

Part I

Python 数据可视化之 matplotlib 实践

Chapter 1

1.1 添加有指示注解和无指示注解

有指示注解是通过箭头指示的方法对绘图区域中的内容进行解释的标注方法。无指示注解是单纯使用文本进行内容解释或是说明的标注方法。

有指示注解和无指示注解的添加方法主要是通过函数 `annotate()` 和 `text()` 来实现的。
通过调用

```
ax.annotate(s, xy, xycoords, xytext, textcoords, weight, color, arrowprops)
```

语句来实现绘制有指示注解的目标，其中参数的含义如下所示。

Chapter 2

划分画布的主要函数

子区顾名思义就是将画布分成若干子画布，这些子画布构成绘图区域，在这些绘图区域上分别绘制图形。因此，子区的本质就是在纵横交错的行列网格中，添加绘图坐标轴。这样就实现了一张画布多张图形分区域展示的效果。这也是组织子区相关代码的逻辑顺序。

2.1 函数 `subplot()`：绘制网格区域中的几何形状相同的子区布局

这个函数是专门用来绘制网格区域中的几何形状相同的子区布局。子区函数的调用签名可以是

```
subplot(numRows, numCols, plotNum)
```

也可以是

```
subplot(CRN)
```

如果子区函数 `subplot()` 的三个参数分别是整数 `C`、整数 `R` 和整数 `P`，即 `subplot(C, R, P)`，那么这三个整数就表示在 `C` 行、`R` 列的网格布局上，子区 `subplot()` 会被放置在第 `P` 个位置上，即为将被创建的子区编号，子区编号从 1 开始，起始于右上角，序号依次向右递增。也就是说，每行的子区位置都是从左向右进行升序计数的，即 `subplot(2, 3, 4)` 是第 2 行的第 1 个子区。

2.2 函数 subplot2grid(): 让子区跨越固定的网格布局

子区函数只能绘制等分画布形式的图形样式，要想按照绘图区域的不同展示目的，进行非等分画布形式的图形展示，需要向画布多次使用子区函数 subplot() 完成非等分画布的展示任务，但是这么频繁地操作显得非常麻烦，而且在划分画布时易于出现疏漏和差错。因此，我们需要用高级的方法使用子区，需要定制化的网格区域，这个函数就是 subplot2grid()，通过使用 subplot2grid() 函数的 rowspan 和 colspan 参数可以让子区跨越固定的网格布局的多个行和列，实现不同的子区布局。

调用函数 subplot2grid(shape, loc)，将参数 shape 所划定的网格布局作为绘图区域，实现在参数 loc 位置处绘制图形的目的。

2.2.1 延伸阅读——模块 gridspec 中的类 GridSpec 的使用方法

在 matplotlib 中，存在一个模块 gridspec。模块 gridspec 是一个可以指定画布中子区位置或者说是布局的“分区”模块。在模块 gridspec 中，有一个类 GridSpec。类 GridSpec 可以指定网格的几何形状，也就是说，可以划定一个子区的网格状的几何结构。我们需要设定网格的行数和列数，以此确定子区的划分结构样式。

2.3 函数 subplots(): 创建一张画布带有多个子区的绘图模式

使用函数 subplots()，只用一句 matplotlib.pyplot.subplots() 调用命令就可以非常便捷地创建 1 行 1 列的网格布局的子区，而且同时创建一个画布对象。也就是说，函数 subplots() 的返回值是一个(fig, ax) 元组，其中，fig 是 Figure 实例，ax 可以是一个 axis 对象，如果是多个子区被创建，那么 ax 可以是一个 axis 对象数组。因此，使用函数 subplots() 可以创建一张画布带有多个子区的绘图模式的网格布局。

如果我们想要改变子区边缘相距画布边缘的距离和子区边缘之间的高度与宽度的距离，可以调用函数 subplots_adjust(*args, **kwargs) 进行设置，其中的关键字参数 left、right、bottom、top、hspace 和 wspace 都有默认值，而且使用 Axes 坐标轴系统度量的，即使用闭区间 [0,1] 的浮点数，前四个关键字参数可以调节子区距离画布的距离，关键字参数 wspace 控制子区之间的宽度距离，关键字参数 hspace 控制子区之间的高度距离。因此，借助函数 subplots_adjust() 可以有效实现子区的画布布局的空间位置的调整。

Chapter 3

共享绘图区域的坐标轴

在使用 matplotlib 实践 Python 数据可视化的过程中，我们都离不开一个重要的呈现载体：画布（figure）。我们的所有数据可视化实践都是在画布上进行操作和展示的。因此，画布的有效和正确地使用就成为需要重点研究的方面。要想实现画布的合理使用，可以借助共享绘图区域的坐标轴实现。因为，坐标轴是图形的重要载体，同时也是划分画布绘图区域的有效展示工具。

