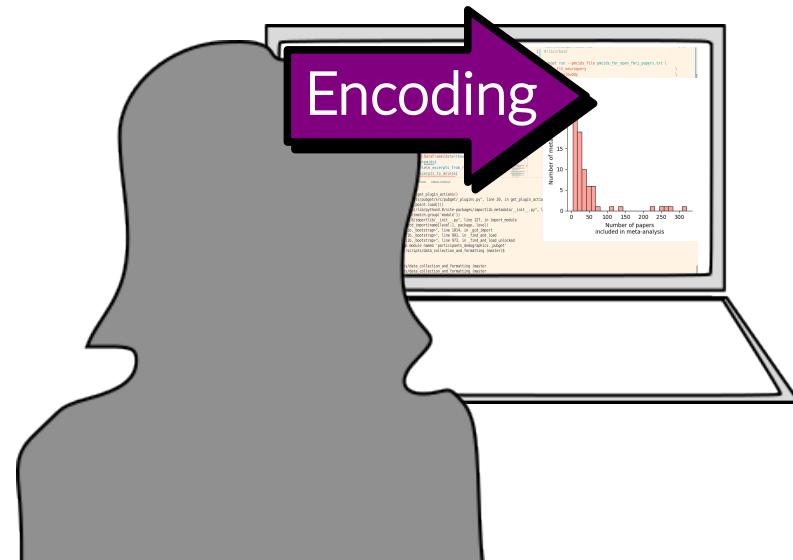


Introduction to Data Visualization

Part 2: Encoding

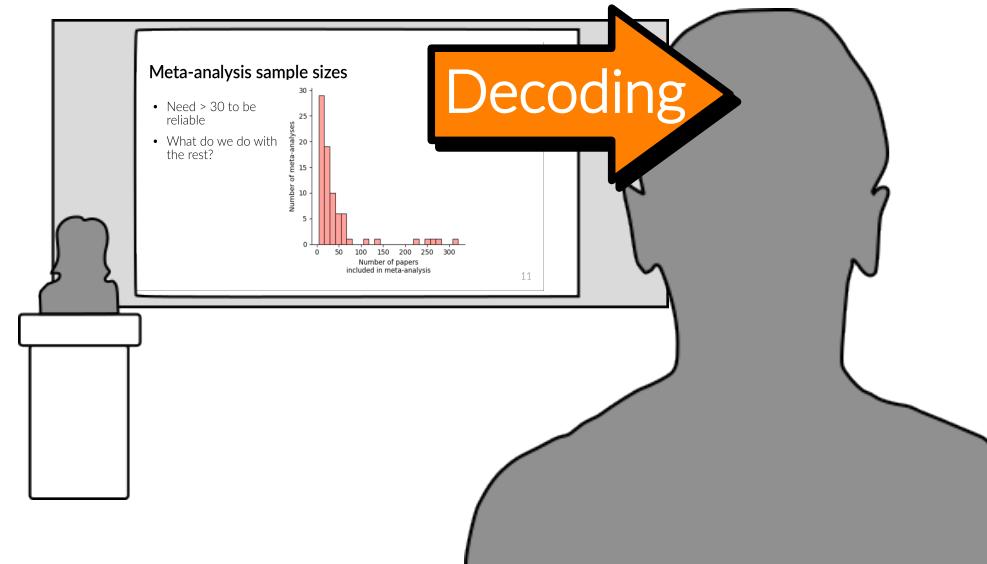
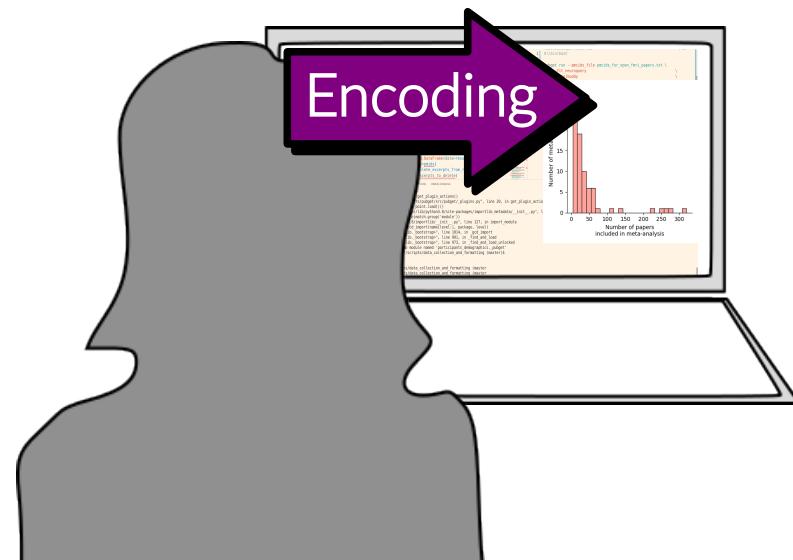
A large grey silhouette of a person's head and shoulders is facing right. Inside the head, a purple arrow points from the word "Encoding" to a computer monitor. The monitor displays a histogram titled "Number of meta-analyses included in meta-analysis". The x-axis is labeled "Number of papers included in meta-analysis" and ranges from 0 to 300. The y-axis is labeled "Number of meta-analyses" and ranges from 0 to 15. The distribution is skewed right, with the highest frequency of 15 meta-analyses occurring at 50 included papers.

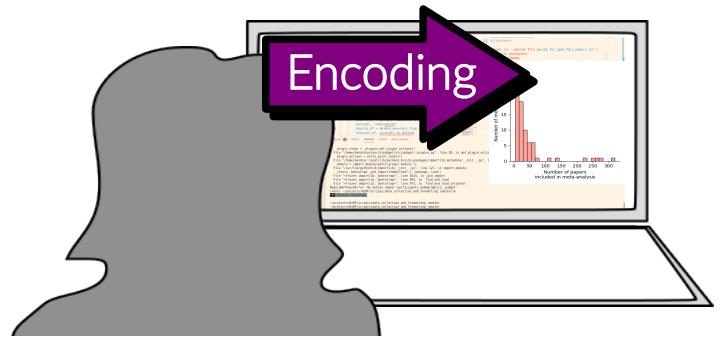
Number of papers included in meta-analysis	Number of meta-analyses
50	15
75	10
100	5
125	3
150	2
175	1
200	1
225	1
250	1
275	1
300	1

Kendra Oudyk

Goal

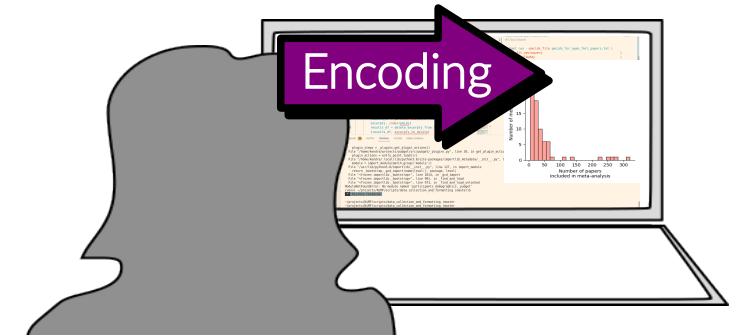
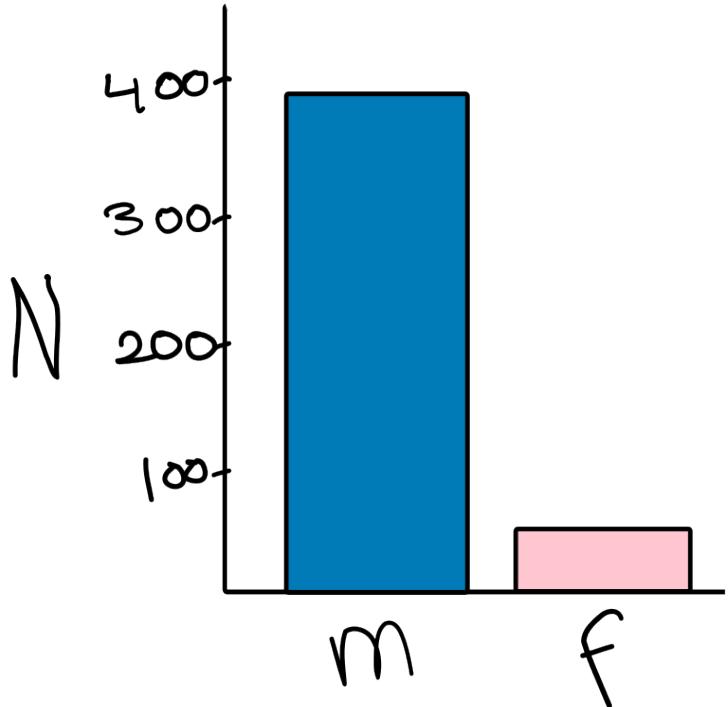
Use principles of visual **encoding** and **decoding**
to **efficiently** create visualizations
that are **effective** and **reproducible**





