第四回:文字图例尽眉目

In [1…

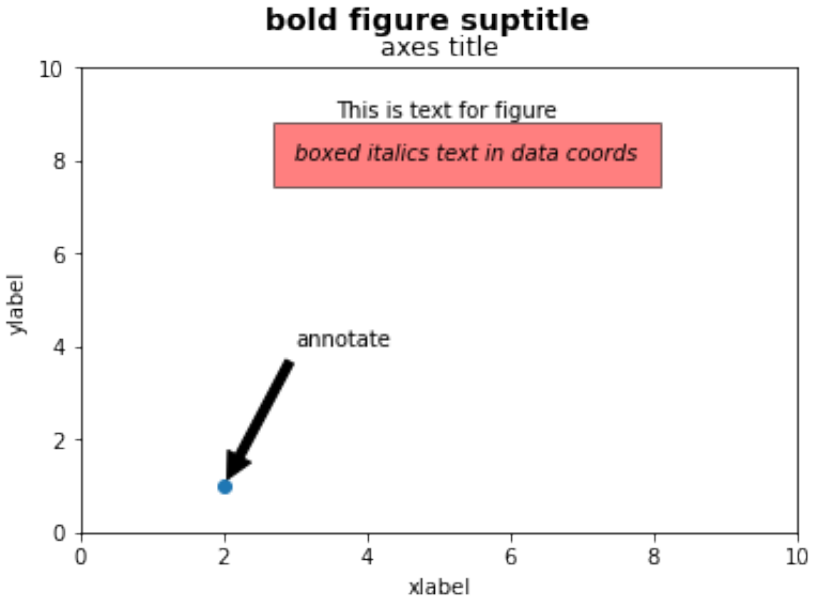
```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.dates as mdates
import datetime
```

一、Figure和Axes上的文本

- Matplotlib具有广泛的文本支持，包括对数学表达式的支持、对栅格和矢量输出的TrueType支持、具有任意旋转的换行分隔文本以及Unicode支持。
  #### 1.文本API示例
- 创建文本的方式包括pyplot API和objected-oriented API.

In [2…

```
# 以OO模式展示这些API是如何控制一个图像中各部分的文本

fig = plt.figure()
ax = fig.add_subplot()


# 分别为figure和ax设置标题，注意两者的位置是不同的
fig.suptitle('bold figure suptitle', fontsize=14, fontweight=
ax.set_title('axes title')

# 设置x和y轴标签
ax.set_xlabel('xlabel')
ax.set_ylabel('ylabel')

# 设置x和y轴显示范围均为0到10
ax.axis([0, 10, 0, 10])

# 在子图上添加文本
ax.text(3, 8, 'boxed italics text in data coords', style='ita
        bbox={'facecolor': 'red', 'alpha': 0.5, 'pad': 10})

# 在画布上添加文本，一般在子图上添加文本是更常见的操作，这种方法很少用
fig.text(0.4,0.8,'This is text for figure')

ax.plot([2], [1], 'o')
# 添加注解
ax.annotate('annotate', xy=(2, 1), xytext=(3, 4),arrowprops=
```

