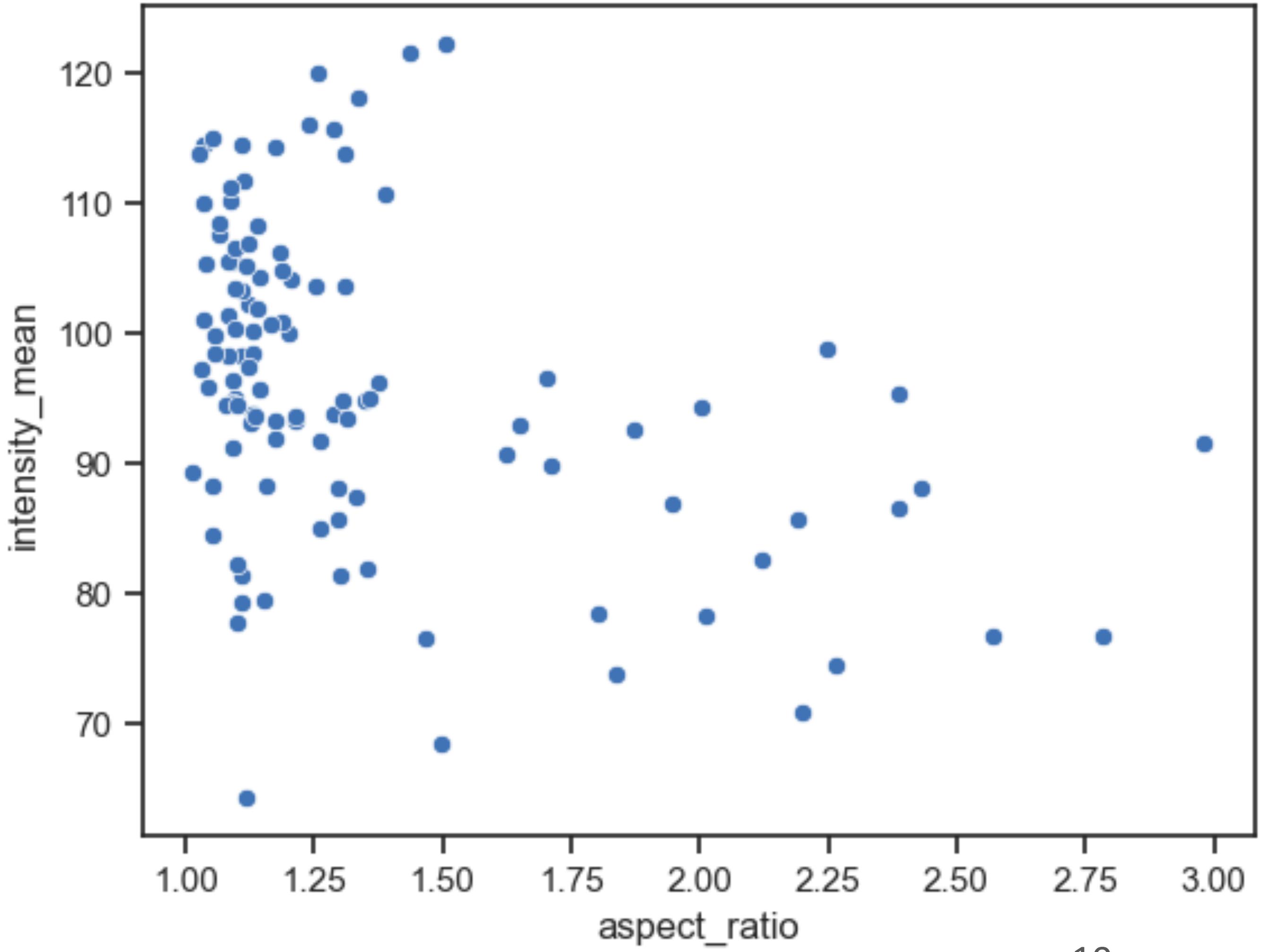
Plotting A Scatter Plot from Tabular Data with Seaborn

```
df = pd.read_csv("../../data/BBBC007_analysis.csv")
df.head()
```