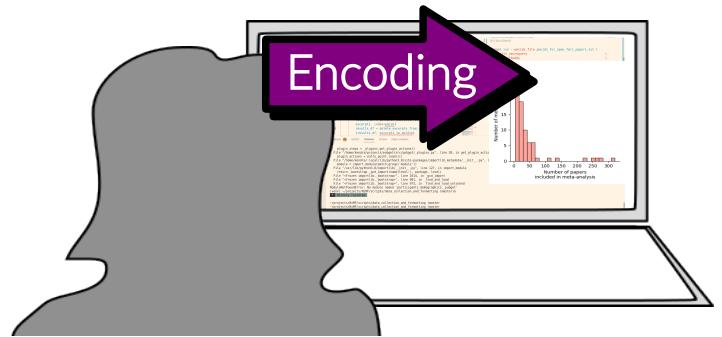
To program a figure efficiently and reproducibly, we should understand

- **Some computer graphics**
 - How the computer makes figure
- **Matplotlib basics**
 - What you and matplotlib need to do
- **Higher-level libraries**
 - More complex visualizations



To program a figure efficiently and reproducibly, we should understand

- Some computer graphics
 - How the computer makes figure
- Matplotlib basics
 - What you and matplotlib need to do
- Higher-level libraries
 - More complex visualizations



To program a figure efficiently and reproducibly, we should understand

- Some computer graphics
 - How the computer makes figure
- Matplotlib basics
 - What you and matplotlib need to do
- Higher-level libraries
 - More complex visualizations

File formats

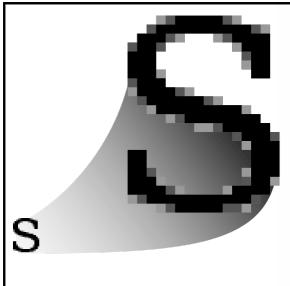
JPG

GIF

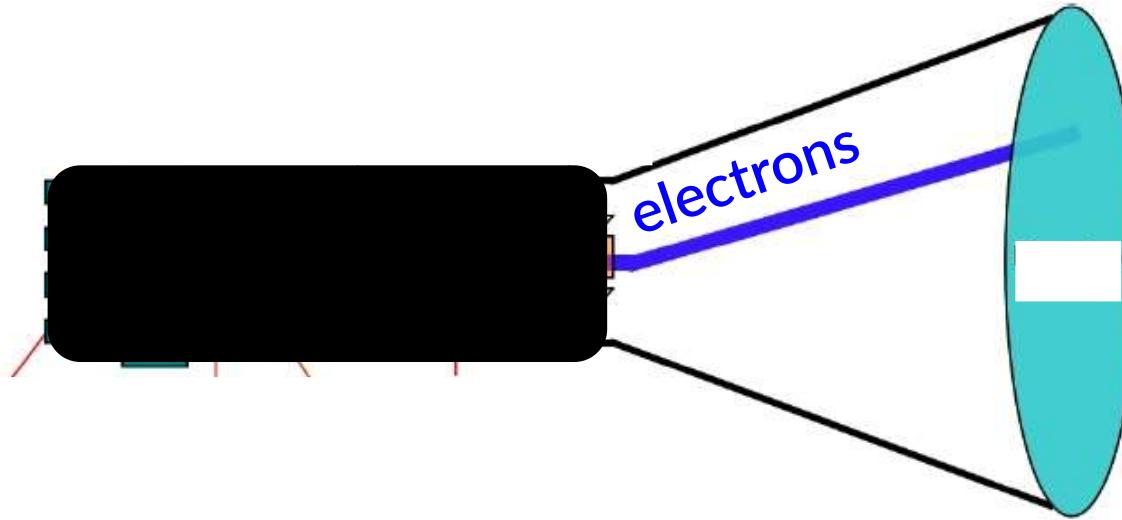
PNG

SVG

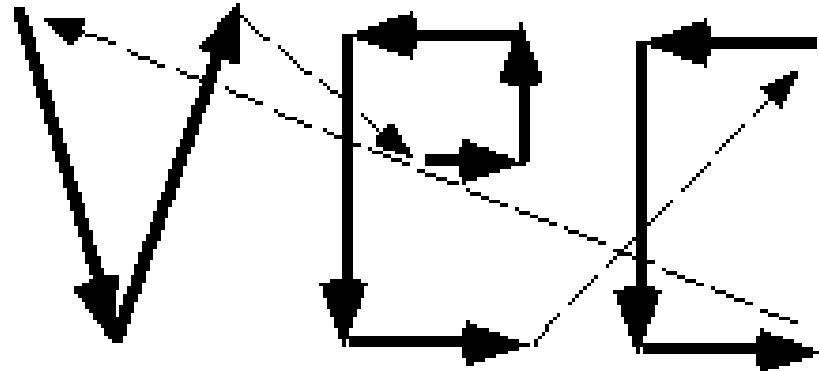
File formats

		JPG	GIF	PNG	SVG
Mathematically-defined shapes		VECTOR			
Pixels		RASTER			
		✓	✓	✓	✓

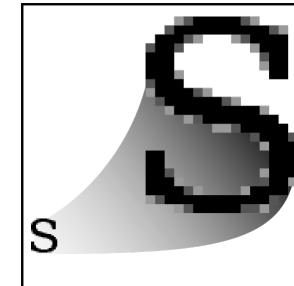
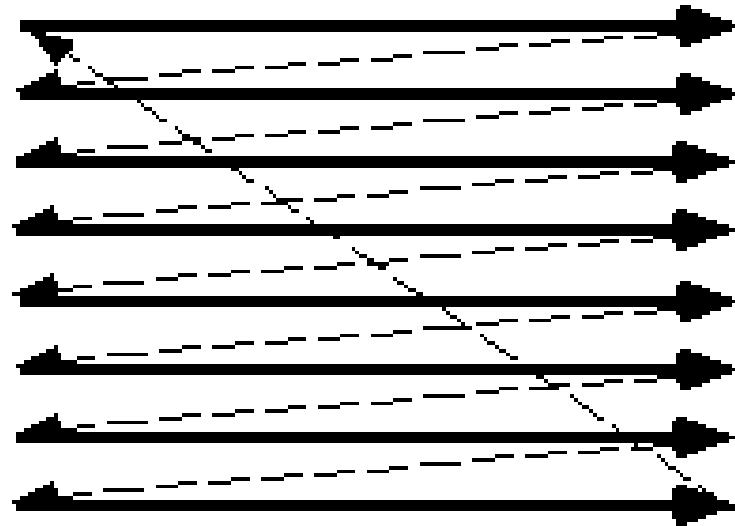
Cathode ray tube