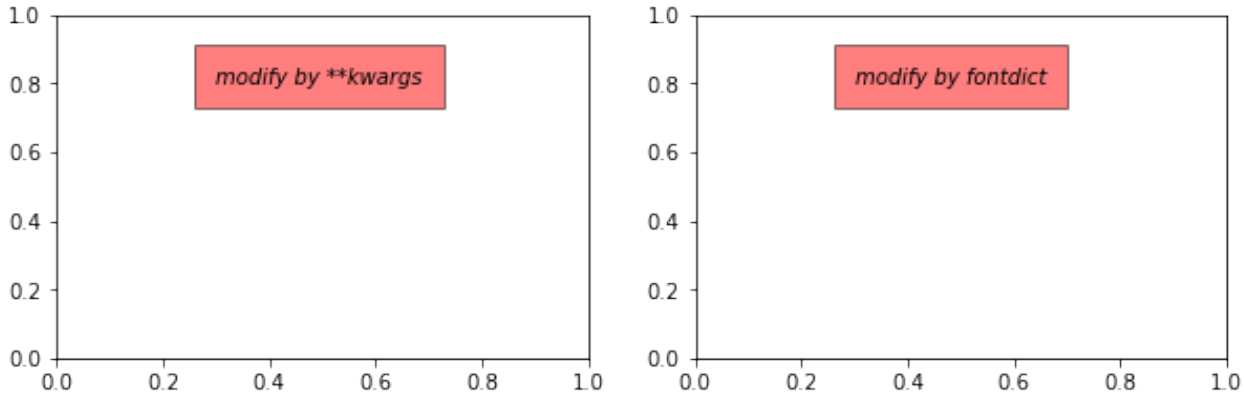


2.text-子图上的文本

- text的调用方式为 `Axes.text(x, y, s, fontdict=None, **kwargs)`
- 其中 x,y 为文本出现的位置，默认状态下即为当前坐标系下的坐标值，
- s 为文本的内容，
- fontdict 是可选参数，用于覆盖默认的文本属性，
- **kwargs 为关键字参数，也可以用于传入文本样式参数

In [3…

```
fig = plt.figure(figsize=(10,3))
axes = fig.subplots(1,2)

# 使用关键字参数修改文本样式
axes[0].text(0.3, 0.8, 'modify by **kwargs', style='italic',
        bbox={'facecolor': 'red', 'alpha': 0.5, 'pad': 10});

# 使用fontdict参数修改文本样式
font = {'bbox':{'facecolor': 'red', 'alpha': 0.5, 'pad': 10}
axes[1].text(0.3, 0.8, 'modify by fontdict', fontdict=font);
```



3.xlabel和ylabel - 子图的x，y轴标签

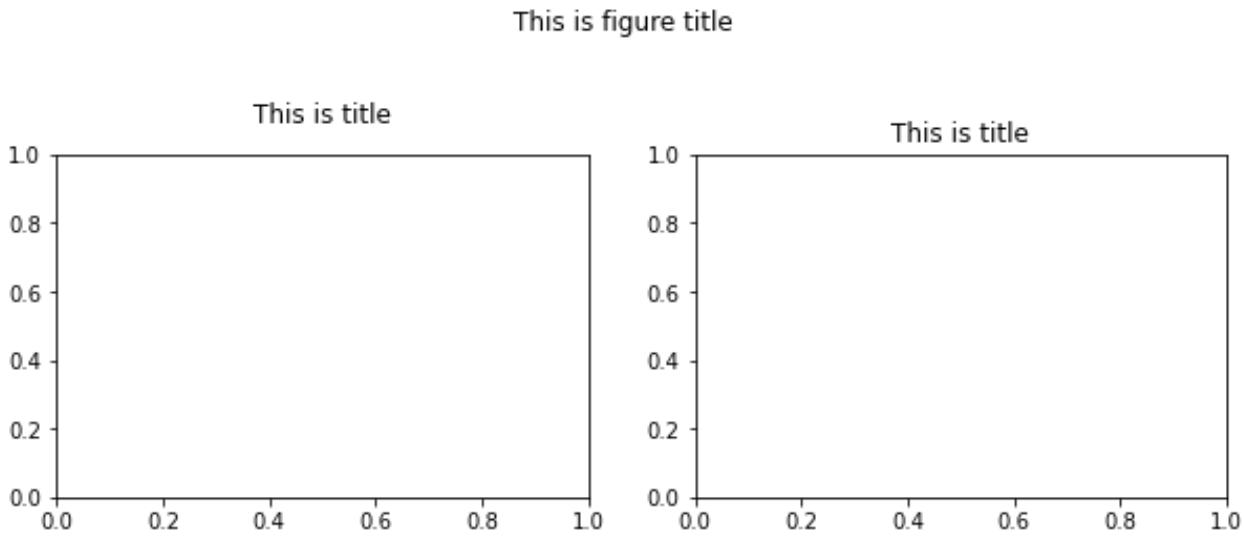
- xlabel的调用方式为Axes.set_xlabel(xlabel, fontdict=None, labelpad=None, *, loc=None, **kwargs)
- ylabel方式类似
- 其中xlabel即为标签内容,
- fontdict和**kwargs用来修改样式，上一小节已介绍,
- labelpad为标签和坐标轴的距离，默认为4,
- loc为标签位置，可选的值为'left', 'center', 'right'之一，默认为居中

4.title和suptitle - 子图和画布的标题

- title的调用方式为 `Axes.set_title(label, fontdict=None, loc=None, pad=None, *, y=None, **kwargs)`
- 其中label为子图标签的内容， `fontdict,loc,**kwargs` 和之前小节相同不重复介绍
- `pad` 是指标题偏离图表顶部的距离，默认为6
- `y` 是title所在子图垂向的位置。默认值为1，即title位于子图的顶部。
- suptitle的调用方式为 `figure.suptitle(t, **kwargs)` 其中 `t` 为画布的标题内容

