10

Fundamentals of Visualization

聊聊可视化

主要了解 Matplotlib、Plotly 两个工具

你能想象的所有东西都是真的。

Everything you can imagine is real.

—— 毕加索 (Pablo Picasso) | 西班牙艺术家 | 1881 ~ 1973

10.1 解剖一幅图

本章和接下来两章介绍如何实现鸢尾花书中最常见的可视化方案。这三章内容本着“够《编程不难》用就好”为原则，不会特别深究某个具体可视化方案中的呈现细节，也不会探究其他高阶的可视化方案。

鸢尾花书《可视之美》专注提供可视化的“家常菜菜谱”。

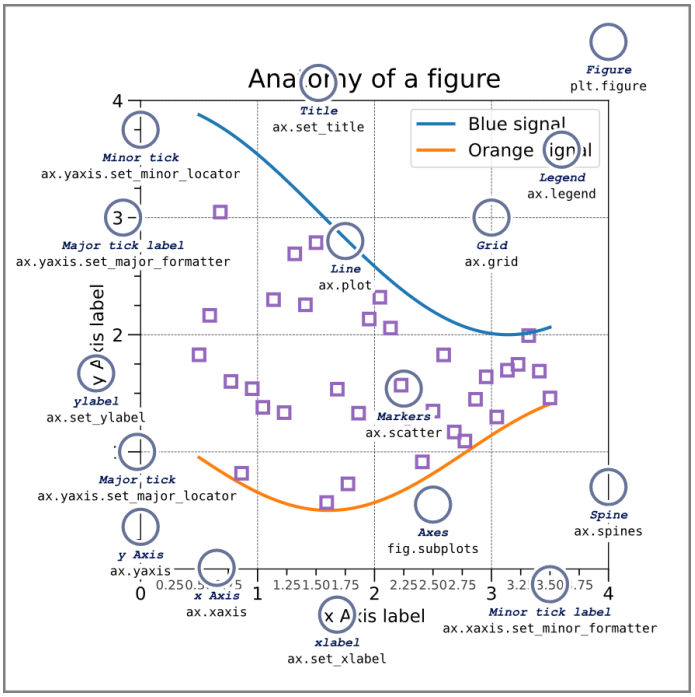

图 1. 解剖一幅图，来源 https://matplotlib.org/stable/gallery/showcase/anatomy.html

如图 1 所示，一幅图的基本构成部分包括以下几个部分：

- **图像区域** (Figure)：整个绘图区域的边界框，可以包含一个或多个子图。
- **子图区域** (Axes)：实际绘图区域，包含坐标轴、绘制的图像和文本标签等。
- **坐标轴** (Axis)：显示子图数据范围并提供刻度标记和标签的对象。
- **脊柱** (Spine)：连接坐标轴和图像区域的线条，通常包括上下左右四条。
- **标题** (Title)：描述整个图像内容的文本标签，通常位于图像的中心位置或上方，用于简要概括图像的主题或内容。
- **刻度** (Tick)：刻度标记，表示坐标轴上的数据值。
- **标签** (Label)：用于描述坐标轴或图像的文本标签。
- **图例** (Legend)：标识不同数据系列的图例，通常用于区分不同数据系列或数据类型。

▶ **艺术家** (Artist)：在 Matplotlib 中，所有绘图元素都被视为艺术家对象，包括图像区域、子图区域、坐标轴、刻度、标签、图例等等。

可视化工具

图 1 这幅图是用 Matplotlib 库绘制。Matplotlib 是 Python 中最基础的绘图工具。鸢尾花书中最常用的绘图库包括：Matplotlib、Seaborn、Plotly。

Matplotlib 可能是 Python 中最常用的绘图库，Matplotlib 具有丰富的绘图功能和灵活的使用方式。Matplotlib 可以绘制多种类型的图形，包括折线图、散点图、柱状图、饼图、等高线图等各种二维、三维图像，还可以进行图像处理和动画制作等。图 17、图 18、图 19 给出 Matplotlib 中常见的可视化方案。

Seaborn 是基于 Matplotlib 的高级绘图库，专注于统计数据可视化。它提供了多种高级数据可视化技术，包括分类散点图、热图 (热力图)、箱线图、分布图等，可以快速生成高质量的统计图表。Seaborn 适用于数据分析、数据挖掘和机器学习等领域。

⚠ 注意，Matplotlib 和 Seaborn 生成的都是静态图，即图片。

Plotly 是一个交互式可视化库，可以生成高质量的静态和动态图表。它提供了丰富的图形类型和交互式控件，可以通过滑块、下拉列表、按钮等方式动态控制图形的显示内容和样式。Plotly 适用于 Web 应用、数据仪表盘和数据科学教育等领域。类似 Plotly 的 Python 库还有 Bokeh、Altair、Pygal 等。

鸢尾花书中，大家会发现 PDF 书稿、纸质书图片一般会使用 Matplotlib、Seaborn 生成的矢量图，配套的 JupyterLab Notebook、Streamlit 则倾向于采用 Plotly。

本书第六大板块“数据”会介绍 Pandas 本身、Seaborn 的统计描述可视化方案。

10.2 使用 Matplotlib 绘制线图

下面我们聊一下如何用 Matplotlib 可视化正弦、余弦函数，图 2 所示代码生成图 3。下面我们逐块讲解这段代码；此外，请大家在 JupyterLab 中复刻这段代码，并绘制图 3。

大家会在鸢尾花书中发现，我们用 Python 代码生成的图像和书中的图像很多细节上并不一致。产生这种偏差的原因有很多。

首先，为了保证矢量图像质量及可编辑性，每幅 Python 代码生成的图形都会经过多道后期处理。后期处理的工具包括 (但不限于) Inkscape、MS Visio、Adobe Illustrator。使用怎样的工具要根据图片类型、图片大小等因素考虑。

也就是说哪怕图 2 这种简单的线图中的所有“艺术家 (artist)”，即所有元素，都被加工过。比如，图中的数字、英文、希腊字母都是手动添加上去的 (为了保证文本可编辑)。此外，从时间角度来看，一些标注、艺术效果用 Python 写代码方生成并不“划算”。