Vector scan



Raster scan



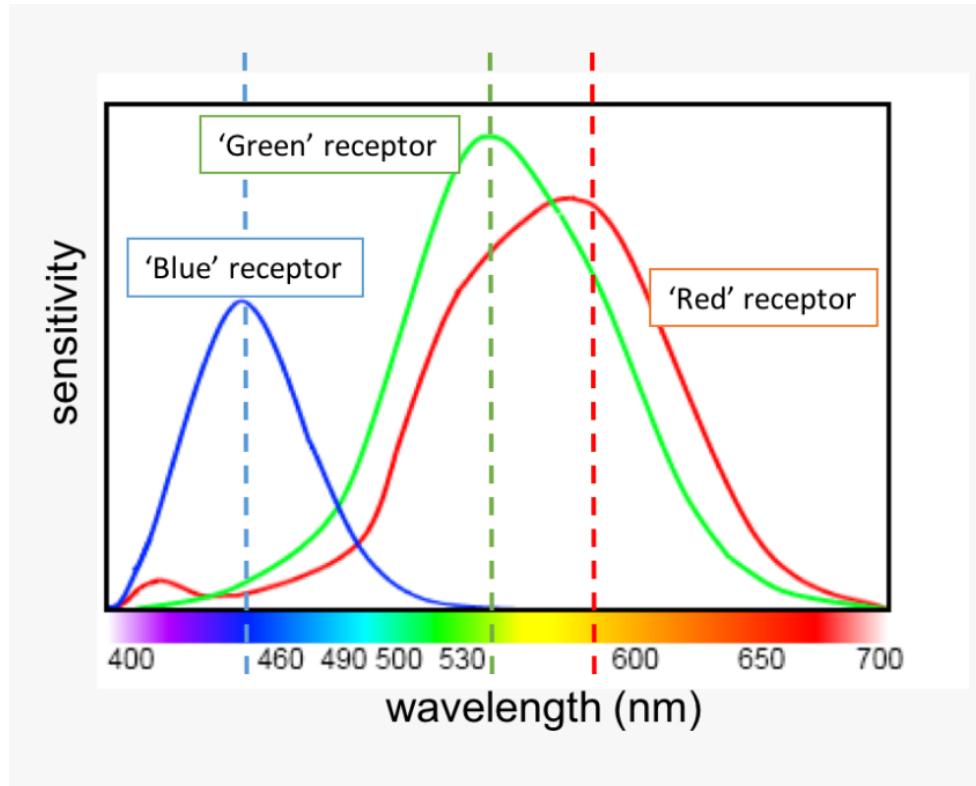
File formats

	JPG	GIF	PNG	SVG
VECTOR				✓
RASTER	✓	✓	✓	
TRANSPARENCY			✓	✓
ANIMATION		✓	✓	✓
LOSSY	✓			

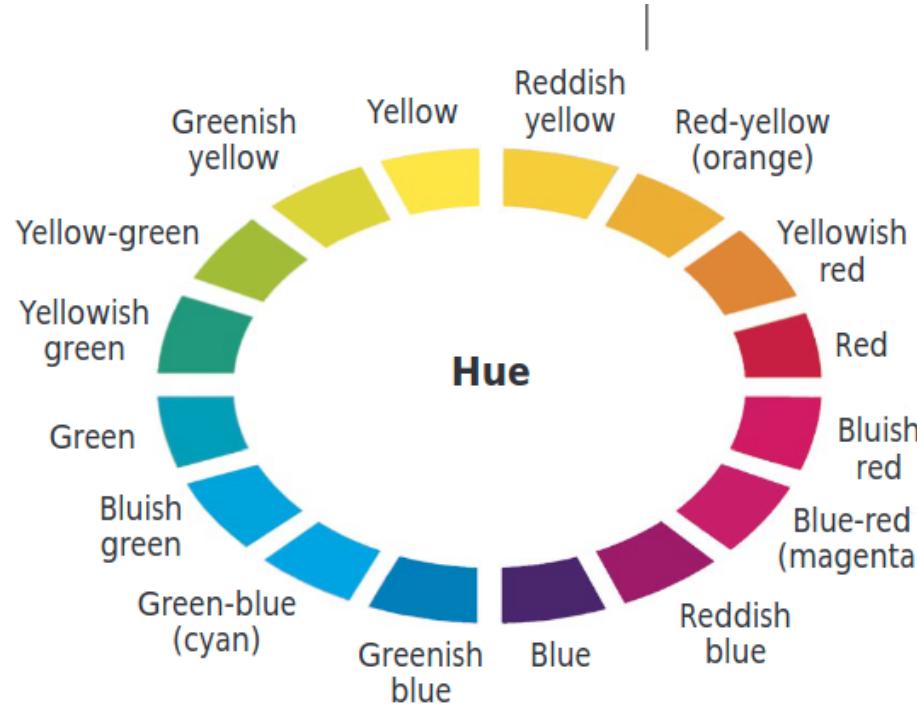
Color: salient but complicated



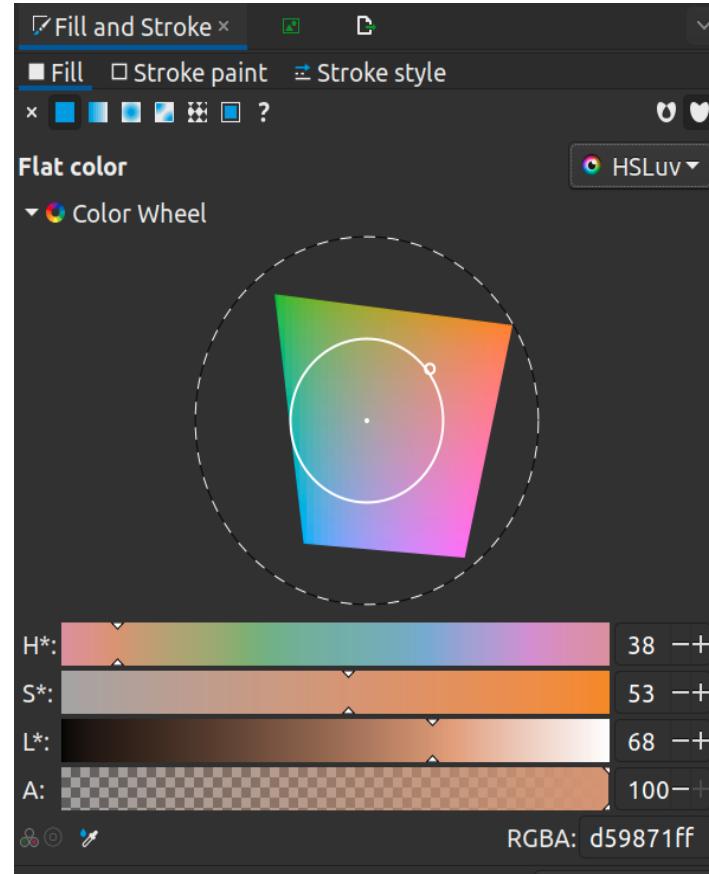
RGB: Types of photoreceptors in the retina