In [4…

```
# 观察pad参数的使用效果
fig = plt.figure(figsize=(10,3))
fig.suptitle('This is figure title',y=1.2) # 通过参数y设置高度
axes = fig.subplots(1,2)
axes[0].set_title('This is title',pad=15)
axes[1].set_title('This is title',pad=6);
```

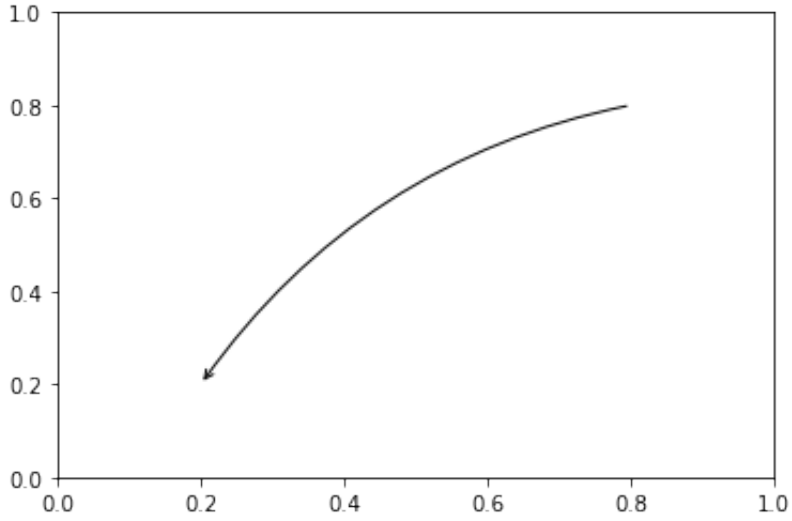


5.annotate - 子图的注解

- annotate的调用方式为 `Axes.annotate(text, xy, *args, **kwargs)`
- 其中 `text` 为注解的内容，
- `xy` 为注解箭头指向的坐标

In [5…

```
fig = plt.figure()
ax = fig.add_subplot()
ax.annotate("",
            xy=(0.2, 0.2), xycoords='data',
            xytext=(0.8, 0.8), textcoords='data',
            arrowprops=dict(arrowstyle="->", connectionstyle=
            );
```



6.字体的属性设置

- 字体设置一般有全局字体设置和自定义局部字体设置两种方法。

二、Tick上的文本

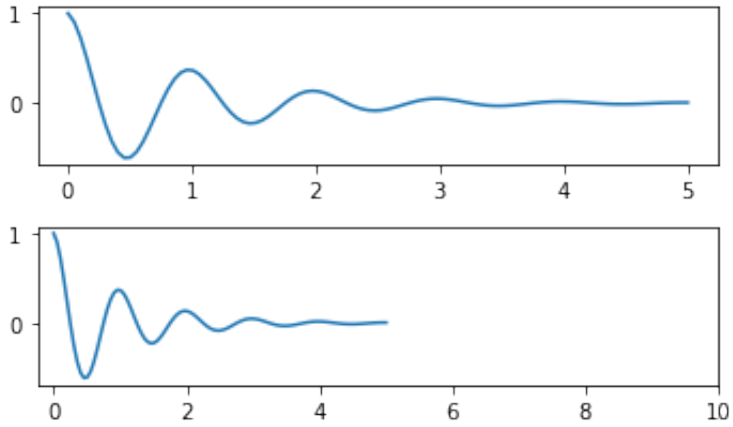
设置tick（刻度）和ticklabel（刻度标签）也是可视化中经常需要操作的步骤，matplotlib既提供了自动生成刻度和刻度标签的模式（默认状态），同时也提供了许多让使用者灵活设置的方式。

1.简单模式

- 可以使用axis的set_ticks方法手动设置标签位置，使用axis的set_ticklabels方法手动设置标签格式

In [6…

```
x1 = np.linspace(0.0, 5.0, 100)
y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
# 使用axis的set_ticks方法手动设置标签位置的例子，该案例中由于tick设置
fig, axs = plt.subplots(2, 1, figsize=(5, 3), tight_layout=T
axs[0].plot(x1, y1)
axs[1].plot(x1, y1)
axs[1].xaxis.set_ticks(np.arange(0., 10.1, 2.));
```



