

# Python 数据分析与科学计算

## 可视化篇

# 第1章 数据可视化简介

## 1.1 学习数据可视化前置知识点

- 1、Python 基础语言
- 2、重点 Python 内置数据结构:列表(list)、字符串(str)、元组(tuple)、集合(set) 和字典(dict)。
- 3、序列的索引和切片操作。
- 4、列表推导式、集合推导式、字典推导式
- 5、lambda 表达式
- 6、Python 三个函数式编程的基础函数: filter()、map() 和 reduce()。
- 7、NumPy 库
- 8、pandas 库

## 1.2 Python 数据可视化库

- 1、Matplotlib (<https://matplotlib.org/>)
- 2、Seaborn (<https://seaborn.pydata.org/>)
- 3、Pandas 中的数据结构自动可视化方法
- 4、Basemap 可视化地理数据 (<https://matplotlib.org/basemap/>)
- 5、pyecharts (<https://pyecharts.org>)

## 1.3 课后练习

- 1、访问 Matplotlib 官网，如何使用官方教程、示例和 API？
- 2、访问 Seaborn 官网，如何使用官方教程、示例和 API？

## 第2章 环境搭建

### 2.1 环境方案 1：手动安装

Python 要求 Python 3.6 以及以上版本。

#### 2.1.1 使用 pip 安装数据可视化等库

还需要安装的库 numpy、scipy、pandas、matplotlib 和 seaborn。

在 Windows 命令提示符中使用 pip 安装指令：

```
pip install matplotlib seaborn
```

#### 2.1.2 安装 IPython

IPython 是一个加强版本的 Python 解释器。

在命令提示符或终端中使用 pip 安装 IPython 指令：

```
pip install ipython
```

### 2.2 环境方案 2：安装 Anaconda

Anaconda 指的是一个开源的 Python 发行版本，其包含了 conda、Python 等 1500 多个科学包。

1、Anaconda 官网：<https://www.anaconda.com>

2、清华大学开源软件镜像站（目前暂停）：

<https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/>

### 2.2.1 Windows 安装 Anaconda

### 2.2.2 Linux 安装 Anaconda

### 2.2.3 macOS 安装 Anaconda

## 2.3 开发工具

Python shell、Python IDEL、IPython shell、IDE 工具（Pycharm、Eclipse Pydev 插件、Visual Studio Code、Spyder）、Jupyter Notebook

### 2.3.1 IPython shell

#### 1、启动

在命令提示符或终端中使用如下指令：

```
ipython
```

#### 2、获得帮助

通过?或??获取获得帮助，示例如下：

### 3、用 Tab 键语法补全

示例如下：

### 4、IPython shell 中的快捷键

表2-1 IPython shell中常用的快捷键

快捷键	说明
Ctrl + p (或向上箭头)	获取前一个历史命令
Ctrl + n (或向下箭头)	获取后一个历史命令
Ctrl + r	对历史命令的反向搜索
Ctrl + L	清除终端屏幕的内容
Ctrl + d	退出IPython会话

示例如下：

### 5、IPython shell 魔法命令

表2-2 IPython shell常用魔法命令

魔法命令	说明
%run	执行外部代码
%timeit	计算代码运行时间
%magic	获得所有可用魔法命令的列表
?	某个魔法命令帮助

示例如下：

## 2.3.2 Jupyter Notebook

Jupyter Notebook 是 IPython shell 基于浏览器的图形界面。Jupyter Notebook 不仅可以执行 Python/IPython 语句，还允许用户编写科技文章。

### 1、安装

- Anaconda 完整安装包括 Jupyter Notebook。
- 手动使用 pip 安装指令：

```
pip install jupyter
```

### 2、启动

在命令提示符或终端中使用如下指令：

```
jupyter notebook
```

示例如下：

## 2.3.3 Spyder

Spyder (Scientific Python Development Environment) 是一个强大的交互式 Python 语言 IDE 开发环境，支持包括 Windows、Linux 和 macOS 系统。Spyder 还集成了很多流行科学软件包，包括 NumPy, SciPy, Pandas, IPython, QtConsole, Matplotlib, SymPy 等。

Spyder 官网：[www.spyder-ide.org](http://www.spyder-ide.org)

### 1、安装

- Anaconda 完整安装包括 Spyder。
- 手动使用 pip 安装指令：

```
pip install spyder
```

~~手动使用安装 Spyder 不推荐！！！！~~

## 2、基本用法

### 2.4 课后练习

- 1、如何使用 pip 或 conda 指令查询 matplotlib 和 seaborn 版本。
- 2、写一个程序来获取 matplotlib 和 seaborn 版本。

参考答案：

```
import matplotlib
print(matplotlib.__version__)

import seaborn as sns
print(sns.__version__)
```

## 第3章 Matplotlib 库

### 3.1 Matplotlib 库介绍

- 1、Matplotlib 是一个支持 Python 的 2D 绘图库，可以绘制各种形式的图表
- 2、Matplotlib 可以绘制的图表有：线图、散点图、条形图、柱状图、图片以及图形动画等
- 3、良好的操作系统兼容性
- 4、绘制印刷级高质量图表
- 5、Matplotlib 可以在 Python 脚本、Python shell、iPython shell、Jupyter notebook、Web 服务器应用中使用。

## 3.2 图表基本构成要素

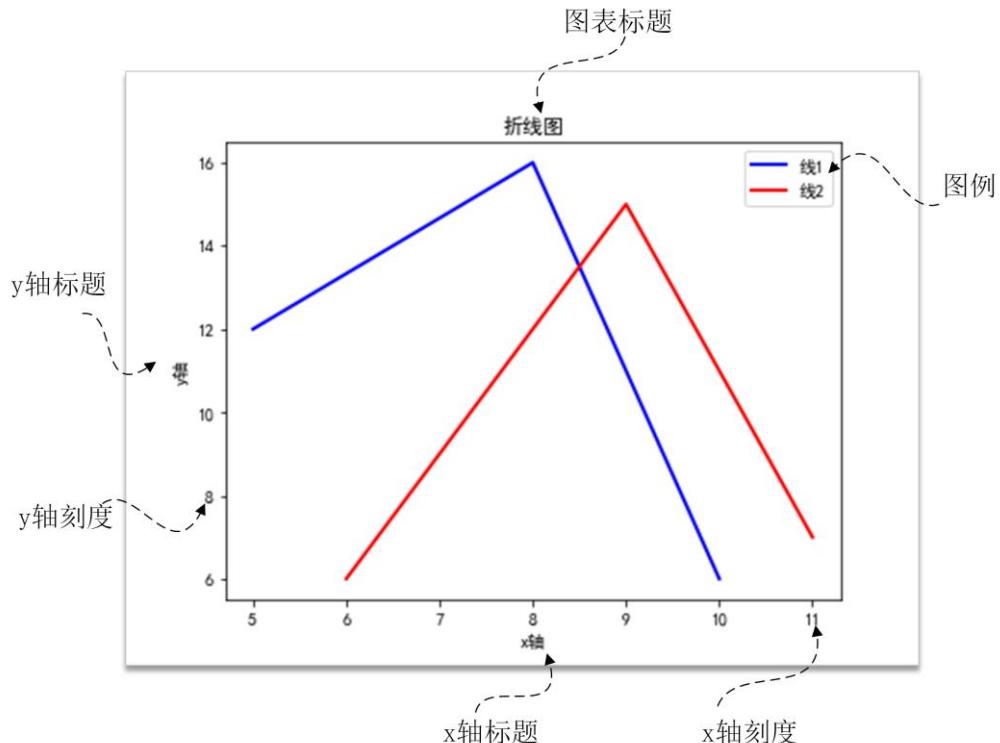


图 3-1 图表基本构成要素

## 3.3 第一个 Matplotlib 绘图程序

### 3.3.1 绘制线图

#### 1、基本绘图过程

Jupyter notebook 代码:

```
import matplotlib.pyplot as plt  
%matplotlib inline  
  
x = [5, 4, 2, 1] # x轴坐标数据  
y = [7, 8, 9, 10] # y轴坐标数据
```

```
# 绘制线段  
plt.plot(x, y)
```

### 2、添加图表标题

```
import matplotlib.pyplot as plt  
%matplotlib inline  
  
# 设置中文字体  
plt.rcParams['font.family'] = ['SimHei']  
  
x = [5, 4, 2, 1] # x轴坐标数据  
y = [7, 8, 9, 10] # y轴坐标数据  
  
# 绘制线段  
plt.plot(x, y)  
plt.title('绘制线图') # 添加图表标题
```

### 3、添加 x 和 y 轴标题

```
import matplotlib.pyplot as plt  
%matplotlib inline  
  
# 设置中文字体  
plt.rcParams['font.family'] = ['SimHei']  
  
x = [5, 4, 2, 1] # x轴坐标数据  
y = [7, 8, 9, 10] # y轴坐标数据  
  
# 绘制线段  
plt.plot(x, y)  
plt.ylabel('y轴') # 添加y轴标题  
plt.xlabel('x轴') # 添加x轴标题
```

#### 4、设置图例

```
import matplotlib.pyplot as plt
%matplotlib inline
# 设置中文字体
plt.rcParams['font.family'] = ['SimHei']

x = [5, 4, 2, 1] # x轴坐标数据
y = [7, 8, 9, 10] # y轴坐标数据

# 绘制线段
plt.plot(x, y, label='线1')
plt.ylabel('y轴') # 添加y轴标题
plt.xlabel('x轴') # 添加x轴标题
plt.legend() # 设置图例
```

#### 5、设置图形大小

```
import matplotlib.pyplot as plt
%matplotlib inline
# 设置中文字体
plt.rcParams['font.family'] = ['SimHei']

x = [5, 4, 2, 1] # x轴坐标数据
y = [7, 8, 9, 10] # y轴坐标数据

# 设置图表大小
plt.figure(figsize=(10, 5))

# 绘制线段
plt.plot(x, y, label='线1')
```

```
plt.ylabel('y轴') # 添加y轴标题  
plt.xlabel('x轴') # 添加x轴标题  
plt.legend() # 设置图例
```

### 3.3.2 显示图形

#### 1、Jupyter Notebook 中显示图形

在 Notebook 嵌入如下代码：

- %matplotlib notebook 会在 Notebook 中启动交互式图形。
- %matplotlib inline 会在 Notebook 中启动静态图形。

#### 2、IPython shell 中显示图形

在启动 IPython shell 时使用如下指令：

- ipython --pylab
- ipython --matplotlib

#### 3、脚本中显示图形

使用 show() 函数

### 3.3.3 显示中文和负号

#### 1、设置显示中文字体

方法一、

```
plt.rcParams['font.family'] = ['SimHei'] # 设置中文字体
plt.rcParams['font.size'] = 20 # 设置字体大小

x = [5, 4, 2, 1] # x轴坐标数据
y = [7, 8, 9, 10] # y轴坐标数据

# 绘制线段
plt.plot(x, y, label='线1')
plt.ylabel('y轴') # 添加y轴标题
plt.xlabel('x轴') # 添加x轴标题
plt.legend() # 设置图例
```

## 方法二、

```
import matplotlib.font_manager as fm

fontPath = r'C:\WINDOWS\Fonts\SIMLI.TTF'
font30 = fm.FontProperties(fname=fontPath, size=30)

x = [5, 4, 2, 1] # x轴坐标数据
y = [7, 8, 9, 10] # y轴坐标数据

# 绘制线段
plt.plot(x, y, 'b', label='线 1', linewidth=2)

plt.title('绘制线图', fontproperties=font30) # 添加图表标题

plt.ylabel('y 轴', fontproperties=font30) # 添加 y 轴标题
plt.xlabel('x 轴', fontproperties=font30) # 添加 x 轴标题

plt.legend(prop=font30) # 设置图例
```

## 方法三、

```
x = [5, 4, 2, 1] # x 轴坐标数据
y = [7, 8, 9, 10] # y 轴坐标数据
```

```
# 绘制线段
plt.plot(x, y, 'b', label='线 1', linewidth=2)

plt.title('绘制折线图', fontproperties='SimHei') # 添加图表标题

plt.ylabel('y 轴', fontproperties='SimHei') # 添加 y 轴标题
plt.xlabel('x 轴', fontproperties='SimHei') # 添加 x 轴标题

plt.legend(prop={'family':'SimHei', 'size':13}) # 设置图例
```

## 2、设置显示负号

```
plt.rcParams['axes.unicode_minus'] = False

x = [5, 4, -2, -1] # x 轴坐标数据
y = [-7, 8, 9, 10] # y 轴坐标数据
# 绘制线段
plt.plot(x, y, label='线 1')
plt.ylabel('y 轴') # 添加 y 轴标题
plt.xlabel('x 轴') # 添加 x 轴标题
plt.legend() # 设置图例
```

### 3.3.4 设置线条颜色和风格

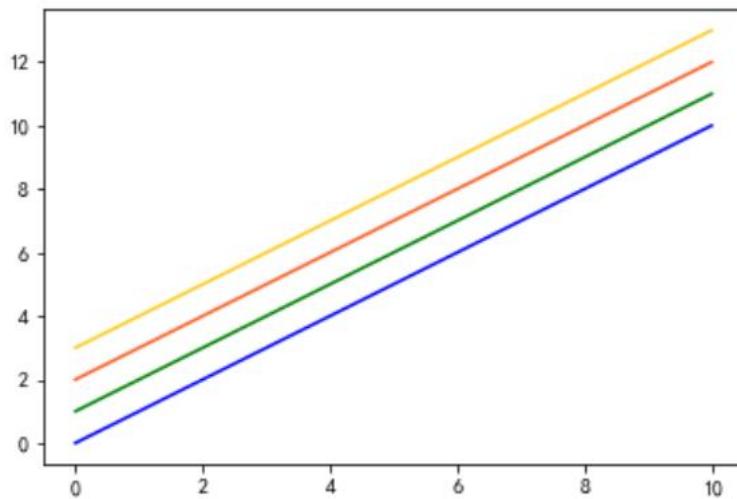
#### 1、设置颜色

设置颜色可以使用 `color` (或 `c`) 参数，它支持各种颜色值的字符串或元组。

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

```
%matplotlib inline

x = np.linspace(0, 10, 1000)
plt.plot(x, x + 0, color='blue')
plt.plot(x, x + 1, color='g')
plt.plot(x, x + 2, c='#FF5D24') # 十六进制 (RRGGBB, 00~FF)
plt.plot(x, x + 3, color=(1.0,0.8,0.1)) # RGB元组, 范围在0~1
```