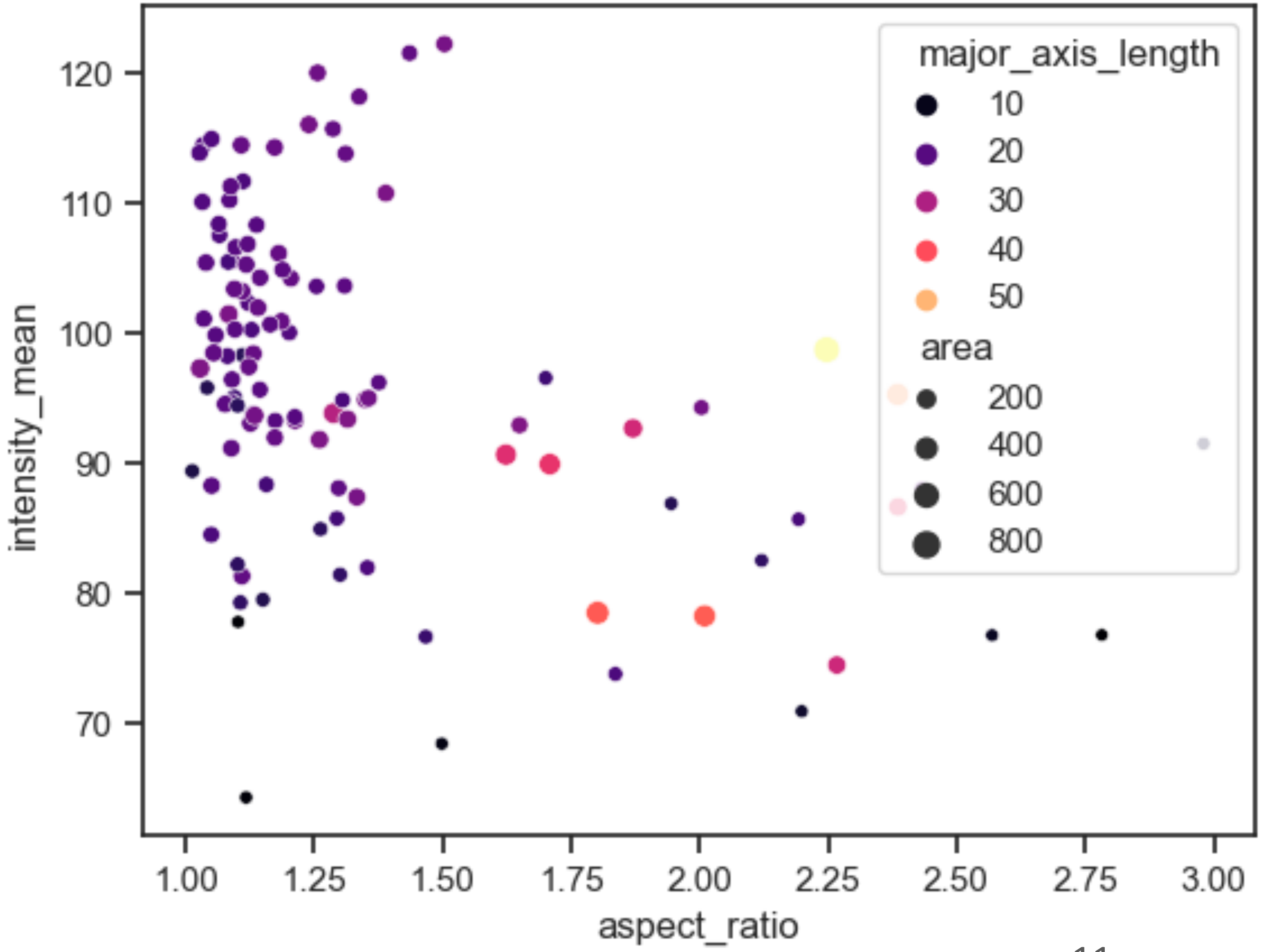
	area	intensity_mean	major_axis_length	minor_axis_length	aspect_ratio	file_name
0	139	96.546763	17.504104	10.292		
1	360	86.613889	35.746808	14.983		
2	43	91.488372	12.967884	4.351		
3	140	73.742857	18.940508	10.314		
4	144	89.375000	13.639308	13.458		