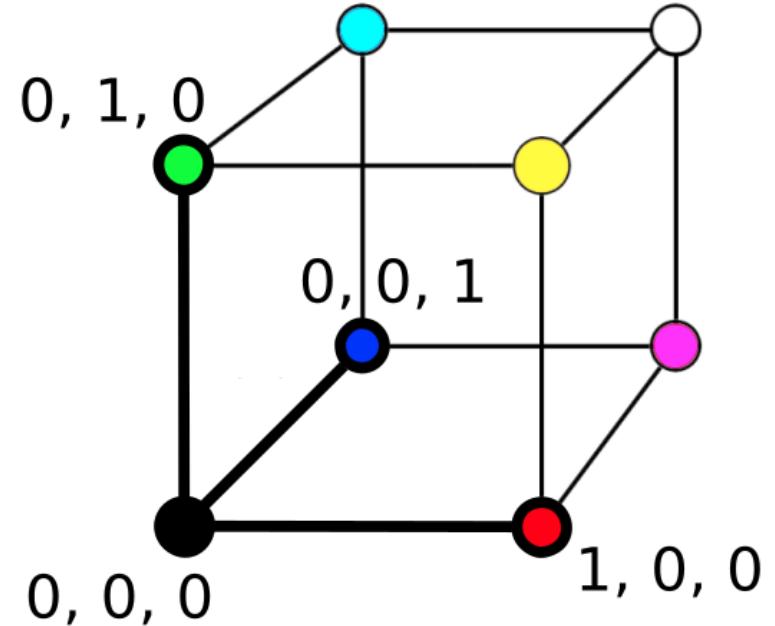
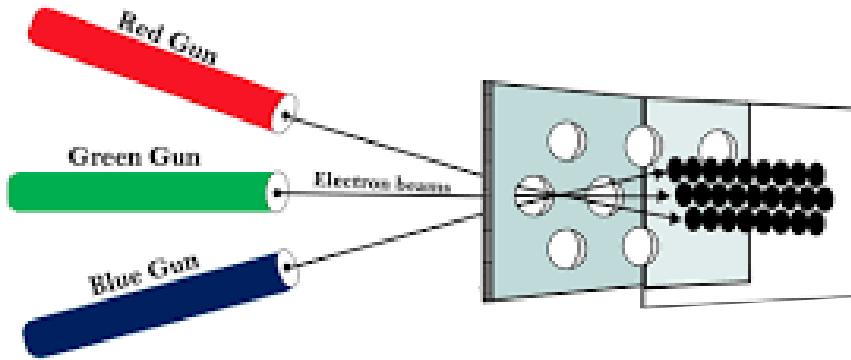
Hue: how we casually talk about color



HSLuv: how we talk about color perception



RGB: How a computer produces color





Section Navigation

- | | |
|----------------------------------|---|
| Introductory | ▼ |
| Intermediate | ▼ |
| Advanced | ▼ |
| Colors | ^ |
| Specifying colors | |
| Customized Colorbars Tutorial | |
| Creating Colormaps in Matplotlib | |
| Colormap Normalization | |
| Choosing Colormaps in Matplotlib | |
| Text | ▼ |
| Toolkits | ▼ |
| Provisional | |

Specifying colors

Color formats

Matplotlib recognizes the following formats to specify a color.

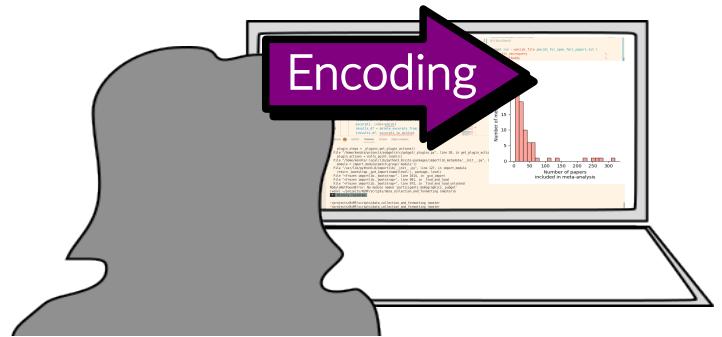
Format	Example
RGB or RGBA (red, green, blue, alpha) tuple of float values in a closed interval [0, 1].	<ul style="list-style-type: none">(0.1, 0.2, 0.5)(0.1, 0.2, 0.5, 0.3)
Case-insensitive hex RGB or RGBA string.	<ul style="list-style-type: none">'#0f0f0f''#0f0f0f80'
Case-insensitive RGB or RGBA string equivalent hex shorthand of duplicated	<ul style="list-style-type: none">'abc' as '#aabbcc''fb1' as '#ffbb11'

 On this page

- Color formats
 - Transparency
 - "**CN**" color selection
 - Comparison between X11/CSS4 and xkcd colors

TL;DR

- Images have different properties/features because of **how they're made**
- The way a computer produces color (**RGB**) is not intuitive
 - You'll probably pick colors using named colors or an HSL tool



To program a figure efficiently and reproducibly, we should understand

- Some computer graphics
 - How the computer makes figure
- Matplotlib basics
 - What you and matplotlib need to do
- Higher-level libraries
 - More complex visualizations

Activities Firefox Web Browser • May 4 2:37 PM • 61% □

open shot inc Capability to i Artist tutorial free clipart pa Art Artist Cra python - Ad X + - ×

https://stackoverflow.com/questions/19125722/adding-a-matplotlib-legend 80% ☆

stackoverflow About Products For Teams Search... Log in Sign up Ask Question

Home PUBLIC Questions Tags Users Companies COLLECTIVES Explore Collectives TEAMS Stack Overflow for Teams – Start collaborating and sharing organizational knowledge. Create a free Team Why Teams?

How can one create a legend for a line graph in Matplotlib's PyPlot without creating any extra variables? 469 Please consider the graphing script below:

```
if __name__ == '__main__':
    PyPlot.plot(total_lengths, sort_times_bubble, 'b-',
                total_lengths, sort_times_ins, 'r-',
                total_lengths, sort_times_merge_r, 'g+',
                total_lengths, sort_times_merge_i, 'p-', )
    PyPlot.title("Combined Statistics")
    PyPlot.xlabel("Length of list (number)")
    PyPlot.ylabel("Time taken (seconds)")
    PyPlot.show()
```

As you can see, this is a very basic use of matplotlib's PyPlot. This ideally generates a graph like the one below:

Combined Statistics

The Overflow Blog

- Don't panic! A playbook for managing any production incident
- How to land a job in climate tech

Featured on Meta

- New blog post from our CEO Prashanth: Community is the future of AI
- We are updating our Code of Conduct and we would like your feedback
- Temporary policy: ChatGPT is banned
- Content Discovery initiative April 13 update: Related questions using a...
- The [connect] tag is being burninated
- Removal of the Census badge

Linked

- Add legends in pyplot with Pandas Dataframe data

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up with email G Sign up with Google S Sign up with GitHub F Sign up with Facebook

Activities Firefox Web Browser • May 4 2:41 PM • 60% ▾

File Home About Products For Teams Search... Log in Sign up

Here's an example to help you out ...

27

```
fig = plt.figure(figsize=(10,5))
ax = fig.add_subplot(111)
ax.set_title('ADR vs Rating (CS:GO)')
ax.scatter(x=data[:,0],y=data[:,1],label='Data')
plt.plot(data[:,0], m*x + b, color='red', label='Our Fitting Line')
ax.set_xlabel('ADR')
ax.set_ylabel('Rating')
ax.legend(loc='best')
plt.show()
```

ADR vs Rating (CS:GO)

The scatter plot displays the relationship between ADR (X-axis) and Rating (Y-axis). The X-axis ranges from approximately 20 to 140, and the Y-axis ranges from 0 to 300. Blue dots represent individual data points labeled 'Data'. A solid red line represents the linear regression fit labeled 'Our Fitting Line'. The plot title is 'ADR vs Rating (CS:GO)'.

