Coordinate system

Guanyao Zhao

2022-10-26

Contents

要想精确的控制画布中每一个元素的位置，坐标系统是个绕不开的话题，它决定着每个元素存在于什么坐标系，以及该怎么在坐标系之间来回转换。

坐标系

参考文档

参考坐标系	transformation object	参考对象	显示范围
data	ax.transData	数学上的横纵坐标轴	xlim, ylim
axes	ax.transAxes	axes 的边框	(0,0)-(1,1)
figure	fig.transFigure	figure 的边框	(0,0)-(1,1)
figure-inches	fig.dpi_scale_trans	figure 的边框 (inhce)	(0,0)-(width, height)
xaxis	ax.get_xaxis_transform()	axes 的 xlim 和 ylim 另一 axes 的边框	xlim and (0,1)
display	None	视窗的边框，像素计算	(0,0)-(width, height)

坐标系设置

导入实用的第三方库

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

绘图

```
fig = plt.figure(figsize=(9, 6), edgecolor="green", linewidth=3)
ax1 = fig.add_subplot(111, xlim=(0, 10), ylim=(0, 10), aspect=1)

#! 以 data 为坐标系
circ1 = patches.Circle(xy=(5, 5), radius=4, edgecolor="r", facecolor="None")
c1 = ax1.add_artist(circ1)
c1.set_transform(ax1.transData)

#! 以 axes 为坐标系
circ2 = patches.Circle(xy=(0.5, 0.5), radius=0.2, edgecolor="b", facecolor="None")
c2 = ax1.add_artist(circ2)
c2.set_transform(ax1.transAxes)

#! 以 figure 为坐标系
c3 = ax1.add_artist(circ2)  # 和以 axes 相比，由于 figure 横纵比不为 1，所以圆显示出来是椭圆形
c3.set_transform(fig.transFigure)

#! 以 figure 为坐标系，添加在 figure 上，，不是 axes 上
circ3 = patches.Circle(xy=(0.1, 0.2), radius=0.2, edgecolor="black", facecolor="None")

# 不遮住
c4 = fig.add_artist(circ3)
c4.set_transform(fig.transFigure)

#! 以 figure 为坐标系，且单位为 inche
circ4 = patches.Circle(xy=(2, 3), radius=1, edgecolor="green", facecolor="None")

```

```python
c5 = fig.add_artist(circ4)
c5.set_transform(fig.dpi_scale_trans)

#! 混合坐标系，x 轴为 Data，y 为 Axes
circ6 = patches.Circle(
    xy=(6, 0.5),
    radius=0.5,  #! 注意，半径纵向仍是按照 Axes，横向按照 Data
    edgecolor="orange",
    facecolor="None",
)
c6 = ax1.add_artist(circ6)
c6.set_transform(ax1.get_xaxis_transform())
```