3.1 共享单一绘图区域的坐标轴

通过使用子区可以在一张画布中创建多个绘图区域，然后在每个绘图区域分别绘制图形。有时候，我们又想将多张图形放在同一个绘图区域，不想在每个绘图区域只绘制一幅图形。这时候，就可以借助共享坐标轴的方法实现在一个绘图区域绘制多幅图形的目的。

3.2 共享不同子区绘图区域的坐标轴

很多时候，我们需要共享不同子区的绘图区域的坐标轴，以求强化绘图区域的展示效果，实现精简绘图区域的目的。这时，我们通过调整函数 `subplots()` 中的参数 `sharey`（或是参数 `sharex`）的不同取值情况，从而实现共享不同子区的绘图区域的坐标轴的需求。

具体而言，参数 `sharex` 和参数 `sharey` 的取值形式有四种，分别是“row”，“col”，“all”和“none”，其中“all”和“none”分别等同于“True”和“False”。

3.3 共享个别子区绘图区域的坐标轴

我们可以对个别子区做出更加细微的局部调整，以求视图展示效果更加理想和美观。

3.3.1 延伸阅读——用函数 `autoscale()` 调整坐标轴范围

如果我们对某一个子区的坐标轴范围和数据范围的搭配比例不是很满意，可以使用函数 `autoscale()` 进行坐标轴范围的自适应调整，以使图形可以非常紧凑地填充绘图区域。调用签名是

```
autoscale(enable=True,axis="both",tight=True)
```

调用签名中的具体参数的含义如下所示。

- `enable`: 进行坐标轴范围的自适应调整。
- `axis`: 使 `x`、`y` 轴都进行自适应调整。
- `tight`: 让坐标轴的范围调整到数据的范围上。

Chapter 4

坐标轴高阶应用

4.1 设置坐标轴的位置和展示形式

不同于使用子区函数 `subplot()`、`subplot2grid()` 和模块 `matplotlib.gridspec` 构建子区的方法，这些方法都只能在规则网格内进行视图布局。也就是说，只能在横纵交错的网格区域绘制子区模式，无法完成子区的交错、覆盖和重叠等视图组合模式。

函数

```
axes(rect, frameon=True, facecolor="y")
```

的参数含义分别如下所示：

- 关键字参数 `rect` 也就是列表 `rect=[left,bottom,width,height]`，列表 `rect` 中的 `left` 和 `bottom` 两个元素分别表示坐标轴的左侧边缘和底部边缘距离画布边缘的距离，`width` 和 `height` 两个元素分别表示坐标轴的宽度和高度，`left` 和 `width` 两个元素的数值都是画布宽度的归一化距离，`bottom` 和 `height` 两个元素的数值都是画布高度的归一化距离。
- 关键字参数 `frameon` 的含义是如果布尔型参数 `frameon` 取值 `True`，则绘制坐标轴的四条轴脊；否则，不绘制坐标轴的四条轴脊。
- 关键字参数 `facecolor` 的含义是填充坐标轴背景的颜色。

我们也可以通过调用函数 `axis()` 实现绘制坐标轴，再绘制图形的可视化需求。

4.2 使用两种方法控制坐标轴刻度的显示

一种方法是利用 `matplotlib` 的面向对象的 `Axes.set_xticks()`和 `Axes.set_yticks()` 实例方法，实现不画坐标轴刻度的需求；另一种方法是调用模块 `pyplot`的 API，使用函数 `setp()` 设置刻度元素（`ticklabel` 和 `tickline`），更新显示属性的属性值为 `False`。

我们可以通过 `Line2D` 实例的方法 `set_attr(attrValue)`实现改变实例属性值的目标，其中，`attr` 代表 `Line2D` 实例的属性，`attrValue` 代表 `Line2D` 实例的 `attr` 属性的属性值。

4.3 控制坐标轴的显示

控制坐标轴显示主要是通过控制坐标轴的载体（轴脊）的显示来实现的，在轴脊上有刻度标签和刻度线，它们共同组成了坐标轴。因此，控制坐标轴显示是综合通过控制轴脊和刻度线的显示来完成的。

在一个绘图区域中，有 4 条轴脊，分别是顶边框、右边框、底边框和左边框，这 4 条轴脊是 4 条坐标轴的载体，起到显示刻度标签和刻度线的作用。

4.4 移动坐标轴的位置

所谓移动坐标轴的位置就是移动坐标轴的载体（轴脊）的位置，进而设置刻度线的位置，从而完成移动坐标轴的位置的任务。

Chapter 5

设置线条类型和标记类型的显示样式

在 matplotlib 的大量实践中，会频繁地进行折线图的线条类型和标记类型的设置工作。更加重要的是，线条类型和标记类型的显示样式的美观与否会极大地影响 Python 数据可视化的效果。

