



# Plotting Data with Seaborn

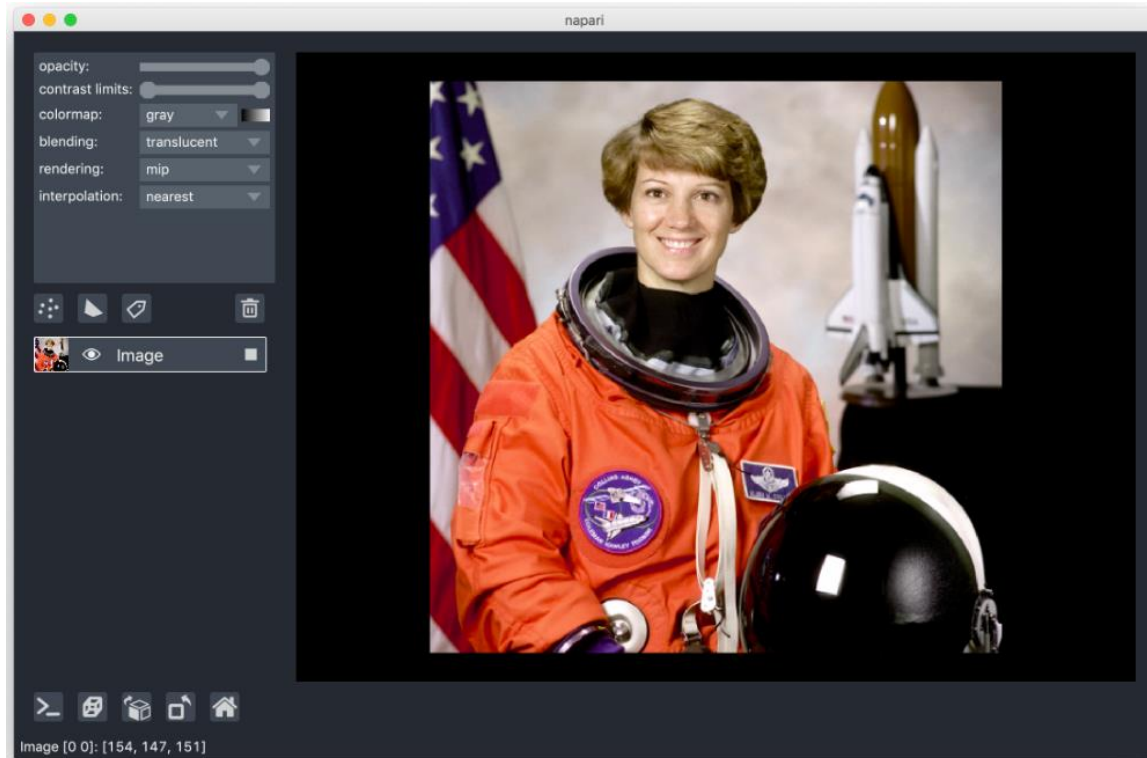
Marcelo Leomil Zoccoler

With material from  
Robert Haase, BiAPoL, PoL TU Dresden

September 2022

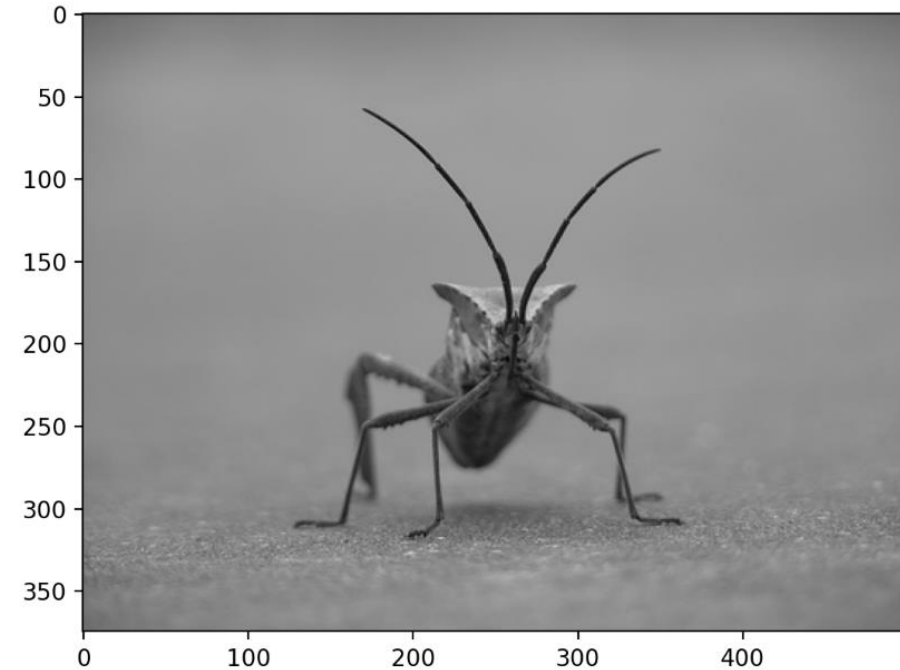
- Displaying images

napari



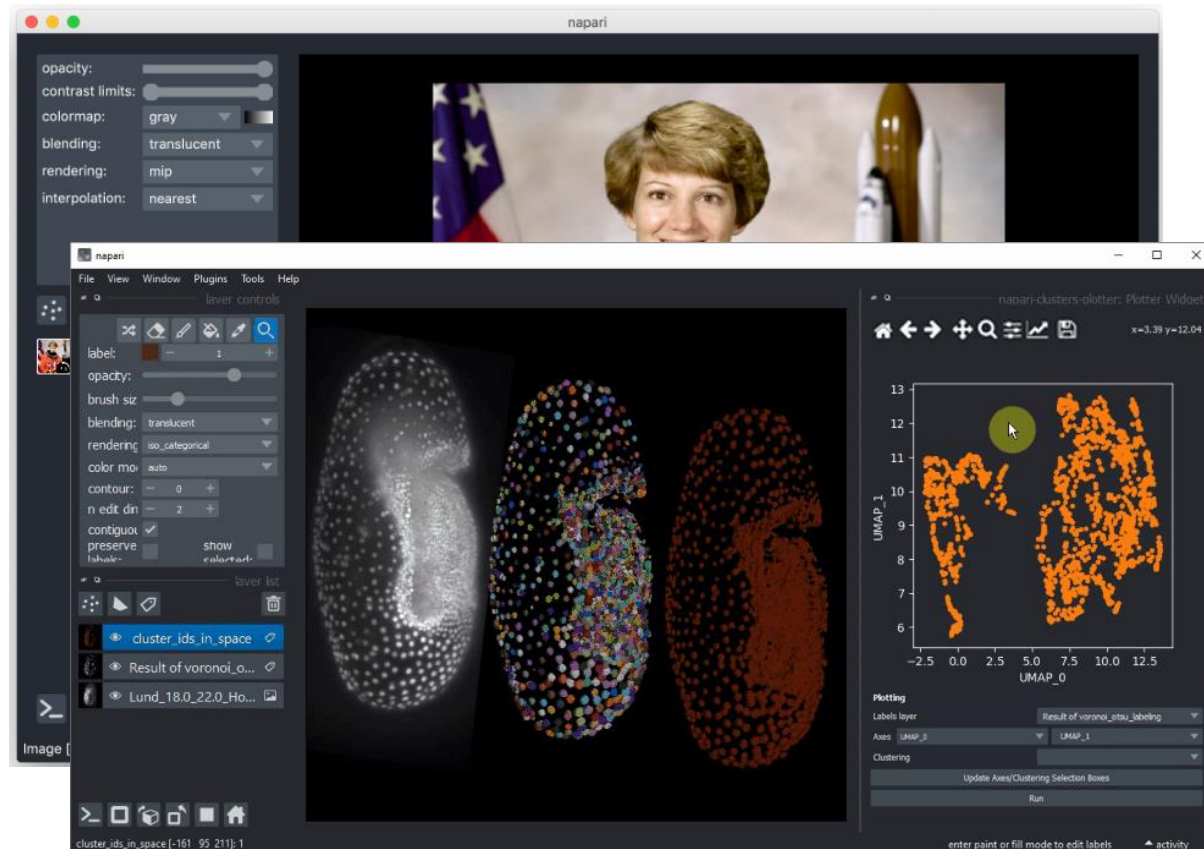
matplotlib

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



- Displaying graphs

napari



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

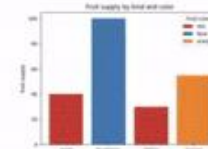
matplotlib

## Examples

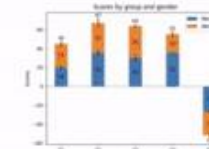
This page contains example plots. Click on any image to see the full image and source code.