但是，加工过程仅仅是为了美化图像，并没有篡改数据本身。不篡改数据是一条铁律，希望大家谨记。

matplotlib 绘制线图

```
# 导入包
import numpy as np
import matplotlib.pyplot as plt

# 生成横轴数据
x_array = np.linspace(0, 2*np.pi, 100)
# 正弦函数数据
sin_y = np.sin(x_array)
# 余弦函数数据
cos_y = np.cos(x_array)

# 设置图片大小
fig, ax = plt.subplots(figsize=(8, 6))

# 绘制正弦和余弦曲线
ax.plot(x_array, sin_y,
        label='sin', color='blue', linewidth=2)
ax.plot(x_array, cos_y,
        label='cos', color='red', linewidth=2)

# 设置标题、横轴和纵轴标签
ax.set_title('Sine and cosine functions')
ax.set_xlabel('x')
ax.set_ylabel('f(x)')

# 添加图例
ax.legend()

# 设置横轴和纵轴范围
ax.set_xlim(0, 2*np.pi)
ax.set_ylim(-1.5, 1.5)

# 设置横轴标签和刻度标签
x_ticks = np.arange(0, 2*np.pi+np.pi/2, np.pi/2)
x_ticklabels = [r'$0$', r'$\frac{\pi}{2}$',
                r'$\pi$', r'$\frac{3\pi}{2}$',
                r'$2\pi$']
ax.set_xticks(x_ticks)
ax.set_xticklabels(x_ticklabels)

# 横纵轴采用相同的scale
ax.set_aspect('equal')

# 将图片存成SVG格式
plt.savefig('正弦_余弦函数曲线.svg', format='svg')

# 显示图形
plt.show()
```

图 2. 用 Matplotlib 绘制正弦、余弦线图

Inkscape 是开源免费的矢量图形编辑软件，支持多种矢量图形格式，适用于绘制矢量图形、图标、插图等。MS Visio 特别适合做示意图、流程图等矢量图像。Adobe Illustrator 是 Adobe 公司

开发的专业矢量图形编辑软件，功能强大，广泛用于图形设计、插图、标志设计等。比如鸢尾花书的封面都是用 Adobe Illustrator 设计，鸢尾花书中复杂的图像也都是在这个软件设计生成。此外，也推荐大家使用 CorelDRAW。CorelDRAW 是 Corel 公司开发的矢量图形编辑软件，具有类似于 Adobe Illustrator 的功能，是一种流行的矢量图形处理工具。

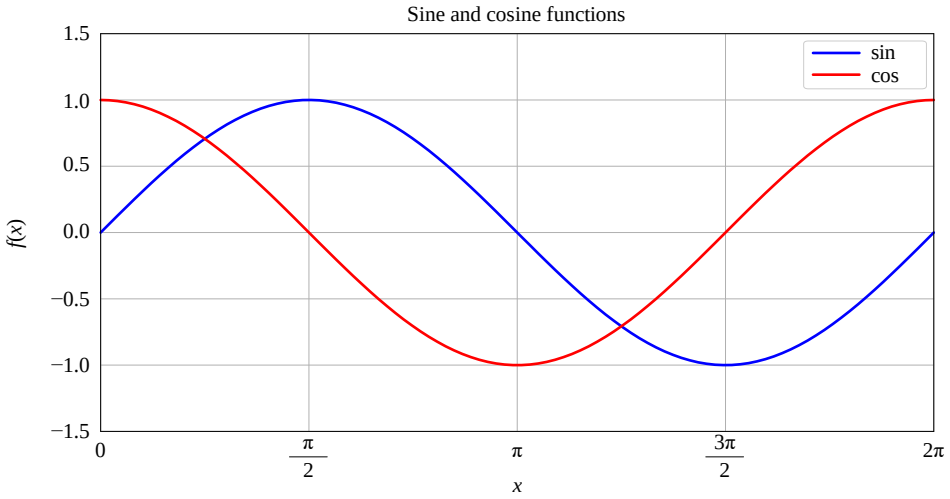

图 3. 正弦、余弦函数线图

产生等差数列

`import numpy as np` 这句代码的意思是将 NumPy (Python 代码中叫 `numpy`) 库导入到当前的 Python 程序中，并为其取一个简短的别名 np。

这意味着我们可以使用 `np` 来代替 `numpy` 来调用 NumPy 库中的函数和方法，例如 `np.linspace()`，`np.sin()`，`np.cos()` 等。这样做的好处是可以简化代码，减少打字量，并且提高代码的可读性。通常，人们将 `numpy` 取别名为 `np`，这是因为它的缩写简短且容易记忆。

`numpy.linspace()` 是 NumPy 库中的一个函数，用于生成在给定范围内等差数列。由于在导入 `numpy` 时，我们将其命名为 `np`，因此代码中大家看到的是 `np.linspace()`。

图 4. 用 numpy.linspace() 生成等差数列

上面的代码中，`0` 是数值序列的起始值，`2*np.pi` 是数值序列的结束值，`100` 是数值序列的数量。因此，`x_array = np.linspace(0, 2*np.pi, 100)` 在 [0, 2π] 闭区间内生成一个 100 个数值等差数列。

numpy.linspace(start, stop, num=50, endpoint=True)

这个函数的重要输入参数：

- start：起始点的值。
- stop：结束点的值。
- num：要生成的数据点数量，默认为 50。
- endpoint：布尔值，指定是否包含结束点。如果为 True，则生成的数据点包括结束点；如果为 False，则生成的数据点不包括结束点。默认为 True。

请大家在 JupyterLab 中自行学习下例。

```
import numpy as np

arr = np.linspace(0, 1, num=11)
print(arr)

arr_no_endpoint = np.linspace(0, 1, num=10, endpoint=False)
print(arr_no_endpoint)
```

什么是 NumPy 数组 array?

NumPy 中最重要的数据结构是 ndarray (n-dimensional array)，即多维数组。一维数组是最简单的数组形式，类似于 Python 中的列表。它是一个有序的元素集合，可以通过索引访问其中的元素。一维数组只有一个轴。二维数组是最常见的数组形式，可以看作是由一维数组组成的表格或矩阵。它有两个轴，通常称为行和列。我们可以使用两个索引来访问二维数组中的元素。多维数组是指具有三个或更多维度的数组。

正弦、余弦

如图 5 所示， numpy.sin() 和 numpy.cos() 是 NumPy 库中的数学函数，用于计算给定角度的正弦和余弦值。这两个函数的输入既可以是弧度值 (比如 numpy.pi/2)，也可以是数组 (一维、二维、多维)。