## CSS Colors

black	bisque	forestgreen	slategrey
dimgray	darkorange	limegreen	lightsteelblue
dimgrey	burlywood	darkgreen	cornflowerblue
gray	antiquewhite	green	royalblue
grey	tan	lime	ghostwhite
darkgray	navajowhite	seagreen	lavender
darkgrey	blanchedalmond	mediumseagreen	midnightblue
silver	papayawhip	springgreen	navy
lightgray	moccasin	mintcream	darkblue
lightgrey	orange	mediumspringgreen	mediumblue
gainsboro	wheat	mediumaquamarine	blue
whitesmoke	oldlace	aquamarine	slateblue
white	floralwhite	turquoise	darkslateblue
snow	darkgoldenrod	lightseagreen	mediumslateblue
rosybrown	goldenrod	mediumturquoise	mediumpurple
lightcoral	cornsilk	azure	rebeccapurple
indianred	gold	lightcyan	blueviolet
brown	lemonchiffon	paleturquoise	indigo
firebrick	khaki	darkslategray	darkorchid
maroon	palegoldenrod	darkslategrey	darkviolet
darkred	darkkhaki	teal	mediumorchid
red	ivory	darkcyan	thistle
mistyrose	beige	aqua	plum
salmon	lightyellow	cyan	violet
tomato	lightgoldenrodyellow	darkturquoise	purple
darksalmon	olive	cadetblue	darkmagenta
coral	yellow	powderblue	fuchsia
orangered	olivedrab	lightblue	magenta
lightsalmon	yellowgreen	deepskyblue	orchid
sienna	darkolivegreen	skyblue	mediumvioletred
seashell	greenyellow	lightskyblue	deppink
chocolate	chartreuse	steelblue	hotpink
saddlebrown	lawngreen	aliceblue	lavenderblush
sandybrown	honeydew	dodgerblue	palevioletred
peachpuff	darkseagreen	lightslategray	crimson
peru	palegreen	lightslategrey	pink
linen	lightgreen	slategray	lightpink

## Base Colors

	b		c		k
	g		m		w
	r		y		

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

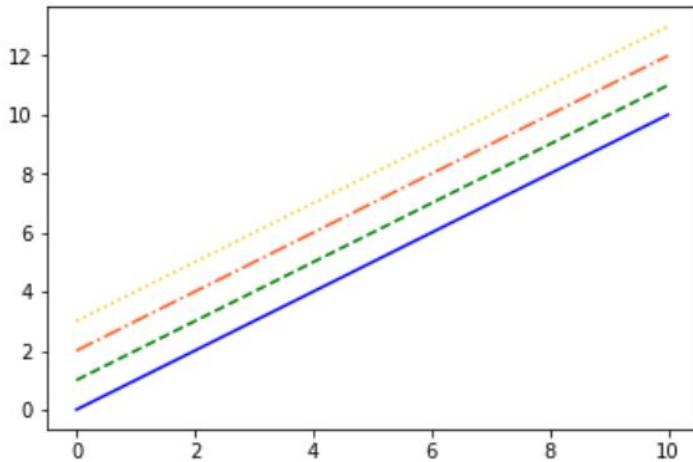
## 2、设置线条风格

设置线条风格可以使用 `linestyle` 参数。

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

%matplotlib inline

x = np.linspace(0, 10, 1000)
plt.plot(x, x + 0, color='blue', linestyle='solid') # 实线
plt.plot(x, x + 1, color='g', linestyle='dashed')    # 虚线
plt.plot(x, x + 2, c='#FF5D24', linestyle='dashdot') # 点划线
plt.plot(x, x + 3, color=(1.0,0.8,0.1), linestyle='dotted') # 实点线
```



character	description
' - '	solid line style
' -- '	dashed line style
' - . '	dash-dot line style
' : '	dotted line style

简写形式：

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

x = np.linspace(0, 10, 1000)
plt.plot(x, x + 0, color='blue', linestyle='-' ) # 实线
plt.plot(x, x + 1, color='g', linestyle='--')    # 虚线
plt.plot(x, x + 2, c='#FF5D24', linestyle='-.') # 点划线
plt.plot(x, x + 3, color=(1.0,0.8,0.1), linestyle=':' ) # 实点线
```

linestyle 和 color 参数编码组合：

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

x = np.linspace(0, 10, 1000)
plt.plot(x, x + 0, 'b-') # 蓝色实线
plt.plot(x, x + 1, '--g') # 绿色虚线
plt.plot(x, x + 2, 'r-.') # 红点划线
plt.plot(x, x + 3, 'y:') # 黄实点线
```

### 3.3.5 保存图片

`savefig()` 函数保存图片，第一个参数图片文件名，第二个参数（`dpi`）是图片的 `dpi` 值。图片默认格式 `png`。

```
x = np.linspace(0, 10, 1000)

# 设置图表大小
plt.figure(figsize=(10,5))

plt.plot(x, np.cos(x), 'b-', label='线 1') # 蓝色实线
plt.plot(x, np.cos(x + 1), '--g', label='线 2') # 绿色虚线
plt.plot(x, np.cos(x + 2), 'r-.', label='线 3') # 红点划线
plt.plot(x, np.cos(x + 3), 'y:', label='线 4') # 黄实点线

plt.title('绘制线图') # 添加图表标题

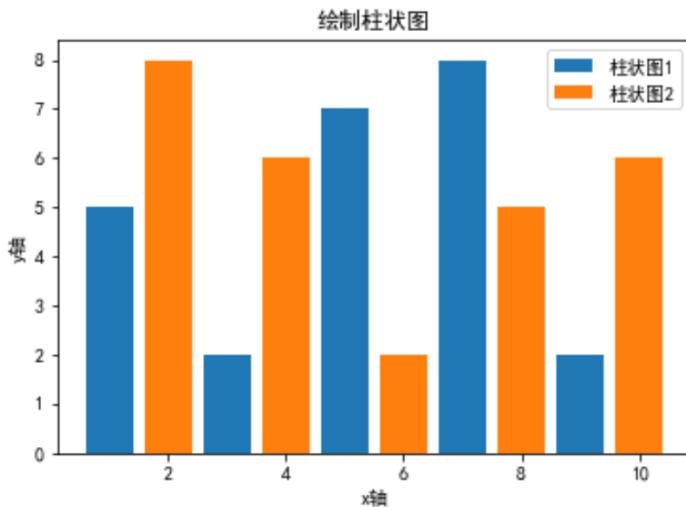
plt.ylabel('y 轴') # 添加 y 轴标题
plt.xlabel('x 轴') # 添加 x 轴标题

plt.legend() # 设置图例

# 以分辨率 72 来保存图片
plt.savefig('线图', dpi=72)
```

## 3.4 绘制柱状图

bar() 函数是绘制柱状图。



```
x1 = [1, 3, 5, 7, 9] # x1 轴坐标数据
y1 = [5, 2, 7, 8, 2] # y1 轴坐标数据

x2 = [2, 4, 6, 8, 10] # x2 轴坐标数据
y2 = [8, 6, 2, 5, 6] # y2 轴坐标数据

# 绘制柱状图
plt.bar(x1, y1, label='柱状图 1')
plt.bar(x2, y2, label='柱状图 2')

plt.title('绘制柱状图') # 添加图表标题

plt.ylabel('y 轴') # 添加 y 轴标题
plt.xlabel('x 轴') # 添加 x 轴标题

plt.legend() # 设置图例
```

### 3.5 绘制饼状图

绘制饼状图函数 `matplotlib.pyplot.pie(x, explode=None, labels=None, colors=None, autopct=None, shadow=False...)`, 主要参数如下:

- x 参数各个扇面数据集
- labels 设置各个扇面标题
- colors 设置各个扇面颜色
- shadow 设置是否有阴影
- explode 设置各个扇面脱离饼主体效果
- autopct 设置各个扇面显示百分比格式

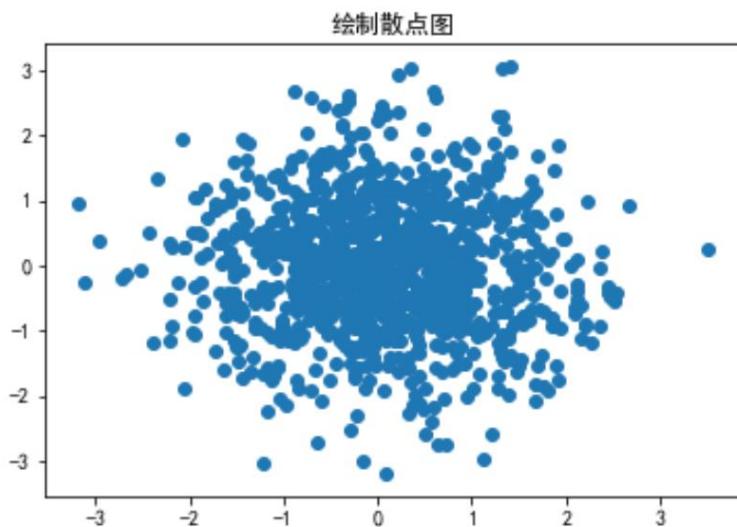


```
# 各种活动标题列表
activities = ['工作', '睡', '吃', '玩']
# 各种活动所占时间列表
slices = [8, 7, 3, 6]
# 各种活动在饼状图中的颜色列表
cols = ['c', 'm', 'r', 'b']
```

```
plt.pie(slices, labels=activities, colors=cols,  
        shadow=True, explode=(0, 0.1, 0, 0), autopct='%.1f%%')  
  
plt.title('绘制饼状图')
```

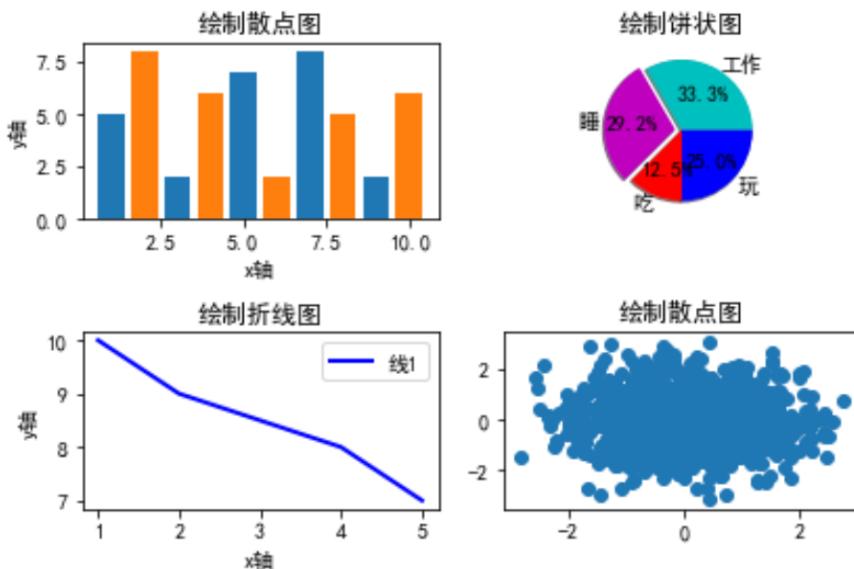
## 3.6 绘制散点图

散点图用于科学计算和数据分析等，使用 scatter() 函数：



```
n = 1024  
x = np.random.normal(0, 1, n)  
y = np.random.normal(0, 1, n)  
  
plt.scatter(x, y)  
  
plt.title('绘制散点图')
```

## 3.7 绘制子图表



subplot() 函数语法如下：

```
subplot(nrows, ncols, index, **kwargs)
```

参数 nrows 是设置总行数；参数 ncols 是设置总列数；index 是要绘制的子图的位置，index 从 1 开始到 nrows × ncols 结束。图 3-2 所示是 2 行 2 列子图表布局，subplot(2, 2, 1) 函数也可以表示为 subplot(221)。

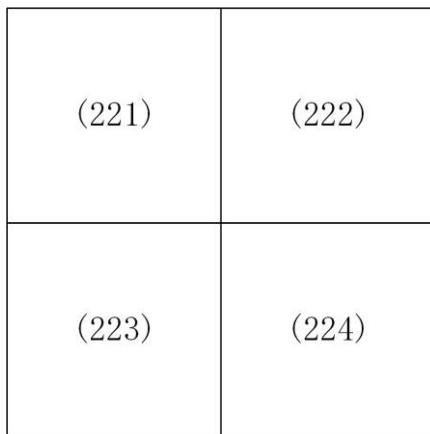


图 3-2 子图表布局

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

# 设置中文字体
plt.rcParams['font.family'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

def drawsubbar():
    """绘制柱状图"""

    x1 = [1, 3, 5, 7, 9] # x1 轴坐标数据
    y1 = [5, 2, 7, 8, 2] # y1 轴坐标数据

    x2 = [2, 4, 6, 8, 10] # x2 轴坐标数据
    y2 = [8, 6, 2, 5, 6] # y2 轴坐标数据

    # 绘制柱状图
    plt.bar(x1, y1, label='柱状图 1')
    plt.bar(x2, y2, label='柱状图 2')

    plt.title('绘制柱状图') # 添加图表标题

    plt.ylabel('y 轴') # 添加 y 轴标题
    plt.xlabel('x 轴') # 添加 x 轴标题


def drawsubpie():
    """绘制饼状图"""

    # 各种活动标题列表
    activities = ['工作', '睡', '吃', '玩']
    # 各种活动所占时间列表
    slices = [8, 7, 3, 6]
    # 各种活动在饼状图中的颜色列表
    cols = ['c', 'm', 'r', 'b']
```

```
plt.pie(slices, labels=activities, colors=cols,
        shadow=True, explode=(0, 0.1, 0, 0), autopct='%.1f%%')

plt.title('绘制饼状图')


def drawsubline():
    """绘制线图"""

    x = [5, 4, 2, 1] # x 轴坐标数据
    y = [7, 8, 9, 10] # y 轴坐标数据

    # 绘制线段
    plt.plot(x, y, 'b', label='线 1', linewidth=2)

    plt.title('绘制折线图') # 添加图表标题

    plt.ylabel('y 轴') # 添加 y 轴标题
    plt.xlabel('x 轴') # 添加 x 轴标题

    plt.legend() # 设置图例


def drawssubscatter():
    """绘制散点图"""

    n = 1024
    x = np.random.normal(0, 1, n)
    y = np.random.normal(0, 1, n)

    plt.scatter(x, y)

    plt.title('绘制散点图')


plt.subplot(2, 2, 1) # 替换(221)
drawsubbar()
```

```
plt.subplot(2, 2, 2) # 替换(222)
drawsubpie()

plt.subplot(2, 2, 3) # 替换(223)
drawsubline()

plt.subplot(2, 2, 4) # 替换(224)
drawssubscatter()

plt.tight_layout() # 调整布局
```