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

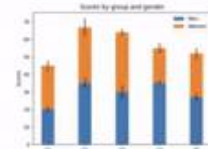
## Lines, bars and markers



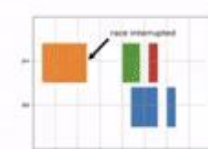
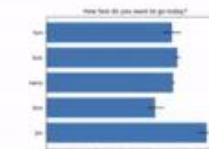
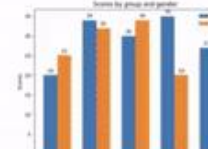
Bar color demo



Bar Label Demo



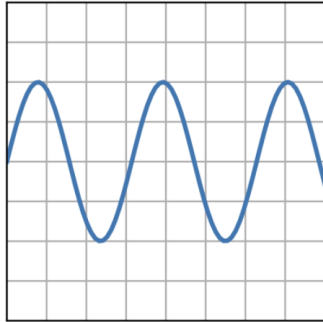
Stacked bar chart



## Basic Plot Types

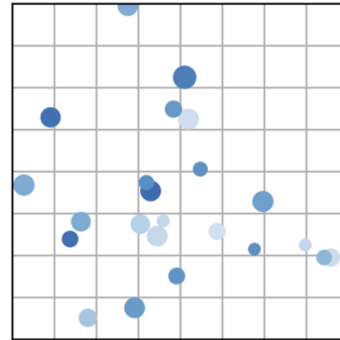
`plot(x, y)`

See `plot`.



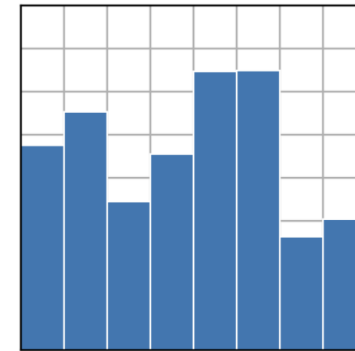
`scatter(x, y)`

See `scatter`.



`bar(x, height) #`

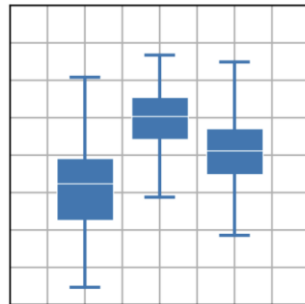
See `bar`.



## Statistical Plots

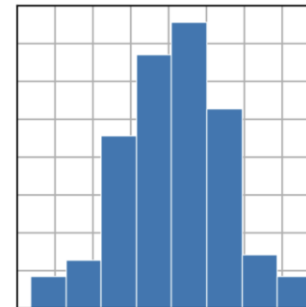
`boxplot(X) #`

See `boxplot`.



`hist(x)`

See `hist`.



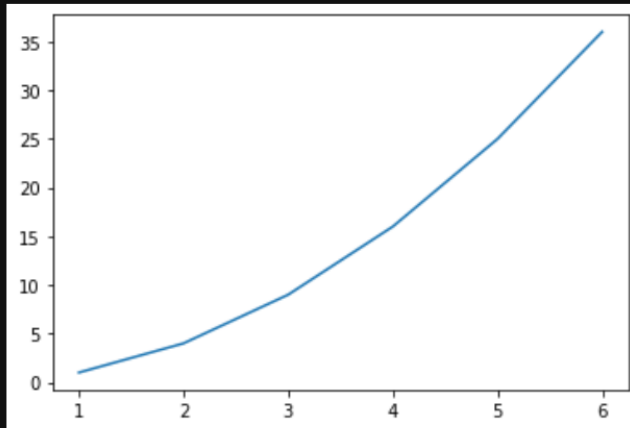
## Individual plots:

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

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

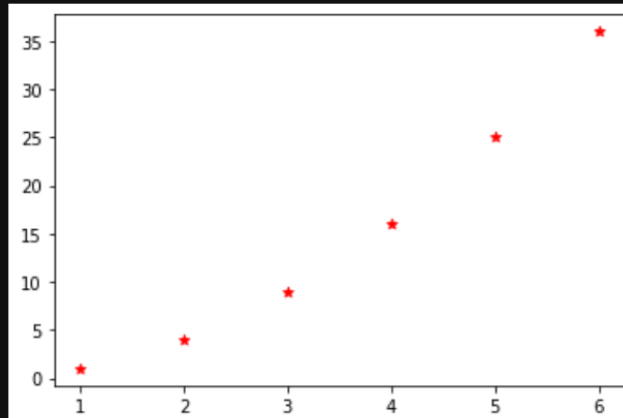
```
plt.plot(x, y)
```

```
[<matplotlib.lines.Line2D at 0x1d01c6c8370>]
```



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

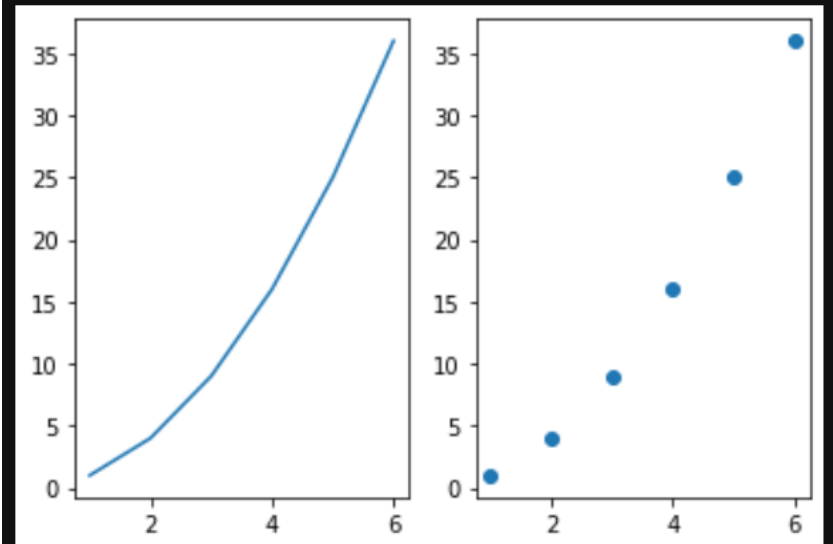
```
<matplotlib.collections.PathCollection at 0x1d01d039730>
```



## Plots with subplots:

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

```
<matplotlib.collections.PathCollection at 0x1d01cf0c1c0>
```





# Plotting Data with Matplotlib

```
median_filtered = filters.median(noisy_mri, disk(1))
mean_filtered = filters.rank.mean(noisy_mri, disk(1))
gaussian_filtered = filters.gaussian(noisy_mri, sigma=1)

fig, axs = plt.subplots(2, 3, figsize=(15,10))

# first row
axs[0, 0].imshow(median_filtered)
axs[0, 0].set_title("Median")
axs[0, 1].imshow(mean_filtered)
axs[0, 1].set_title("Mean")
axs[0, 2].imshow(gaussian_filtered)
axs[0, 2].set_title("Gaussian")

# second row
axs[1, 0].imshow(median_filtered[50:100, 50:100])
axs[1, 1].imshow(mean_filtered[50:100, 50:100])
axs[1, 2].imshow(gaussian_filtered[50:100, 50:100])
```