⚠ 注意，NumPy 中 numpy.deg2rad()将角度转换为弧度，numpy.rad2deg()将弧度转换为角度。

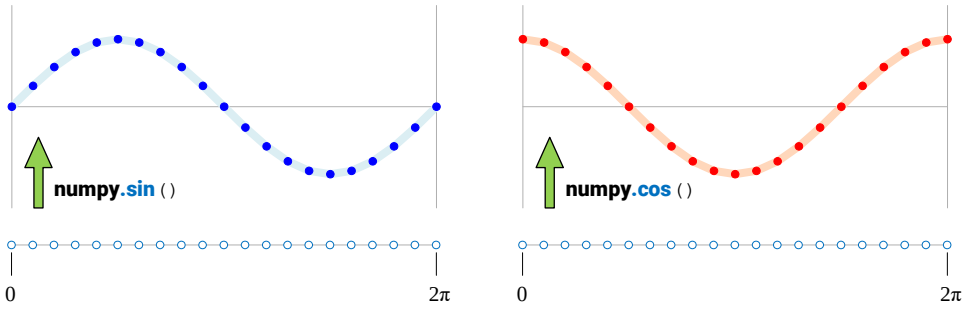

图 5. 生成正弦、余弦数据

创建图形、轴对象

`fig, ax = plt.subplots(figsize=(8, 6))` 用于创建一个新的 Matplotlib 图形 `fig` 和一个轴 `ax` 对象，并设置图形的大小为 (8, 6)，单位为英寸。

通过创建图形和轴对象，我们可以在轴上绘制图表、设置轴的标签和标题、调整轴的范围等。`fig, ax = plt.subplots()` 这一句代码常常是开始绘图的第一步，它创建了一个具有指定大小的图形和轴对象，为后续绘图操作提供了一个可用的基础。

需要注意的是，`plt` 是 Matplotlib 的一个常用的别名，通常通过 `import matplotlib.pyplot as plt` 引入。所以在使用 `plt.subplots()` 函数之前，需要确保已经正确导入了 Matplotlib 库。

