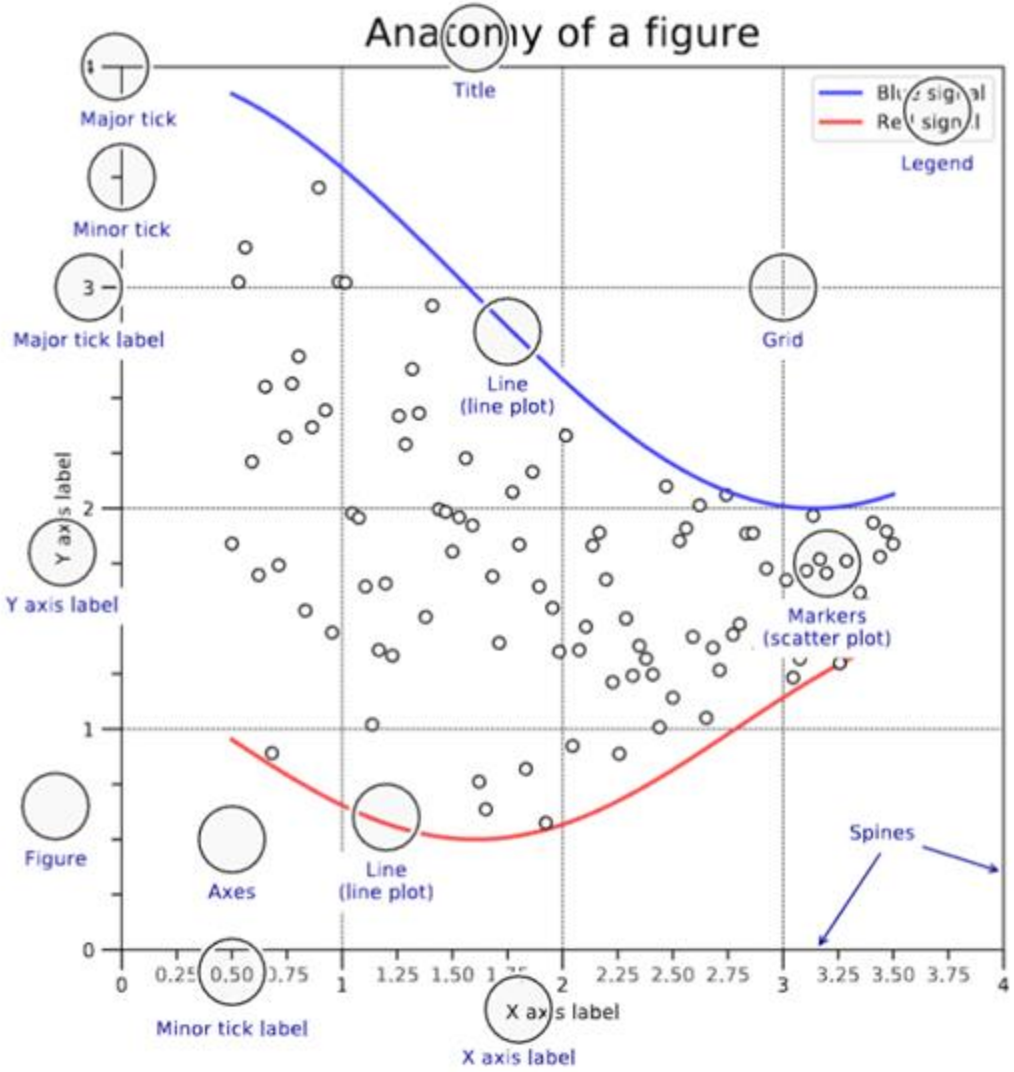
Matplotlib suggested practices

by Jian Huang
Chen lab

Outline

1. Basic overview on the concepts and hierarchy
2. Good practices for layout control
3. Dimension and resolution
4. Coordination system
5. Annotations
6. Other suggestions and animation

Basic concepts and Hierarchy



0 Figure: the canvas itself

(canvas size, facecolor, suptitle)

1 Axes: the plot region where your data is rendered (also called “subplot”)

(spines, ticks, labels, legend)

2 Axis: decorated spines, including xaxis and yaxis

(spine, major/minor ticks, ticks labels, axis labels)

3 Spine: lines connecting the axis tick marks

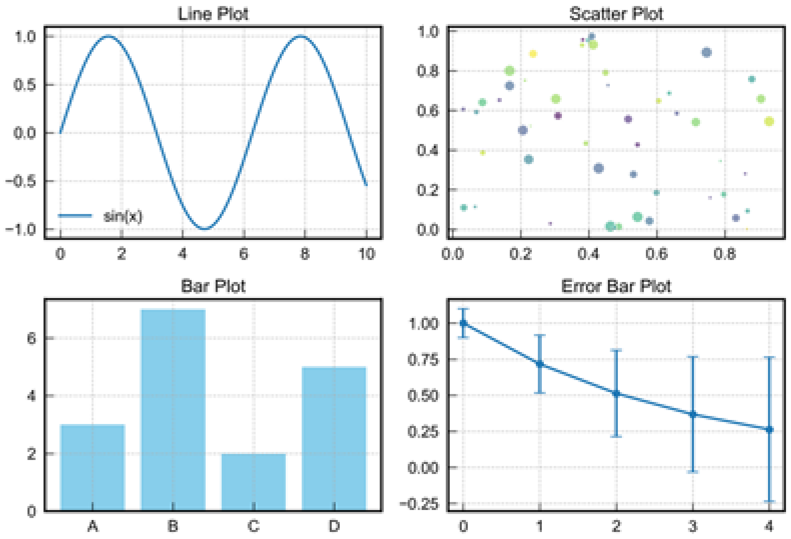
(position, visibility)

4 Artist: every element on the figure

(This is relevant when you use “tight_layout()”)

Define the overall plot style

```
import matplotlib as mpl
import matplotlib.pyplot as plt

plt.style.use(“my_style.style”)
```



Basic setup

```
# font settings
font.size: 14
font.family: Arial

# grid setting
axes.grid: True
grid.linestyle: --
grid.linewidth: 0.8
grid.alpha: 0.75

# global axes
axes.labelsize: 14
axes.titlesize: 16
axes.linewidth: 2.0
# figure.autolayout: True

# legend settings
legend.fontsize: 14
legend.frameon: False
# legend.fancybox: True
legend.facecolor: 'none'
legend.edgecolor: 'none'
```

Default plots

```
lines.linewidth: 2.0

# error bars
errorbar.capsize: 2

# bar plot
patch.linewidth: 1
patch.edgecolor: ‘white’

# other options

xtick.labelsize: 14
ytick.labelsize: 14
xtick.direction: in
xtick.major.size: 8
xtick.major.pad: 8
ytick.direction: in
ytick.major.size: 8
ytick.major.pad: 8

mathtext.default: regular
```

Practices for overall style control

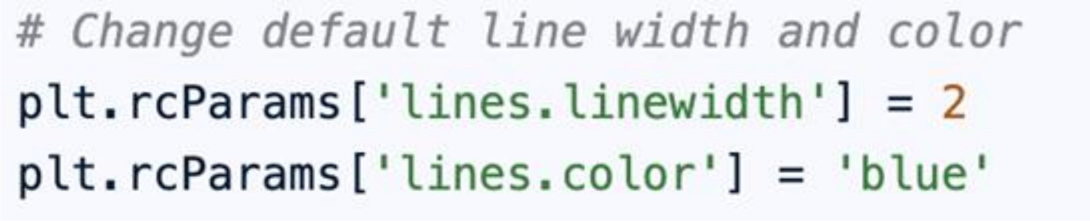
What to do in your style file:

1. Figure-level: default font family and size
2. Axes-level: linewidth, label/title font size, legend properties, grid properties
3. Axis-level: ticks label size and pad, direction, visibility

What you should not:

1. very specificalized settings that can only be determined according to your data

However, sometimes we do want to override the default style settings:



```
# Change default line width and color
plt.rcParams['lines.linewidth'] = 2
plt.rcParams['lines.color'] = 'blue'
```

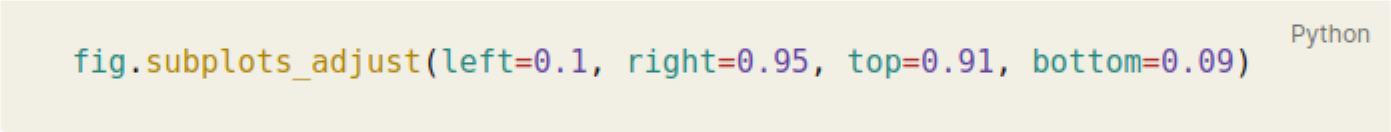
Subplots layout control

There are three ways to control the whole layout (spacing, padding and positions etc.).