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



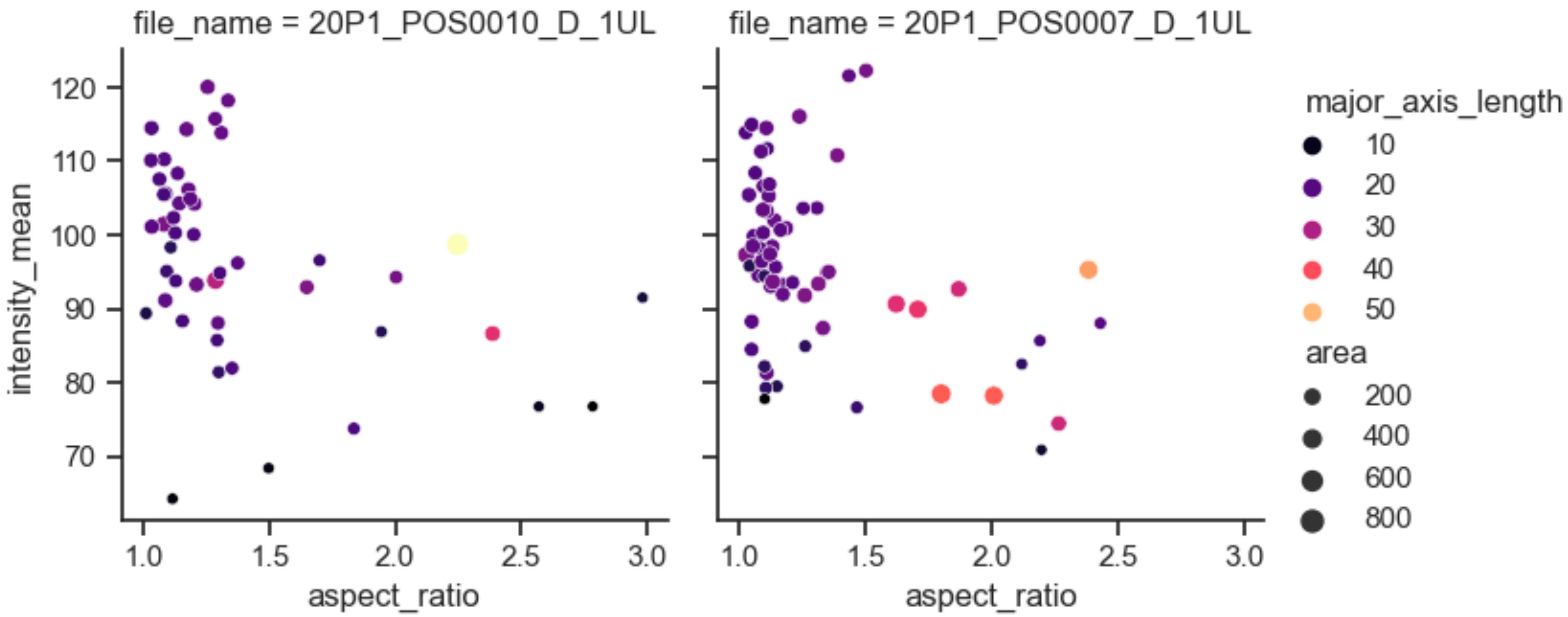
Symbol Size and Color can be Defined by Data

```
sns.scatterplot(data=df,
x="aspect_ratio",
y="intensity_mean",
size="area",
hue="major_axis_length",
palette='magma')
```