### 3.8 课后练习

- 1、使用 Matplotlib 库绘制两个或更多线条图，要求线条有不同的宽度、颜色和风格，并带有图例。

参考答案：

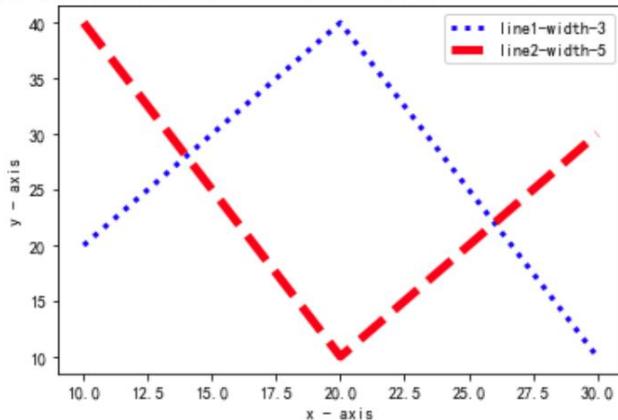
```
import matplotlib.pyplot as plt
# line 1 points
x1 = [10,20,30]
y1 = [20,40,10]
# line 2 points
x2 = [10,20,30]
y2 = [40,10,30]
# Set the x axis label of the current axis.
plt.xlabel('x - axis')
# Set the y axis label of the current axis.
plt.ylabel('y - axis')
# Set a title
plt.title('Two or more lines with different widths and colors with
suitable legends ')
# Display the figure.
plt.plot(x1,y1, color='blue', linewidth = 3, label = 'line1-width-
3', linestyle='dotted')
```

```

plt.plot(x2,y2, color='red', linewidth = 5, label = 'line2-width-5',
linestyle='dashed')
# show a legend on the plot
plt.legend()

```

Two or more lines with different widths and colors with suitable legends



## 2、使用 Matplotlib 库绘制编程语言流行度的柱状图。

示例数据：

编程语言：Java, Python, PHP, JavaScript, C#, C ++

人气：22.2, 17.6, 8.8, 8, 7.7, 6.7

参考答案：

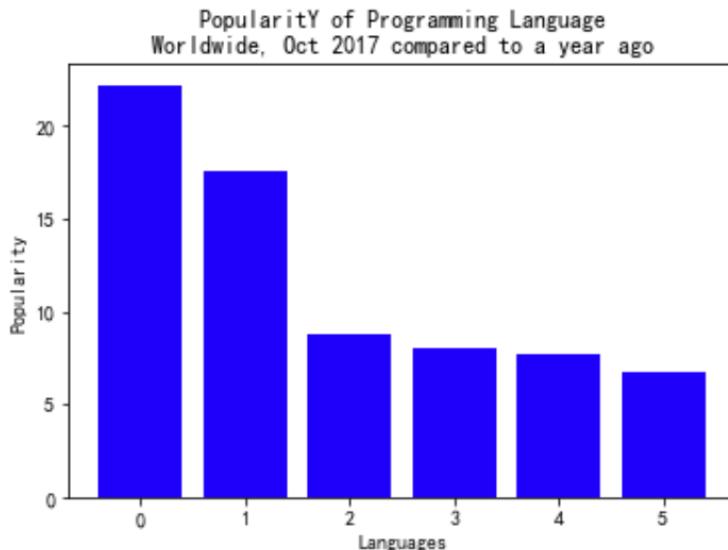
```

import matplotlib.pyplot as plt

x = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
x_pos = [i for i, _ in enumerate(x)]

plt.bar(x_pos, popularity, color='blue')
plt.xlabel("Languages")
plt.ylabel("Popularity")
plt.title("Popularity of Programming Language\n" + "Worldwide, Oct
2017 compared to a year ago")

```



3、在第 2 题的基础上，为柱状图中的条柱设置不同颜色。

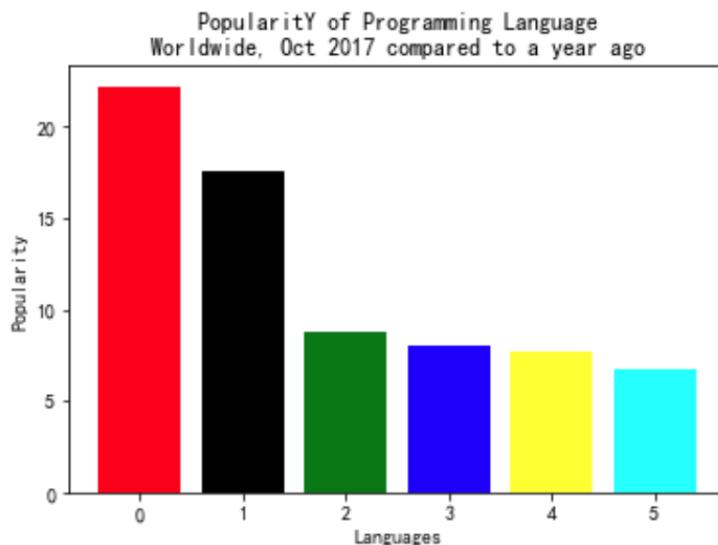
参考答案：

```
import matplotlib.pyplot as plt

x = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
x_pos = [i for i, _ in enumerate(x)]

plt.bar(x_pos, popularity, color=['red', 'black', 'green', 'blue',
'yellow', 'cyan'])

plt.xlabel("Languages")
plt.ylabel("Popularity")
plt.title("Popularity of Programming Language\n" + "Worldwide, Oct
2017 compared to a year ago")
```



4、使用 Matplotlib 库绘制编程语言流行度的饼状图。

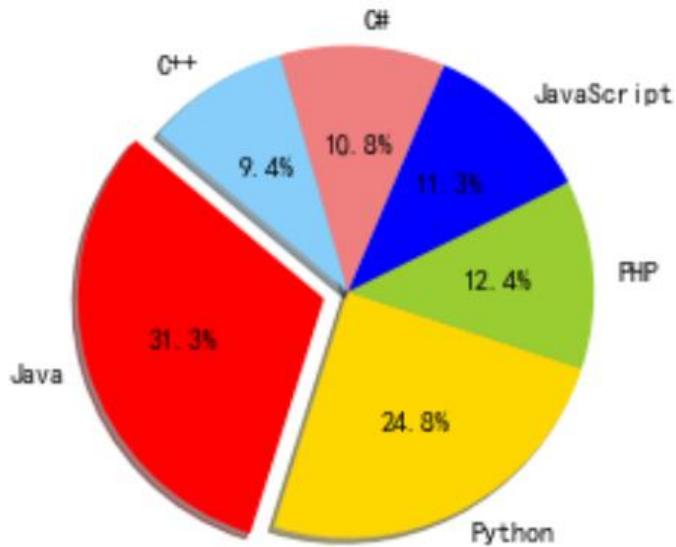
示例数据：

编程语言：Java, Python, PHP, JavaScript, C#, C ++

人气：22.2, 17.6, 8.8, 8, 7.7, 6.7

参考答案：

```
import matplotlib.pyplot as plt
# Plot data
languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
colors = ['red', 'gold', 'yellowgreen', 'blue', 'lightcoral',
'lightskyblue']
explode = (0.1, 0, 0, 0, 0, 0)
# Plot
plt.pie(popularity, explode=explode, labels=languages,
colors=colors, autopct='%.1f%%', shadow=True)
```



5、使用 Matplotlib 库绘制按组和性别划分的【分数】柱状图，要求男女在同一图表中。

样本数据：

均值（男性） = (22, 30, 35, 35, 26)  
均值（女性） = (25, 32, 30, 35, 29)

参考答案：

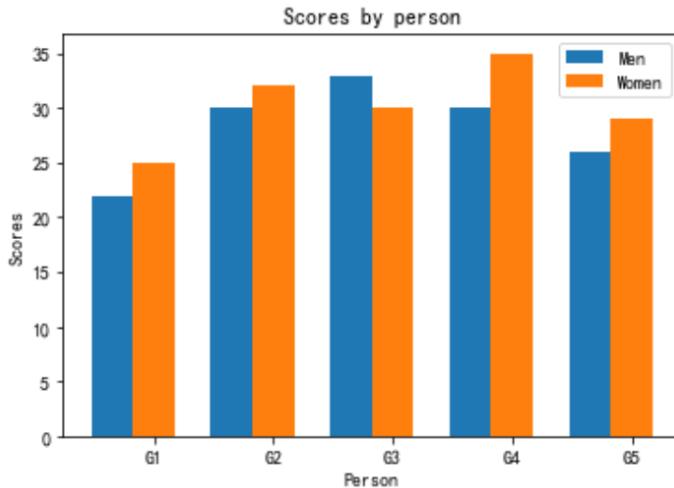
```
import numpy as np
import matplotlib.pyplot as plt

# 绘图数据
men_means = (22, 30, 33, 30, 26)
women_means = (25, 32, 30, 35, 29)
n_groups = 5
index = np.arange(n_groups)
bar_width = 0.35

rects1 = plt.bar(index, men_means, bar_width, label='Men')
rects2 = plt.bar(index + bar_width, women_means,
bar_width, label='Women')
```

```
plt.xlabel('Person')
plt.ylabel('Scores')
plt.title('Scores by person')
plt.xticks(index + bar_width, ('G1', 'G2', 'G3', 'G4', 'G5'))

plt.legend()
```

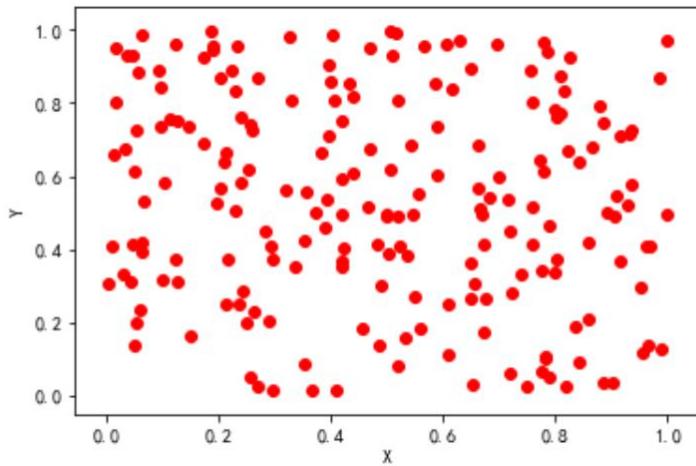


6、用随机数函数生成 200 个点，并使用 Matplotlib 库绘制散点图，要求将颜色设置为红色。

参考答案：

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

x = np.random.rand(200)
y = np.random.rand(200)
plt.scatter(x, y, color='r')
plt.xlabel("X")
plt.ylabel("Y")
```

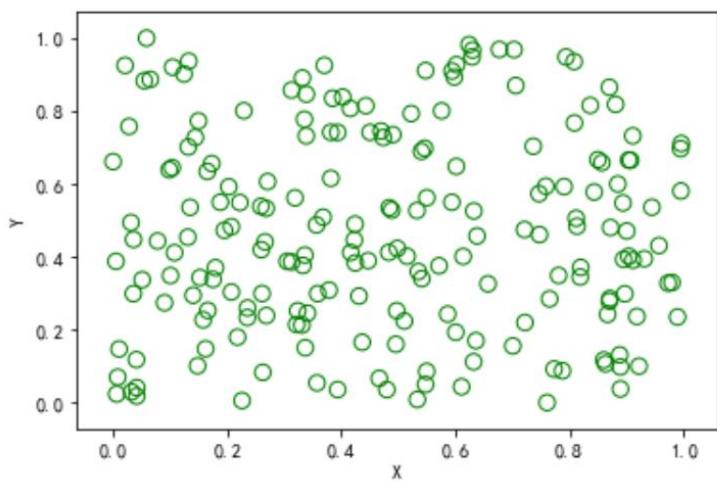


7、用随机数函数生成 200 个点，并使用 Matplotlib 库绘制散点图，要求用空心圆绘制点。

参考答案：

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

x = np.random.rand(200)
y = np.random.rand(200)
plt.scatter(x, y, s=70, facecolors='none', edgecolors='g')
plt.xlabel("X")
plt.ylabel("Y")
```



# 第4章 Seaborn 库

## 4.1 Seaborn 库介绍

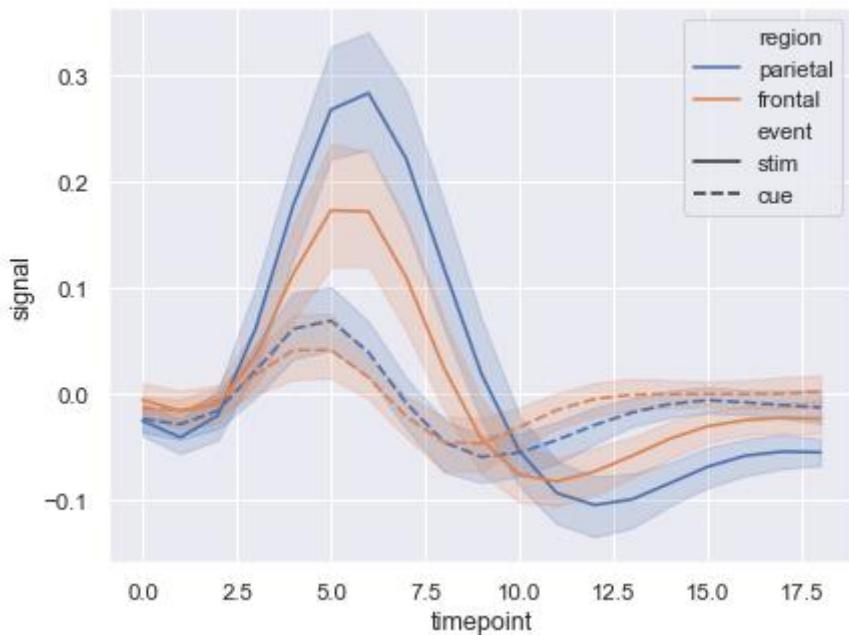
Seaborn 是一个基于 Matplotlib 的数据可视化库。它用于绘制有吸引力且信息丰富的统计图形。

### 1、Matplotlib 库存在问题？

- Matplotlib 绘制的图表不漂亮
- Matplotlib 要进行数据分析需要编写很多代码
- Matplotlib 不能与 Pandas 很好结合

### 2、Seaborn 优势

- Seaborn 绘制漂亮图表
- Seaborn 是数据分析而设计的绘图库
- Seaborn 能与 Pandas 很好结合



### 3、Seaborn 绘制图表分类

- 分类图：柱状图 barplot、箱线图 boxplot、小提琴图 violinplot、散点图 (stripplot、swarmplot)，以及分面网格 (FacetGrid) 分类图 catplot。
- 关联图：散点图 scatterplot、线图 lineplot，以及分面网格 (FacetGrid) 关联图 relplot。
- 分布图：单变量分布图 distplot、密度图 kdeplot。
- 矩阵图：热力图 heatmap、聚类图 clustermap。
- 回归图：线性回归图 regplot 和分面网格 (FacetGrid) 线性回归图 lmplot。
- 分面网格图：FacetGrid。