1. quick and dirty but works really well for most cases: (when you are using **subplot/subplots**)
plt.tight_layout()

2. Combine **GridSpec** + **constrained_layout [Most Recommended]**

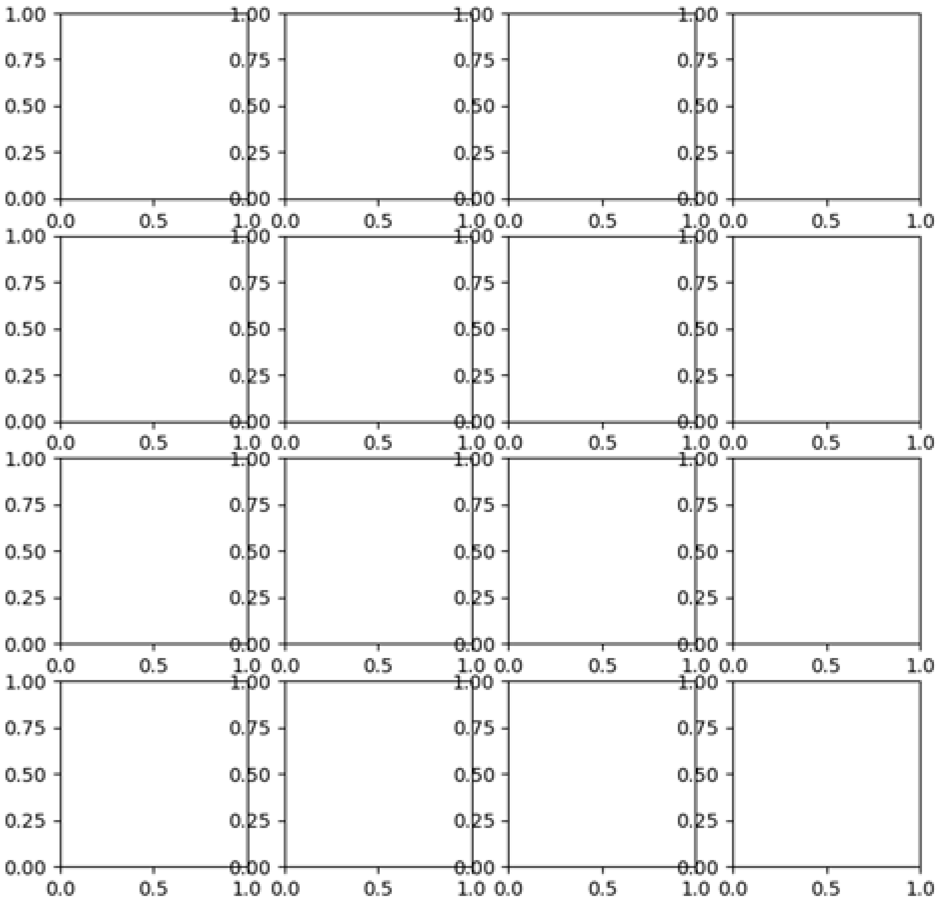
3. Combine **subplots_adjust** with **GridSpec** or **subplots** [Global control; Recommended]



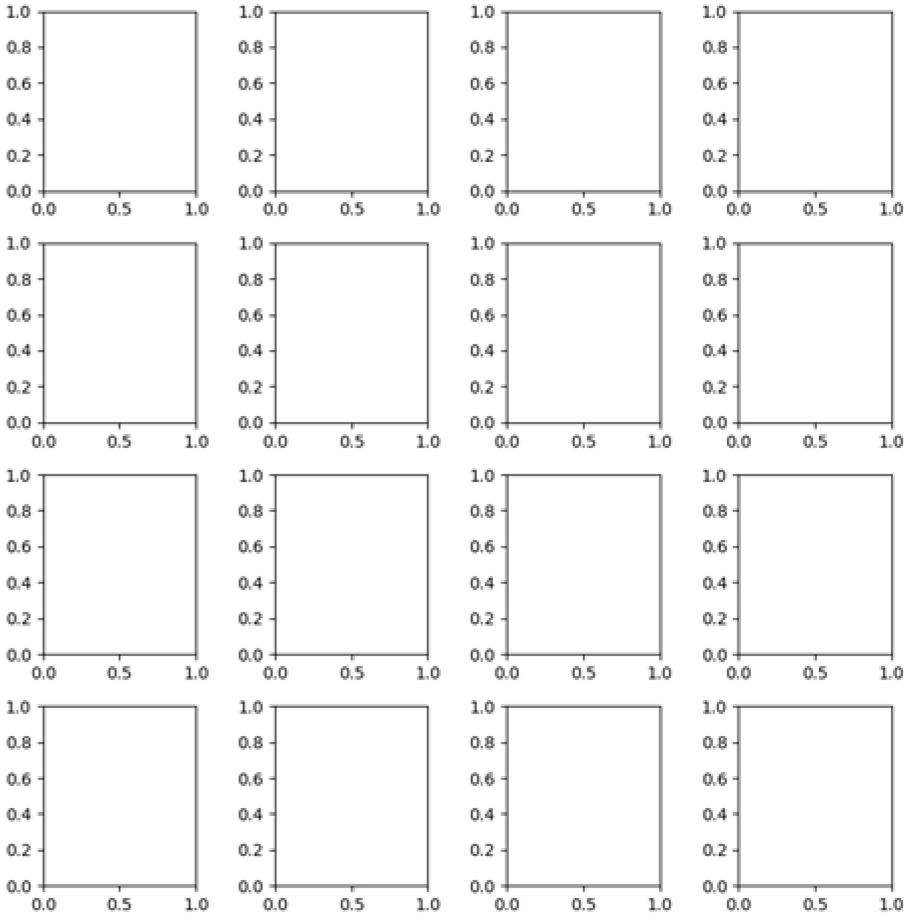
```python
fig.subplots_adjust(left=0.1, right=0.95, top=0.91, bottom=0.09)
```

tight_layout()

without tight_layout()

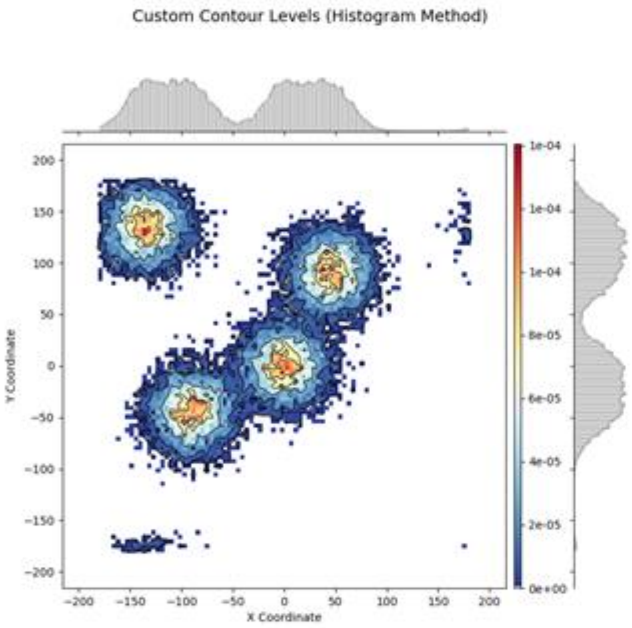


with tight_layout()