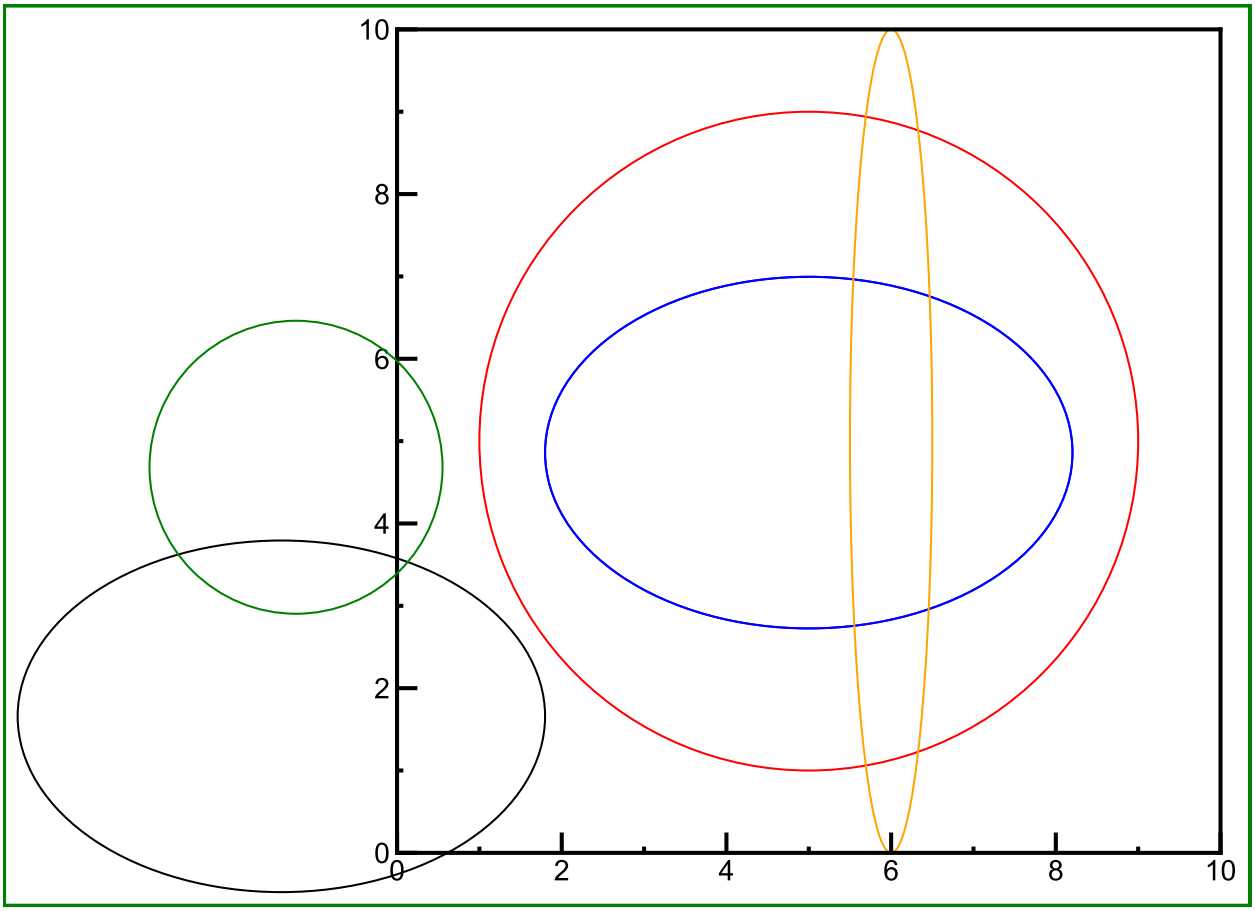

Fig. 1: coordinates

混合坐标系

有时候会需要在特定坐标系下进行一定的偏移，在绘图过程中这是个非常实用的方法。

```
from matplotlib.transforms import blended_transform_factory, ScaledTranslation

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

ax = fig.add_subplot(111)
ax.set_xlim(0, 10)
ax.set_xticks(range(11))
ax.set_ylim(0, 5)
ax.set_xticks(range(11))

inches_to_point = 72  # point to inch
fontsize = 12  # unit: point
dx, dy = 0, -1.5 * fontsize / inches_to_point  # 偏移量：x 不变，y 向下移动 X inches 距离
offset = ScaledTranslation(dx, dy, fig.dpi_scale_trans)  #! figure 的 inches 为坐标系
transform = blended_transform_factory(  #! 混合坐标转换工厂
    ax.transData, ax.transAxes + offset
)  #! ax.transData 表示 x 轴不变；ax.transAxes+offset 表示 y 改变

for x in range(11):
    plt.text(
        x,
        0,
        "↑",
        transform=transform,
        ha="center",
        va="top",
        color="red",
        fontsize=fontsize,
    )  # 在此转变坐标，str 为上箭头
```