## 4.2 Seaborn 内置数据集

Seaborn 内置数据集可以通过 load\_dataset 函数加载数据集，返回 DataFrame 对象，语法如下：

```
seaborn.load_dataset(name, cache=True, data_home=None, **kws)
```

- name 参数是数据集名字，<https://github.com/mwaskom/seaborn-data> 定义数据集名。
- cache 参数是否提供缓存。
- data\_home 参数是指定缓存路径，默认当前用户 home 下的 seaborn-data 目录中。
- sns.get\_dataset\_names() 获得数据集名字

```
import seaborn as sns

tips = sns.load_dataset('tips')
tips.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

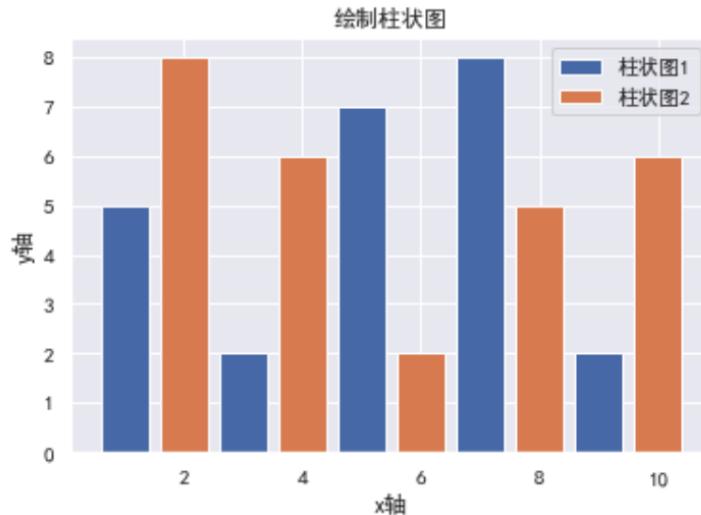
tips 是消费数据集，各个列说明如下：

- total\_bill: 一顿饭的餐费金额
- tip: 该顿饭给的小费
- sex: 服务生性别
- smoker: 服务生是否吸烟
- day: 一周中哪个一天吃的饭
- time: 吃饭时间，午餐或晚餐
- size: 吃饭人数

## 4.3 Seaborn 的样式控制

1、在 matplotlib 中设置 Seaborn 样式：

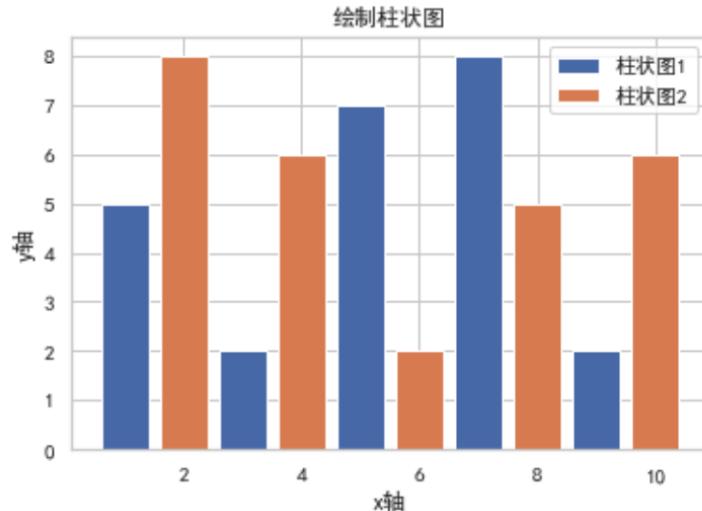
```
import seaborn as sns  
sns.set()
```



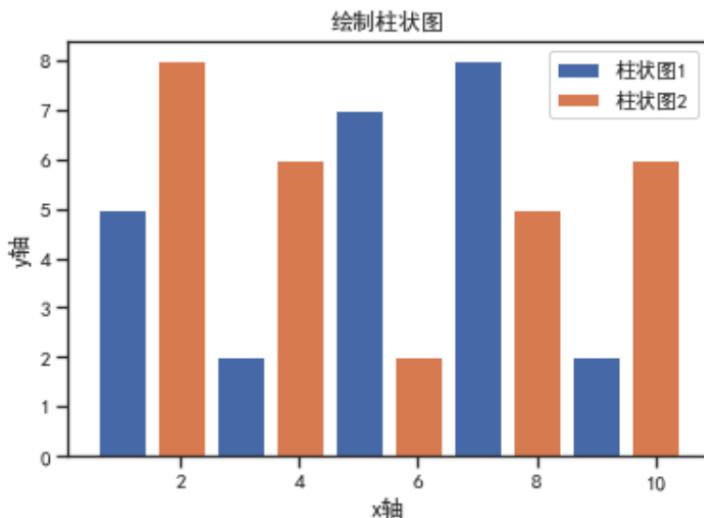
2、Seaborn 内置样式 5 种：

darkgrid、whitegrid、dark、white 和 ticks

```
import matplotlib.pyplot as plt  
%matplotlib inline  
import seaborn as sns  
  
def plot_mybar():  
    ...  
  
    sns.set_style("whitegrid")  
    plot mybar()
```

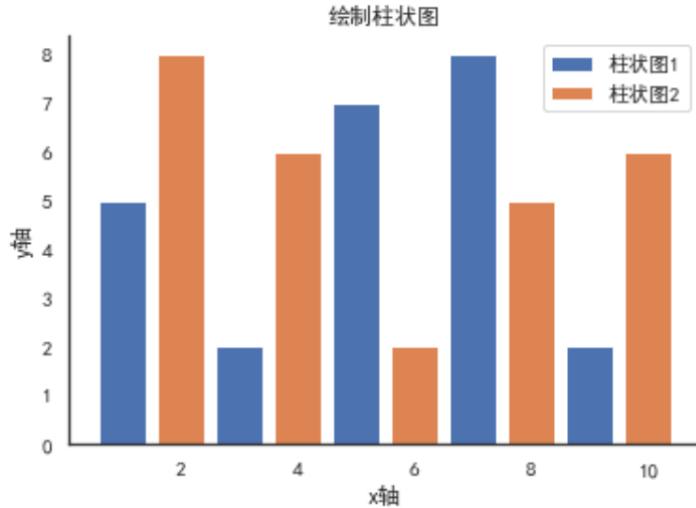


```
sns.set_style("ticks")
plot_mybar()
```

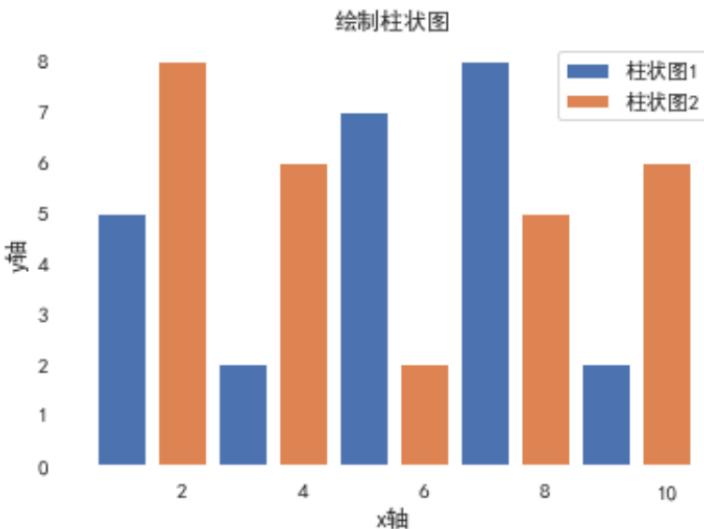


3、去除 Seaborn 图脊，使用 `despine()` 函数：

```
sns.set_style("white")
plot_mybar()
sns.despine()
```



```
sns.set_style("white")
plot_mybar()
sns.despine(left=True, bottom=True)
```

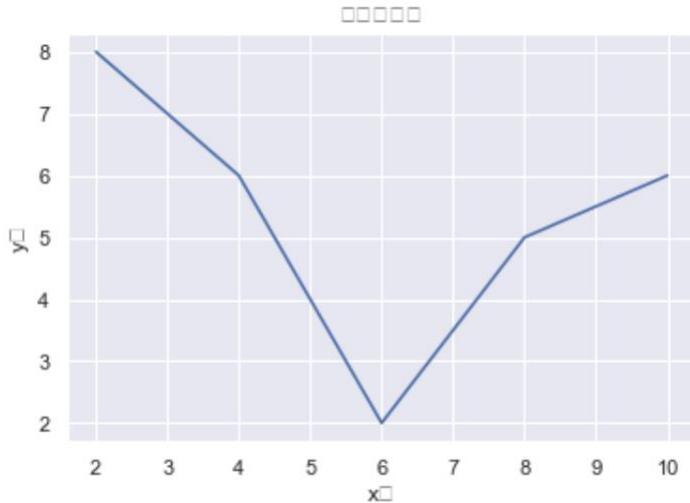


#### 4、设置 Seaborn 样式引起中文乱码问题：

```
sns.set_style('darkgrid')
x2 = [2, 4, 6, 8, 10] # x2轴坐标数据
```

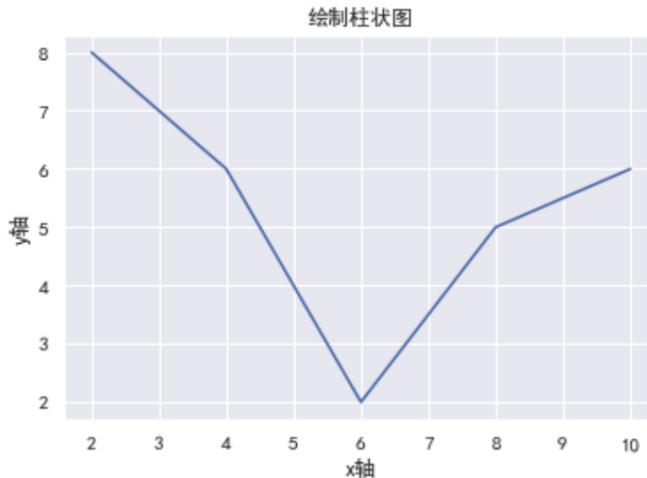
```
y2 = [8, 6, 2, 5, 6] # y2轴坐标数据

sns.lineplot(x=x2, y=y2)
plt.title('绘制柱状图') # 添加图表标题
plt.ylabel('y轴') # 添加y轴标题
plt.xlabel('x轴') # 添加x轴标题
```



```
sns.set_style('darkgrid',{'font.sans-serif':['SimHei','Arial']})
x2 = [2, 4, 6, 8, 10] # x2轴坐标数据
y2 = [8, 6, 2, 5, 6] # y2轴坐标数据

sns.lineplot(x=x2, y=y2)
plt.title('绘制柱状图') # 添加图表标题
plt.ylabel('y轴') # 添加y轴标题
plt.xlabel('x轴') # 添加x轴标题
```



## 4.4 分类图

### 4.4.1 柱状图

Seaborn 中绘制柱状图函数是 `seaborn.barplot`, 它的主要参数:

```
seaborn.barplot(x=None, y=None, hue=None, data=None)
```

- `hue` 参数, 通过颜色对数据进行分类。
- `data` 参数, 指定数据集, 可以是 DataFrame、NumPy 数组或列表等类型。
- 如果 `data` 设定, 则 `x`、`y`、`hue` 取值是 `data` 中列名。

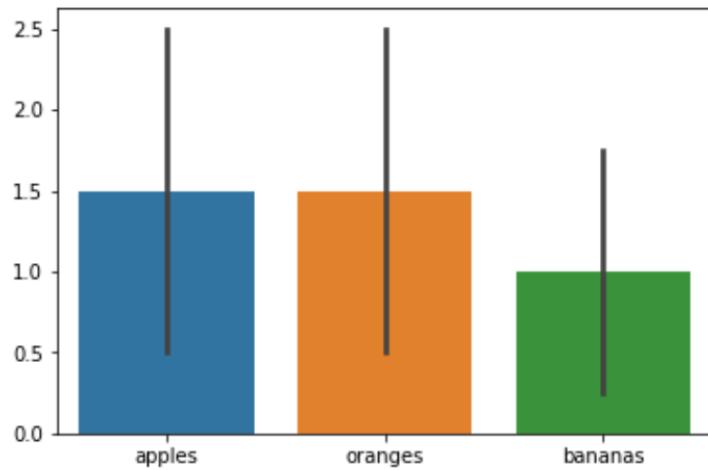
1、

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

```
data = {  
    'apples': [3, 2, 0, 1],  
    'oranges': [0, 1, 2, 3],  
    'bananas': [1, 2, 1, 0]  
}  
df = pd.DataFrame(data, index=['June', 'Robert', 'Lily', 'David'])  
sns.barplot(data=df)
```



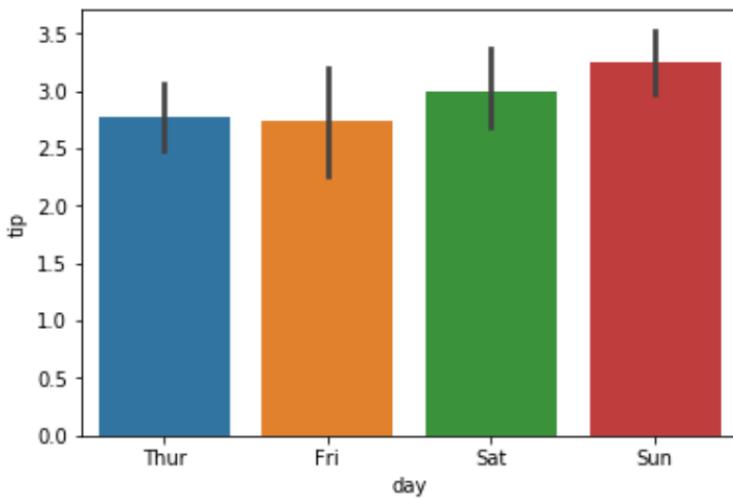
---

提示 黑色线是误差条 (error bar)。每一个“柱”最大值是一组数据的平均值。

---

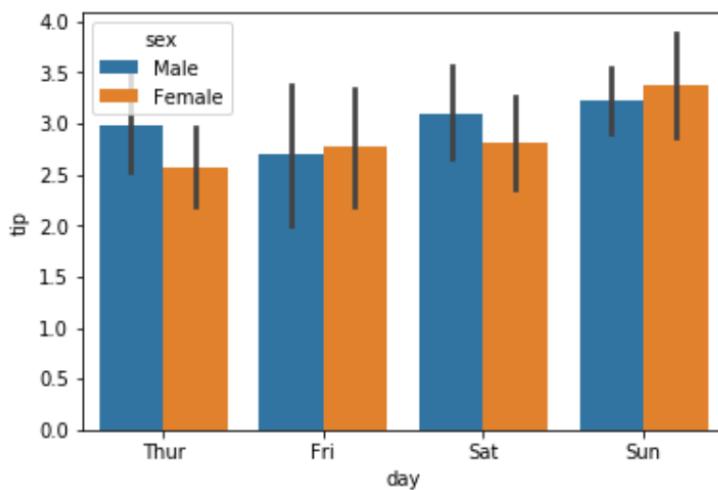
2、