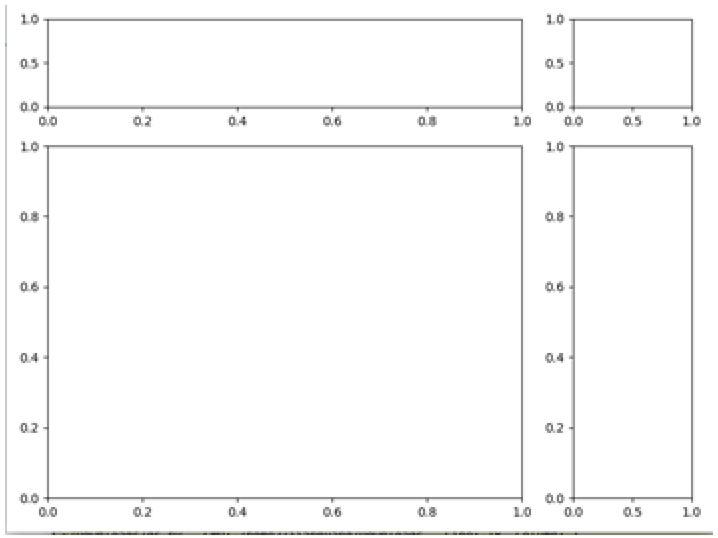


Internally, tight_layout() will calculate the locations of all Artists based on their constrained
Caveat: constraints have priorities and tight_layout() can be un-deterministic

GridSpec for complicated layout

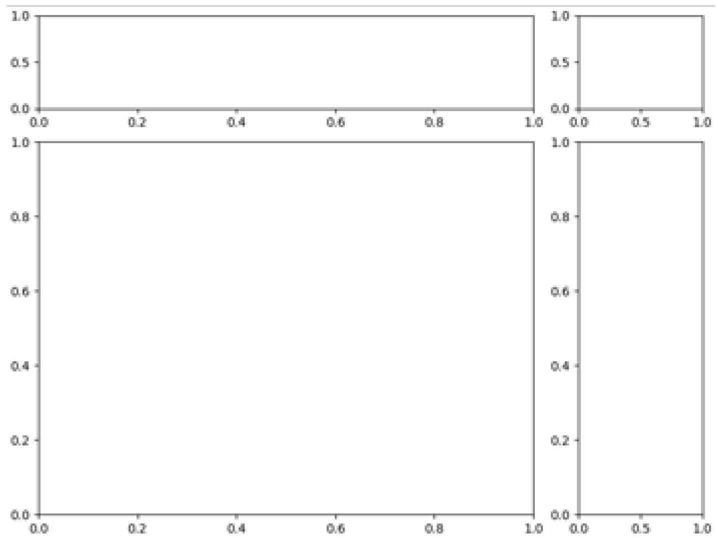


tight_layout()

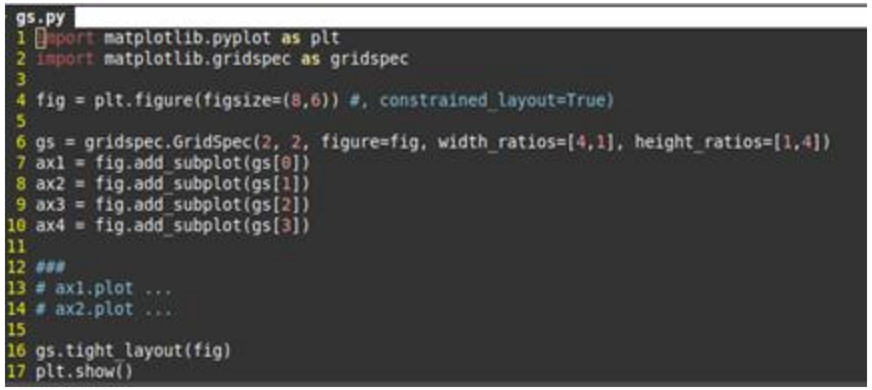


(automatically adjust wspace, hspace & left, right, top, bottom)

constrained_layout=True

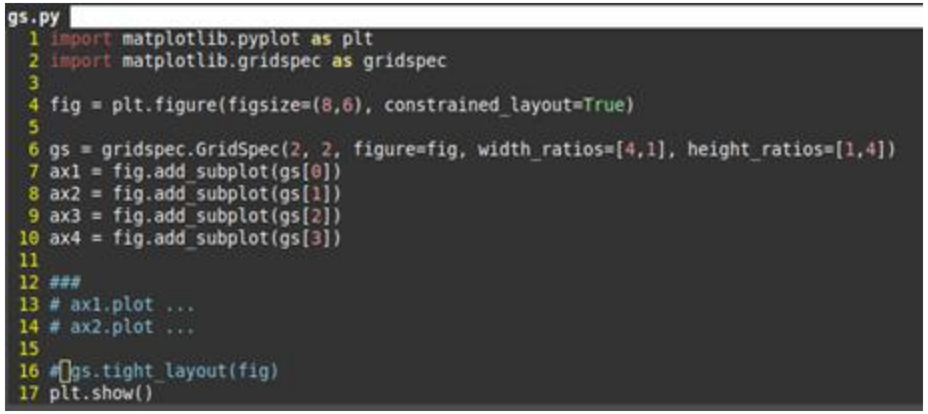


Personal favorite



gs.py

```
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

fig = plt.figure(figsize=(8,6)) #, constrained_layout=True)

gs = gridspec.GridSpec(2, 2, figure=fig, width_ratios=[4,1], height_ratios=[1,4])
ax1 = fig.add_subplot(gs[0])
ax2 = fig.add_subplot(gs[1])
ax3 = fig.add_subplot(gs[2])
ax4 = fig.add_subplot(gs[3])

###
# ax1.plot ...
# ax2.plot ...

gs.tight_layout(fig)
plt.show()
```

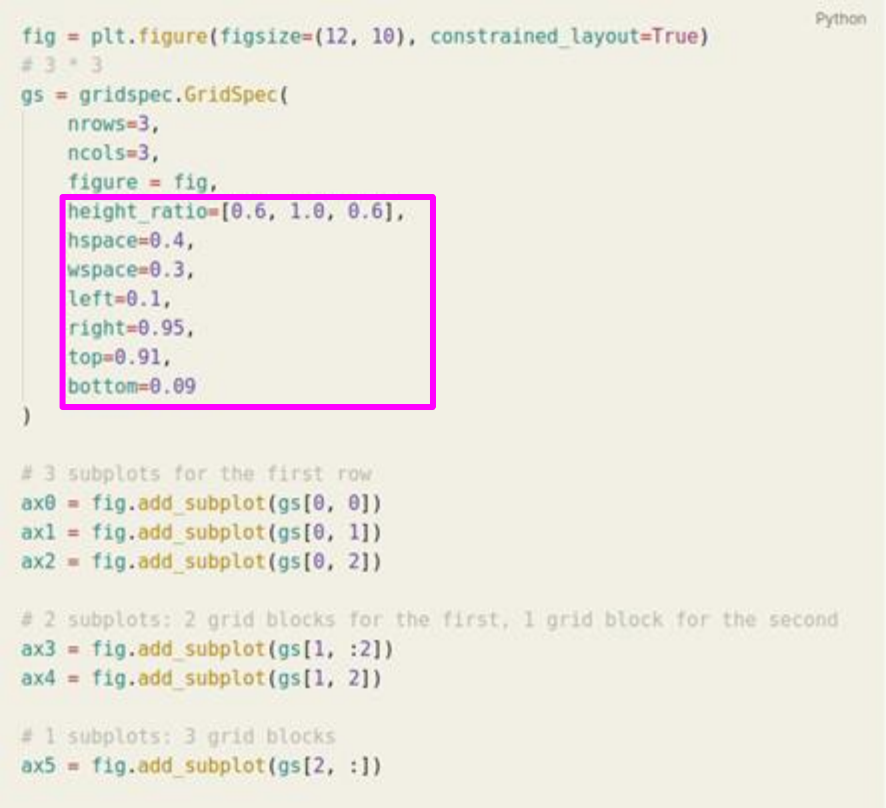


gs.py

```
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

fig = plt.figure(figsize=(8,6), constrained_layout=True)

gs = gridspec.GridSpec(2, 2, figure=fig, width_ratios=[4,1], height_ratios=[1,4])
ax1 = fig.add_subplot(gs[0])
ax2 = fig.add_subplot(gs[1])
ax3 = fig.add_subplot(gs[2])
ax4 = fig.add_subplot(gs[3])

###
# ax1.plot ...
# ax2.plot ...

#gs.tight_layout(fig)
plt.show()
```