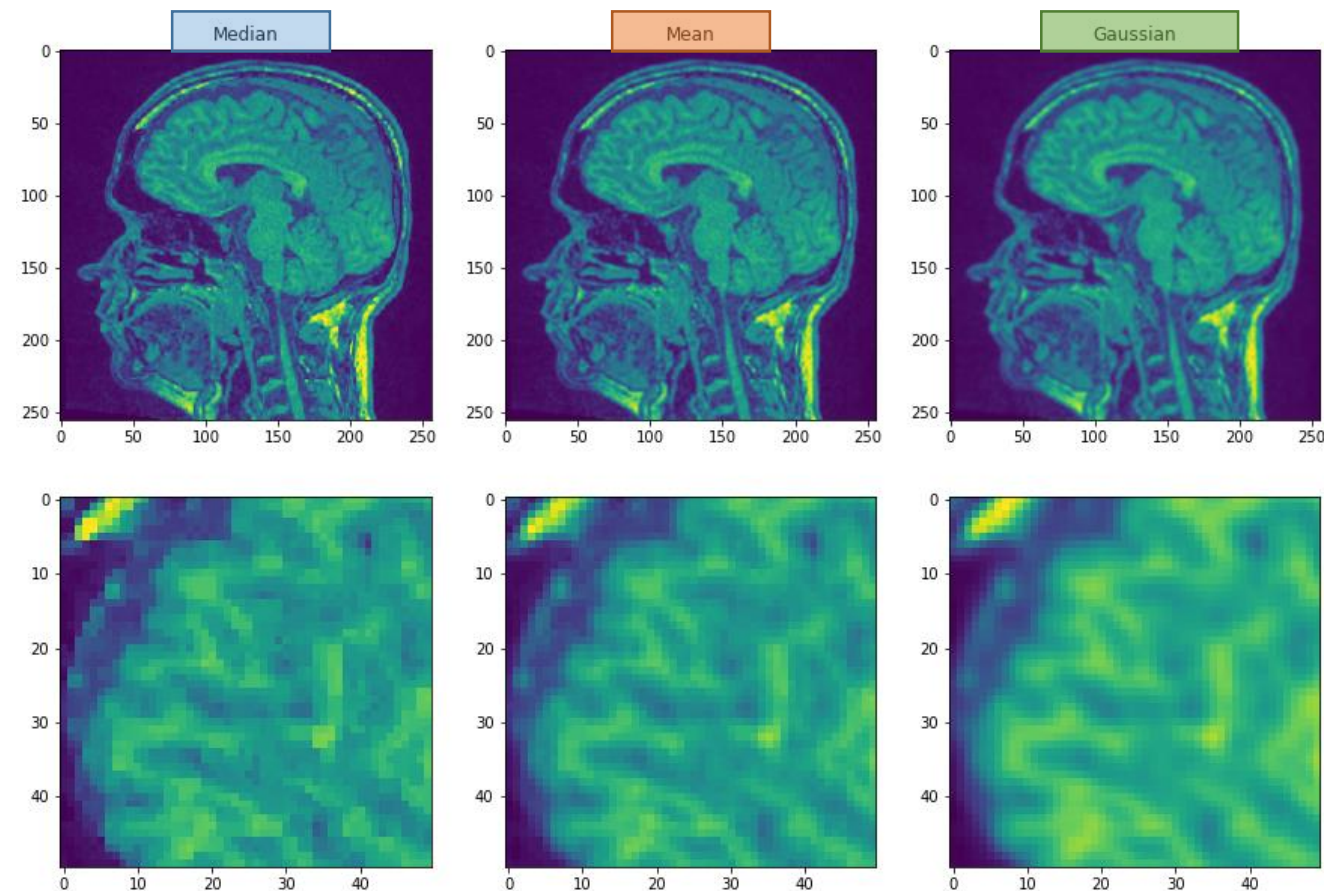
rows

columns

row

column

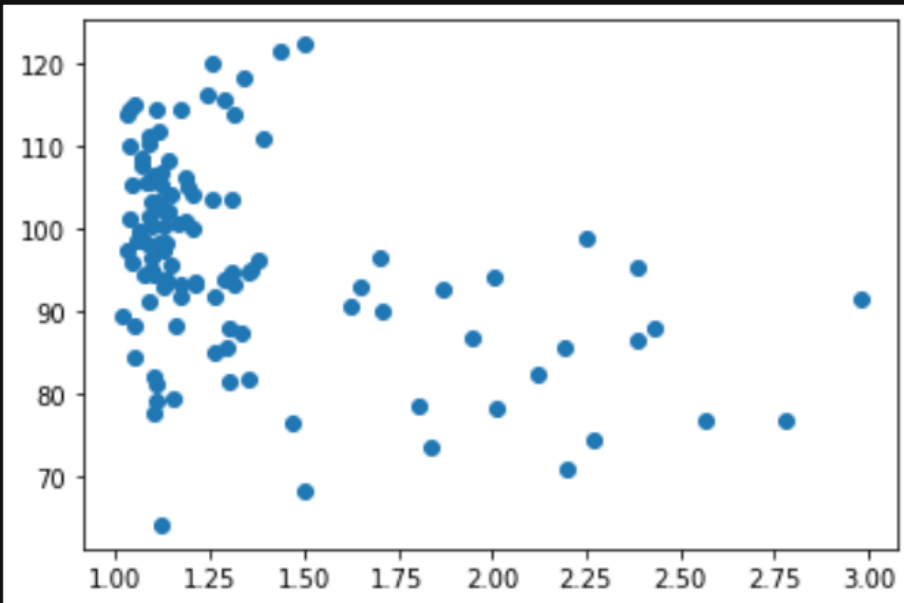
[https://matplotlib.org/stable/api/as\\_gen/matplotlib.pyplot.subplots.html](https://matplotlib.org/stable/api/as_gen/matplotlib.pyplot.subplots.html)



## Plotting tabular data:

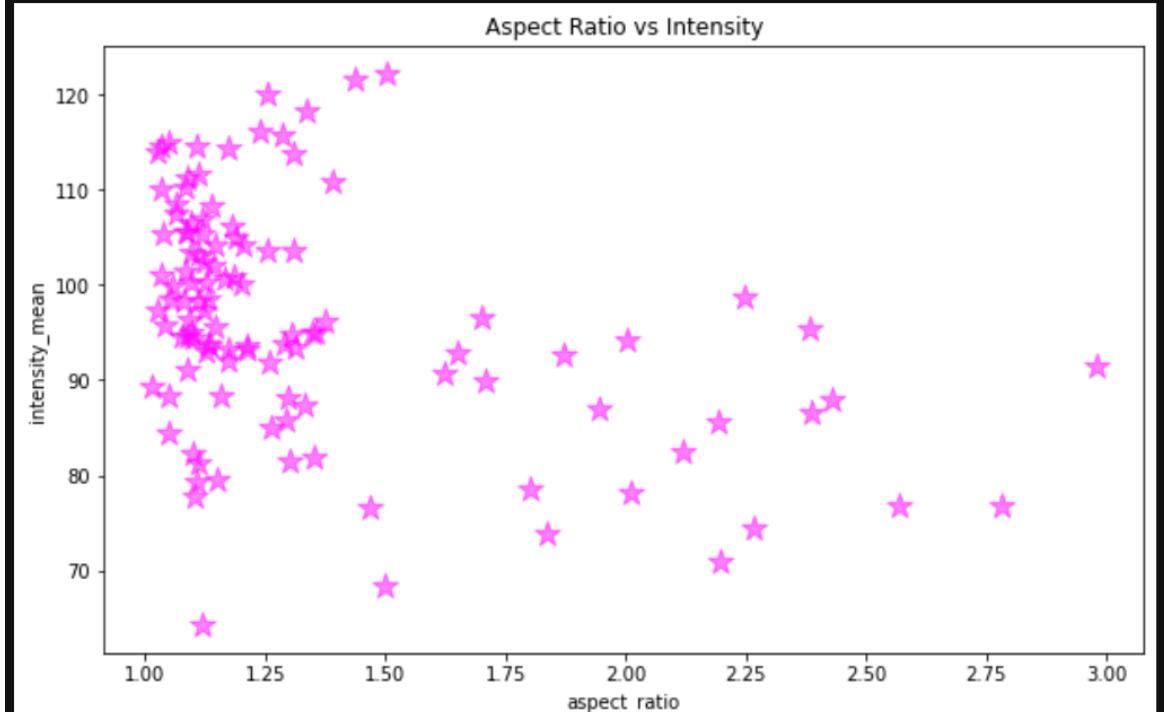
```
x = df['aspect_ratio']  
y = df['intensity_mean']  
  
plt.scatter(x, y)
```