```
tips = sns.load_dataset('tips')  
sns.barplot(x='day', y='tip', data=tips)
```



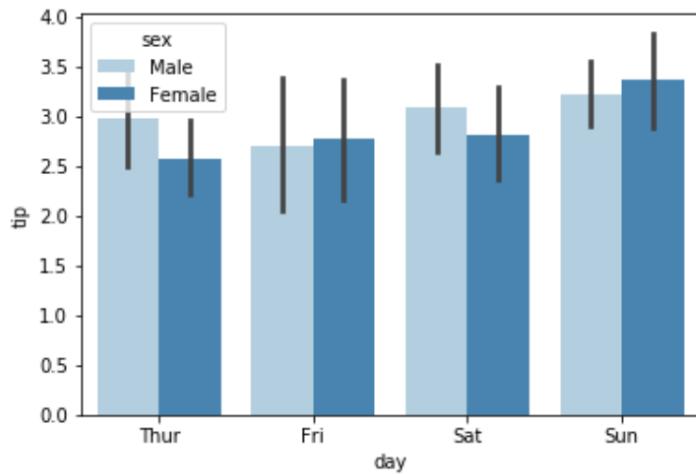
### 3、设置 hue 参数

```
sns.barplot(x='day', y='tip', data=tips, hue='sex')
```



### 4、设置颜色面板，使用 palette 参数。

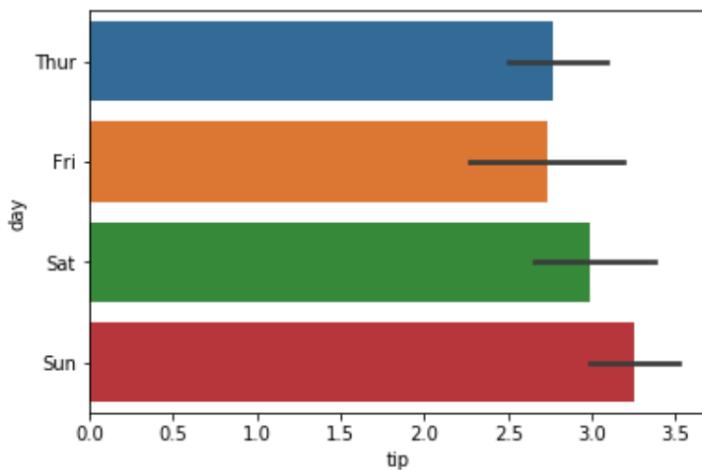
```
sns.barplot(x='day', y='tip', data=tips, hue='sex', palette='Blues')
```



[https://seaborn.pydata.org/tutorial/color\\_palettes.html](https://seaborn.pydata.org/tutorial/color_palettes.html)

## 5、设置条形图

```
sns.barplot(x='tip', y='day', data=tips)
```



### 4.4.2 箱形图

箱形图又称为盒须图、盒式图或箱线图，是一种显示数据分散情况统计图。

常见于品质管理。

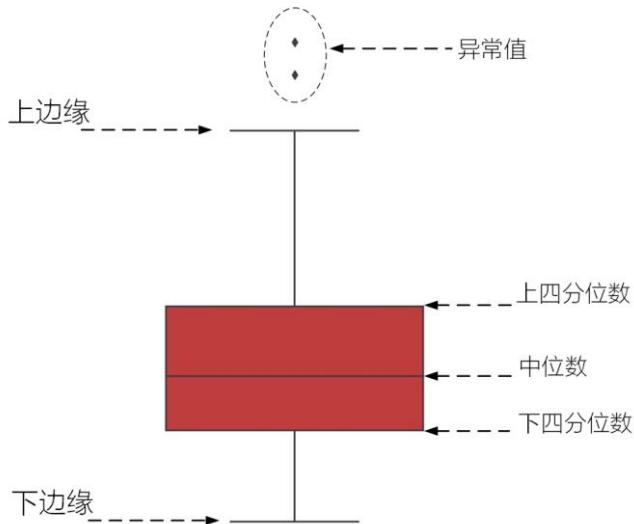


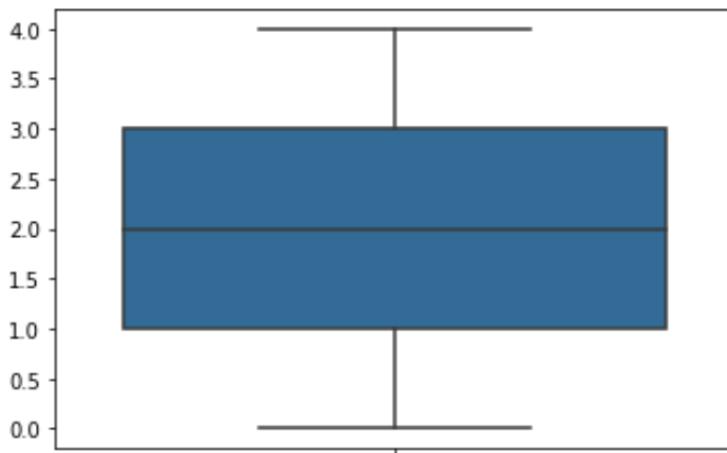
图 4-1 箱形图

- 上四分位数，又称“第一个四分位数”(Q1)，等于该样本中所有数值由小到大排列后第25%的数字。
- 中位数，又称“第二个四分位数”(Q2)，等于该样本中所有数值由小到大排列后第50%的数字。
- 下四分位数，又称“第三个四分位数”(Q3)，等于该样本中所有数值由小到大排列后第75%的数字。

Seaborn 中绘制箱形图函数是 `seaborn.boxplot`，它的主要参数参考柱状图。

## 1、最简单箱形图

```
L=[3, 2, 0, 1, 4]  
sns.boxplot(y=L)
```

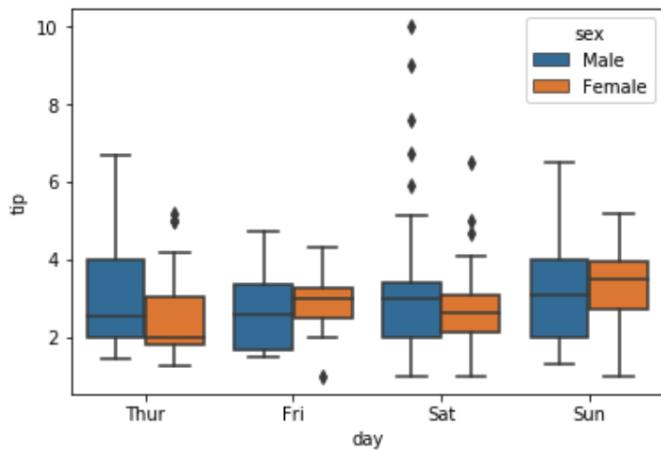


```
输入: s = pd.Series(L)
      print('中位数', s.median())
      print('上四分位数', s.quantile(0.25))
      print('中位数', s.quantile(0.5))
      print('下四分位数', s.quantile(0.75))
```

```
输出: 中位数 2.0
      上四分位数 1.0
      中位数 2.0
      下四分位数 3.0
```

## 2、更多参数箱形图

```
tips = sns.load_dataset('tips')
sns.boxplot(x='day', y='tip', hue='sex', data=tips)
```



#### 4.4.3 小提琴图

小提琴图是箱形图和密度图结合。

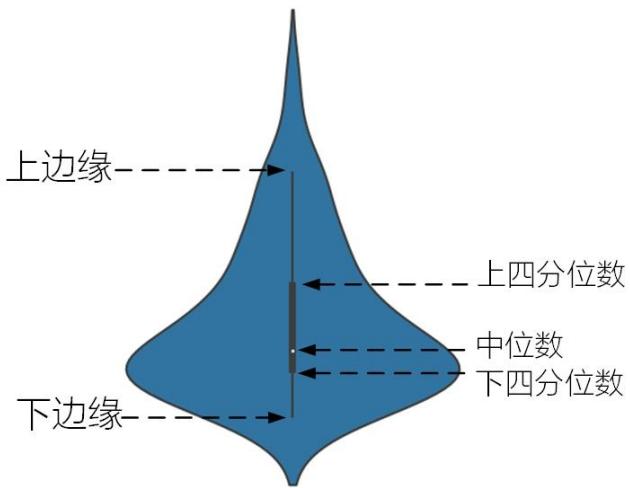
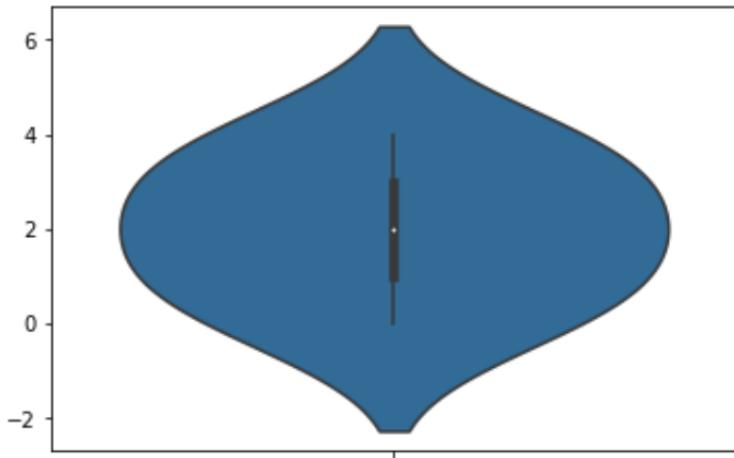


图 4-2 小提琴图

Seaborn 中绘制小提琴图函数是 `seaborn.violinplot`, 它的主要参数参考柱状图。

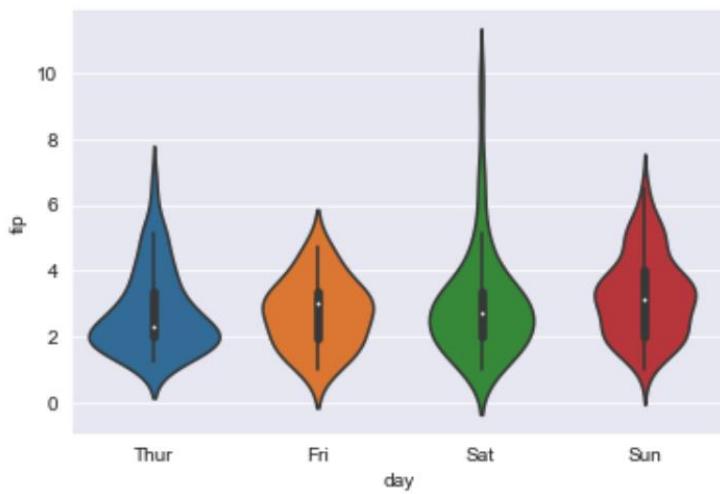
#### 1、最简小提琴图

```
L=[3, 2, 0, 1, 4]  
sns.violinplot(y=L)
```



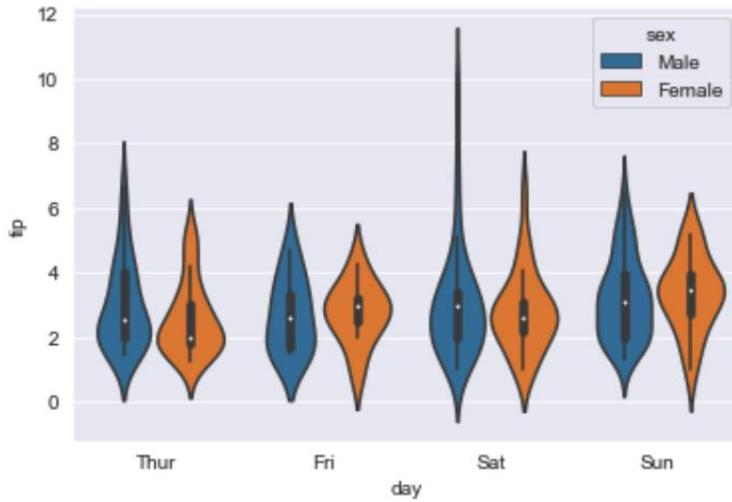
## 2、更多参数箱形图

```
tips = sns.load_dataset('tips')  
sns.set_style("darkgrid")  
sns.violinplot(x='day', y='tip', data=tips)
```



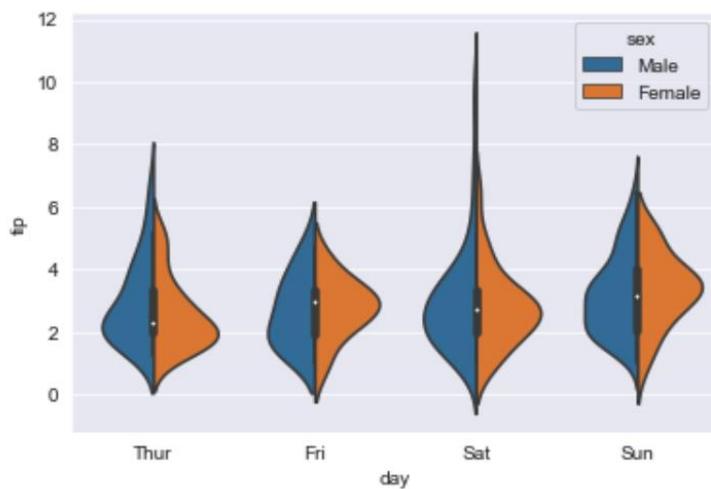
### 3、设置 hue 参数

```
sns.violinplot(x='day', y='tip', hue='sex', data=tips)
```



### 4、设置 split 参数