More controls in GridSpec

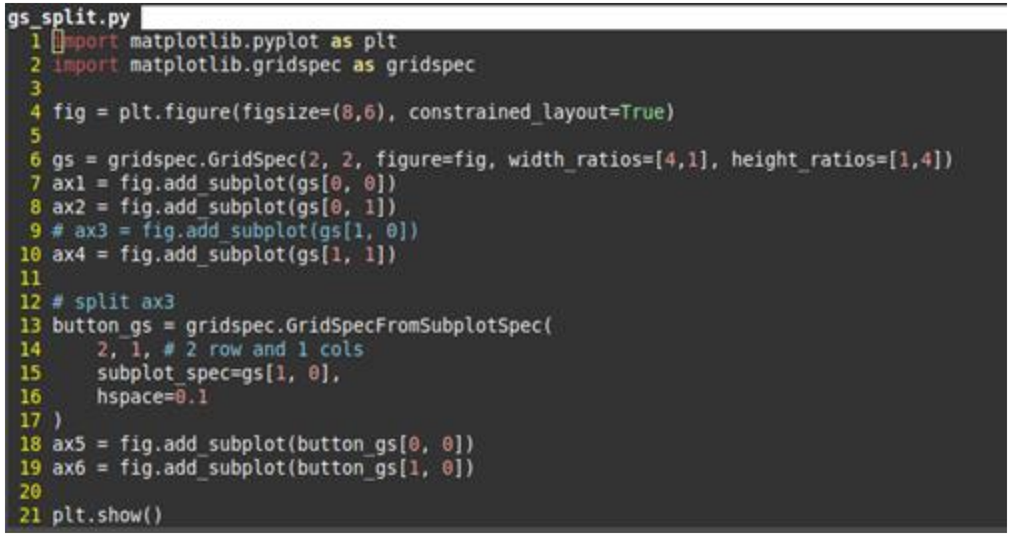
more keyword arguments in GridSpec



Python

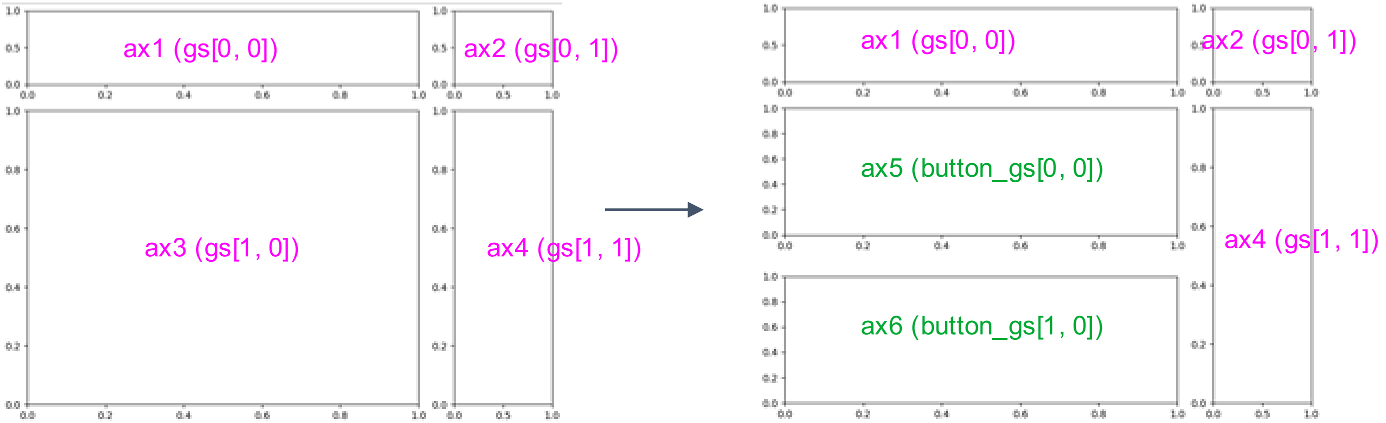
```python
fig = plt.figure(figsize=(12, 10), constrained_layout=True)
# 3 * 3
gs = gridspec.GridSpec(
    nrows=3,
    ncols=3,
    figure = fig,
    height_ratio=[0.6, 1.0, 0.6],
    hspace=0.4,
    wspace=0.3,
    left=0.1,
    right=0.95,
    top=0.91,
    bottom=0.09
)

# 3 subplots for the first row
ax0 = fig.add_subplot(gs[0, 0])
ax1 = fig.add_subplot(gs[0, 1])
ax2 = fig.add_subplot(gs[0, 2])

# 2 subplots: 2 grid blocks for the first, 1 grid block for the second
ax3 = fig.add_subplot(gs[1, :2])
ax4 = fig.add_subplot(gs[1, 2])

# 1 subplots: 3 grid blocks
ax5 = fig.add_subplot(gs[2, :])
```

Further splitting of a grid block



gs_split.py

```python
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

fig = plt.figure(figsize=(8,6), constrained_layout=True)

gs = gridspec.GridSpec(2, 2, figure=fig, width_ratios=[4,1], height_ratios=[1,4])
ax1 = fig.add_subplot(gs[0, 0])
ax2 = fig.add_subplot(gs[0, 1])
# ax3 = fig.add_subplot(gs[1, 0])
ax4 = fig.add_subplot(gs[1, 1])

# split ax3
button_gs = gridspec.GridSpecFromSubplotSpec(
    2, 1, # 2 row and 1 cols
    subplot_spec=gs[1, 0],
    hspace=0.1
)
ax5 = fig.add_subplot(button_gs[0, 0])
ax6 = fig.add_subplot(button_gs[1, 0])

plt.show()
```



Advanced topic: Dimension and Resolution

figsize: usually in the unit of **inches** (1 inch = 2.54 cm)

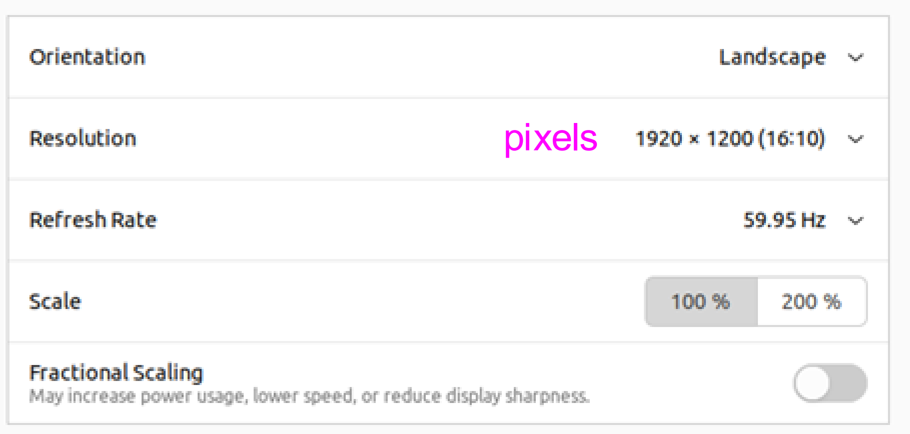
```
plt.subplots(2, 2, figsize=(8,6))
# determine the physical size of the figure
```

resolution: **dpi** (dots per inch) and **pixels**

```
plt.savefig(‘tmp.png’, dpi=100)
# for fixed inches, more dpi means higher resolution

# in your screen
```

the figure will be shown in (800, 600) **pixels (=dpi*inches)**

```
# Visually, the figure size is determined by your PC resolution
```