<matplotlib.collections.PathCollection at 0x190e2bd1790>



```
fig, ax = plt.subplots(figsize = [10,6])  
ax.scatter(x, y, color = 'magenta', marker = '*', s = 200, alpha = 0.5)  
ax.set_xlabel('aspect_ratio')  
ax.set_ylabel('intensity_mean')  
ax.set_title('Aspect Ratio vs Intensity')
```

Text(0.5, 1.0, 'Aspect Ratio vs Intensity')

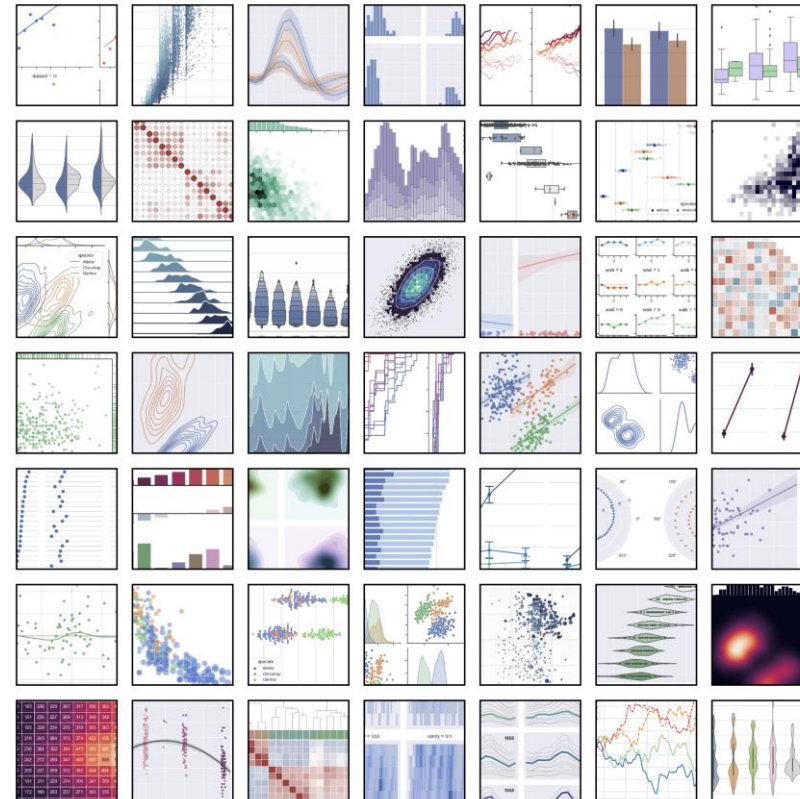


<https://seaborn.pydata.org/tutorial/introduction.html>

## 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





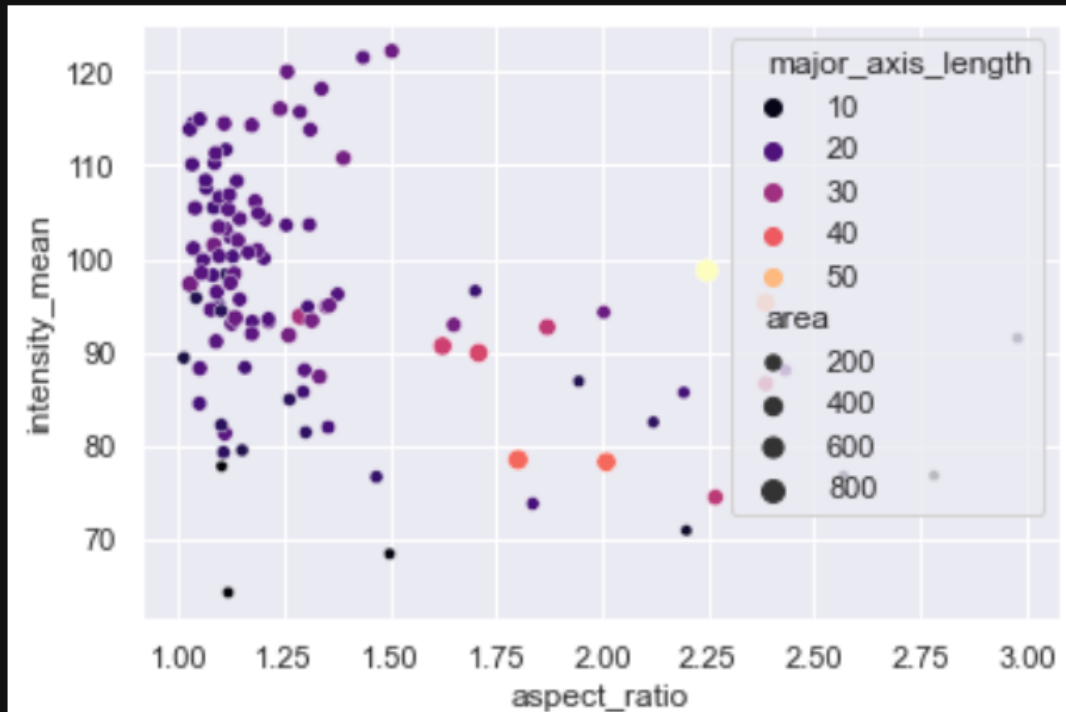
## Plotting tabular data: scatterplots

```