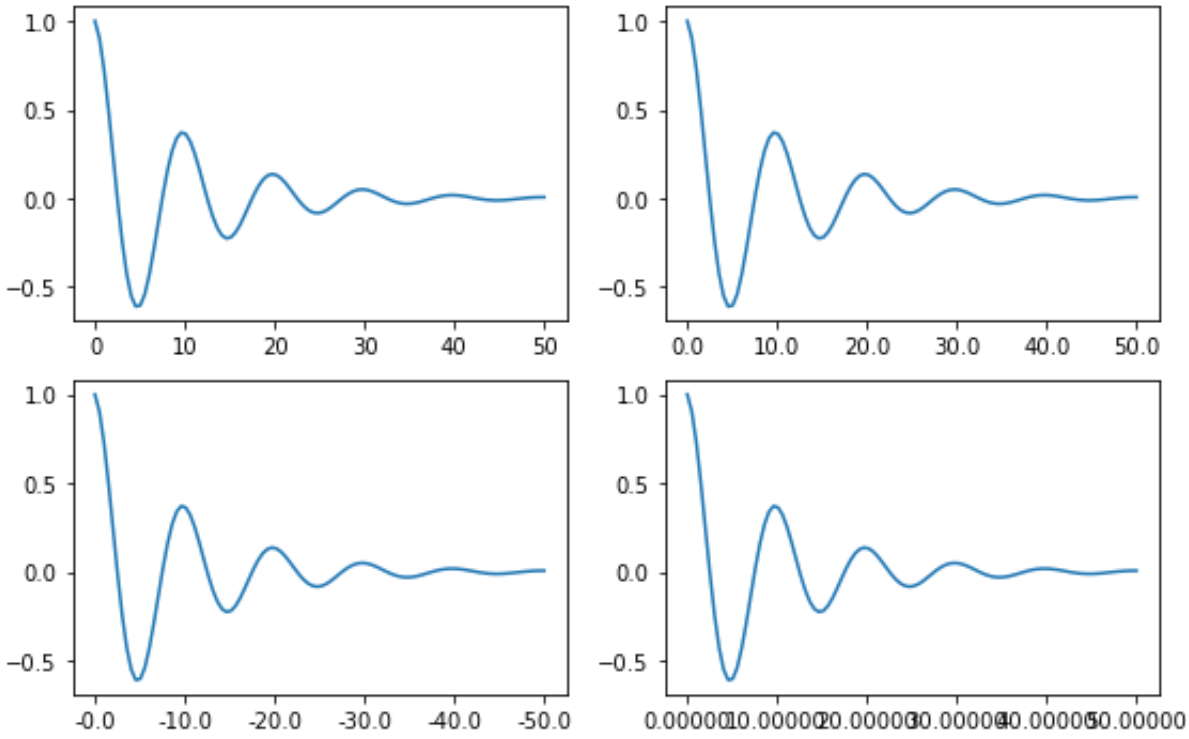
2.Tick Locators and Formatter

- 除了上述的简单模式，还可以使用Tick Locators and Formatters完成对于刻度位置和刻度标签的设置。这种方式的好处是不用显式地列举出刻度值列表。

- (a)Tick Formatters

In [7…

```
# 接收字符串格式的例子
fig, axs = plt.subplots(2, 2, figsize=(8, 5), tight_layout=T
for n, ax in enumerate(axs.flat):
    ax.plot(x1*10., y1)

formatter = matplotlib.ticker.FormatStrFormatter('%1.1f')
axs[0, 1].xaxis.set_major_formatter(formatter)

formatter = matplotlib.ticker.FormatStrFormatter('-%1.1f')
axs[1, 0].xaxis.set_major_formatter(formatter)

formatter = matplotlib.ticker.FormatStrFormatter('%1.5f')
axs[1, 1].xaxis.set_major_formatter(formatter);
```