Orientation Landscape

Resolution 1920 × 1200 (16:10)

pixels

Refresh Rate 59.95 Hz

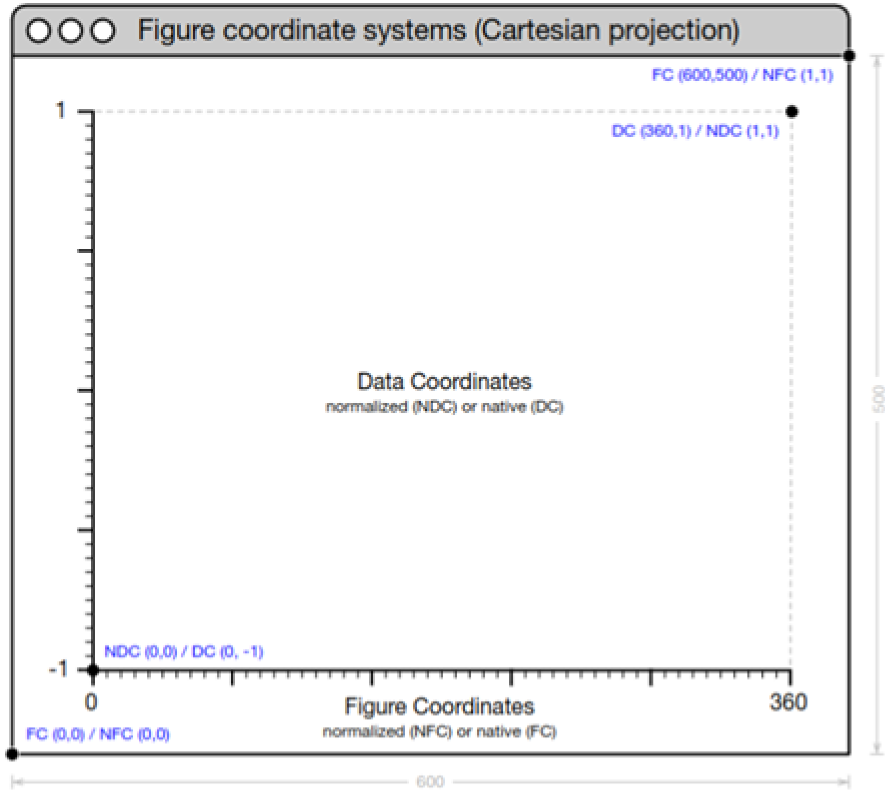
Scale 100 % 200 %

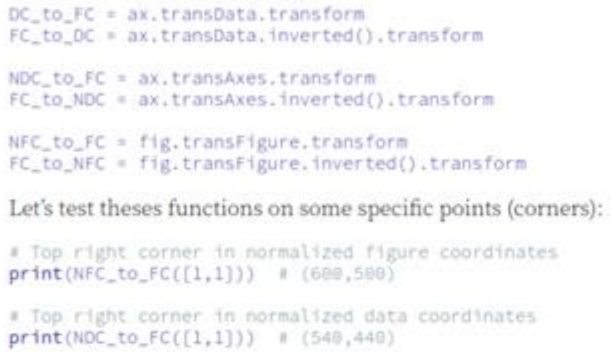
Fractional Scaling
May increase power usage, lower speed, or reduce display sharpness.

Advice for determining your figure size for publications (if size restrictions do apply)

1. determine the physical size in inches (referring to page size, A4 or A5, also margins)
2. use high dpi values for high resolution (suggested: 300 or 600)

Advanced topic: Coordinates system





```
DC_to_FC = ax.transData.transform
FC_to_DC = ax.transData.inverted().transform

NDC_to_FC = ax.transAxes.transform
FC_to_NDC = ax.transAxes.inverted().transform

NFC_to_FC = fig.transFigure.transform
FC_to_NFC = fig.transFigure.inverted().transform
```

Let's test theses functions on some specific points (corners):

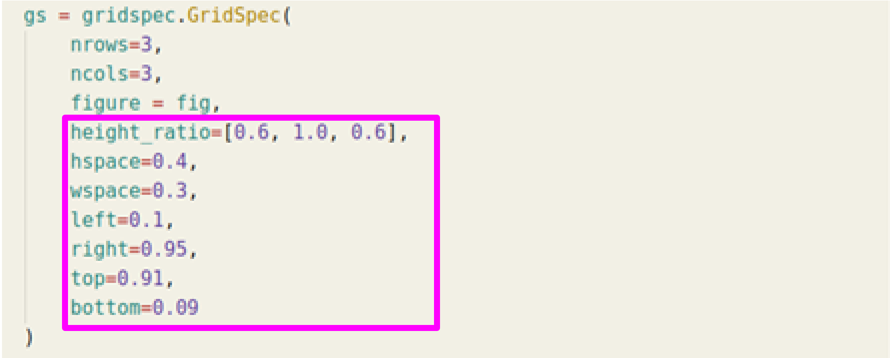
```
# Top right corner in normalized figure coordinates
print(NFC_to_FC([1,1]))  # (600,500)

# Top right corner in normalized data coordinates
print(NDC_to_FC([1,1]))  # (540,440)
```

Data coordinate (DC): in data units
Normalized data coordinate (NDC) (0,0) → (1,1)

Figure coordinate (FC): in **pixels**
Normalized figure coordinate (NFC) (0,0) → (1,1)

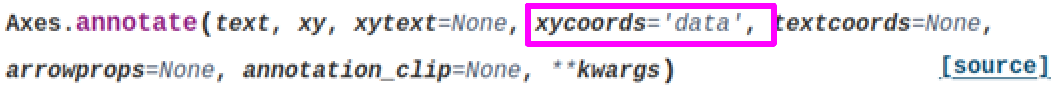
Why is it important to know?

1. some arguments may use NFC



```
gs = gridspec.GridSpec(
    nrows=3,
    ncols=3,
    figure = fig,
    height_ratio=[0.6, 1.0, 0.6],
    hspace=0.4,
    wspace=0.3,
    left=0.1,
    right=0.95,
    top=0.91,
    bottom=0.09
)
```

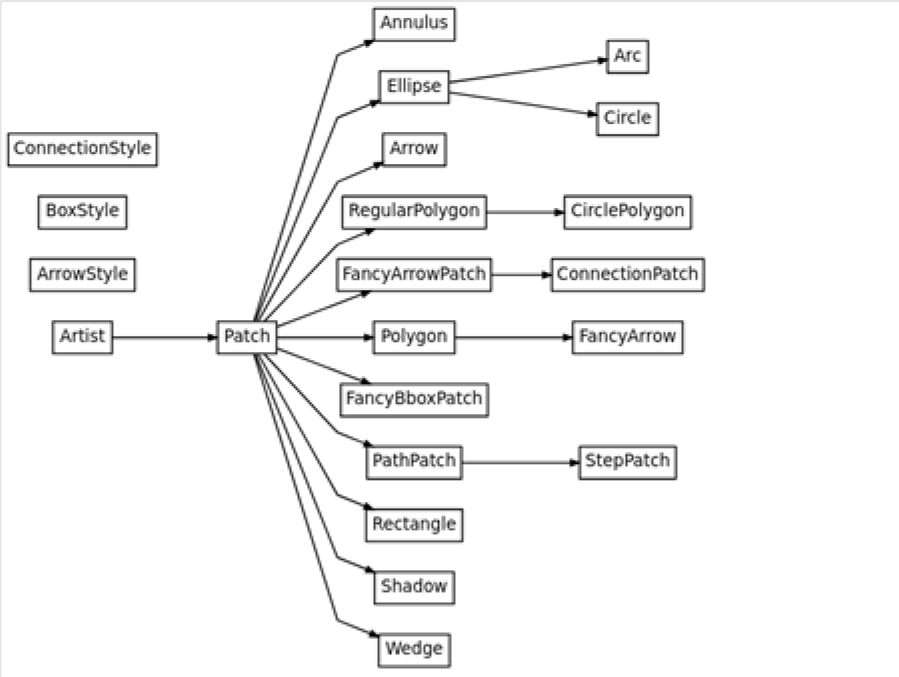
1. For precisely control locations of annotations