添加子图

此外，我们还可以使用使用 `add_subplot()` 方法创建一个新的子图对象，并指定其所在的行、列、编号等属性。

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

x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, y)
plt.show()
```

add_subplot() 创建新的子图对象

图 6. add_subplot() 方法创建一个新的子图对象

在这个例子中，我们使用 add_subplot() 方法创建了一个新的子图对象，并将其添加到 Figure 对象中。其中，1, 1, 1 参数表示子图在 1 行 1 列的第 1 个位置，即占据整个 Figure 对象的空间。然后，我们在子图中绘制了一个正弦曲线。最后，使用 plt.show()函数显示 Figure 对象，即可在屏幕上显示绘制的图像。

绘制曲线

`ax.plot(x_array, sin_y, label='sin', color='blue', linewidth=2)` 用于在轴对象 ax 上绘制正弦曲线。x_array 为 x 轴数据，sin_y 为 y 轴数据。

`label='sin'` 设置了曲线的标签为 `'sin'`，`color='blue'` 设置曲线的颜色为蓝色，`linewidth=2` 设置曲线的线宽为 2。在 Matplotlib 中，linewidth 参数表示线条的宽度。它的单位是点 (point, pt)，通常用于测量线条、字体等绘图元素的大小。在 Matplotlib 中，默认情况下，一个点等于 1/72 inch。

颜色

在 Matplotlib 中，可以使用多种方式指定线图的颜色，包括 RGB 值、预定义颜色名称、十六进制颜色码和灰度值。

可以使用 RGB (R 是 red，G 是 green，B 是 blue) 来指定颜色，其中每个元素的值介于 0 到 1 之间。例如，(1, 0, 0) 表示纯红色，(0, 1, 0) 表示纯绿色。使用 RGBA 值指定“颜色 + 透明度 (A)”。

如图 8 所示，RGB 三原色模型实际上构成了一个色彩“立方体”——一个色彩空间。

鸢尾花书《矩阵力量》将会用 RGB 三原色模型讲解线性代数中向量空间 (vector space) 这个重要概念。

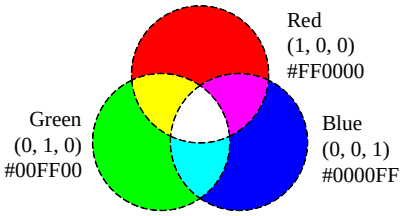

图 7. RGB 三原色模型

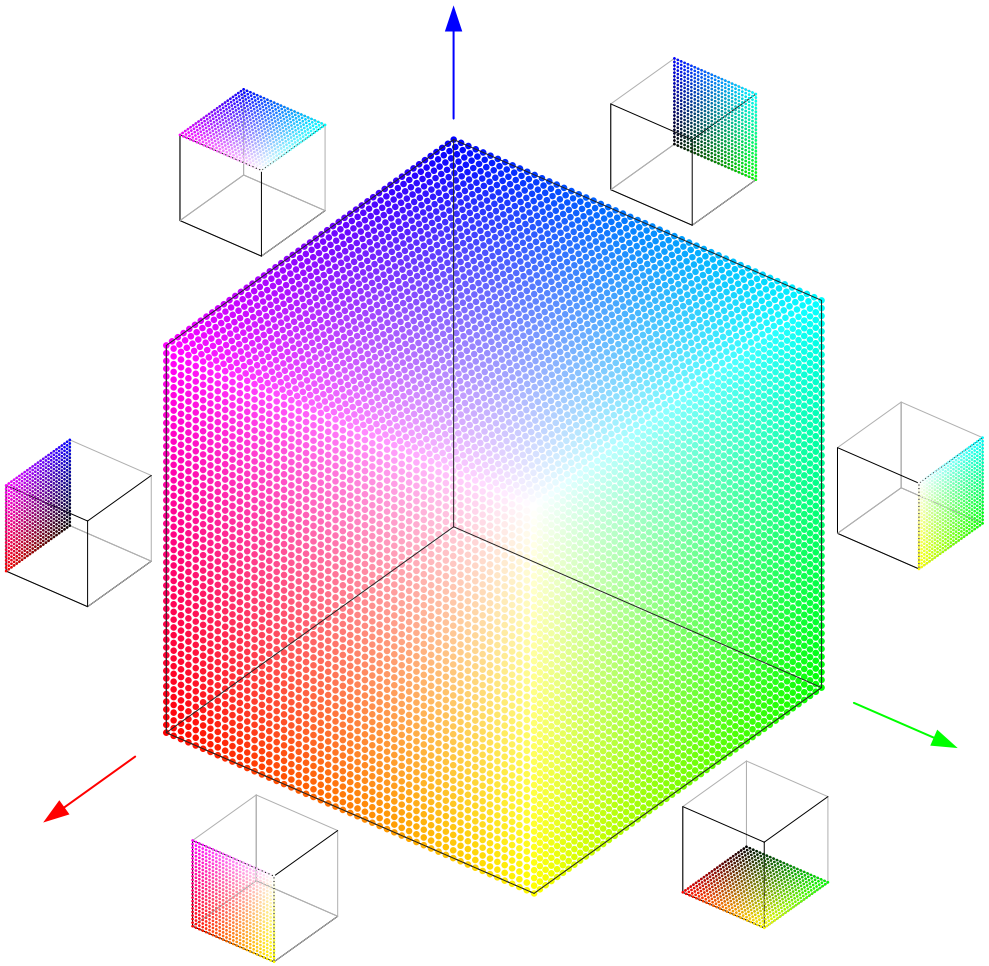

图 8. RGB 三原色模型“立方体”

什么是 RGB 颜色模式？

RGB (红绿蓝) 颜色模式是一种使用红、绿、蓝三个基本颜色通道来表示颜色的方法。在 RGB 模式中，通过调整每个通道的强度 (从 0 到 255 的值，Matplotlib 中 0 到 1 的值) 来创建各种颜色。通过组合不同强度的红、绿和蓝，可以形成几乎所有可见光颜色。RGB 颜色模式被广泛应用于计算机图形、数字图像处理和网页设计等领域，它提供了一种直观、灵活且广泛支持的方式来表示和操作颜色。

Matplotlib 提供了一些常见颜色的预定义名称，例如 'red'、'green'、'blue' 等。图 16 所示为在 Matplotlib 中已经预定义名称的颜色。

大家还可以使用十六进制颜色码来指定颜色。它以 '#' 开头，后面跟着六位十六进制数。例如，'#FF0000' 表示纯红色，'#00FF00' 表示纯绿色。

我们还可以使用灰度值来指定颜色，取值介于 0 到 1 之间，表示不同的灰度级别。'0' 表示黑色，'1' 表示白色。比如，color='0.5' 代表灰度值为 0.5 的灰色。

使用色谱

Matplotlib 中还有一种渐变配色方案——colormap。在 Matplotlib 中，colormap 用于表示从一个端到另一个端的颜色变化。这个变化可以是连续的，也可以是离散的。

在 Matplotlib 中，colormap 主要用于绘制二维图形，如热图、散点图、等高线图等。它用于将数据值映射到不同的颜色，以显示数据的变化和模式。Colormap 可以直译为“色彩映射”，鸢尾花书一般称之为“色谱”。图 9 所示为几种常见的色谱。鸢尾花书中最常用的色谱为 RdYlBu。

《可视之美》将专门讲解色谱。

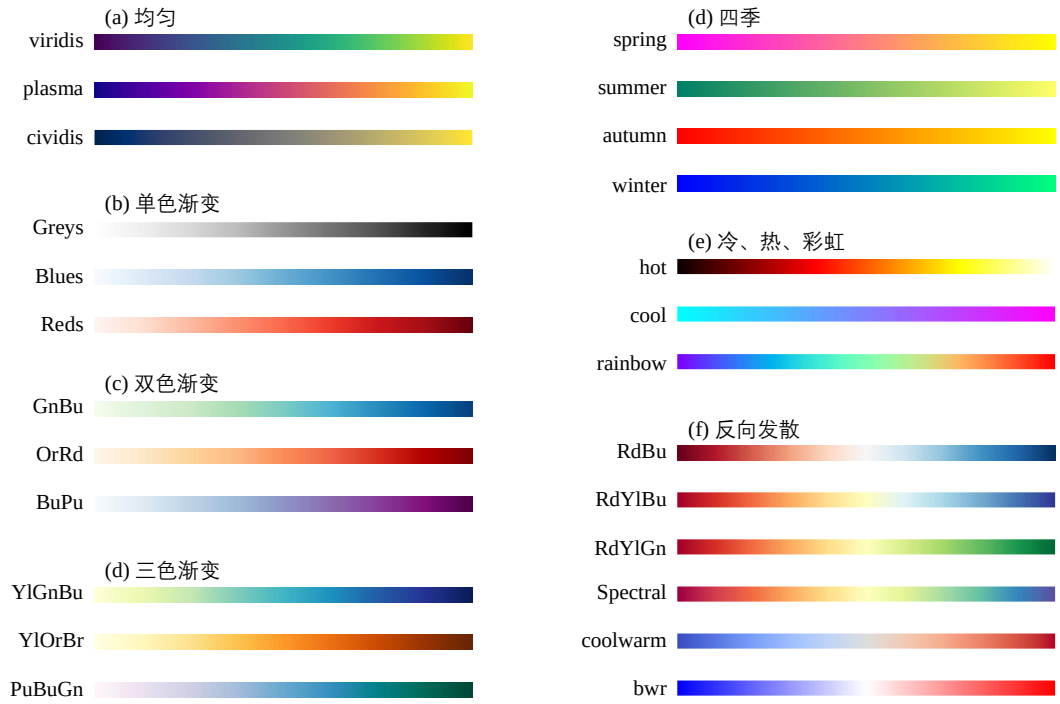