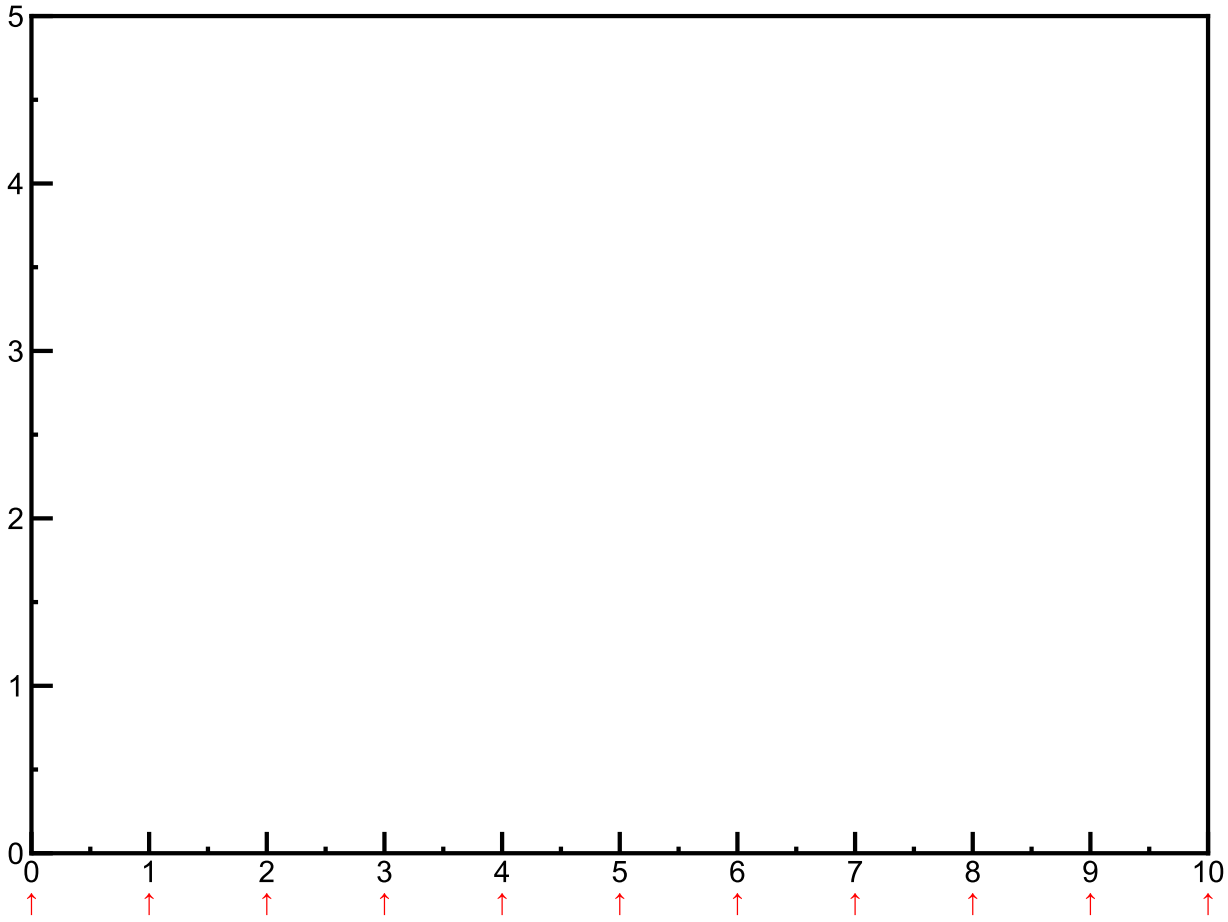

Fig. 2: ScaledTranslation

坐标系转换

在 matplotlib 的坐标系系统中，一切都是以 FC 为中心互相转换：

B[FC] C[NDC] –> B D[NFC] –> B B –> A B –> C B –> D

```

具体转换代码如下：

``` python
fig = plt.figure(figsize=(6, 5), dpi=100)
ax1 = fig.add_subplot(111, xlim=(0, 360), ylim=(-1, 1))