```
sns.violinplot(x='day', y='tip', hue='sex', data=tips, split=True)
```

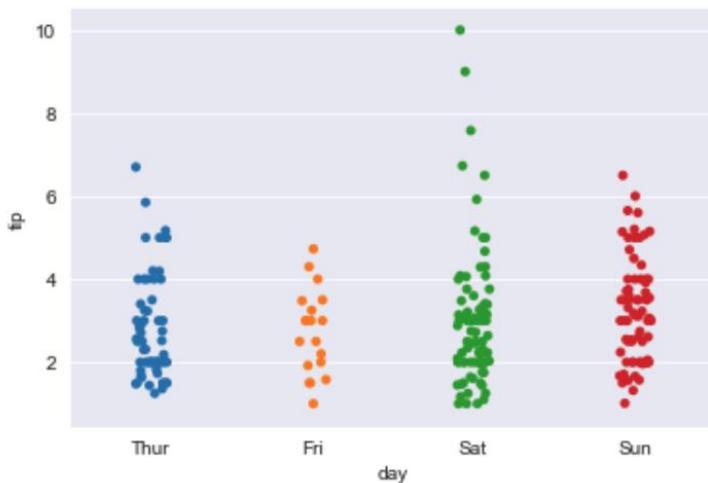


#### 4.4.4 分类散点图——Strip 图

分类图有两种散点图：`Strip`(带状)图和 `Swarm` (蜂群状) 图

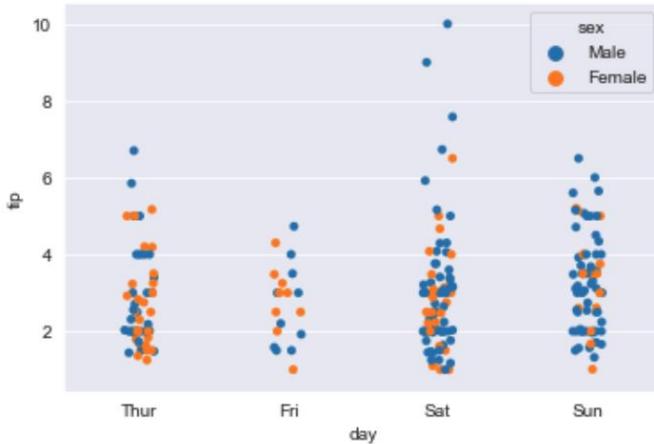
##### 1、`Strip` 图

```
tips = sns.load_dataset('tips')
sns.set_style("darkgrid")
sns.stripplot(x='day', y='tip', data=tips)
```



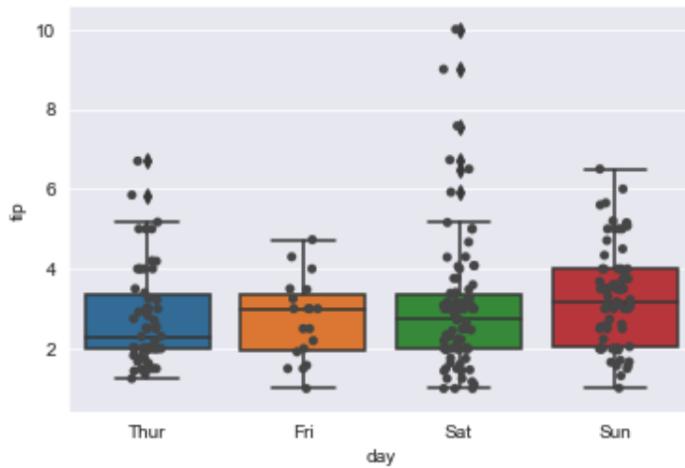
##### 2、设置 `hue` 参数

```
sns.stripplot(x='day', y='tip', data=tips, hue='sex')
```



### 3、与箱形图结合

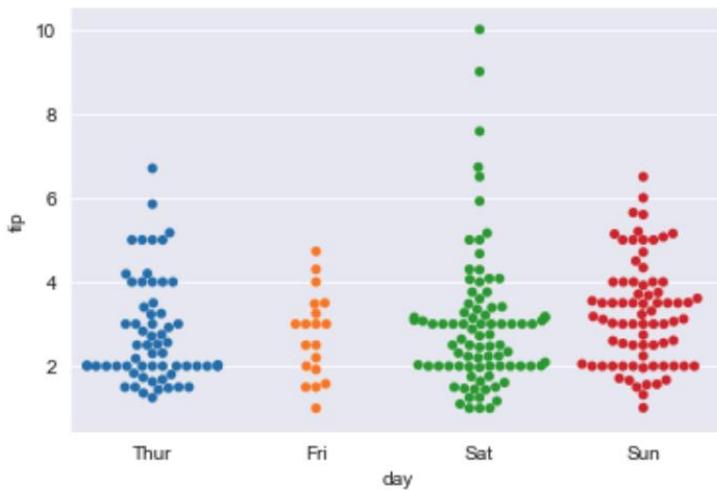
```
sns.boxplot(x='day', y='tip', data=tips)  
sns.stripplot(x='day', y='tip', data=tips, color='0.3')
```



### 4.4.5 分类散点图——Swarm 图

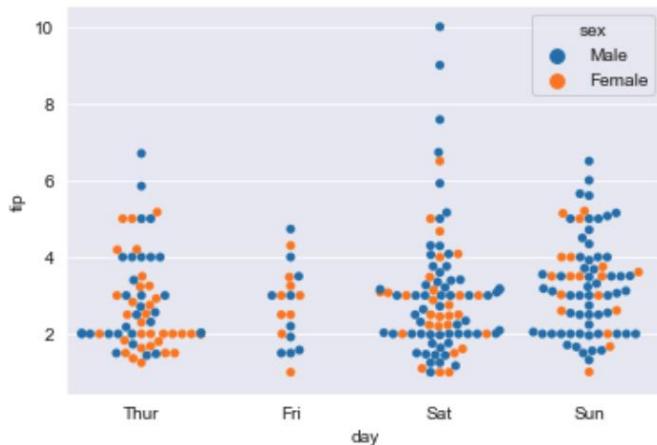
#### 1、Swarm 图

```
sns.swarmplot(x='day', y='tip', data=tips)
```



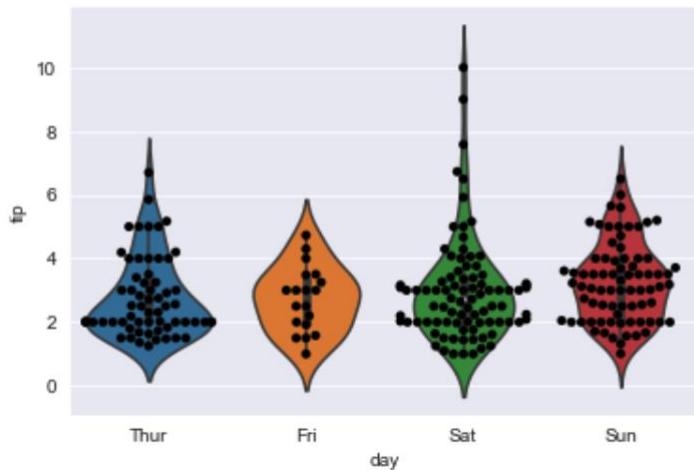
## 2、设置 hue 参数

```
sns.swarmplot(x='day', y='tip', data=tips, hue='sex')
```



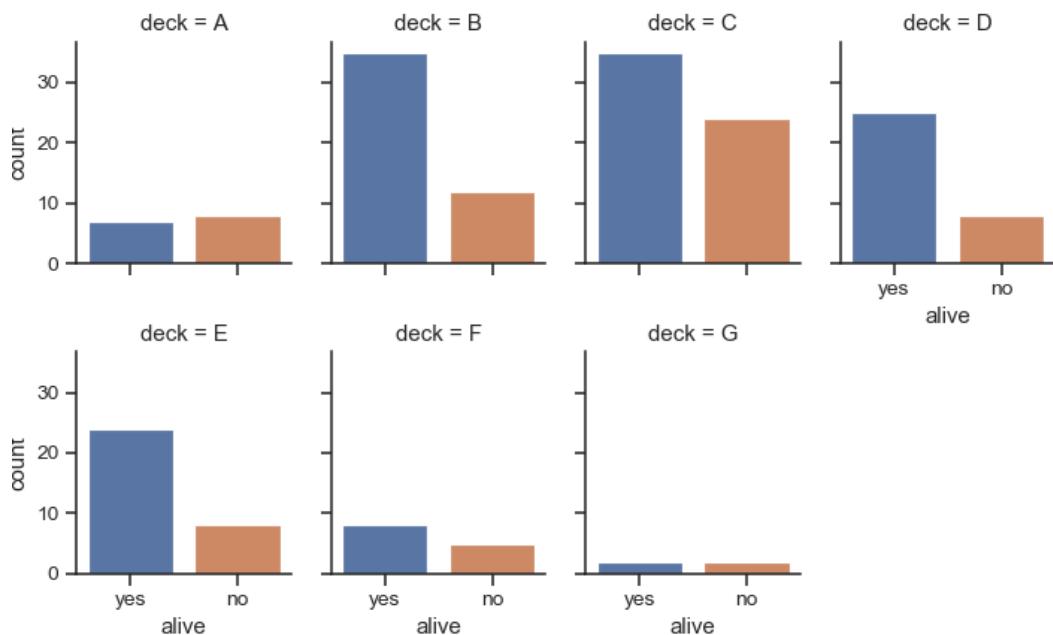
## 3、与小提琴图结合

```
sns.violinplot(x='day', y='tip', data=tips)
sns.swarmplot(x='day', y='tip', data=tips, color='black')
```



#### 4.4.6 分面网格分类图

分面网格 (FacetGrid) 可以绘制多个子图，这个网格是一个大图，有 x 和 y 两个坐标轴。



在分面网格中绘制分类图使用 catplot 函数，catplot 函数称为“图级函

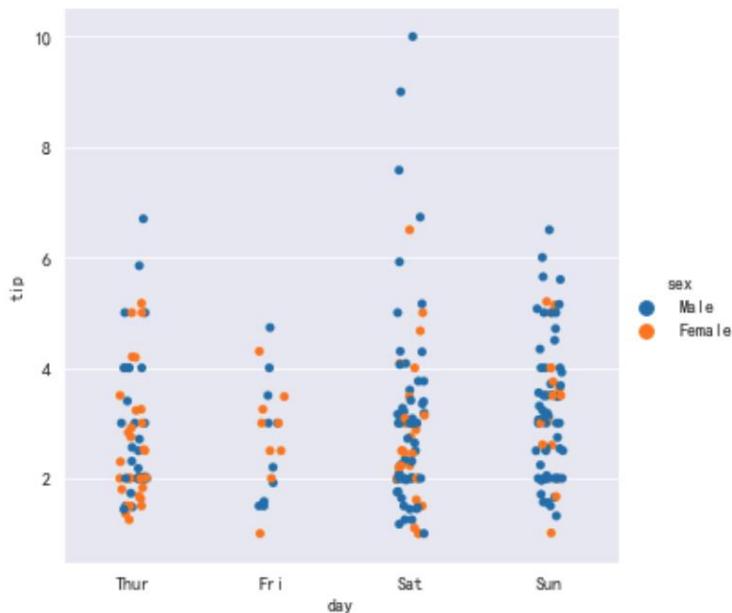
数”，而 barplot、boxplot、violinplot 等函数是分面网格 x 和 y 轴上进行绘制，这些函数称为“轴级函数”。

catplot 函数除了具有“轴级函数”的参数外，还有如下主要参数：

- row: 在 x 轴上绘制的数据。
- col: 在 y 轴上绘制的数据。
- col\_wrap: 在 x 轴上绘制子图的最大个数。
- kind: 绘制子图类型，主要有"bar"、"strip"、"swarm"、"box"、"violin" 或"boxen"，其中"strip"是默认值。

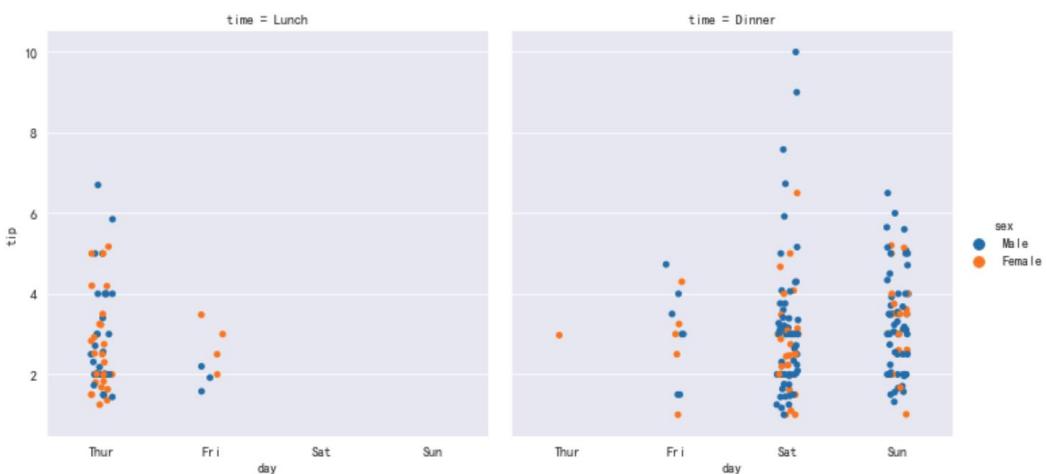
### 1、默认绘制

```
sns.catplot(x='day', y='tip', hue='sex', data=tips)
```



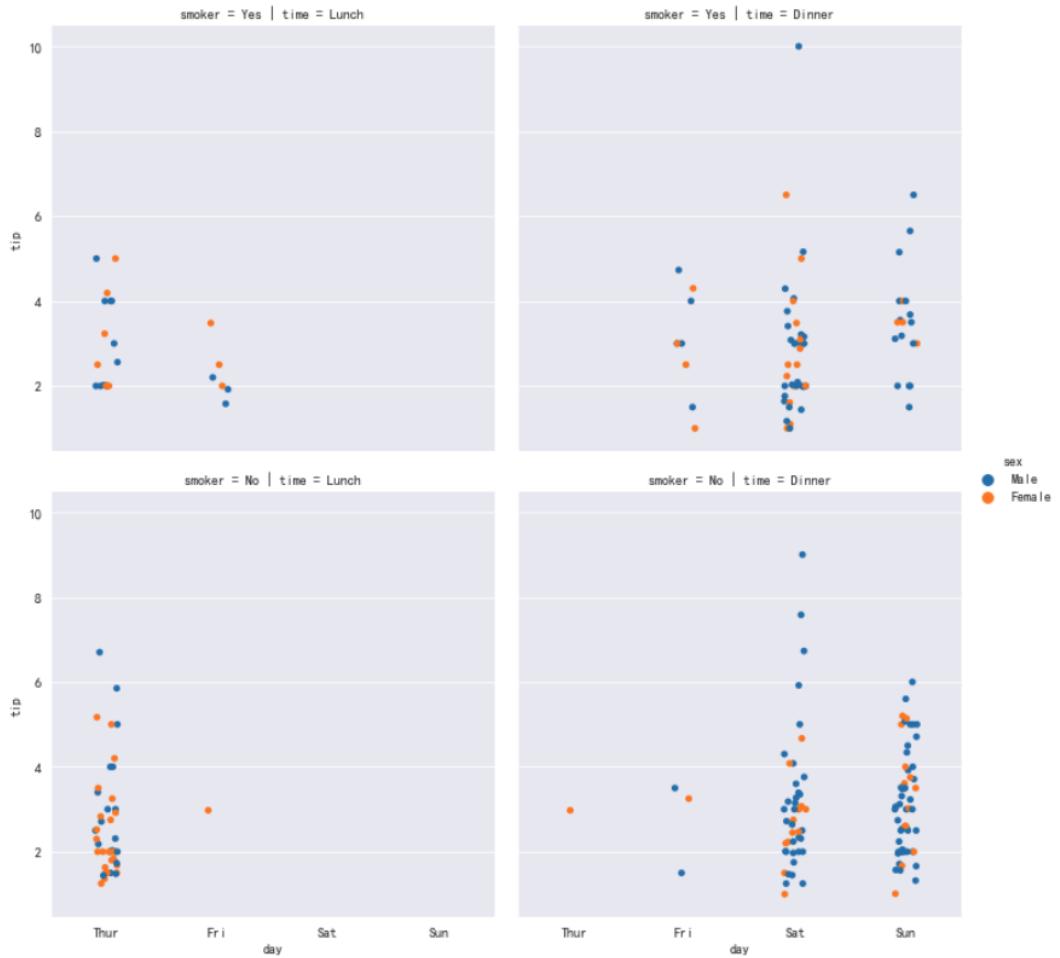
### 2、设置 col 参数