图 9. 几种常用色谱

图 10 所示为利用色谱渲染一组曲线。图 10 左图所示为一元高斯概率密度分布曲线随均值 μ 变化，图 10 右图所示为曲线随标准差 σ 变化。大家可以在本章配套 Jupyter Notebook 找到对应代码。

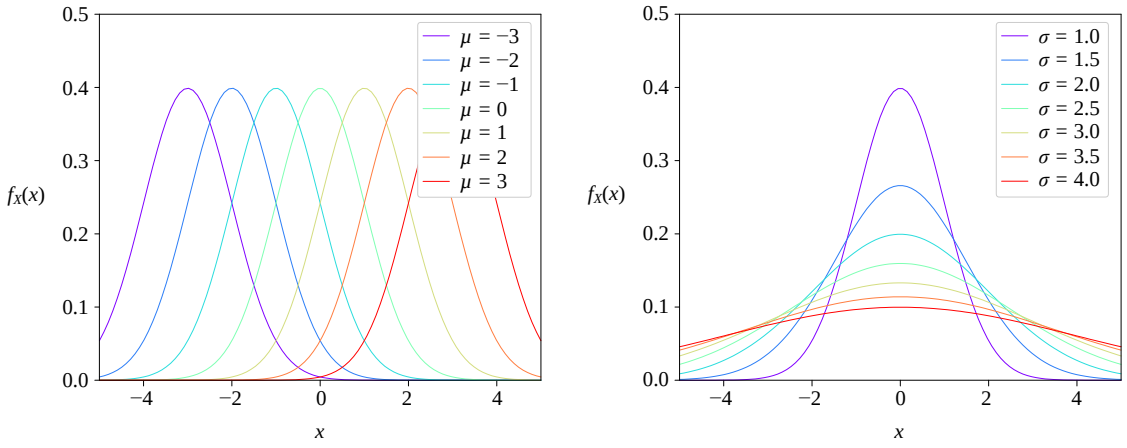

图 10. 用色谱渲染一组曲线

什么是高斯分布？

高斯分布 (Gaussian distribution)，也称为正态分布 (Normal distribution)，是统计学中常用的概率分布模型之一。它具有钟形曲线的形状，呈对称分布。高斯分布的概率密度函数可以由两个参数完全描述：均值 (mean) 和标准差 (standard deviation)。均值决定了分布的中心位置，标准差决定了分布的展开程度。

高斯分布在自然界和社会现象中广泛存在，例如身高、体重、温度等连续型随机变量常常服从高斯分布。中心极限定理也说明了许多独立同分布的随机变量的总和趋向于高斯分布。

高斯分布在统计学和数据分析中有着重要的应用，可用于描述数据集的分布特征、进行假设检验、构建回归模型等。在机器学习和人工智能领域，高斯分布在概率密度估计、聚类分析、异常检测等算法中被广泛使用。

什么是概率密度函数?

概率密度函数 (Probability Density Function，简称 PDF) 是概率论和统计学中用于描述连续型随机变量的概率分布的函数。它表示了变量落在某个特定取值范围内的概率密度，而不是具体的概率值。

一元连续随机变量的概率密度函数是非负函数，并且在整个定义域上的积分等于 1。对于给定的连续型随机变量，通过 PDF 可以计算出在不同取值范围内的概率密度值，从而了解变量的分布特征和概率分布形状。

以正态分布为例，其概率密度函数即高斯函数，可以描述变量取值的概率密度。在某个特定取值处，概率密度函数的值越高，表示该取值的概率越大。概率密度函数在统计分析、数据建模、概率推断等领域广泛应用，可用于计算概率、推断参数、生成模拟数据等。

其他细节美化

图 2 中还提供图片美化命令，下面逐一说明。

- `ax.set_title('Sine and cosine functions')` 设置图表的标题为 "Sine and cosine functions"，即正弦和余弦函数。
- `ax.set_xlabel('x')` 设置横轴标签为 "x"。`ax.set_ylabel('f(x)')` 设置纵轴标签为 "f(x)"。
- `ax.legend()` 添加图例 legend，用于标识不同曲线或数据系列。
- `ax.set_xlim(0, 2*np.pi)` 设置横轴范围从 0 到 2π。`ax.set_ylim(-1.5, 1.5)` 设置纵轴范围从 −1.5 到 1.5。
- `x_ticks = np.arange(0, 2*np.pi+np.pi/2, np.pi/2)` 生成横轴刻度的位置，从 0 到 2π，间隔为 π/2。
- `x_ticklabels = [r'$0$', r'$\frac{\pi}{2}$', r'$\pi$', r'$\frac{3\pi}{2}$', r'$2\pi$']` 设置横轴刻度的标签，分别为 0, π/2, π, 3π/2, 2π。在代码中，`r'$\frac{\pi}{2}$'` 是一个特殊的字符串，用于表示数学公式中的文本。在这个字符串前面的 `r` 前缀表示该字符串是一个“原始字符串”，即不对字符串中的特殊字符进行转义。
- 在这个特殊字符串中，使用了 LaTeX 符号来表示一个分数。具体来说，`\frac{\pi}{2}` 表示一个分数，分子是π，分母是 2。当这个字符串被用作横轴刻度的标签时，它会在图表中显示为 "π/2" 的形式。这种表示方法可以用于在图表中显示复杂的数学公式或符号。
- `ax.set_xticks(x_ticks)` 设置横轴刻度的位置。
- `ax.set_xticklabels(x_ticklabels)` 设置横轴刻度的标签。
- `ax.set_aspect('equal')` 设置横纵轴采用相同的比例，保持图形在绘制时不会因为坐标轴的比例问题而产生形变。

图片输出格式

Matplotlib 可以输出多种格式的图片，其中一些是矢量图。以下是一些常见的输出格式及其特点：