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	10.292770	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

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

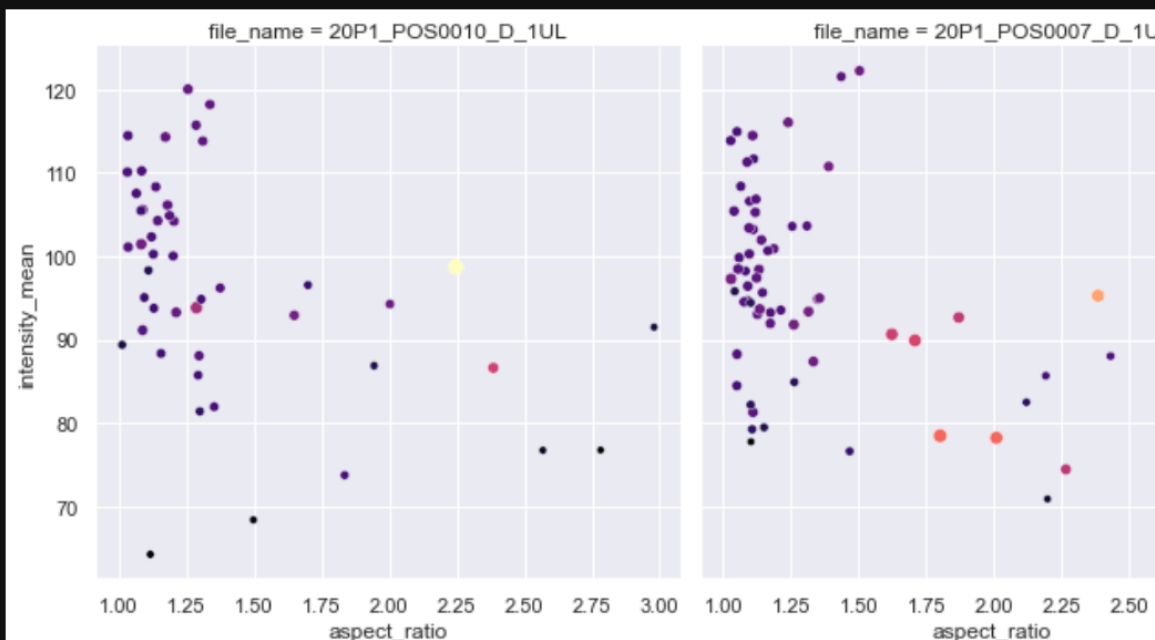
<AxesSubplot:xlabel='aspect\_ratio', ylabel='intensity\_mean'>



## Plotting categorical variables splitted:

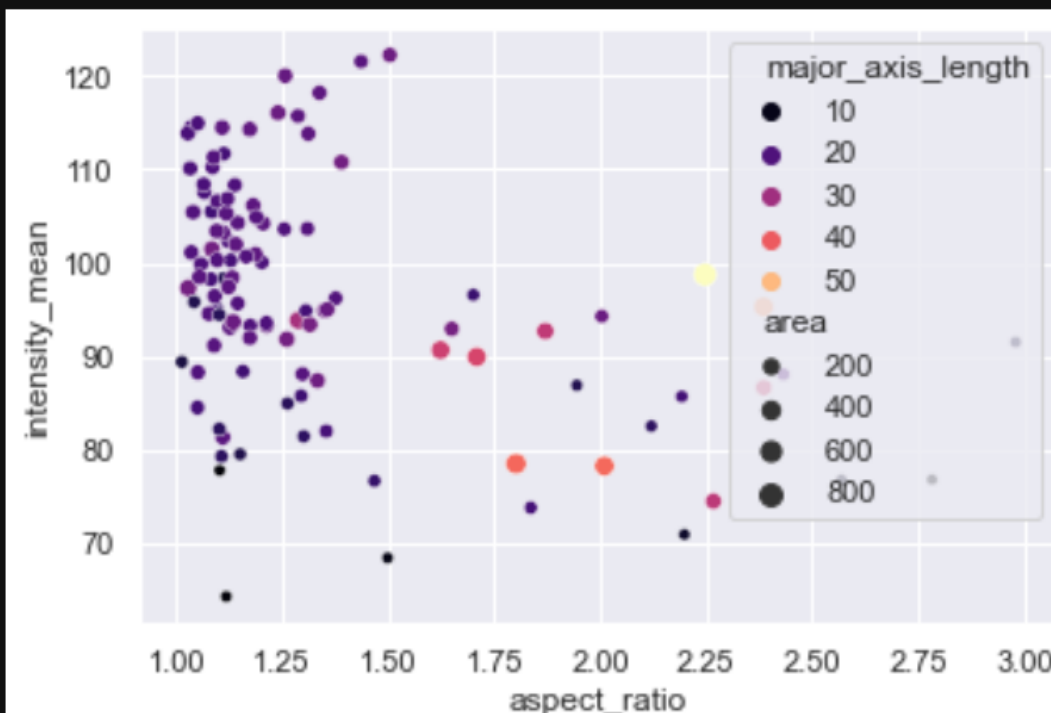
```
sns.relplot(data=df,  
            x = "aspect_ratio",  
            y = "intensity_mean",  
            size = "area",  
            hue = "major_axis_length",  
            col = "file_name",  
            palette = 'magma')
```

<seaborn.axisgrid.FacetGrid at 0x1b97a2c8d30>

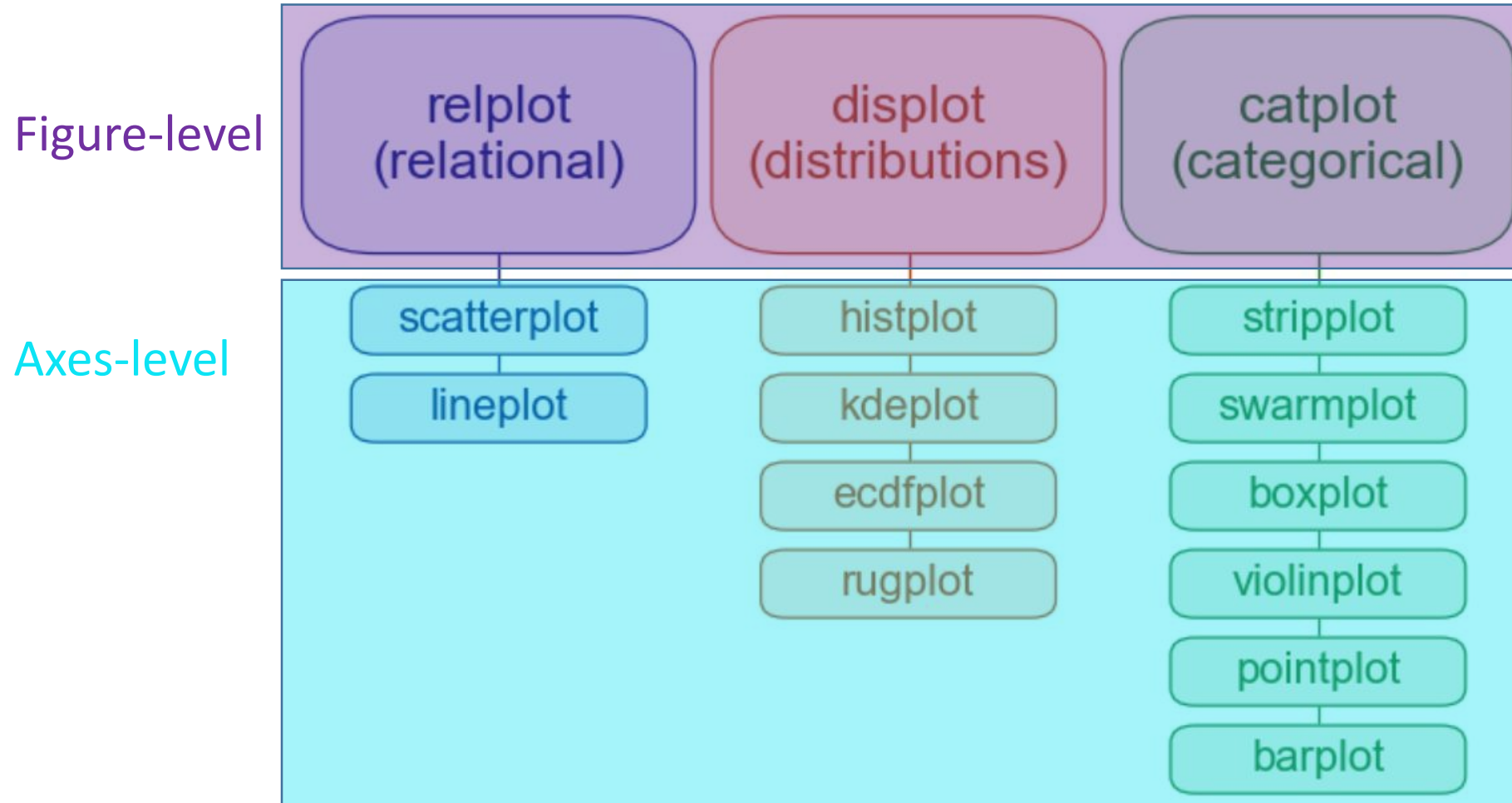