Share Improve this answer Follow answered Dec 6, 2017 at 7:00 Akash Akash Kandpal

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up with email Sign up with Google Sign up with GitHub Sign up with Facebook

Activities Firefox Web Browser • May 4 2:42 PM • 59%

File Home About Products For Teams Search... Log in Sign up

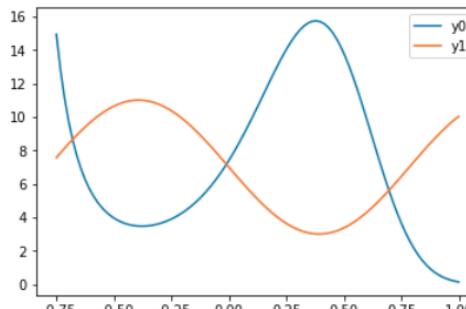
Add a comment

You can access the Axes instance (`ax`) with `plt.gca()`. In this case, you can use

62 `plt.gca().legend()`

You can do this either by using the `label=` keyword in each of your `plt.plot()` calls or by assigning your labels as a tuple or list within `legend`, as in this working example:

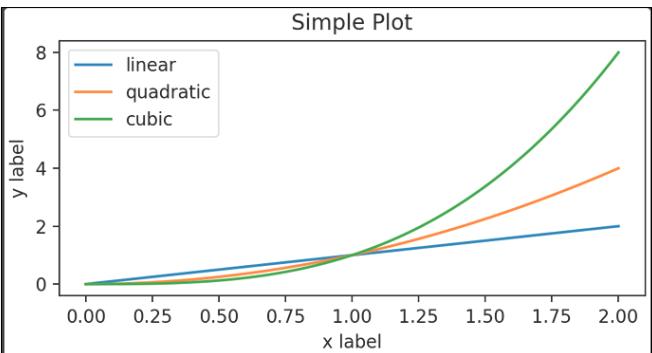
```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-0.75, 1, 100)
y0 = np.exp(2 + 3*x - 7*x**3)
y1 = 7 - 4*np.sin(4*x)
plt.plot(x, y0, x, y1)
plt.gca().legend(['y0', 'y1'])
plt.show()
```



Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up with email Sign up with Google Sign up with GitHub Sign up with Facebook

Coding styles



Explicit (object-oriented)

```
x = np.linspace(0, 2, 100) # Sample data.

# Note that even in the OO-style, we use `plt.subplots(figsize=(5, 2.7), layout='constrained')
fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
ax.plot(x, x, label='linear') # Plot some data on the axes.
ax.plot(x, x**2, label='quadratic') # Plot more data on the axes...
ax.plot(x, x**3, label='cubic') # ... and some more.
ax.set_xlabel('x label') # Add an x-label to the axes.
ax.set_ylabel('y label') # Add a y-label to the axes.
ax.set_title("Simple Plot") # Add a title to the axes.
ax.legend() # Add a legend.
```

Implicit

```
x = np.linspace(0, 2, 100) # Sample data.

plt.figure(figsize=(5, 2.7), layout='constrained')
plt.plot(x, x, label='linear') # Plot some data on the (implicit) axes.
plt.plot(x, x**2, label='quadratic') # etc.
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()
```

Some objects

Figure

- The top-level container for all elements of the plot

Axes

- An **Artist** subclass attached to a **Figure**
- Incl. All elements of an individual (sub-)plot

Artist

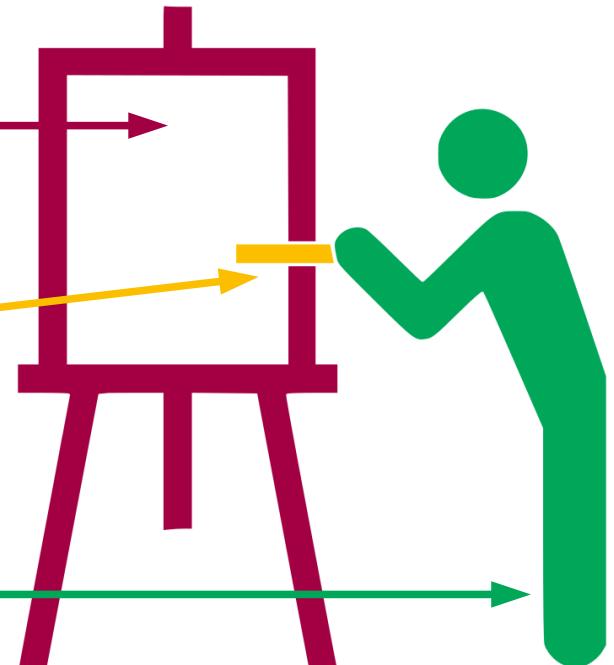
- Abstract base class
- Everything visible is a subclass of **Artist**

Explicit (object-oriented)

```
x = np.linspace(0, 2, 100) # Sample data.  
  
# Even in the OO-style, we use `plt.subplots` to create the Figure.  
fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')  
  
ax.plot(x, x, label='linear') # Plot some data on the axes.  
ax.plot(x, x**2, label='quadratic') # Plot more data on the axes...  
ax.plot(x, x**3, label='cubic') # ... and some more.  
ax.set_xlabel('x label') # Add an x-label to the axes.  
ax.set_ylabel('y label') # Add a y-label to the axes.  
ax.set_title("Simple Plot") # Add a title to the axes.  
ax.legend() # Add a legend.
```

3 layers of matplotlib

- “matplotlib.backend_bases.FigureCanvas”
is the **area** onto which the figure is drawn
 - matplotlib.backend_bases.Renderer
- is the object which knows how to **draw** on the FigureCanvas
- matplotlib.artist.Artist
- is the object that knows how to **use a renderer** to paint onto the canvas.”