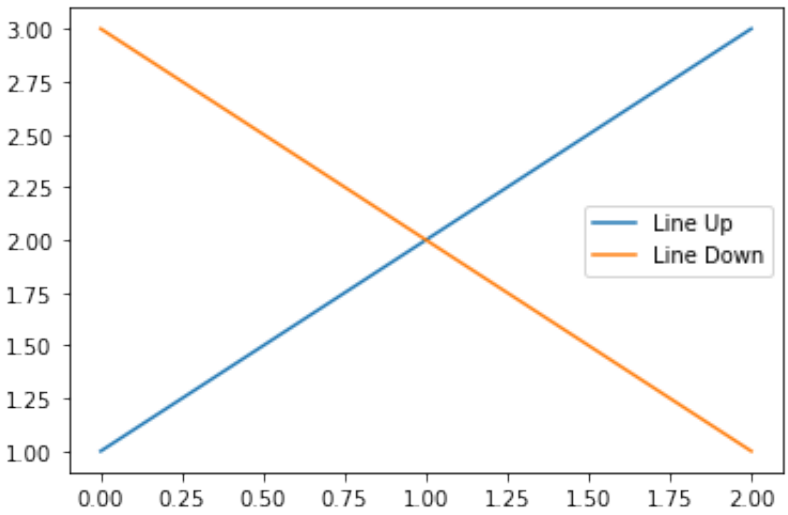
- (b) Tick Locators
- 在普通的绘图中，我们可以直接通过上图的set_ticks进行设置刻度的位置，缺点是需要自己指定或者接受matplotlib默认给定的刻度。当需要更改刻度的位置时，matplotlib给了常用的几种locator的类型。如果要绘制更复杂的图，可以先设置locator的类型，然后通过 `axs.xaxis.set_major_locator(locator)` 绘制即可
- 此外 `matplotlib.dates` 模块还提供了特殊的设置日期型刻度格式和位置的方式

三、legend（图例）

- 在具体学习图例之前，首先解释几个术语：
  - legend entry（图例条目):每个图例由一个或多个legend entries组成。一个entry包含一个key和其对应的label。
  - legend key（图例键):每个legend label左面的colored/patterned marker（彩色/图案标记）
  - legend label（图例标签):描述由key来表示的handle的文本
  - legend handle（图例句柄):用于在图例中生成适当图例条目的原始对象
- 图例的绘制同样有OO模式和pyplot模式两种方式，写法都是一样的，使用legend()即可调用。

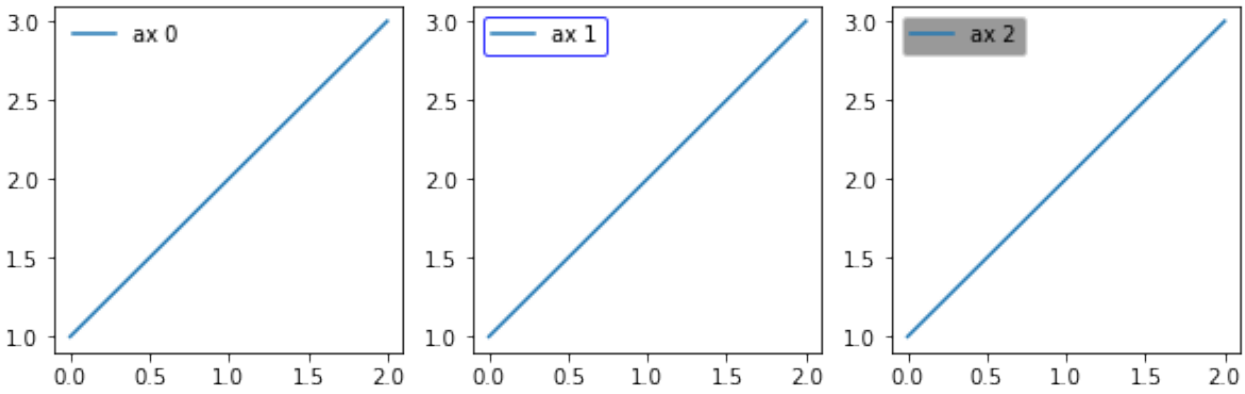
In [8…

```
fig, ax = plt.subplots()
line_up, = ax.plot([1, 2, 3], label='Line 2')
line_down, = ax.plot([3, 2, 1], label='Line 1')
ax.legend(handles = [line_up, line_down], labels = ['Line Up
```



In [9…

```
# 设置图例边框及背景
fig = plt.figure(figsize=(10,3))
axes = fig.subplots(1,3)
for i, ax in enumerate(axes):
    ax.plot([1,2,3],label=f'ax {i}')
axes[0].legend(frameon=False) #去掉图例边框
axes[1].legend(edgecolor='blue') #设置图例边框颜色
axes[2].legend(facecolor='gray'); #设置图例背景颜色,若无边框,参数
```



思考题

- 请尝试使用两种方式模仿画出下面的图表(重点是柱状图上的标签)，本文学习的text方法和matplotlib自带的柱状图标签方法bar_label

In […