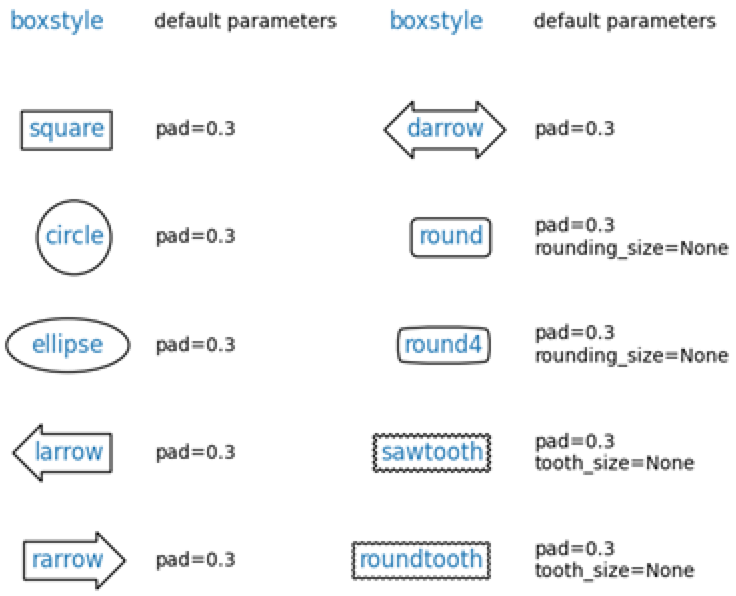
```
Axes.annotate(text, xy, xytext=None, xycoords='data', textcoords=None,
arrowprops=None, annotation_clip=None, **kwargs)                [source]
```

Advanced topic: Annotations

matplotlib.patches



Shapes



boxstyle	default parameters	boxstyle	default parameters
square	pad=0.3	darrow	pad=0.3

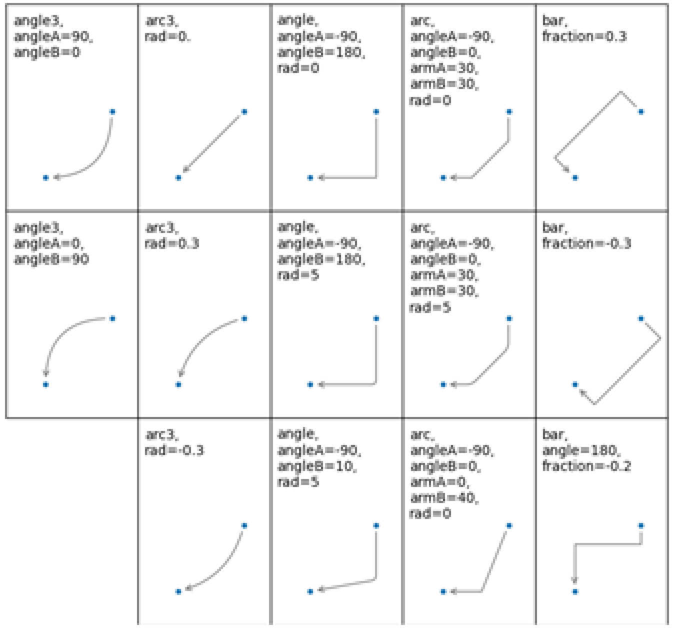
circle	pad=0.3	round	pad=0.3
			rounding_size=None

ellipse	pad=0.3	round4	pad=0.3
			rounding_size=None

larrow	pad=0.3	sawtooth	pad=0.3
			tooth_size=None

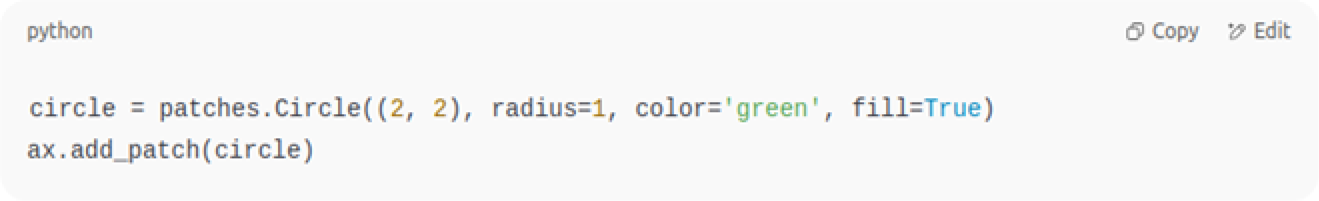
rarrow	pad=0.3	roundtooth	pad=0.3
			tooth_size=None

Connections


angle3, angleA=90, angleB=0
arc3, rad=0.
angle, angleA=-90, angleB=180, rad=0
arc, angleA=-90, angleB=0, armA=30, armB=30, rad=0
bar, fraction=0.3
angle3, angleA=0, angleB=90
arc3, rad=0.3
angle, angleA=-90, angleB=180, rad=5
arc, angleA=-90, angleB=0, armA=30, armB=30, rad=5
bar, fraction=-0.3
arc3, rad=-0.3
angle, angleA=-90, angleB=10, rad=5
arc, angleA=-90, angleB=0, armA=0, armB=40, rad=0
bar, angle=180, fraction=-0.2