```
sns.catplot(x='day', y='tip', hue='sex', col='time', data=tips)
```



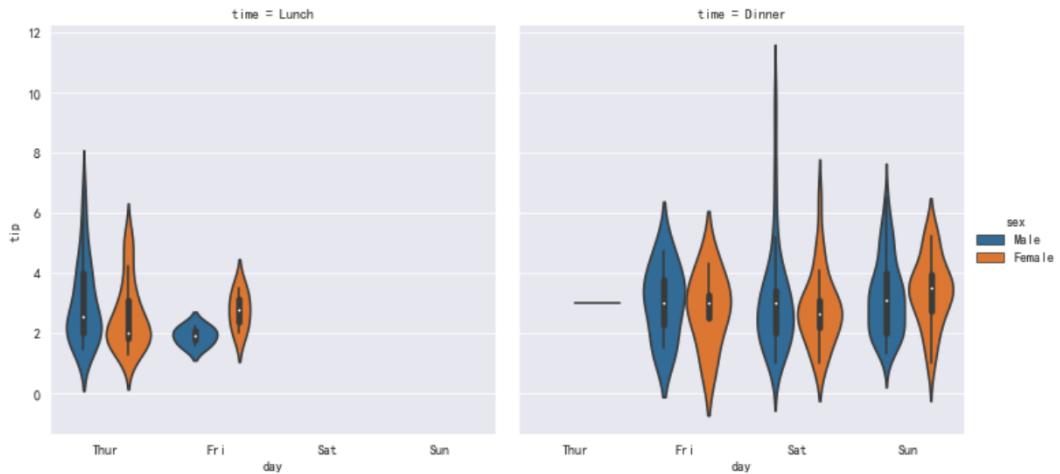
### 3、设置 col 和 row 参数

```
sns.catplot(x='day', y='tip', hue='sex', col='time', row='smoker', data=tips)
```



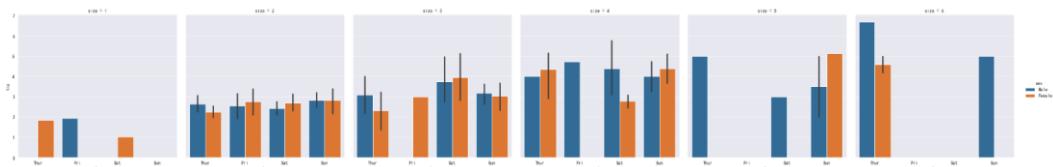
## 4、设置 kind 参数

```
sns.catplot(x='day', y='tip', hue='sex', col='time', kind="violin",  
            data=tips)
```

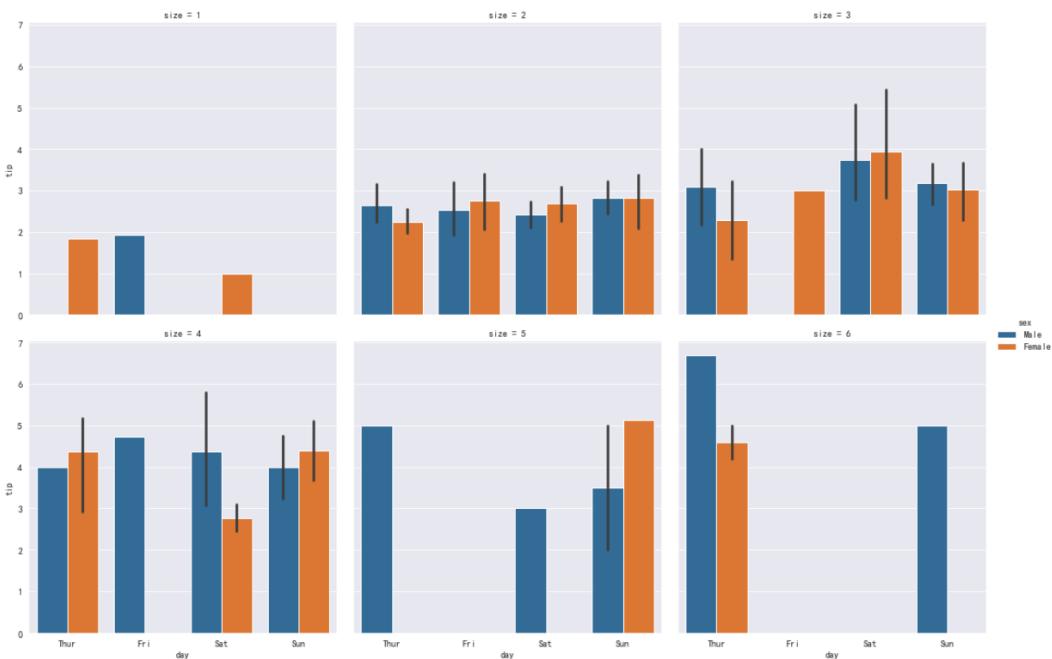


## 5、设置 col\_wrap 参数

```
sns.catplot(x='day', y='tip', hue='sex', col='size', kind="bar",  
            data=tips)
```



```
sns.catplot(x='day', y='tip', hue='sex', col='size', col_wrap=3,  
            kind="bar", data=tips)
```



## 4.5 关联图

关联图包括：关联散点图和关联线图

### 4.5.1 关联散点图

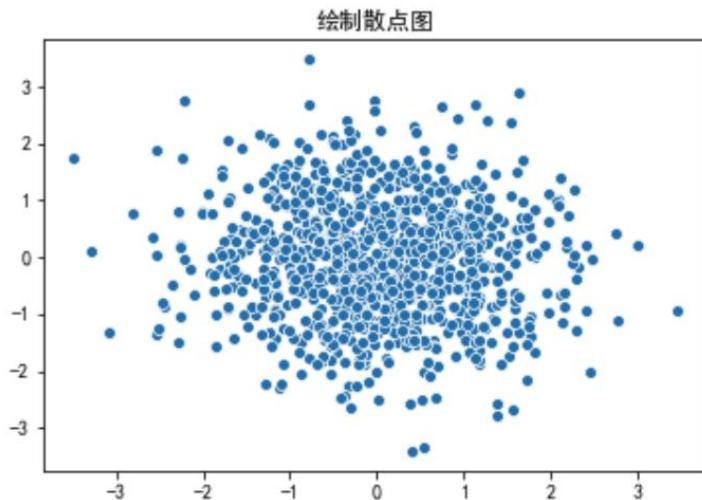
Seaborn 中关联散点图函数是 `seaborn.scatterplot`, 它的主要参数：

```
seaborn.scatterplot (x=None, y=None, hue=None, style=None,
size=None, data=None)
```

- x 和 y 是有关联的两个变量数据集。
- hue、size、style 能够显示不同的数据集。
- 如果 data 设定，则 x、y、hue、size、style 取值是 data 中列名。

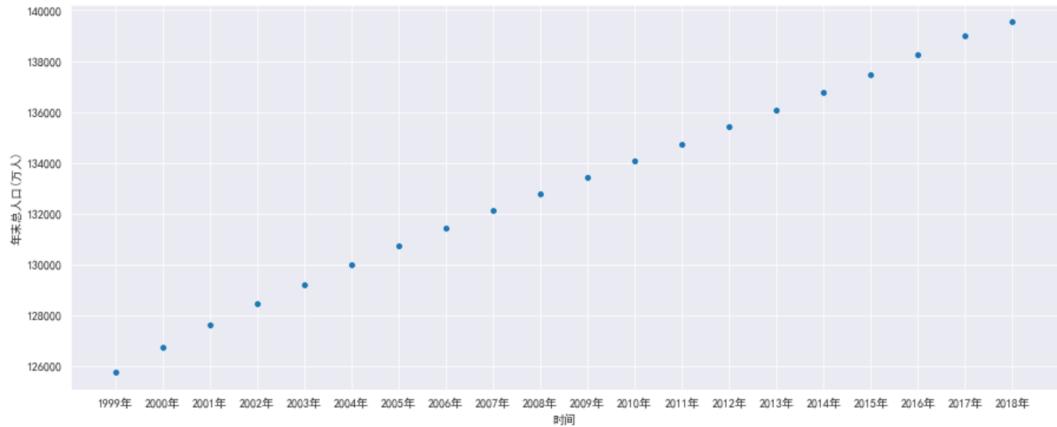
1、

```
n = 1024  
x = np.random.normal(0, 1, n)  
y = np.random.normal(0, 1, n)  
sns.scatterplot(x=x, y=y)  
plt.title('绘制散点图', fontproperties='SimHei')
```



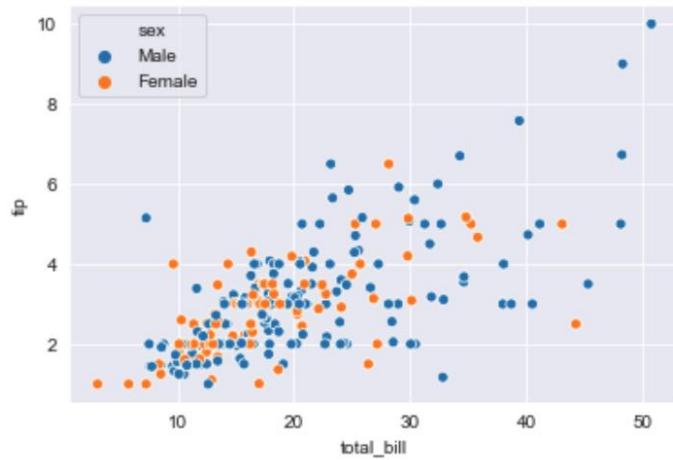
## 2、使用外部数据

```
sns.set_style("darkgrid", {'font.family':'simhei'})  
file_path='data\\'  
df20 = pd.read_excel(file_path+'全国总人口数据.xls', sheet_name='20年  
数据', skiprows=2, skipfooter=2)  
df20 = df20.sort_values('时间')  
plt.figure(figsize=(15, 6))  
sns.scatterplot(x='时间', y='年末总人口(万人)', data=df20)
```

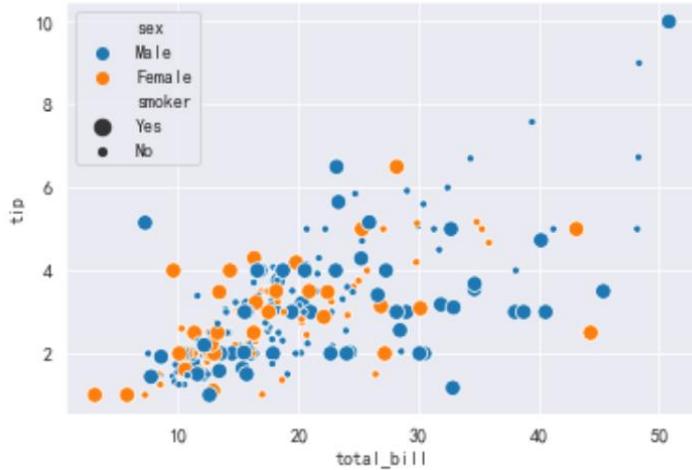


### 3、设置 hue、size、style 参数

```
tips = sns.load_dataset('tips')
sns.scatterplot(x='total_bill', y='tip', hue='sex', data=tips)
```



```
sns.scatterplot(data=tips, x='total_bill', y='tip', hue='sex',
size='smoker')
```

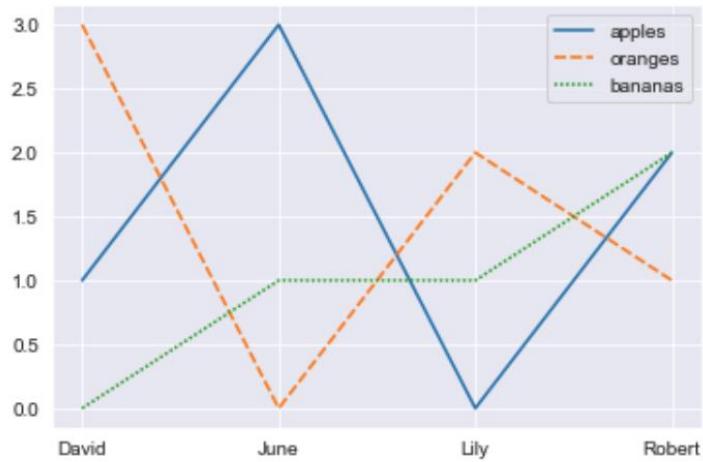


#### 4.5.2 关联线图

Seaborn 中关联线图函数是 `seaborn.lineplot`, 它的主要参数参考关联散点图。

1、

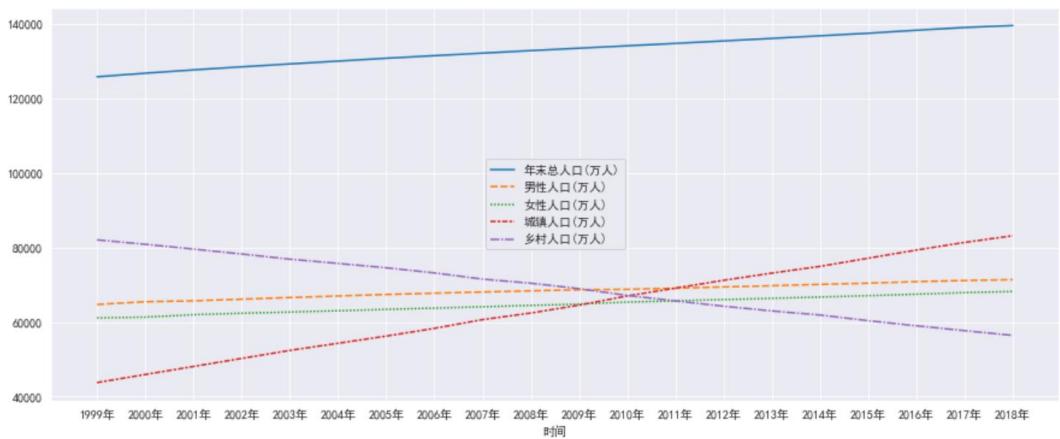
```
data = {
    'apples': [3, 2, 0, 1],
    'oranges': [0, 1, 2, 3],
    'bananas': [1, 2, 1, 0]
}
df = pd.DataFrame(data, index=['June', 'Robert', 'Lily', 'David'])
sns.lineplot(data=df)
```



## 2、使用外部数据

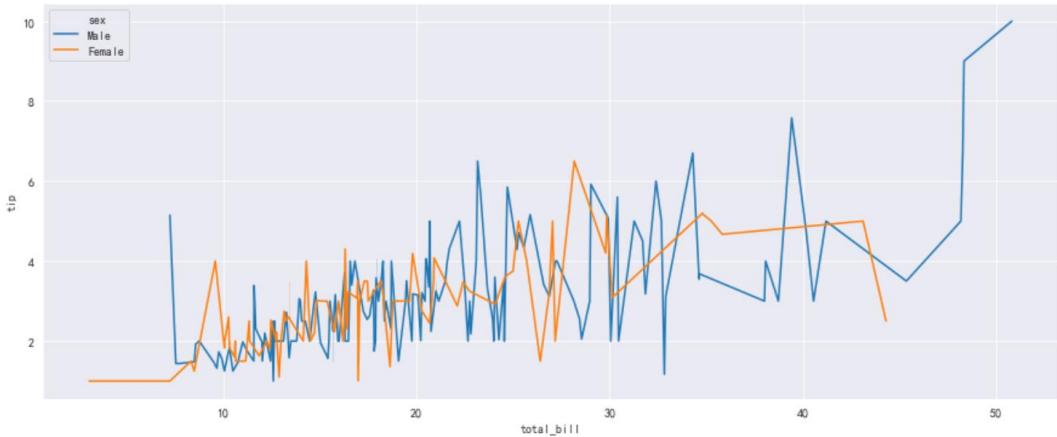
```
file_path='data\\'
df20 = pd.read_excel(file_path+'全国总人口数据.xls', sheet_name='20年
数据', skiprows=2, skipfooter=2, index_col=0)