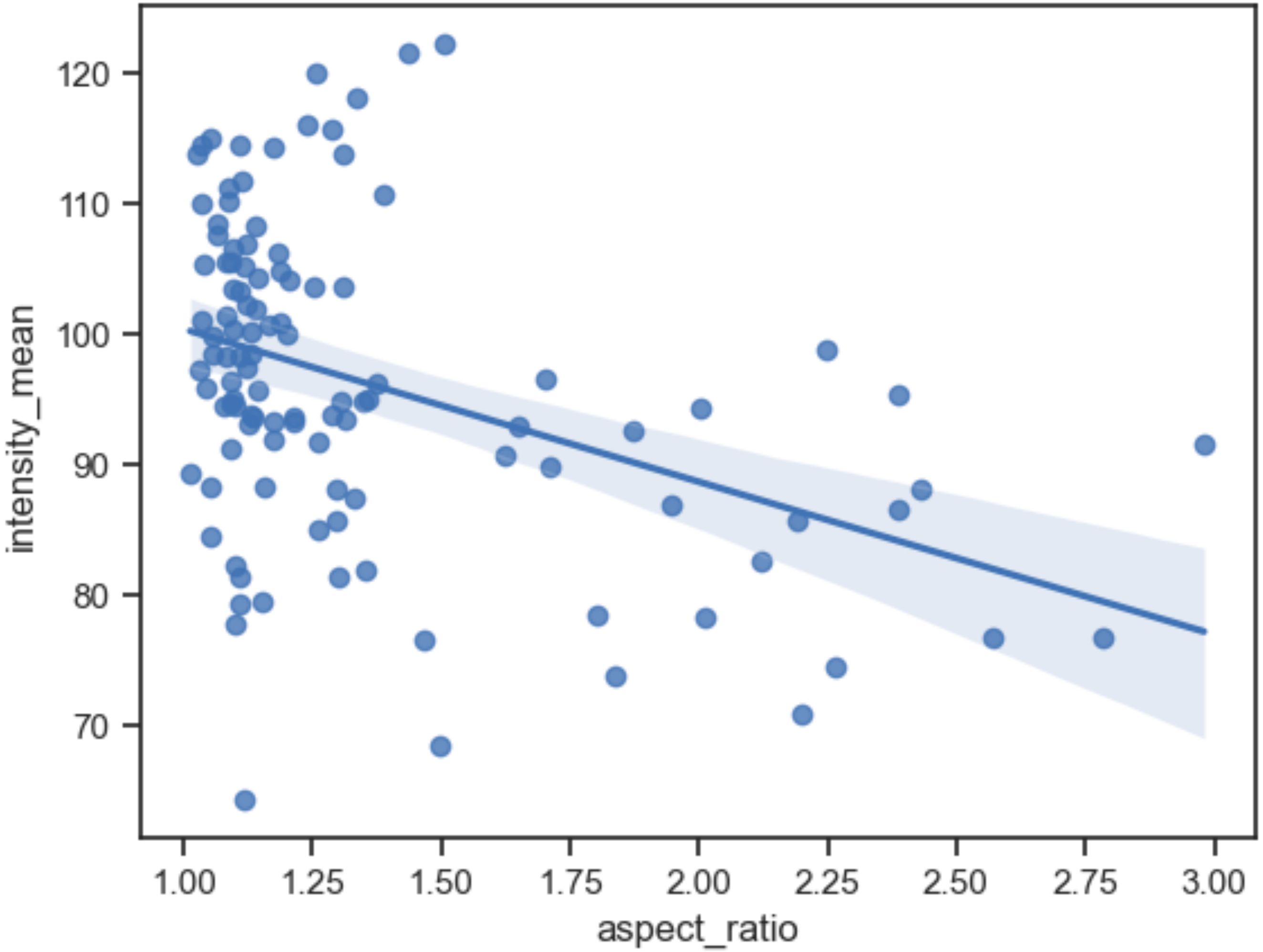
relplot: Split Plot by Categorical Variable

```
sns.relplot(data=df, x="aspect_ratio", y="intensity_mean", size="area",
hue="major_axis_length", col="file_name", palette='magma')
```