```
sns.scatterplot(data=df,  
                x = "aspect_ratio",  
                y = "intensity_mean",  
                size = "area",  
                hue = "major_axis_length",  
                palette = 'magma')
```

<AxesSubplot:xlabel='aspect\_ratio', ylabel='intensity\_mean'>



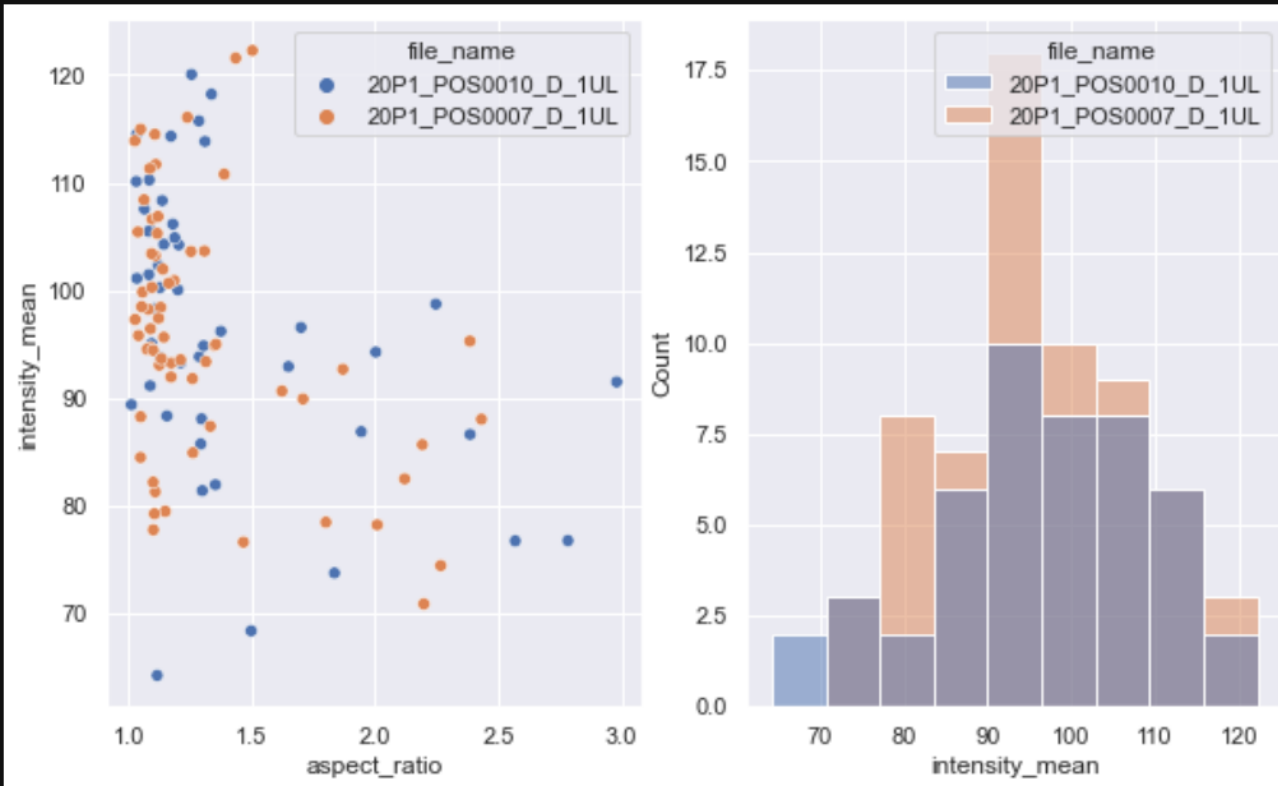
## Figure-level functions vs axes-level functions



## Figure-level functions vs axes-level functions

```
fig, ax = plt.subplots(1,2, figsize=(10,6))
sns.scatterplot(data=df, x="aspect_ratio", y="intensity_mean", hue = 'file_name', ax=ax[0])
sns.histplot(data = df, x="intensity_mean", hue="file_name", ax=ax[1])
```

<AxesSubplot:xlabel='intensity\_mean', ylabel='Count'>

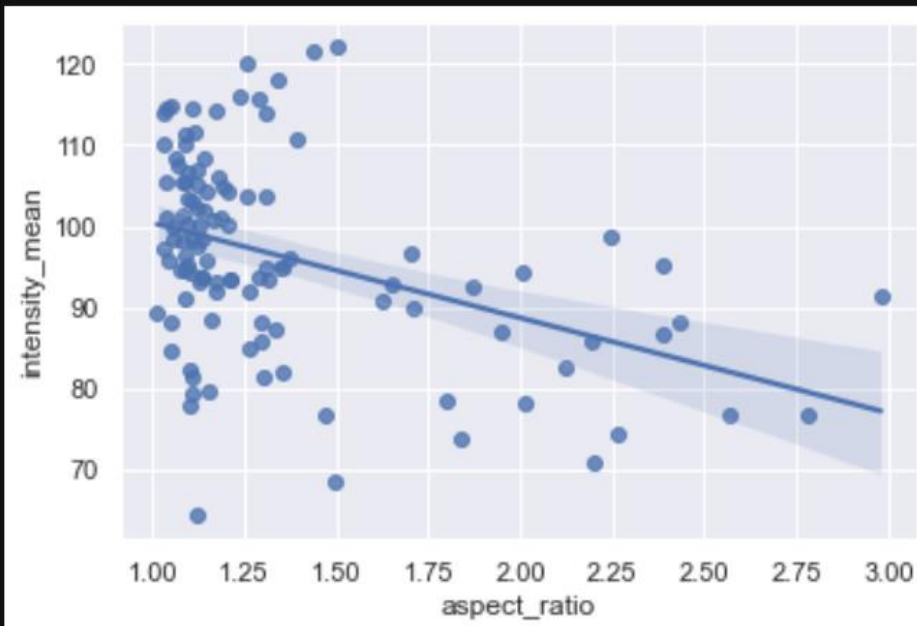


## Adding a line regression to a scatter plot

### Axes-level

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

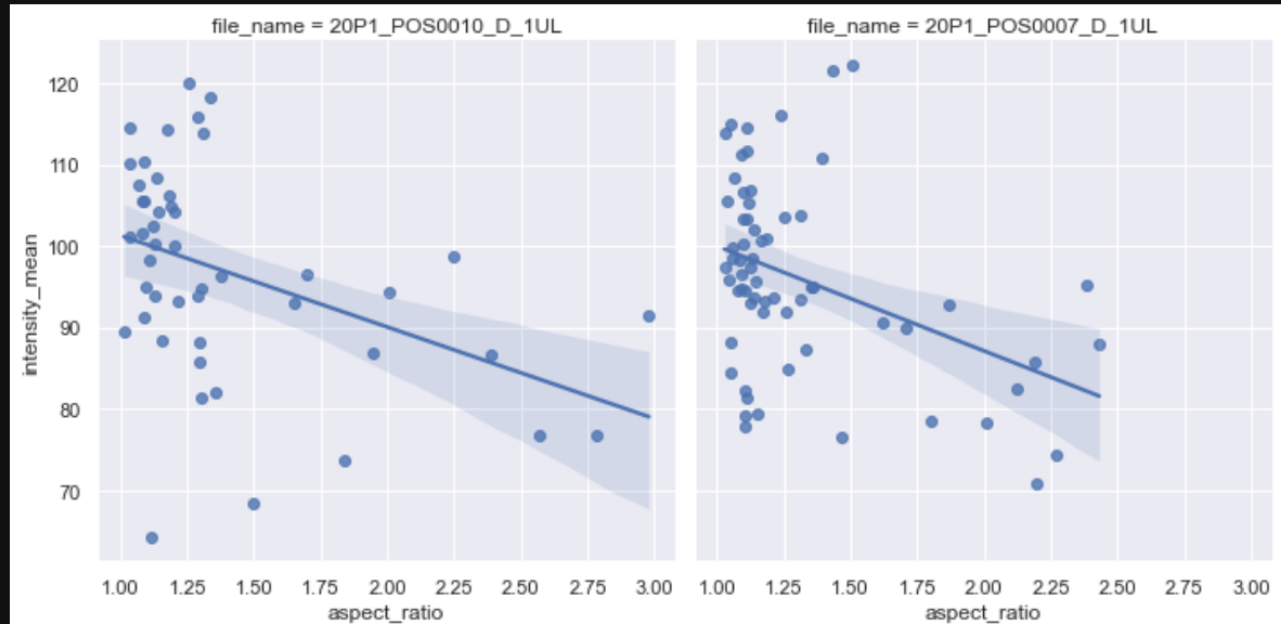
```
<AxesSubplot:xlabel='aspect_ratio', ylabel='intensity_mean'>
```



### Figure-level