Connection styles for annotations

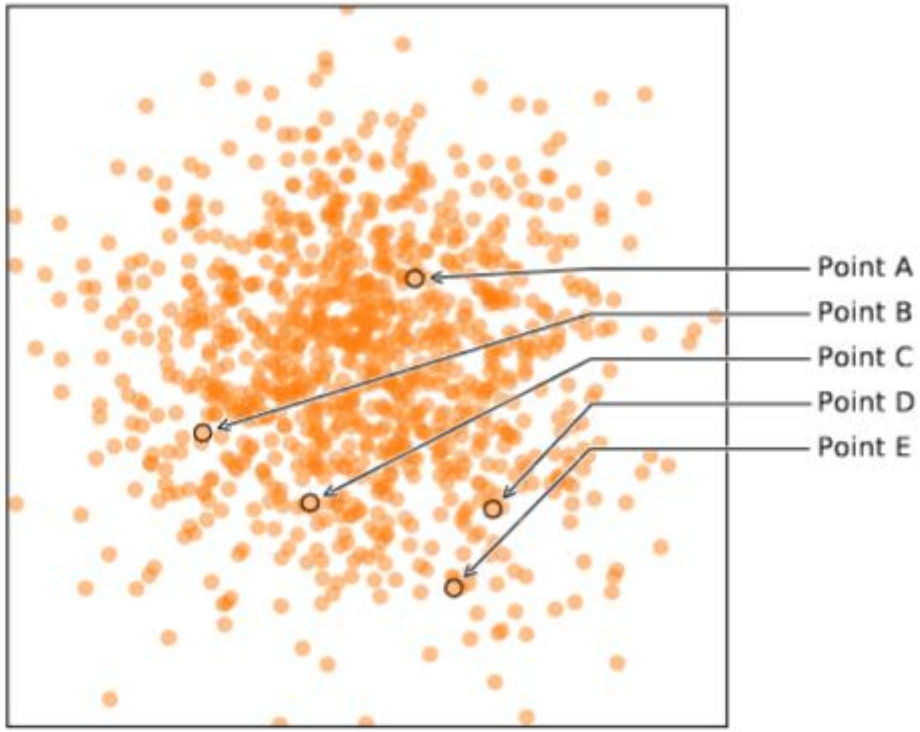
Circle



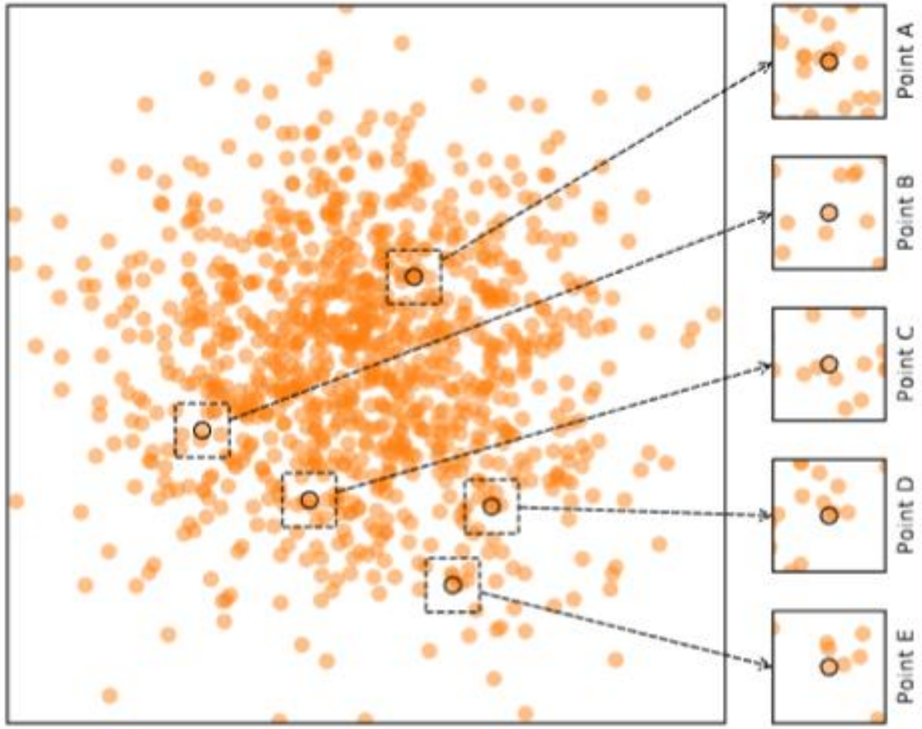
```python
circle = patches.Circle((2, 2), radius=1, color='green', fill=True)
ax.add_patch(circle)
```

https://matplotlib.org/stable/users/explain/text/annotations.html#coordinate-systems-for-annotations

Advanced topic: Annotation



https://github.com/rougier/scientific-visualization-book/blob/master/code/ornaments/annotation-side.py



https://github.com/rougier/scientific-visualization-book/blob/master/code/ornaments/annotation-zoom.py

```
ax.text(
        1.1,
        0.5,
        "Point " + chr(ord("A") + i),
        rotation=90,
        size=8,
        ha="left",
        va="center",
        transform=ax.transAxes,)
```

using NDC

Annotations:

1. ax.text: text annotation
2. ax.annotate(): text annotation
3. patches: circles, rectangles, polygons etc

Guess what kind of practices I used for the following

GridSpec for grinding subplots
(use constrianed_layout)

mplstyle: global control for
font, grid, ticks

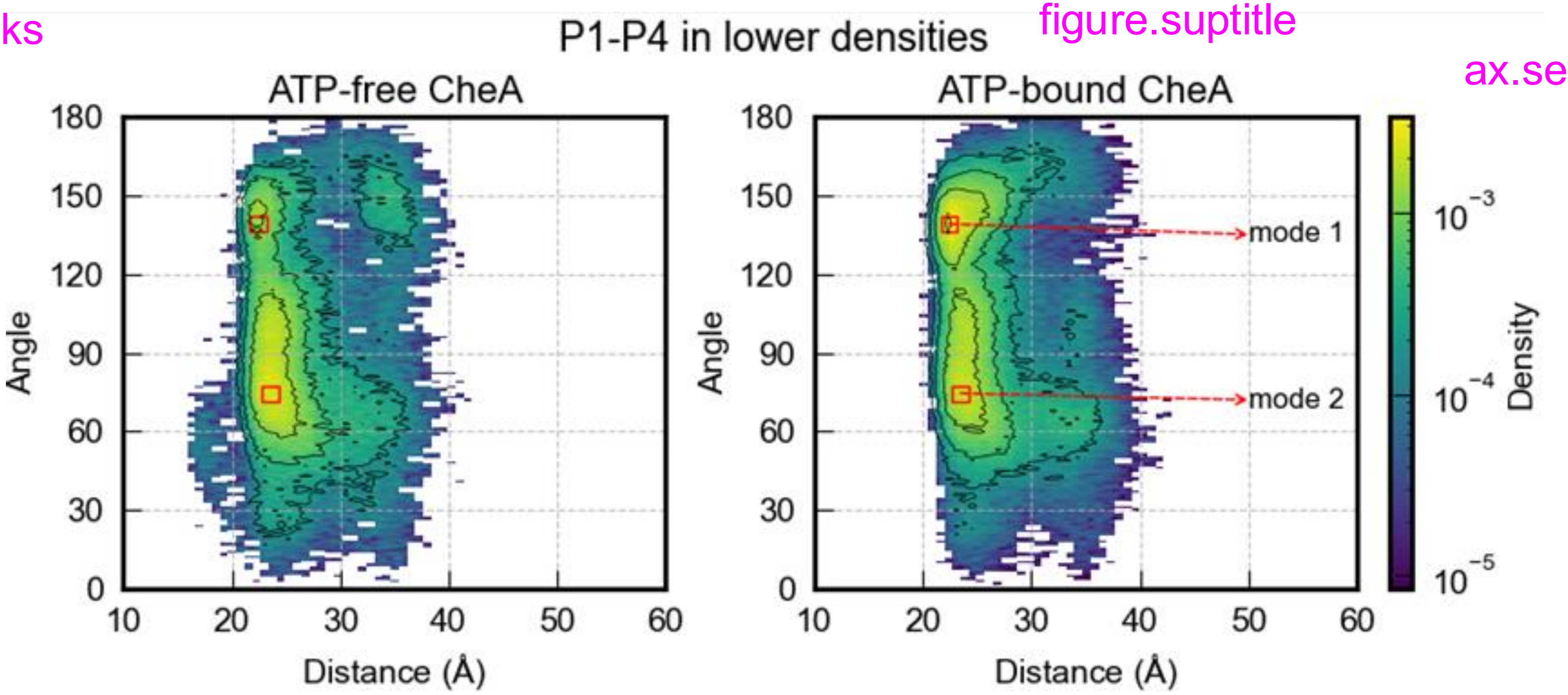


figure.suptitle

ax.set_title

ax.text
(ax.transAxes)

rectangle patches
(ax.transData)

ConnectionPatches
(for the arrow)

Lastly, use Jian’s package & check some samples

Use **seaborn**

Use the “**plot**” module from **EnsembleAnalysis**

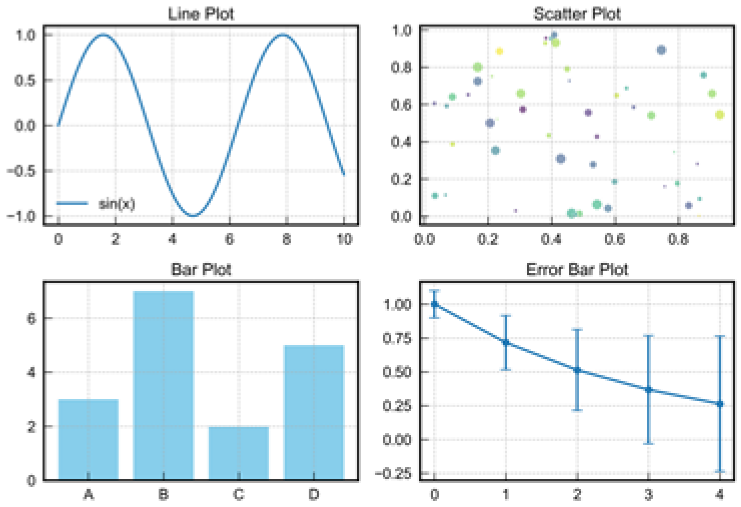
https://github.com/huangjianhuster/EnsembleAnalysis