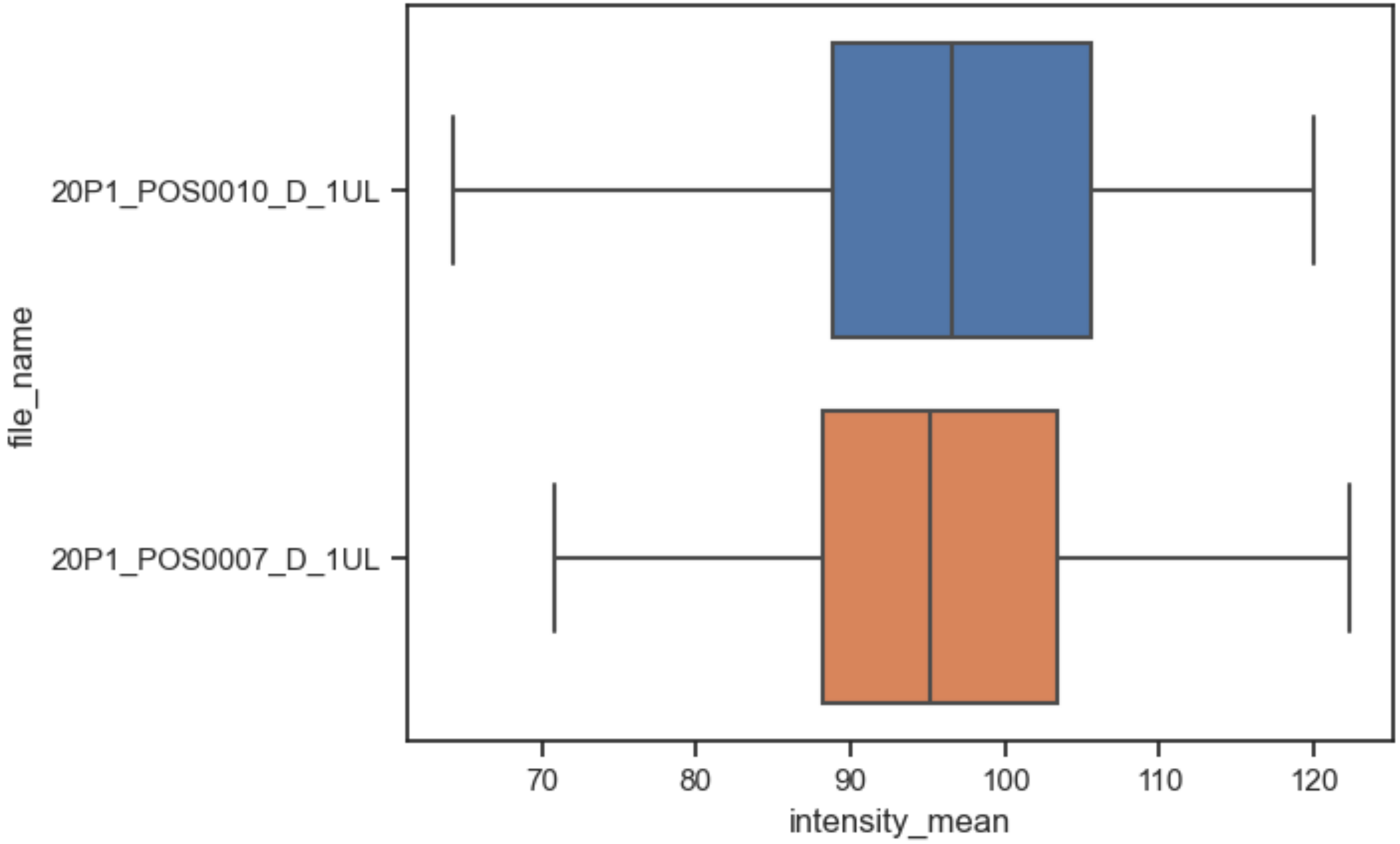
regplot: Line Regression

```
sns.regplot(data=df,
x="aspect_ratio",
y="intensity_mean")
```