**What you'll be working
with 95% of the time**

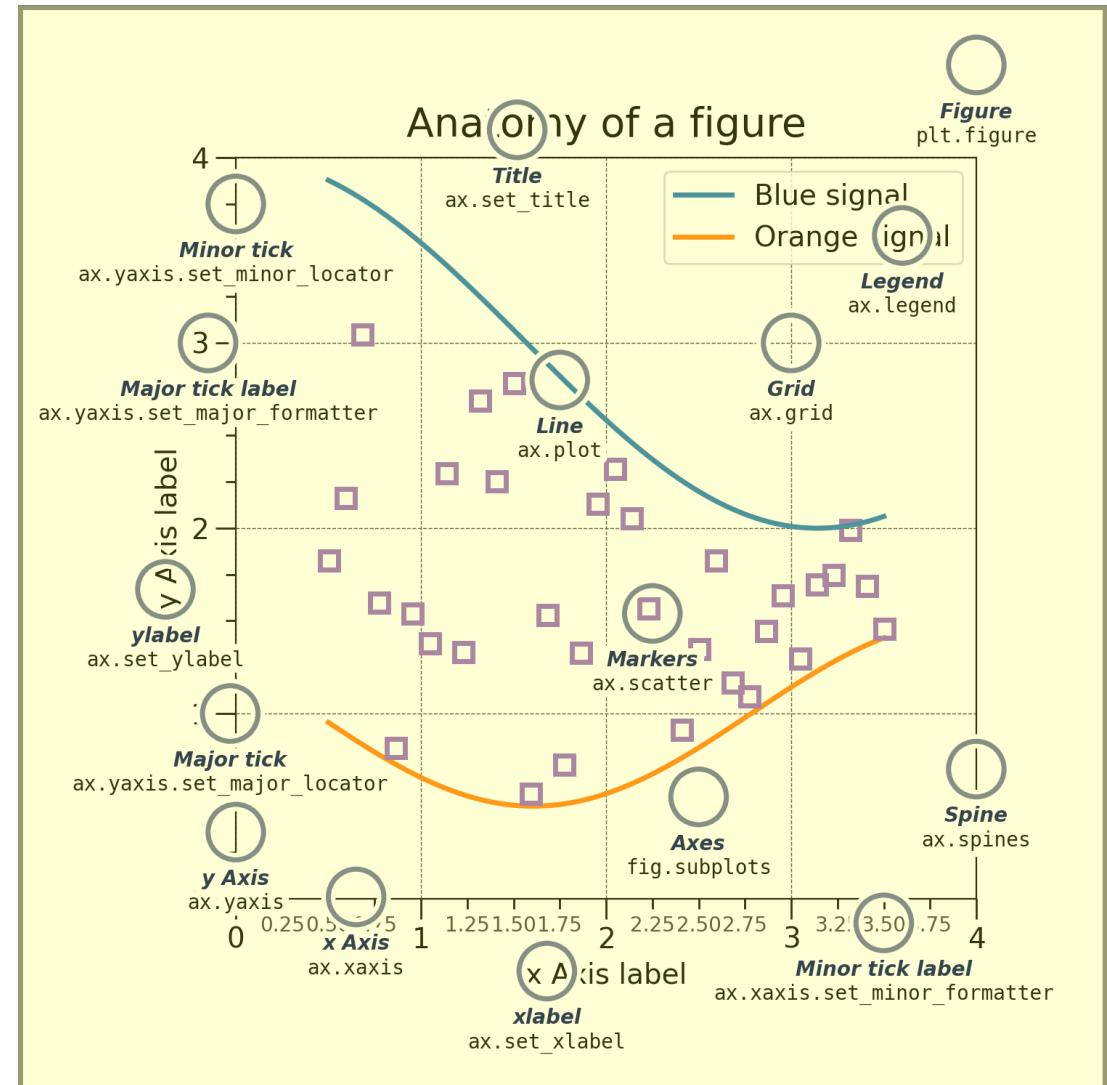
2 types of artists

1) Primatives: things to draw

- Line2D
- Rectangle
- Text
- AxesImage

2) Containers: where to draw them

- Figure
- Axes
- Axis



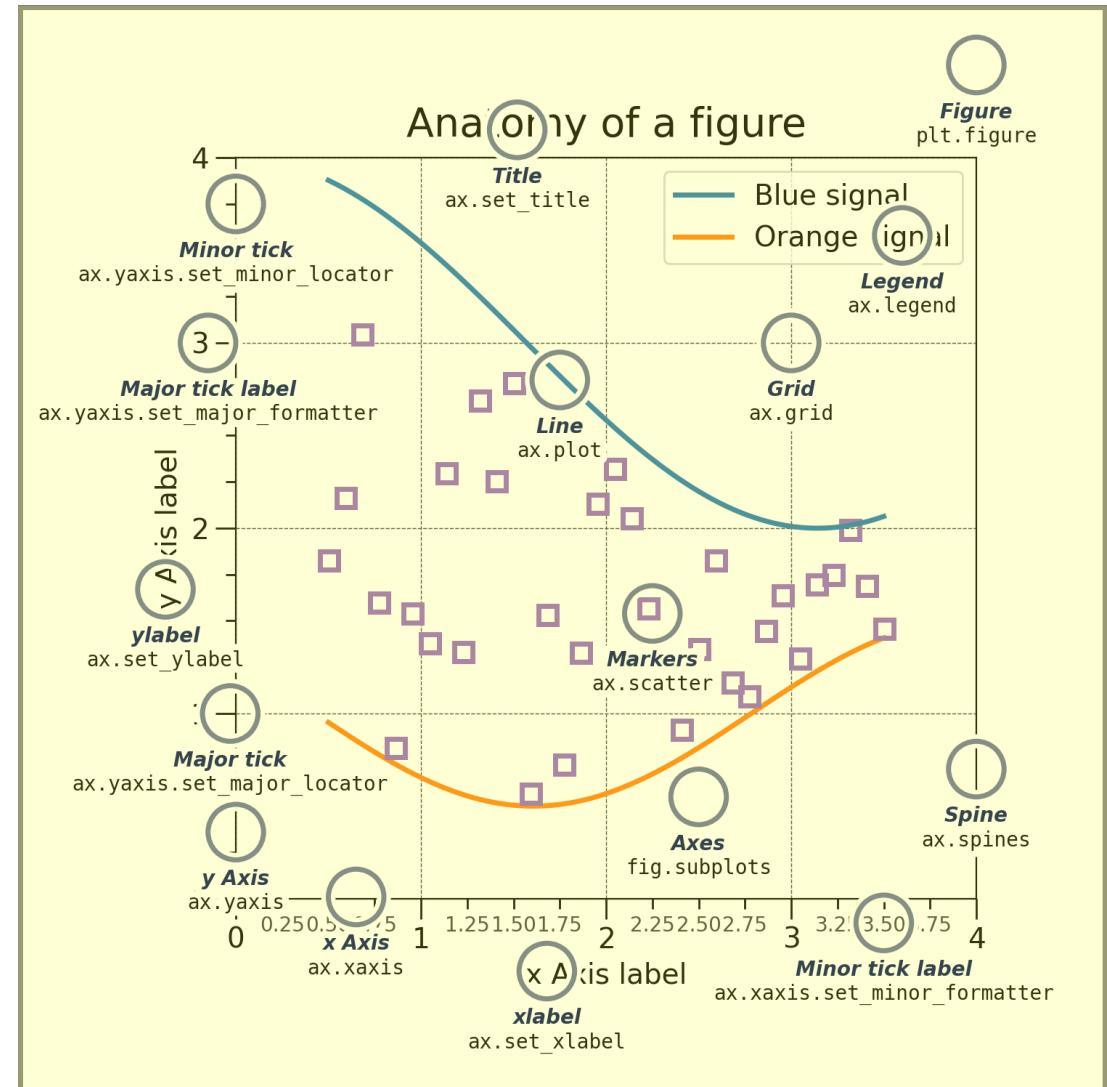
2 types of artists

1) Primatives: things to draw

- Line2D
 - Rectangle
 - Text
 - AxesImage

2) Containers: where to draw them

- Figure
 - Axes
 - Axis



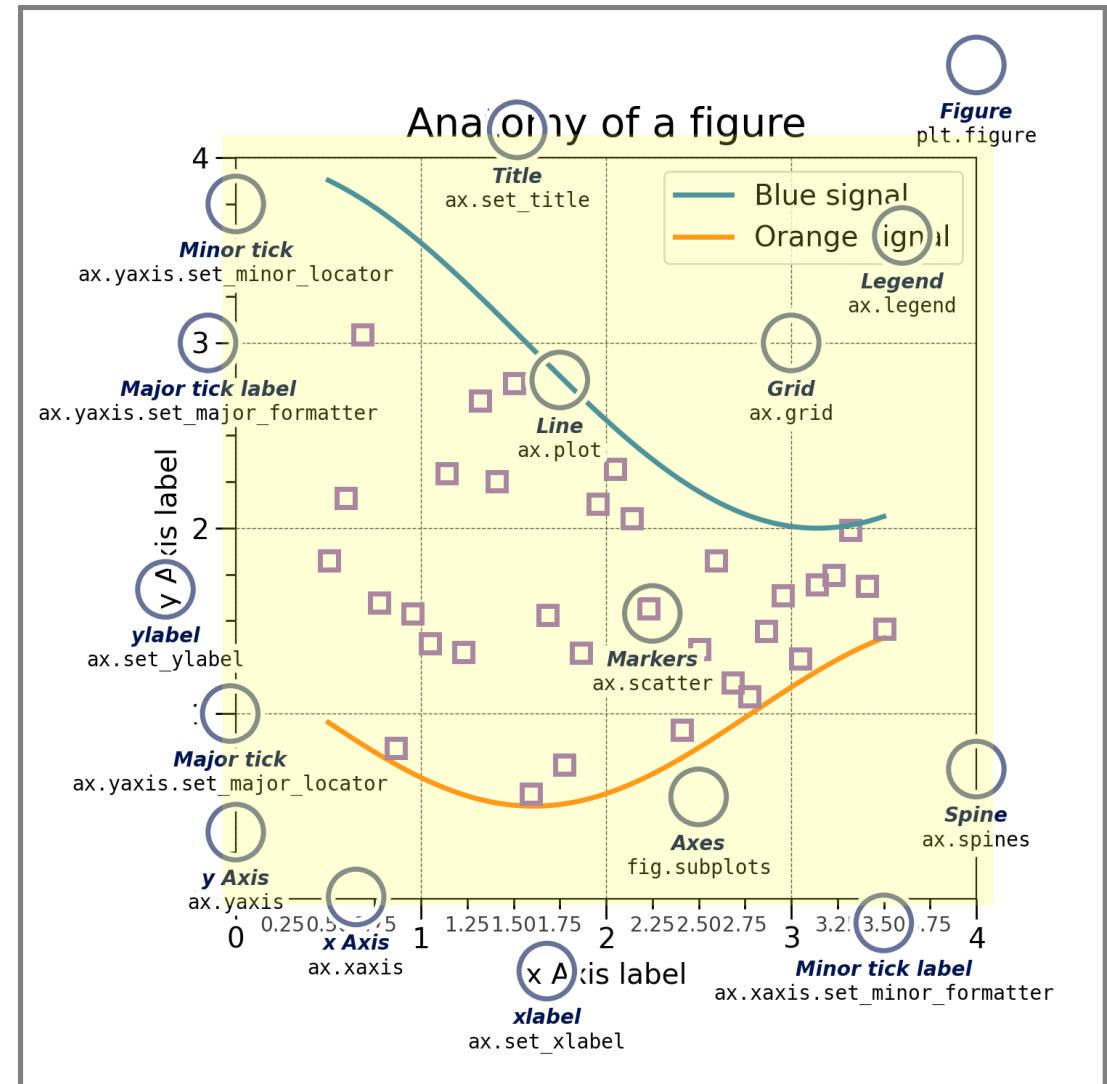
2 types of artists

1) Primitives: things to draw

- Line2D
- Rectangle
- Text