- PNG (Portable Network Graphics)：PNG 是一种常见的位图格式，支持透明度和压缩。PNG 格式输出的图片不是矢量图，因此在放大时会失去清晰度，但是可以保持较高的分辨率和细节。
- JPG/JPEG (Joint Photographic Experts Group)：JPG 是一种常见的有损压缩位图格式，用于存储照片和复杂的图像。与 PNG 不同，JPG 格式输出的图片是有损的，压缩率高时会失去一些细节，但是文件大小通常较小。
- EPS (Encapsulated PostScript)：EPS 是一种矢量图格式，可以在很多绘图软件中使用。EPS 格式输出的图片可以无限放大而不失真，适合于需要高品质图像的打印和出版工作。
- PDF (Portable Document Format)：PDF 是一种常见的文档格式，可以包含矢量图和位图。与 EPS 类似，PDF 格式输出的图片也是矢量图，可以无限放大而不失真，同时具有可编辑性和高度压缩的优势。存成 PDF 很方便插入 Latex 文档。
- SVG (Scalable Vector Graphics)：SVG 是一种基于 XML 的矢量图格式，可以用于网页和打印等多种用途。SVG 格式输出的图片可以无限放大而不失真，且文件大小通常较小。鸢尾花书的图片首选 SVG 格式。

注意，EPS、PDF 和 SVG 是矢量图格式，可以无限放大而不失真 (比如图 11 (b))，适合于需要高品质图像的打印和出版工作。在需要高品质图像的场合，最好使用这些矢量图格式。

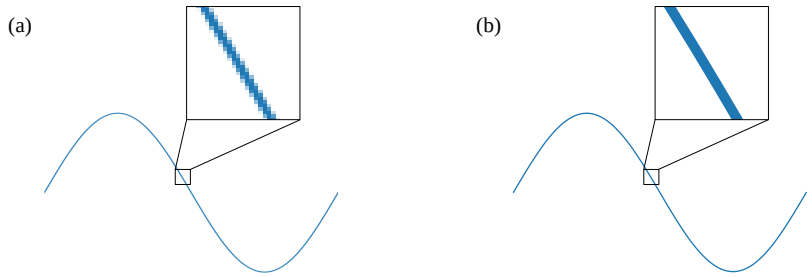

图 11. 比较非矢量、矢量图

子图

图 12 所示一行两列子图。请大家在 JupyterLab 中给图 13 代码逐行添加注释，并复刻图 12。

《可视之美》将介绍更多子图可视化方案。

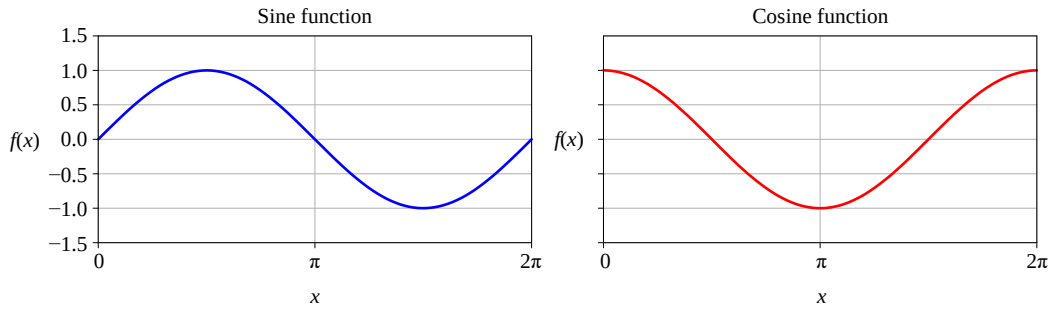

图 12. 一行、两列子图

subplots() 一行两列子图

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

x = np.linspace(0, 2 * np.pi, 100)
y_sin = np.sin(x)
y_cos = np.cos(x)

# 创建图形对象和子图布局
fig, (ax1, ax2) = plt.subplots(1, 2,
                  figsize=(10, 4),
                  sharey=True)

# 在左子图中绘制正弦函数曲线，设置为蓝色
ax1.plot(x, y_sin, color='blue')
ax1.set_title('Sine function')
ax1.set_xlabel('x')
ax1.set_ylabel('f(x)',
               rotation='horizontal',
               ha='right')
ax1.set_xlim(0, 2*np.pi)
ax1.set_ylim(-1.5, 1.5)
x_ticks = np.arange(0, 2*np.pi+np.pi/2, np.pi)
x_ticklabels = [r'$0$', r'$\pi$', r'$2\pi$']
ax1.set_xticks(x_ticks)
ax1.set_xticklabels(x_ticklabels)
ax1.grid(True)
ax1.set_aspect('equal')

# 在右子图中绘制余弦函数曲线，设置为红色
ax2.plot(x, y_cos, color='red')
ax2.set_title('Cosine function')
ax2.set_xlabel('x')
ax2.set_ylabel('f(x)',
               rotation='horizontal',
               ha='right')
ax2.set_xlim(0, 2*np.pi)
ax2.set_ylim(-1.5, 1.5)
ax2.set_xticks(x_ticks)
ax2.set_xticklabels(x_ticklabels)
ax2.grid(True)
ax2.set_aspect('equal')

# 调整子图之间的间距
plt.tight_layout()

# 显示图形
plt.show()
```