boxplot

```
sns.boxplot(data=df,
x="intensity_mean",
y="file_name")
```

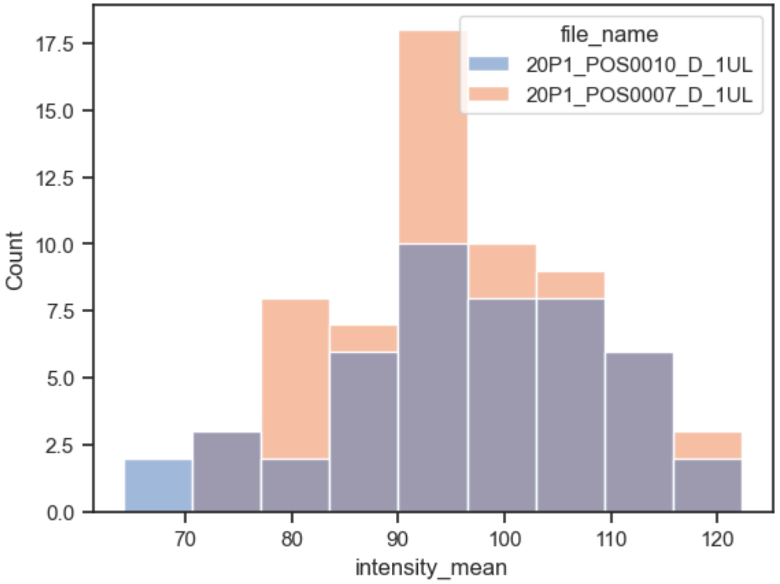


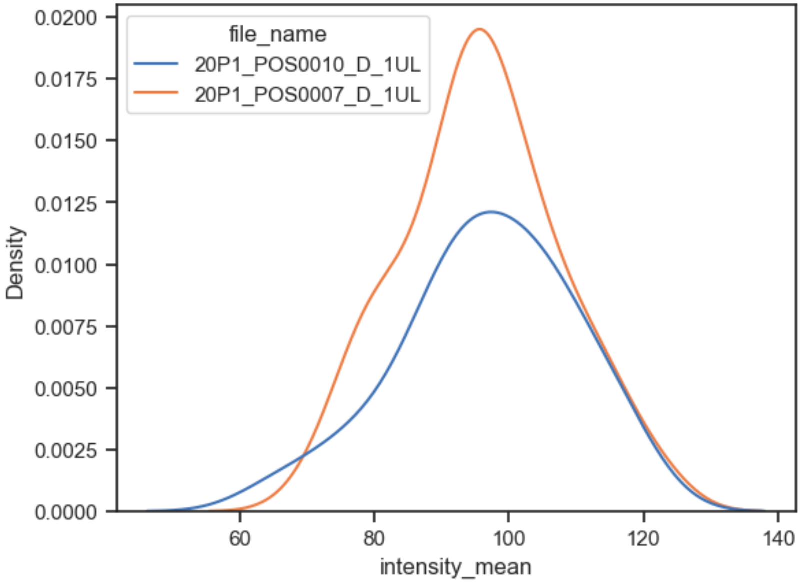
Histograms

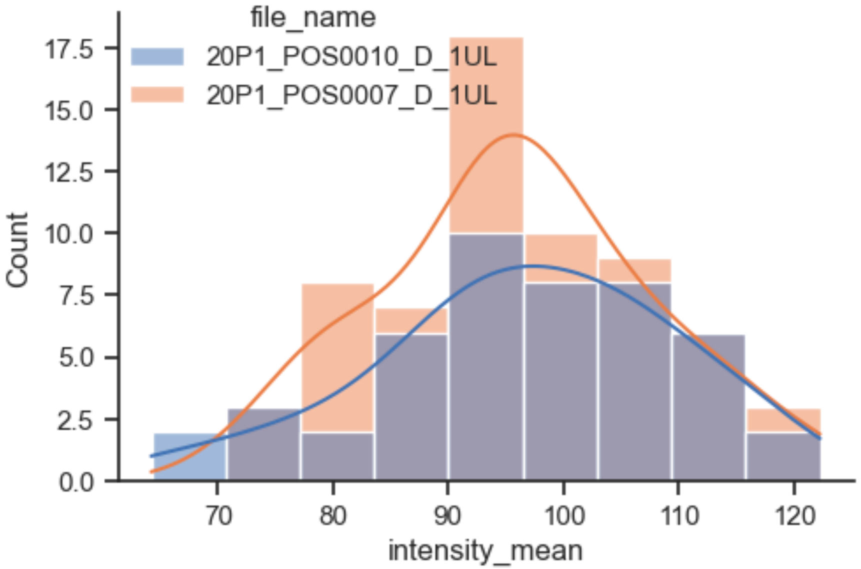
```
sns.histplot(data=df,
x="intensity_mean",
hue="file_name")
```

```
sns.kdeplot(data=df,
x="intensity_mean",
hue="file_name")
```

```
sns.displot(data=df,
x="intensity_mean",
hue="file_name",
kde=True)
```