- AxesImage

2) Containers: where to draw them

- Figure
- Axes
- Axis



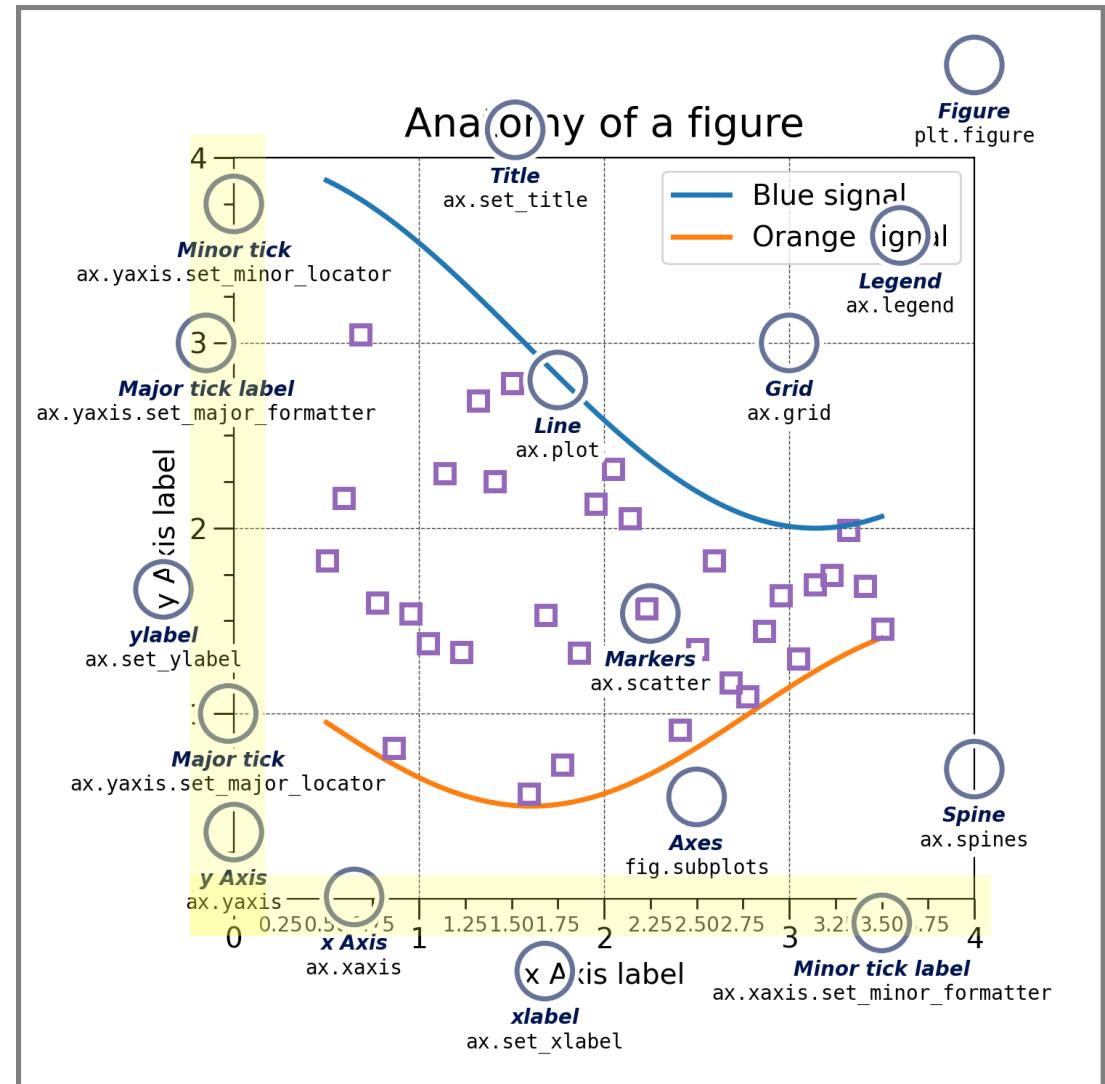
2 types of artists

1) Primatives: things to draw

- Line2D
- Rectangle
- Text
- AxesImage

2) Containers: where to draw them

- Figure
- Axes
- Axis



What actually is an Artist??

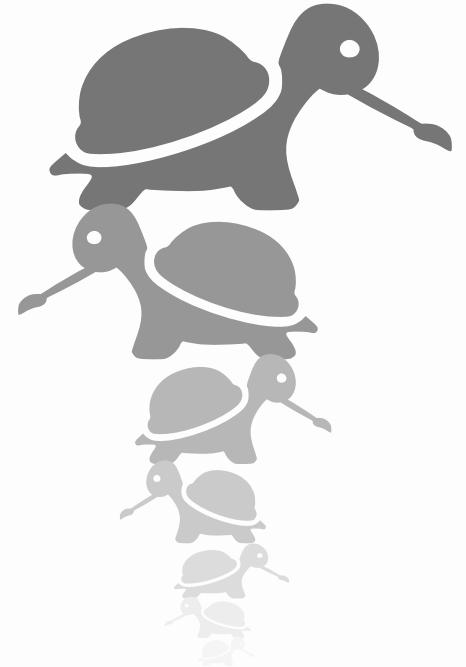
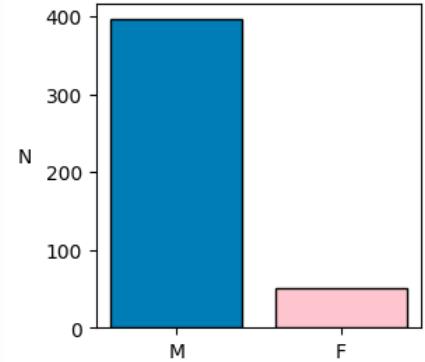
Artist class

```
class matplotlib.artist.Artist
```

Abstract base class for objects that render into a FigureCanvas.

Typically, all visible elements in a figure are subclasses of Artist.