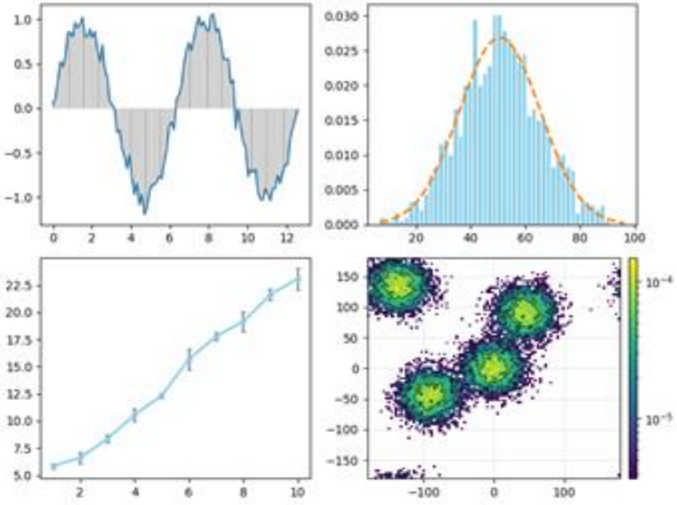
Use Jian’s mplstyle:

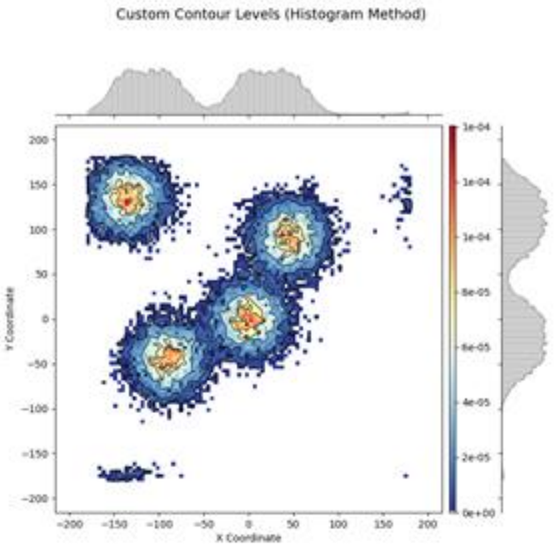
https://github.com/huangjianhuster/toolbox/tree/main/plot

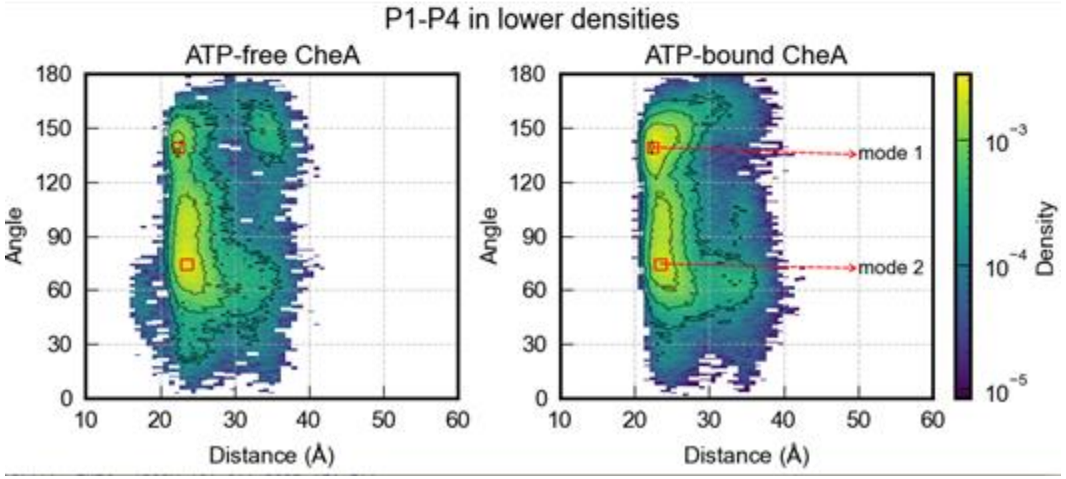
Lastly, for **manuscript figures**:

1. plot all elements using matplotlib (doable but sometimes can be lots of work for annotations, images, alignments etc.)
2. plot each subplots individually and use PPT or AI to further adjust it. (alignment; format-rich annotations etc.) However this should be as minimal as possible.

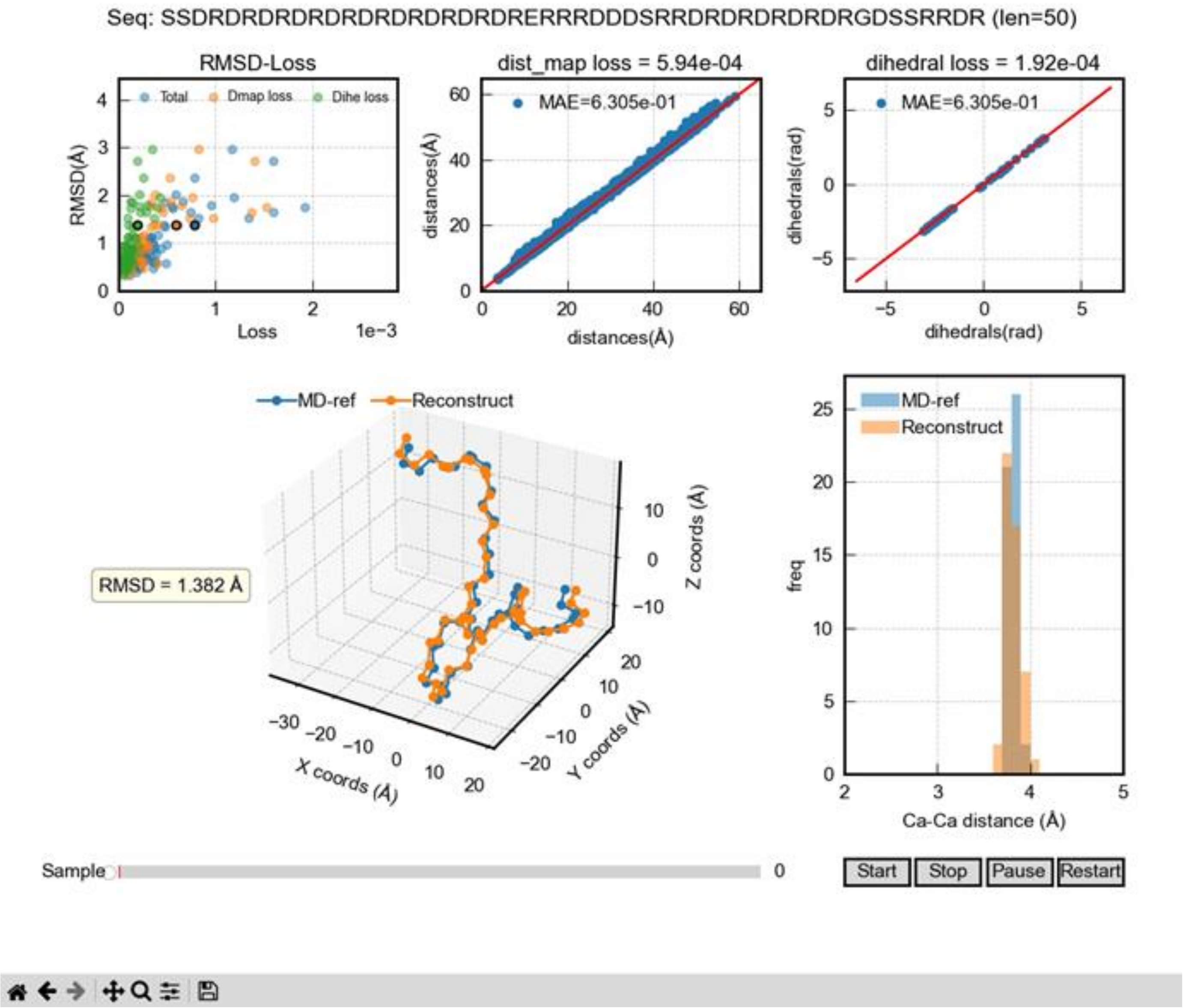








SI: Animation



GridSpec
Annotation
3D plot
Slider
Animation

(will not be covered here…Happy to discuss in private)

Thank you.