图 13. 绘制一行两列子图

10.3 使用 Plotly 绘制线图

此外，我们还可以用 Plotly 绘制具有交互属性的图形，比如图 14，对应的代码如图 15。

`plotly.graph_objects` 是 Plotly 库中的一个模块，它提供了创建和操作图形对象的类和方法。通过 `go` 的别名，我们可以方便地使用 `plotly.graph_objects` 模块中的各种类和函数。在 `plotly.graph_objects` 模块中，有许多类可用于创建各种类型的图形，如 `scatter`、`bar`、`surface` 等。

通过 `go` 模块，我们可以创建一个 `Figure` 对象，用于容纳和管理我们的图形。Figure 对象是一个图形容器，可以添加多个轨迹 (trace)，设置整体布局和样式，并最终显示或保存图形。

`fig.add_trace(go.Scatter(x=x, y=y_sin, mode='lines', name='Sine'))` 的作用是向图形对象 fig 中添加一个轨迹，其中包含了一条正弦曲线的数据和样式。`go.Scatter` 创建了一个散点图 (scatter plot) 的轨迹对象。`x=x` 指定了横轴的数据，即之前生成的 x 值数组。`y=y_sin` 指定了纵轴的数据，即正弦函数的 `y` 值数组。`mode='lines'` 设置了散点图的显示模式为连线模式，表示将数据点用线连接起来。`name='Sine'` 设置了轨迹的名称为 `"Sine"`，在图例中显示。通过 `add_trace` 方法，我们将该轨迹添加到 fig 图形对象中，使得该正弦曲线在图形中显示出来。

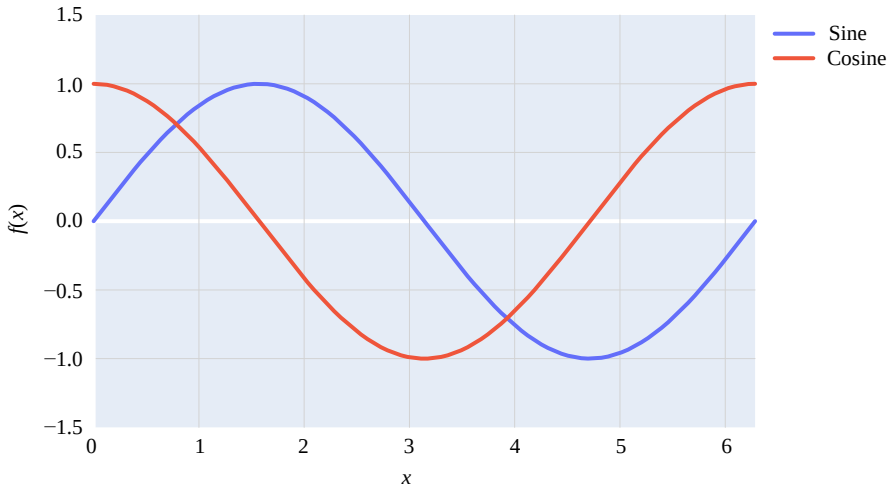

图 14. 用 Plotly 绘制具有交互性质的曲线

用Plotly绘制线图

```
import numpy as np
import plotly.graph_objects as go
import plotly.io as pio
pio.kaleido.scope.default_format = "svg"

# 生成 x 值
x = np.linspace(0, 2 * np.pi, 100)

# 生成正弦和余弦函数的 y 值
y_sin = np.sin(x)
y_cos = np.cos(x)

# 创建图形对象
fig = go.Figure()

# 添加正弦曲线
fig.add_trace(go.Scatter(x=x, y=y_sin, mode='lines',
name='Sine'))

# 添加余弦曲线
fig.add_trace(go.Scatter(x=x, y=y_cos, mode='lines',
name='Cosine'))

# 设置横轴和纵轴范围
fig.update_layout(xaxis_range=[0, 2 * np.pi],
                  yaxis_range=[-1.5, 1.5])

# 设置横轴和纵轴标签
fig.update_xaxes(title_text='x')
fig.update_yaxes(title_text='f(x)')

# 添加网格
fig.update_xaxes(showgrid=True, gridwidth=0.25,
gridcolor='lightgray')
fig.update_yaxes(showgrid=True, gridwidth=0.25,
gridcolor='lightgray')

# 显示图形
fig.show()

# 将fig保存为SVG格式
fig.write_image("fig.svg")
```