TL;DR for our purposes,
it's Artists all the way down

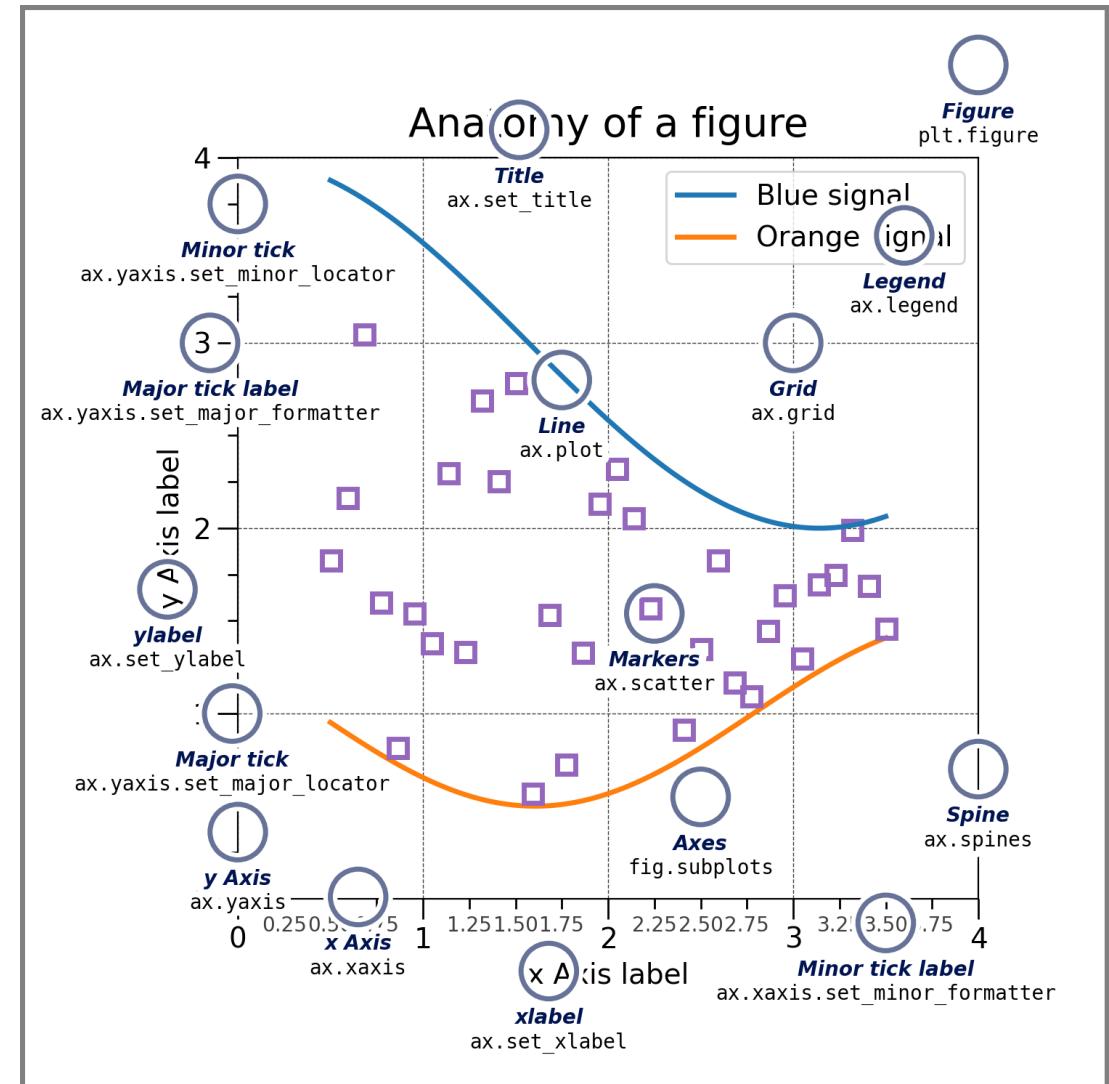


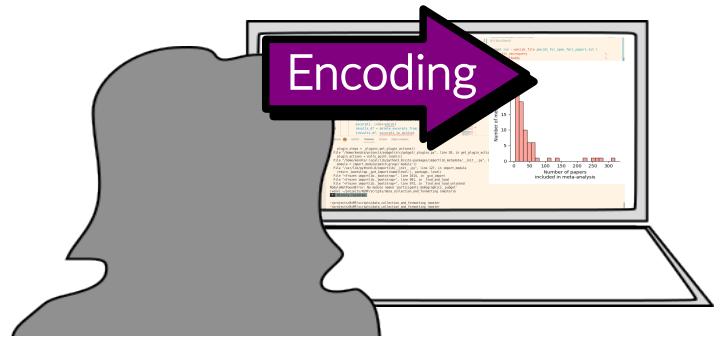
Axes helper methods for plot types

- `ax.plot()` # line graph
- `ax.bar()` # bar graph
- `ax.imshow()` # image

So you don't have to construct figures from shapes

Helper methods for customization

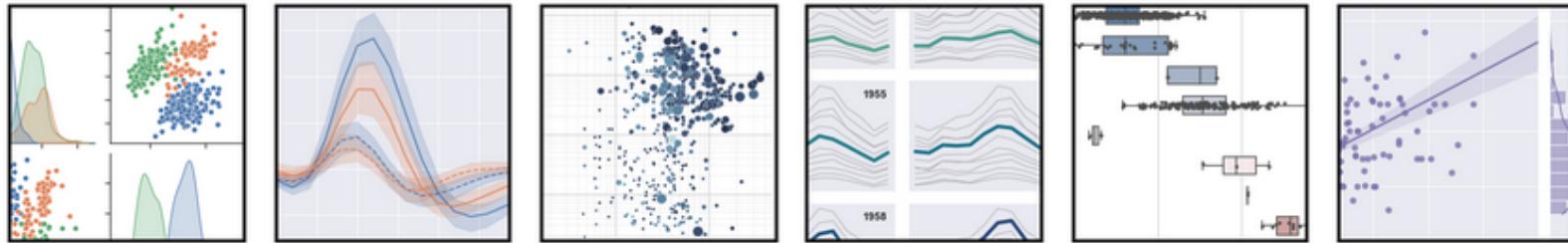




To program a figure efficiently and reproducibly, we should understand

- Some computer graphics
 - How the computer makes figure
- Matplotlib basics
 - What you and matplotlib need to do
- Higher-level libraries
 - More complex visualizations

seaborn: statistical data visualization



Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Features of Seaborn

- Works well with `pandas.DataFrame()` objects
- Default colors are from **perceptually uniform** pallettes
- Can **calculate stats** as well as visualize
- More easily visualize **complex data** (e.g, multivariate)

https://nilearn.github.io/stable/plotting/index.html

Nilearn

Search

Quickstart

Examples

User guide

1. Introduction

2. What is nilearn ?

3. Using nilearn for the first time

4. Machine learning applications to Neuroimaging

5. Decoding and MVPA:
predicting from brain images

6. Functional connectivity

7. Plotting brain images

In this section, we detail the general tools to visualize neuroimaging volumes and surfaces with nilearn.

Nilearn comes with plotting function to display brain maps coming from Nifti-like images, in the [nilearn.plotting](#) module.

Code examples

Nilearn has a whole section of the example gallery on plotting.

A small tour of the plotting functions can be found in the example [Plotting tools in nilearn](#).

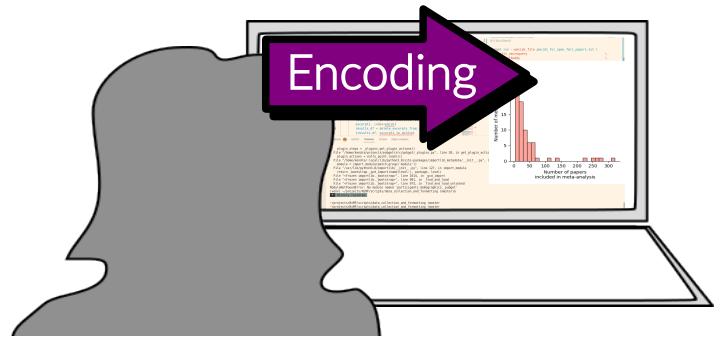
Finally, note that, as always in the nilearn documentation, clicking on a figure will take you to the code that generates it.

7.1. Different plotting functions

Nilearn has a set of plotting functions to plot brain volumes that are fined tuned to specific applications. Amongst other things, they use different heuristics to find cutting coordinates.

plot_anat

Plotting an anatomical image



To program a figure efficiently and reproducibly, we should understand

- **Some computer graphics**
 - How the computer makes figure
- **Matplotlib basics**
 - What you and matplotlib need to do
- **Higher-level libraries**
 - More complex visualizations