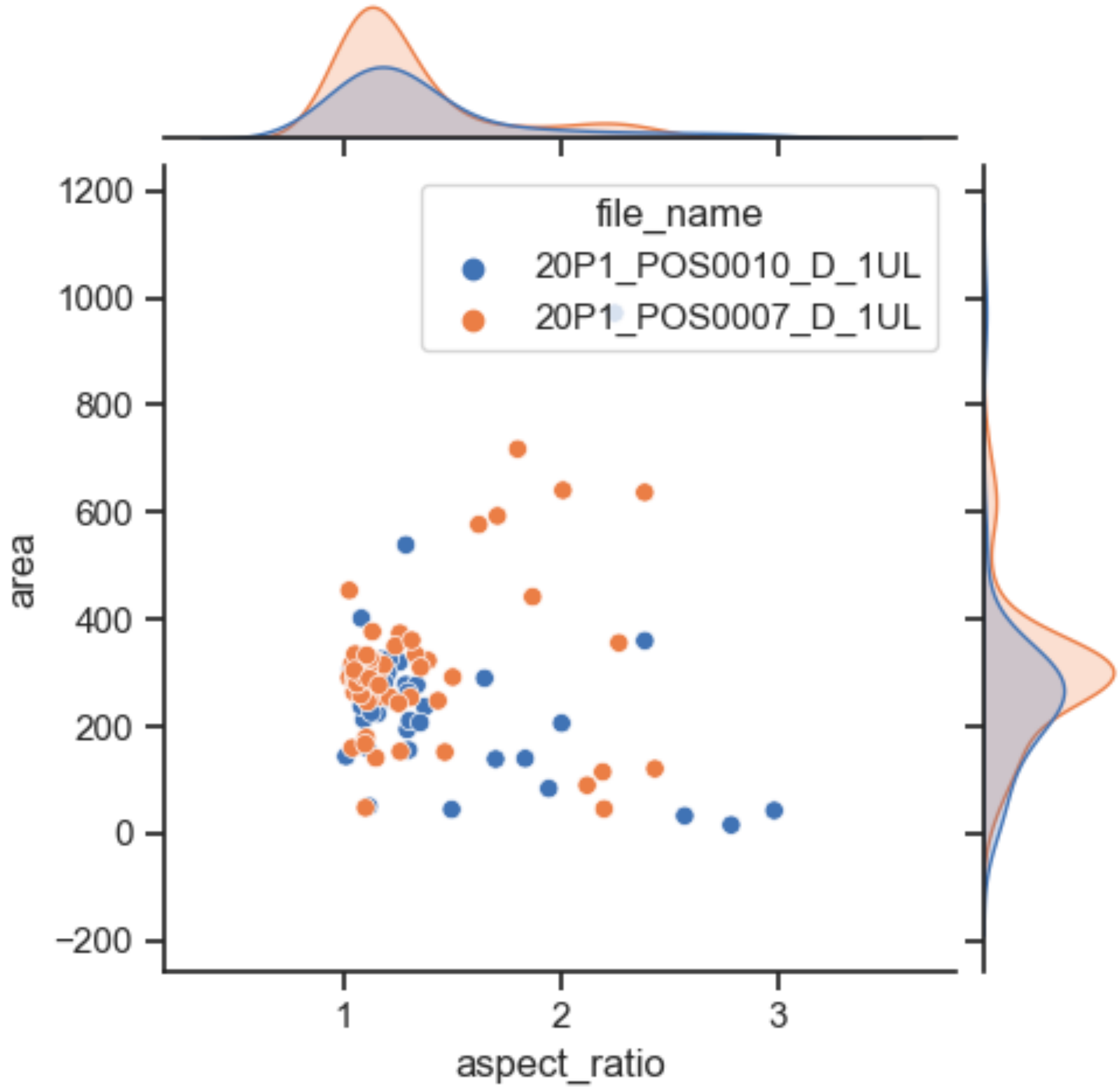






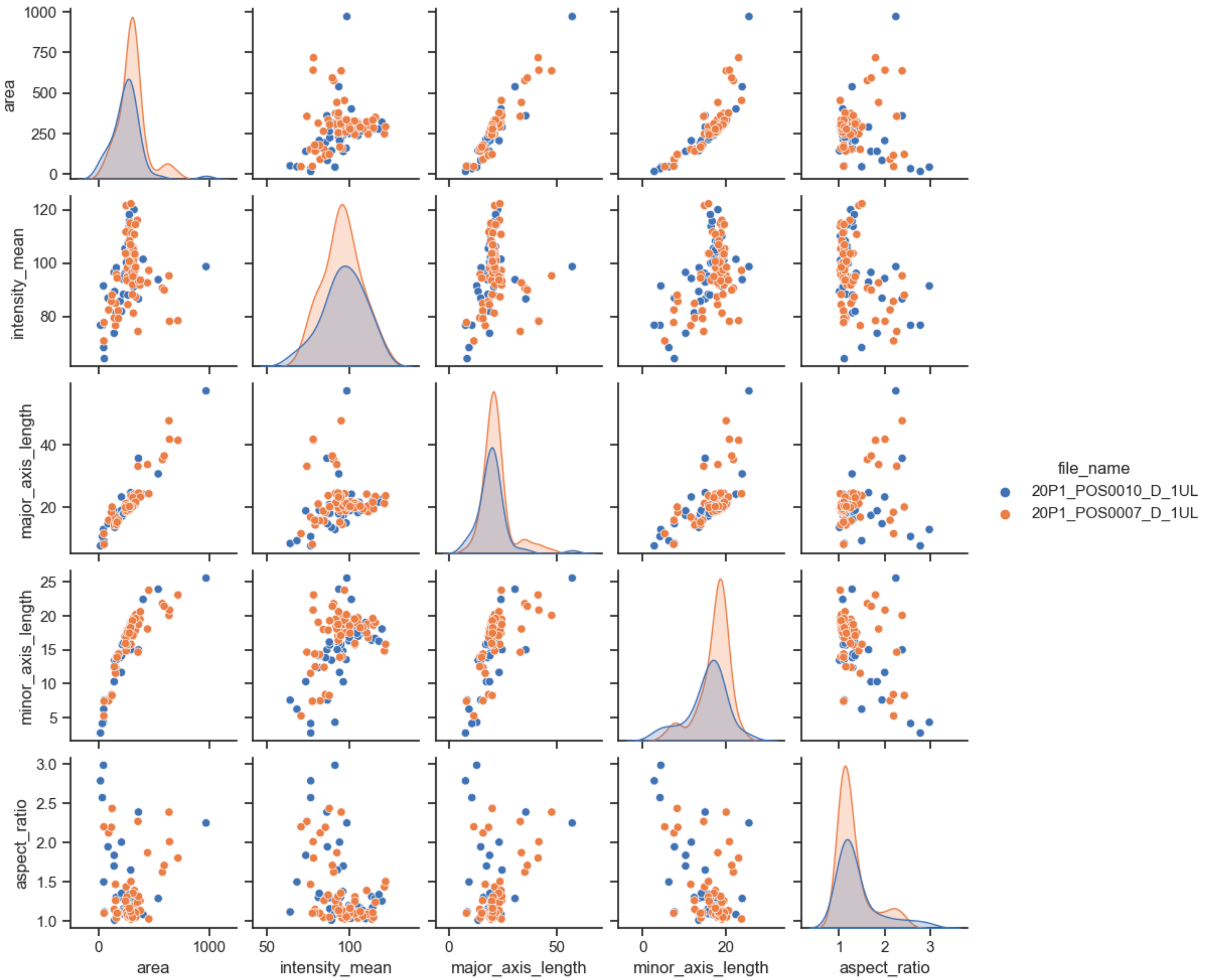
jointplot: Visualizing Distributions of a Scatter plot

```
sns.jointplot(data=df,
x="aspect_ratio",
y="area",
hue='file_name')
```



pairplot: Comparing Many Properties

```
sns.pairplot(data=df,
hue="file_name")
```



@TillKorten

Exercises

- The examples presented here can be found in the Jupyter notebook “Plotting.ipynb”
- Exercises
  - Exercise 1: line regression
  - Exercise 2: ECDF
  - Exercise 3: non-redundant pairplot
  - Exercise 4: Plot images + data