```
sns.lmplot(data = df,  
           x = "aspect_ratio",  
           y = "intensity_mean",  
           col = "file_name")
```

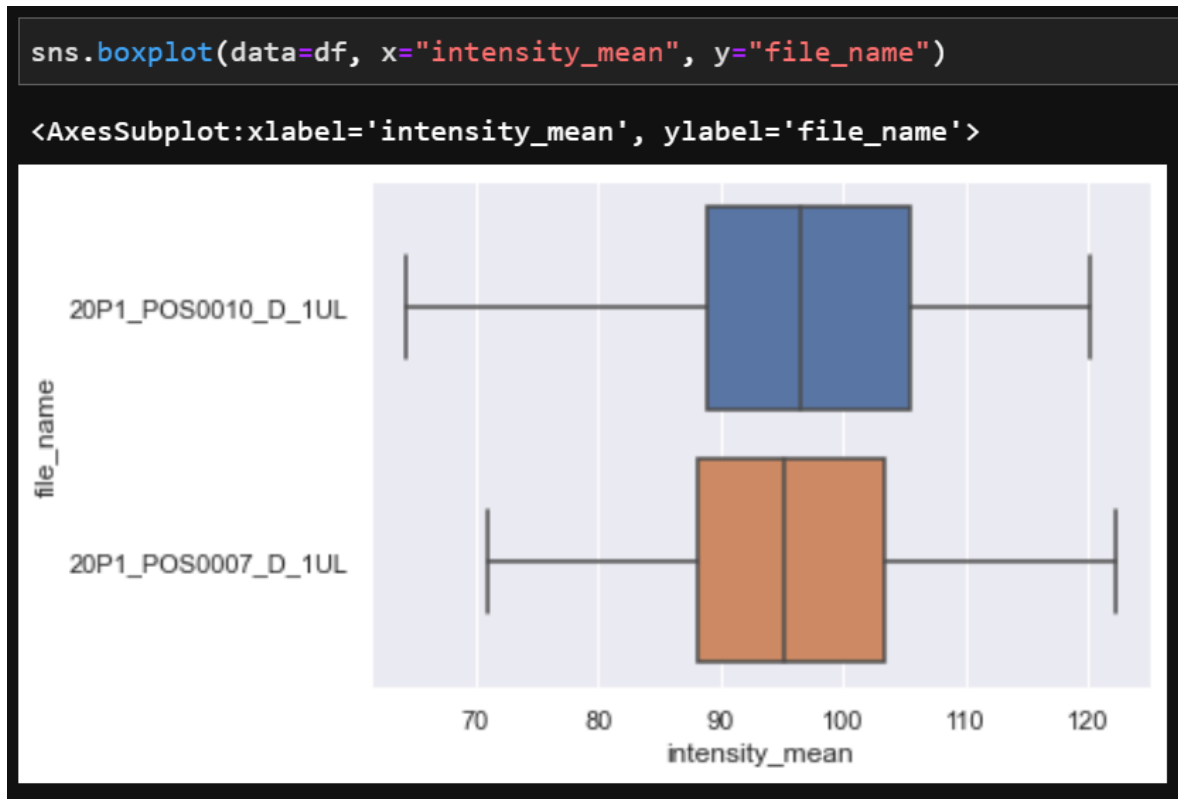
```
<seaborn.axisgrid.FacetGrid at 0x1b97a479970>
```



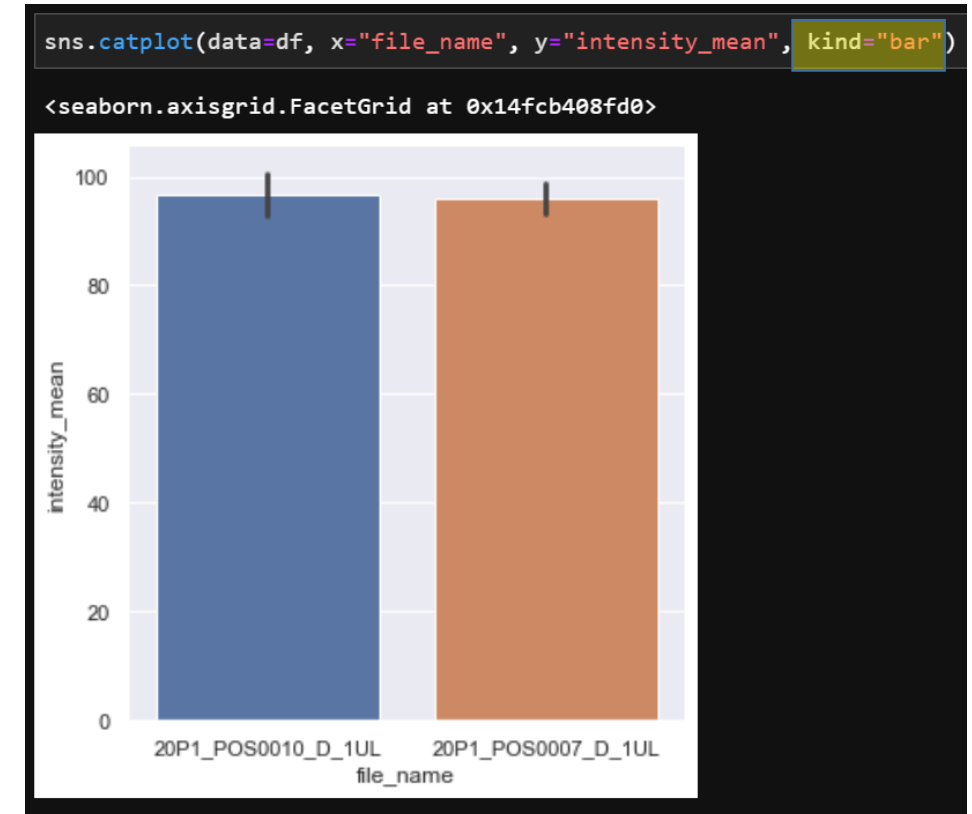


## Boxplots

### Axes-level



### Figure-level



## Histograms

Axes-level

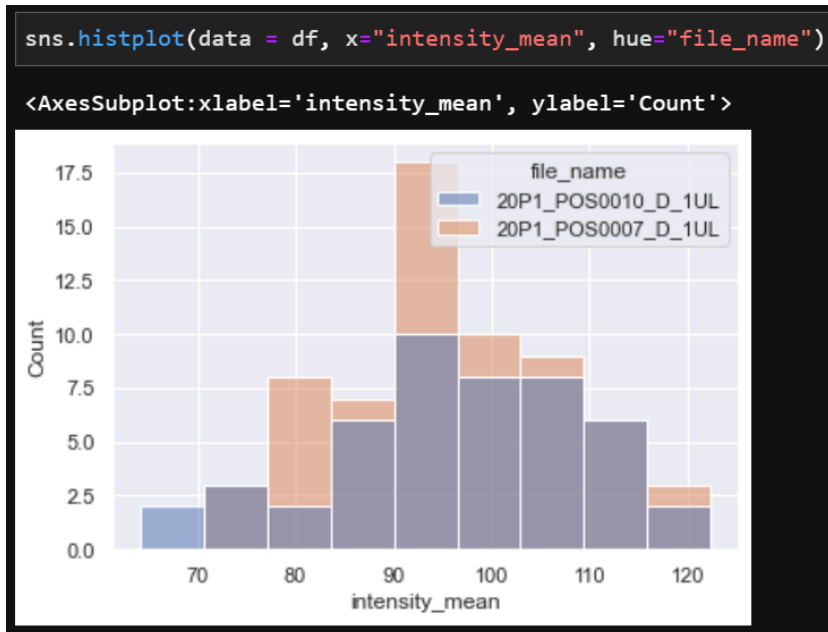
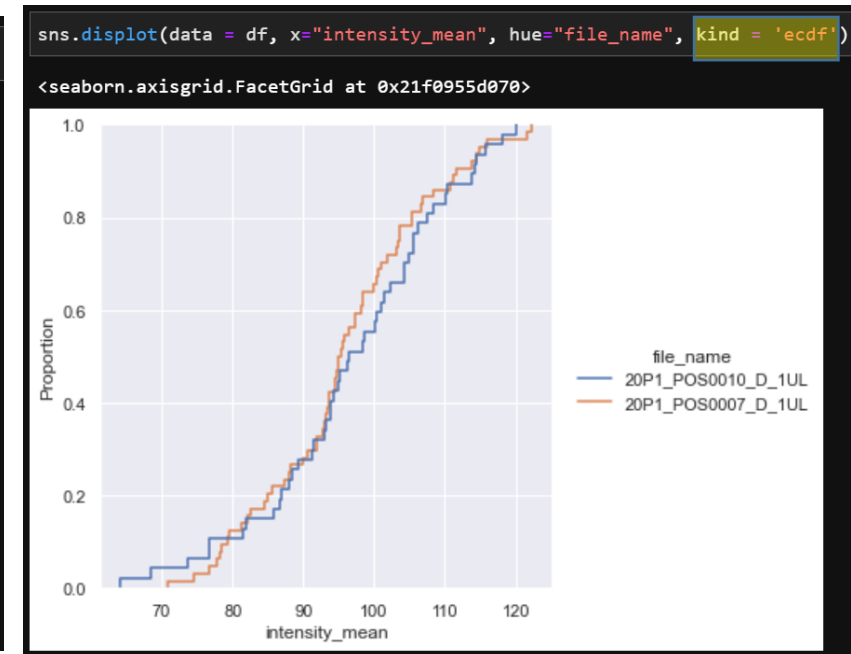
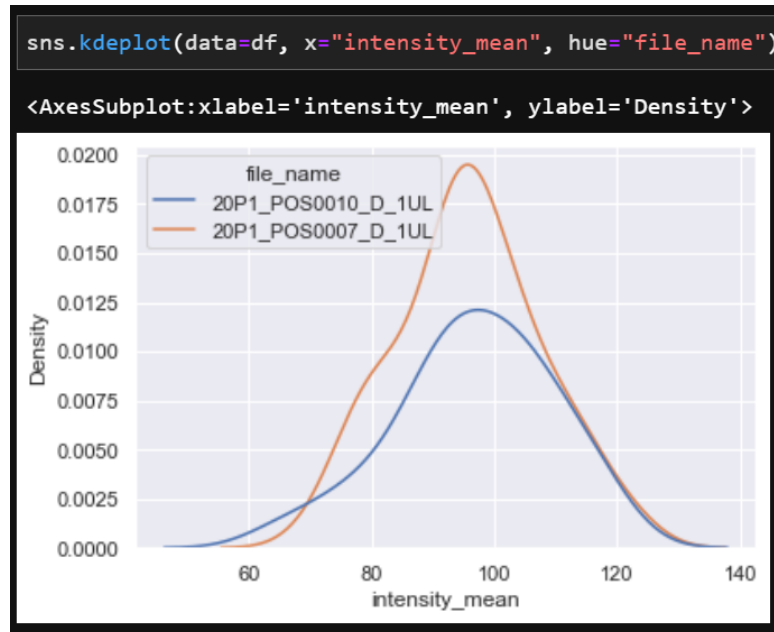


Figure-level

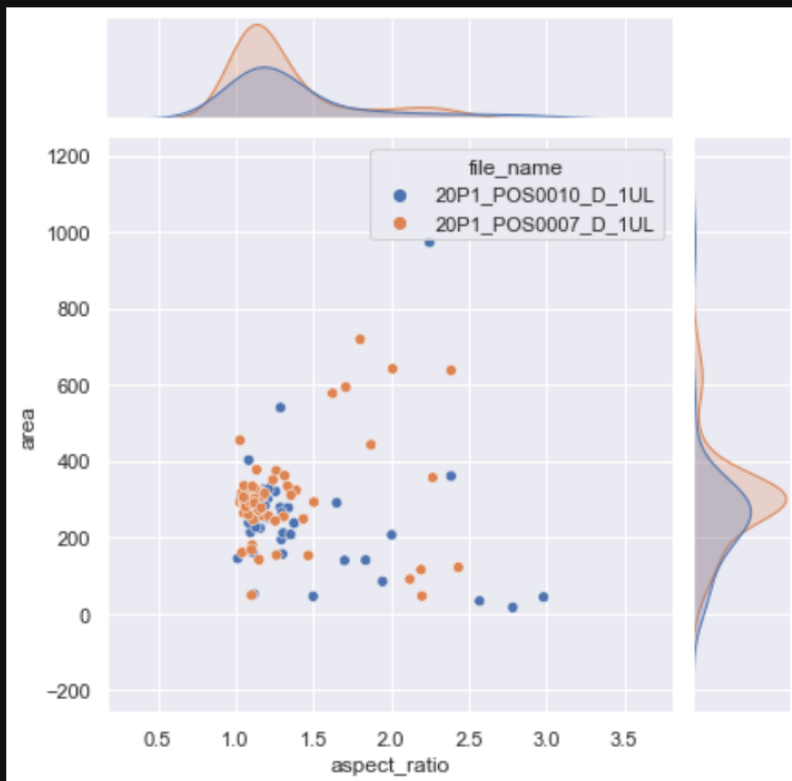


## Joint distributions and pair-plotting multiple properties

### Figure-level

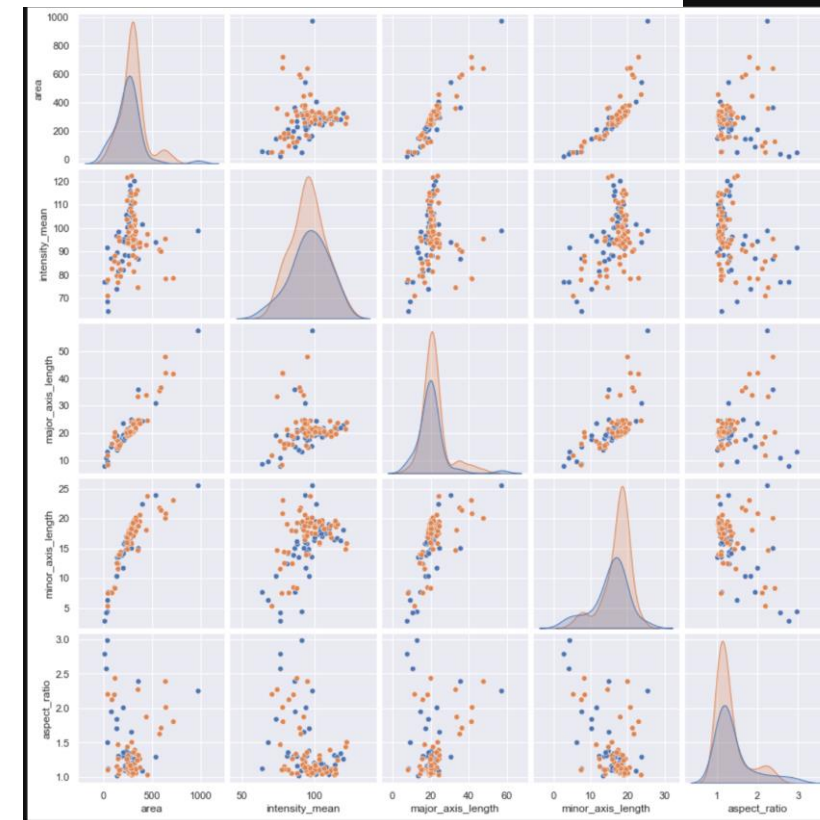
```
sns.jointplot(data=df, x="aspect_ratio", y="area", hue = 'file_name')
```

<seaborn.axisgrid.JointGrid at 0x12edd860970>



### Figure-level

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



## Pros and cons of Figure-level functions:

Advantages	Drawbacks
Easy faceting by data variables	Many parameters not in function signature
Legend outside of plot by default	Cannot be part of a larger matplotlib figure
Easy figure-level customization	Different API from matplotlib
Different figure size parameterization	Different figure size parameterization