#! ax1
```
```

```
DC_to_FC = ax1.transData.transform
FC_to_DC = ax1.transData.inverted().transform

#! ax1
NDC_to_FC = ax1.transAxes.transform
FC_to_NDC = ax1.transAxes.inverted().transform

#! fig
NFC_to_FC = fig.transFigure.transform
FC_to_NFC = fig.transFigure.inverted().transform

print(NFC_to_FC([1, 1]))
print(NDC_to_FC([1, 1]))
print(DC_to_FC([360, 1]))

#! 以上都是相对于 FC 的转换，其余的转换可通过 FC 再加一次转换即可达到目的，比如 DC_to_NDC
print(FC_to_NDC(DC_to_FC([180,0])))
```

Exercise-1

通过以下例子弄清楚 pixel, point, inches 的关系。在此之前需要弄清楚一个重要的概念：ppi，其指的是 pixels per inche，即每英尺有多少个像素点。在绘图的时候可以将 pixel 理解为相对值，而 point 和 inche 的长度是绝对的，1cm=28.3464567pt，1inche=72pt，而 1pt 为多少个 pixel 自己确定，值越大，清晰度越高，也就是我们所说的分辨率越高。

4K 27inche 的显示器分辨率（ppi）怎么计算？4K 指的是分辨率为（3840，2160）个 pixels：

$$\text{ppi} = \frac{\sqrt{X^2+Y^2}}{\text{屏幕尺寸}} = \frac{3840^2+2160^2}{27} = 163$$

```
from matplotlib.patches import Circle

fig = plt.figure(figsize=(8, 2), edgecolor="green", linewidth=2, dpi=100)

ymin, ymax = [0, 2]
xmin, xmax = [0, 8]
ax1 = fig.add_subplot(111, xlim=(xmin, xmax), ylim=(ymin, ymax), aspect=1)
```

```python

#! 方法 1
for i in range(8):
    circ = Circle(xy=(i + 0.5, 1), radius=0.5, edgecolor="green", facecolor="None")
    ax1.add_artist(circ)

#! 方法 2
point = fig.dpi / 72  # 1 pt 有多少 pixel
X = 0.5 + np.arange(8)
Y = np.full(shape=len(X), fill_value=0.5)

# pixel to point
DC_to_PT = (
    lambda x: ax1.get_window_extent().width / (xmax - xmin) / point
)  # get_window_extent().width 单位是 pixel，此处要计算的是将 pixel 转换为 point，因为 scatter 默认的是 po

S = DC_to_PT(1) ** 2  # 散点图大小，单位为 point
ax1.scatter(X, Y, s=S, edgecolor="red", facecolor="None")
plt.show()
```

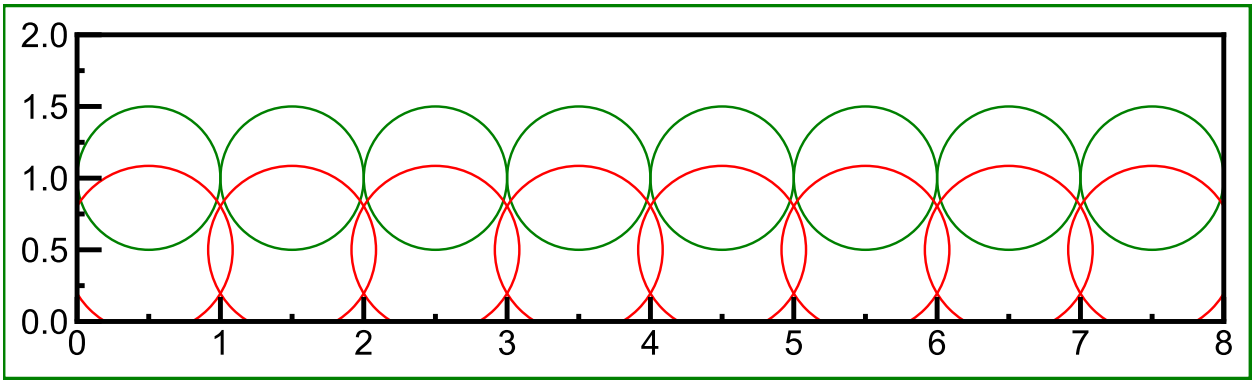

Fig. 3: Exercise-1