这部分会涉及字典数据结构作为关键字参数的使用方法、线条类型的设置方法和标记类型的设置方法。

5.1 不同调用签名形式的字典使用方法

在 Python 数据可视化的代码实现中，大量运用了字典数据结构。在函数或是实例方法的调用签名中，字典数据结构经常作为关键字参数值进行调用而传入代码块中。通过使用字典设置相应属性的属性值，大大提高了代码的简洁程度，并减少了重复设置的烦琐工作。

5.2 线条类型的显示样式设置方法

在折线图中，我们通过函数或是实例方法 `plot()` 的关键字参数 `linestyle(ls)` 设置线条类型的显示样式。

图 5.1: 标记类型的显示样式



5.3 标记类型的显示样式设置方法

5.4 延伸阅读

5.4.1 “破折号” 线条样式的不同展现形式的设置方法

线条类型是“破折号”样式的折线呈现多种展现形式，实现多种展现形式的关键是关键字参数 `dashes` 的使用。折线是由若干个数据点所组成的，如果我们将这些数据点中的一些数据点有规律地抹掉，就会出现“破折号”样式的折线。因此，控制数据点的抹去模式就可以实现“破折号”样式的折线的多种展现形式。

语句

```
ax.plot(x, y + 0.4, dashes=[2, 2, 8, 2])
```

里的关键字参数 `dashes` 的取值含义是：折线组成单元是由 2 个数据点的线段、2 个数据点的间隔、8 个数据点的线段和 2 个数据点的间隔所组成的结构单元。

5.4.2 标记填充样式的设置方法

??已经介绍过有关标记类型的显示样式的相关内容。现在，我们再进一步考虑标记样式能否通过标记填充样式得以展现，也就是说，借助标记填充样式的选择也可以同样实现标记

显示样式的设置需求，而且同种标记类型会由于标记填充样式的不同而呈现出更加丰富的展示效果，这就极大地丰富了标记展示样式的内容。

5.4.3 函数 `plot()` 的调用签名的设置方法

函数 `plot()` 用来绘制有序数对的折线和标记的绘图函数。函数 `plot()` 的典型调用签名如下。

```
plot([x], y, [fmt], **kwargs)
plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)
```

参数 `x` 和 `y` 是输入值，然而参数 `x` 是选择输入值，如果省略 `x` 输入值，`x` 输入值就是列表 `[0, 1, ..., N-1]`，其中 `N` 是输入值 `y` 的元素个数。一般 `x` 和 `y` 输入值是长度 `N` 的数组，也可以是常数值组成的列表。对于参数 `fmt`，我们可以使用参数 `fmt` 控制线条颜色、标记样式和线条风格，也就是说，`fmt=[color][marker][linestyle]`。对于参数 `fmt` 的使用而言，这是格式化折线图的基本方式。对于线条颜色、标记样式和线条风格而言，我们可以选择其中的一种格式化方式或是多种格式化方式。因此，参数 `fmt` 是一种便捷的字符串式注释。

说起折线图 `plot()` 的关键字参数（keyword arguments），就不得不提实例 `Line2D`。实例 `Line2D` 的属性可以作为关键字参数用来控制折线图的展现样式。例如，线条标签（`label`）、线条宽度（`linewidth`）、标记样式（`linestyle`）、标记颜色（`markerfacecolor`）等。也就是说，实例具有的属性是生成实例的类、函数或方法的关键字参数。因此，查找实例具有的工作就可以通过遍历实例对应的类、函数或方法的关键字参数来完成。而且，实例 `Line2D` 的属性作为关键字参数经常和参数 `fmt` 混合使用，共同完成控制折线图的展示效果的任务。

值得注意的是，在参数 `fmt` 和关键字参数存在冲突时，关键字参数优先执行绘图样式。

间序列图可以理解成折线图的一种变形或是特例。也就是说，时间序列图是将 `x` 轴或是 `y` 轴用日期（`date`）标示，反映数据随时间延伸的趋势变化或是规律。在 `matplotlib` 中，时间序列图是包含日期的折线图，实现函数是 `plot_date()`，函数 `plot_date()` 的参数和函数 `plot()` 的参数类似，只是坐标轴的刻度标签被格式化为日期数据。实例 `Line2D` 的属性依然可以作为函数 `plot_date()` 中的关键字参数 `kwargs`，绘图格式化参数 `fmt` 依然可以使用，如果关键字参数 `xdate` 或是 `ydate` 取值是 `True`，那么参数 `x` 和参数 `y` 的取值就会被理解成 `matplotlib` 中的日期。使用函数 `plot_date()` 实现的时间跨度可以任意设定。