图 15. 用 Plotly 绘制线图

请大家完成下面 3 道题目。

Q1. 大家可以在本章配套代码中找到图 1 对应的 Matplotlib 官方提供的代码文件。本书将 Python 代码文件命名为 Q1_Assignment_Anatomy_of_a_figure.py。请大家给这个代码文件中的代码逐行中文注释，并在 JupyterLab 中进行探究式学习。

Q2. Matplotlib 提供丰富的可视化方案实例，图 17、图 18、图 19 大部分子图对应的代码都在如下链接中，请大家在 JupyterLab 复刻每幅子图，并补充必要注释。

https://matplotlib.org/stable/plot_types/index.html

* 本章习题不提供答案。

图 16. Matplotlib 已定义名称的颜色

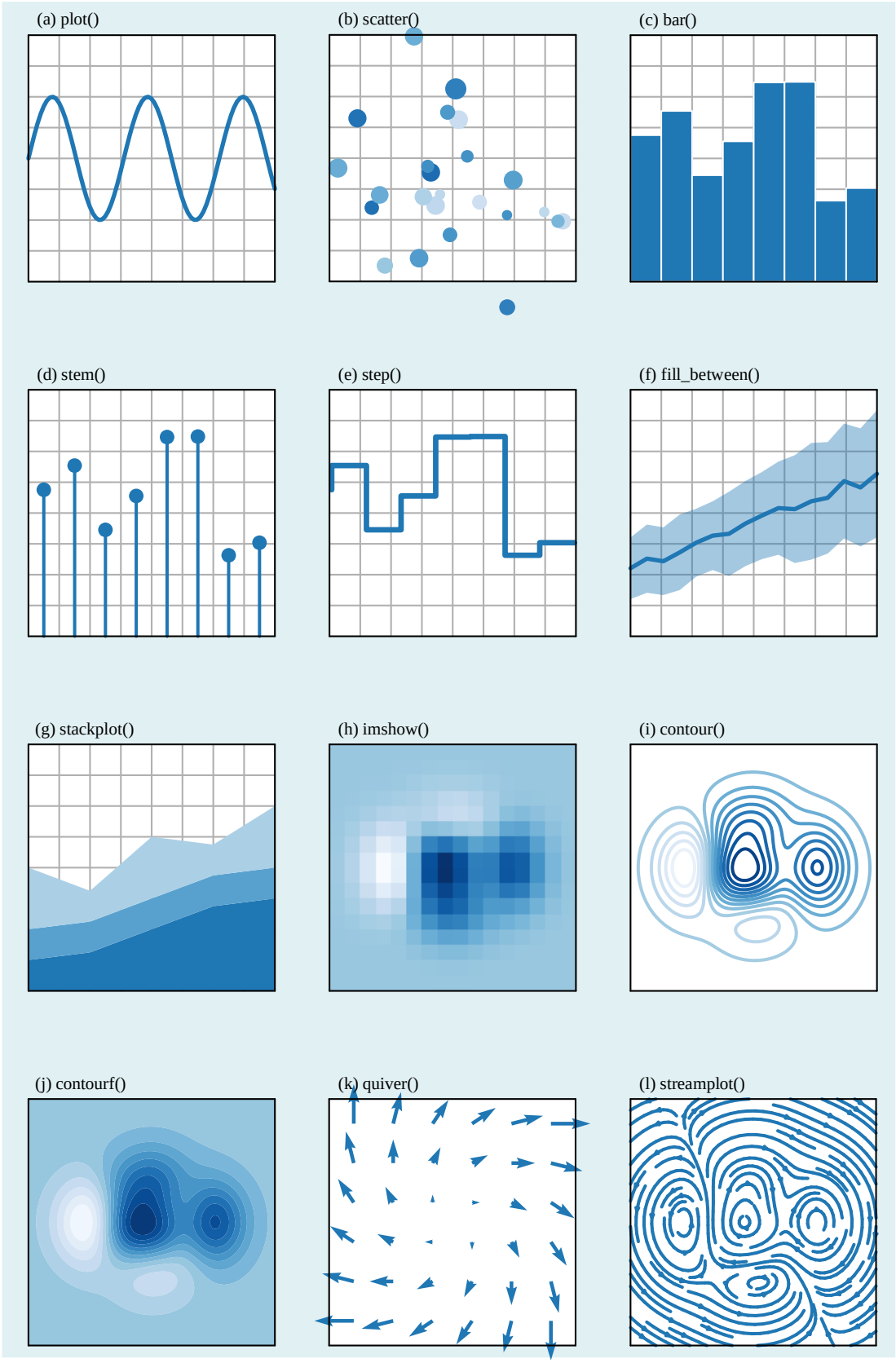

图 17. Matplotlib 常见可视化方案，第一组

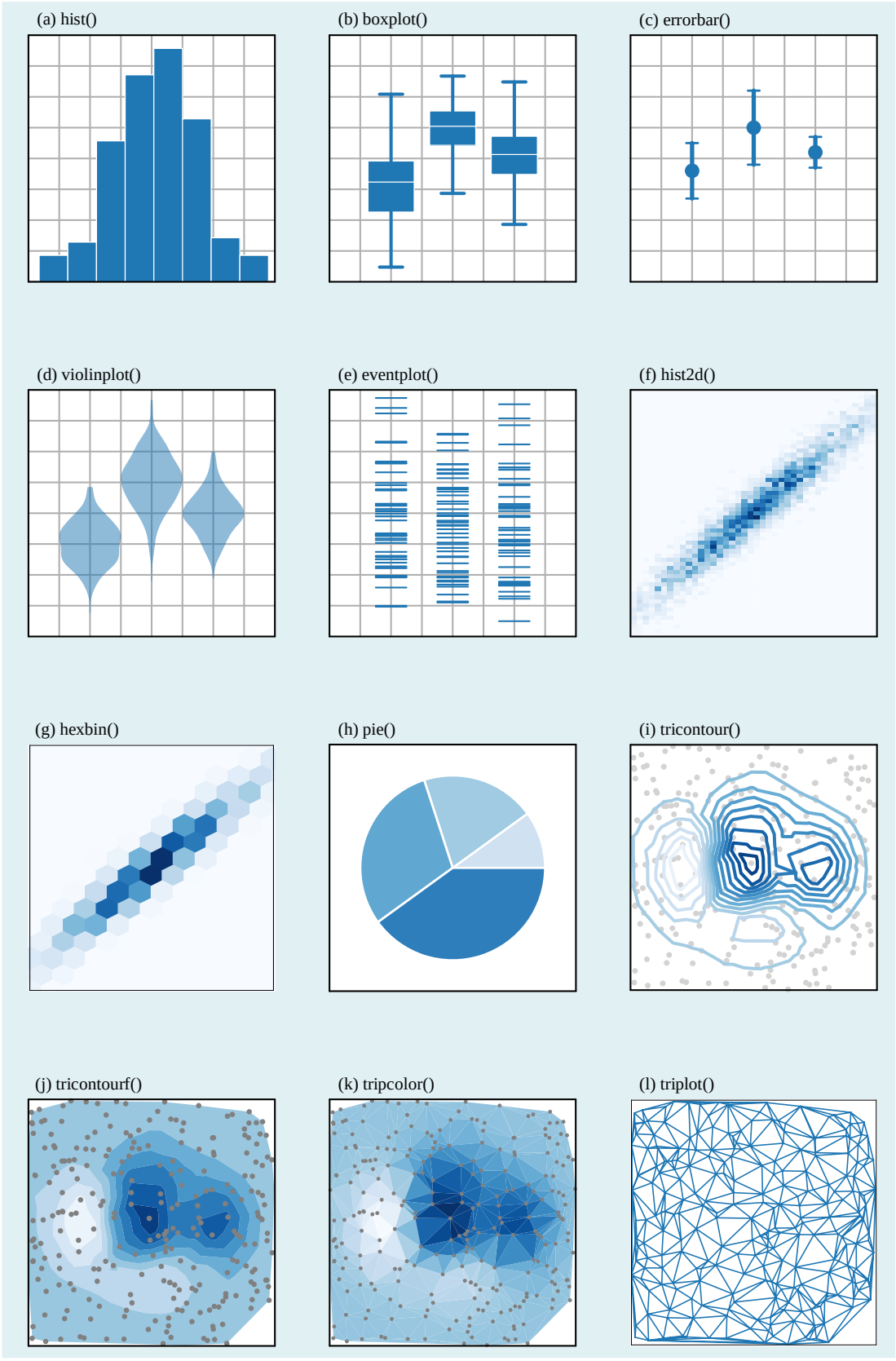

图 18. Matplotlib 常见可视化方案，第二组

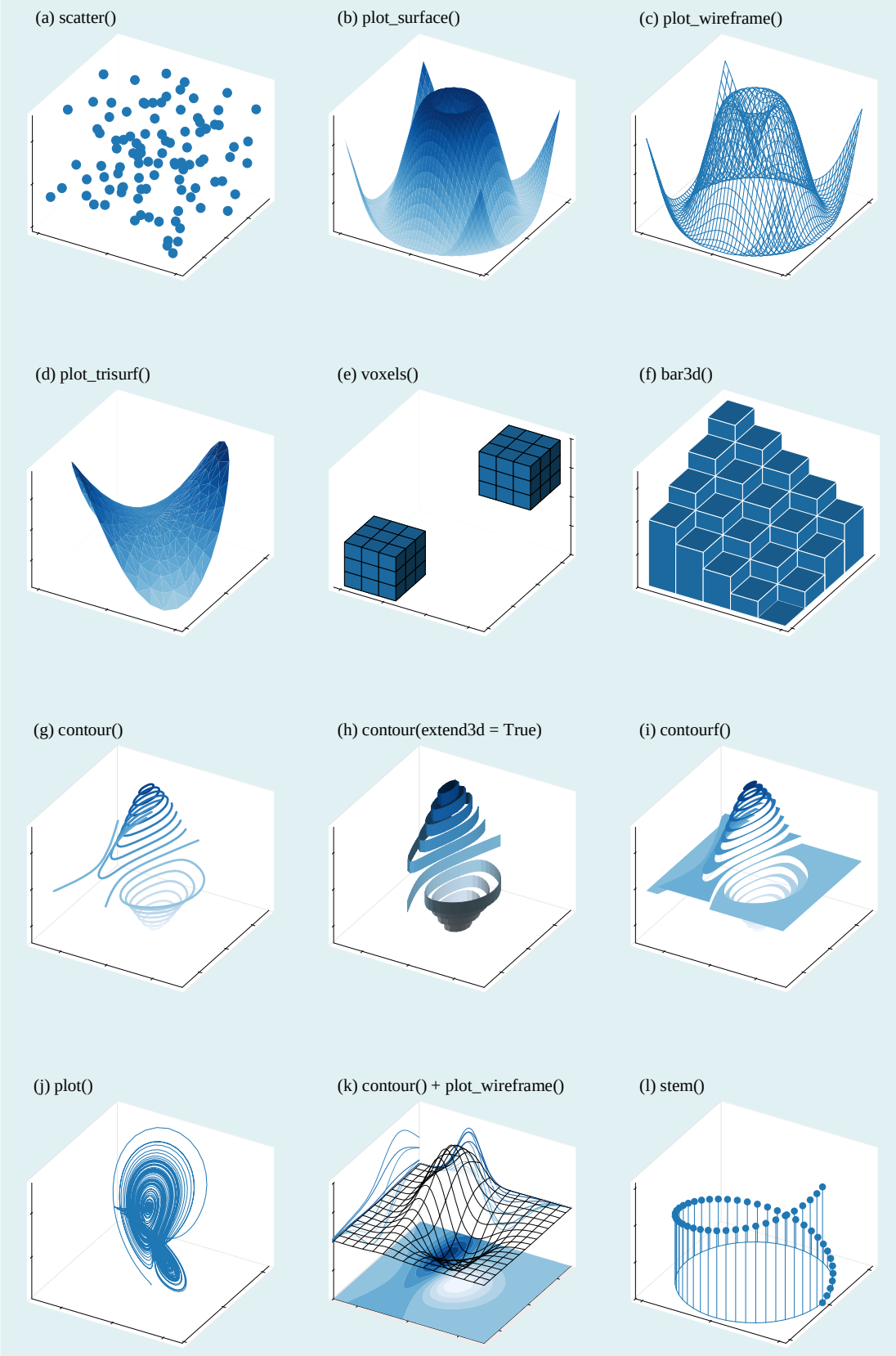

图 19. Matplotlib 常见可视化方案，第三组