plt.figure(figsize=(15, 6))
ax = sns.lineplot(data=df20)
```



## 3、设置 hue、size、style 参数

```
tips = sns.load_dataset('tips')
plt.figure(figsize=(15, 6))
sns.lineplot(x='total_bill', y='tip', hue='sex', data=tips)
```



```
plt.figure(figsize=(15, 6))
sns.lineplot(x='total_bill', y='tip', hue='sex',
size='smoker', style='time', data=tips)
```



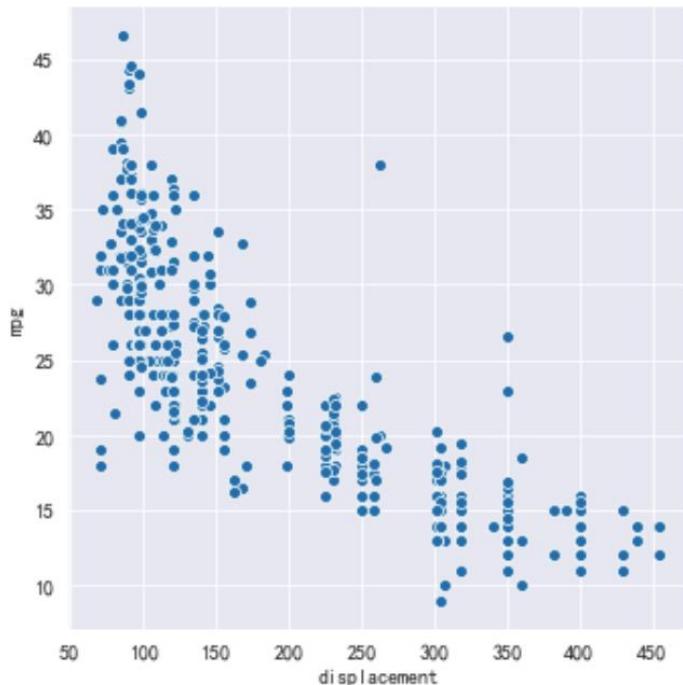
### 4.5.3 分面网格关联图

在分面网格中绘制关联图使用 relplot 函数，relplot 参数参考 4.4.6 节。

#### 1、默认绘制

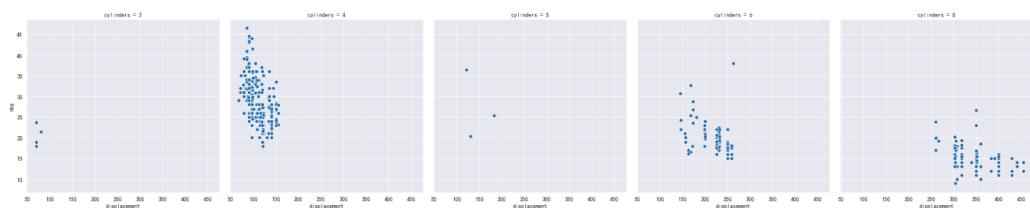
```
# 加载seaborn数据集mpg (小汽车每加仑燃料在城市公路行驶英里数)
# 字段说明
#   1. mpg:          每加行驶10英里数
#   2. cylinders:    汽缸数量
#   3. displacement: 排量
#   4. horsepower:   马力
#   5. weight:        重量
#   6. acceleration: 加速度
#   7. model year:   生产年份
#   8. origin:        原产地
#   9. car name:     车名
mpg_df = sns.load_dataset("mpg")

g = sns.relplot(x="displacement", y="mpg", data=mpg_df)
```



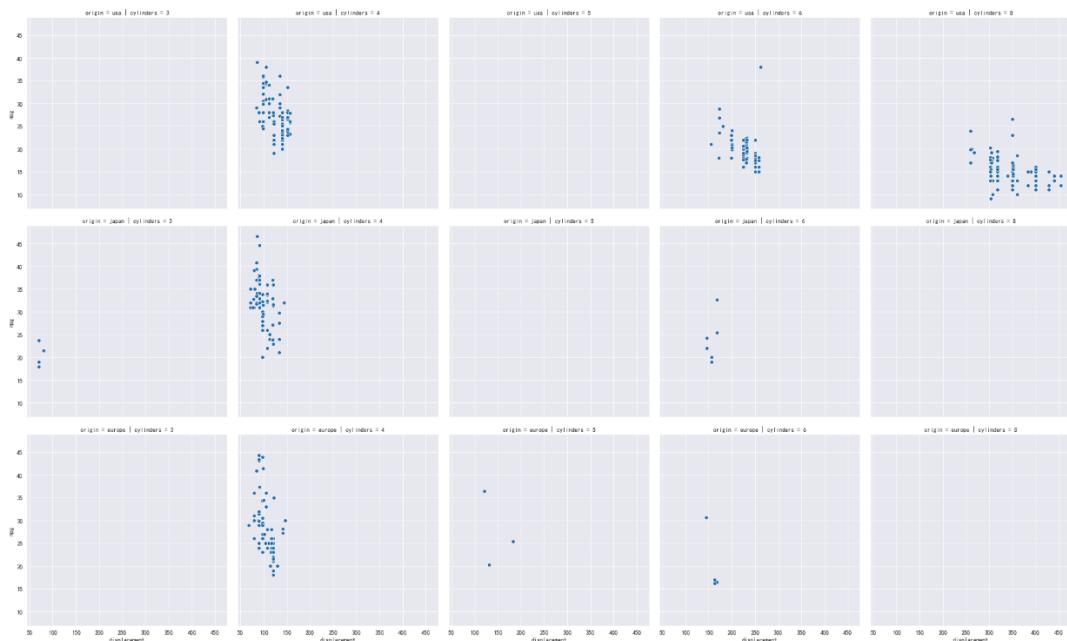
### 2、设置 col 参数

```
g = sns.relplot(x="displacement", y="mpg", col='cylinders',  
data=mpg_df)
```



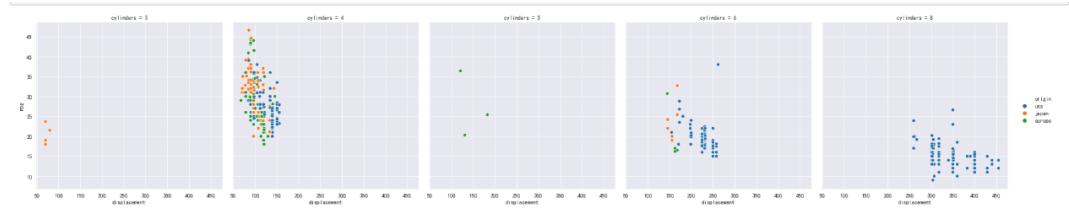
### 3、设置 col 和 row 参数

```
g = sns.relplot(x="displacement", y="mpg", col='cylinders',  
row='origin', data=mpg_df)
```



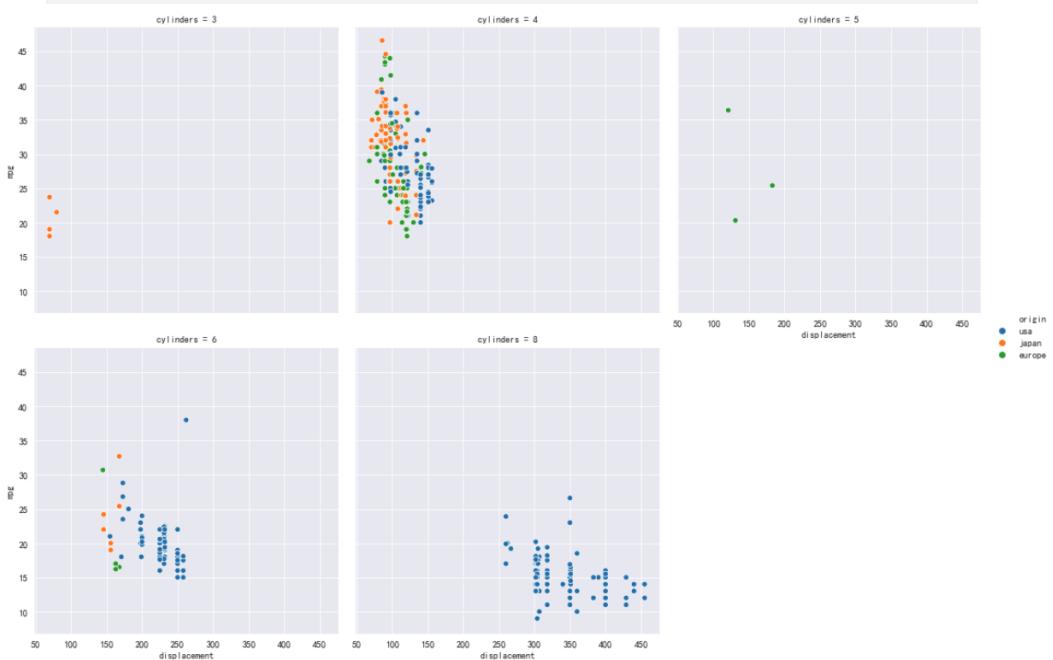
### 4、设置 hue 参数

```
g = sns.relplot(x="displacement", y="mpg", col='cylinders',  
hue='origin', data=mpg_df)
```



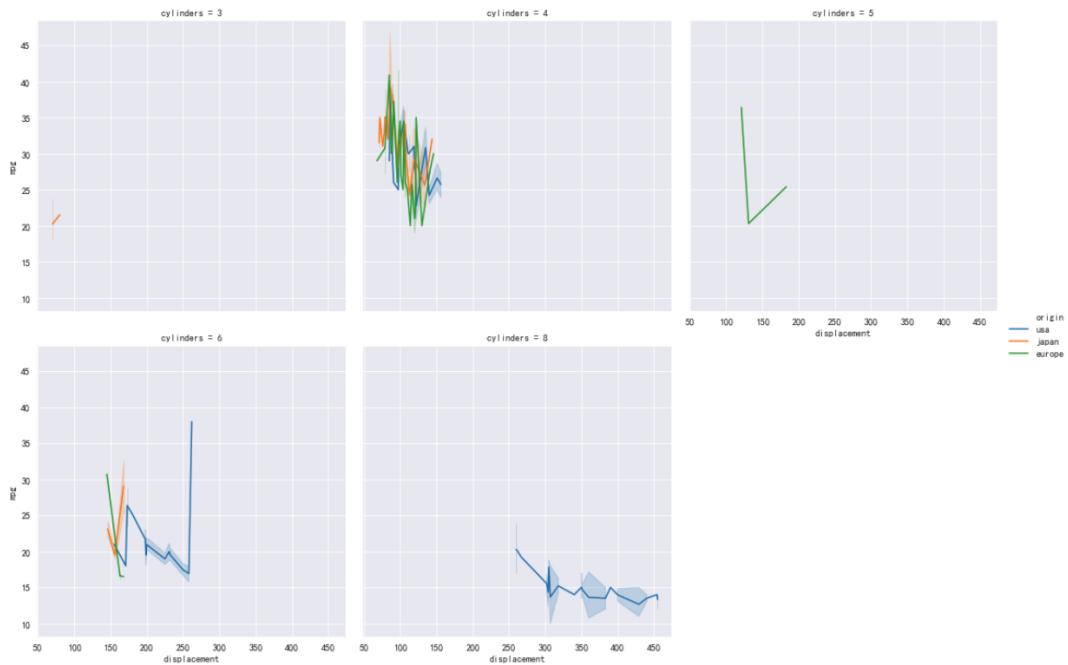
## 5、设置 col\_wrap 参数

```
g = sns.relplot(x="displacement", y="mpg", col='cylinders',
col_wrap=3, hue='origin', data=mpg_df)
```



## 6、设置 kind 参数

```
g = sns.relplot(x="displacement", y="mpg", col='cylinders',
col_wrap=3, hue='origin', kind="line", data=mpg_df)
```



## 4.6 分布图

分布图包括：直方图和密度图（KDE）。

### 4.6.1 Dist 图

Seaborn 中 distplot 函数可以绘制 Dist 图，事实上统计学并不存在 Dist 图，Seaborn 中 Dist 图是单变量的直方图和密度图结合体。

distplot 函数的主要参数：

```
seaborn.distplot (a, bins = None, hist = True, kde = True)
```

- a 参数是单变量数据，他可以是 Series、一维数组或列表。
- bins 参数是直方图中柱体的个数。
- hist 参数是否绘制直方图。

- kde 参数是否绘制密度图。

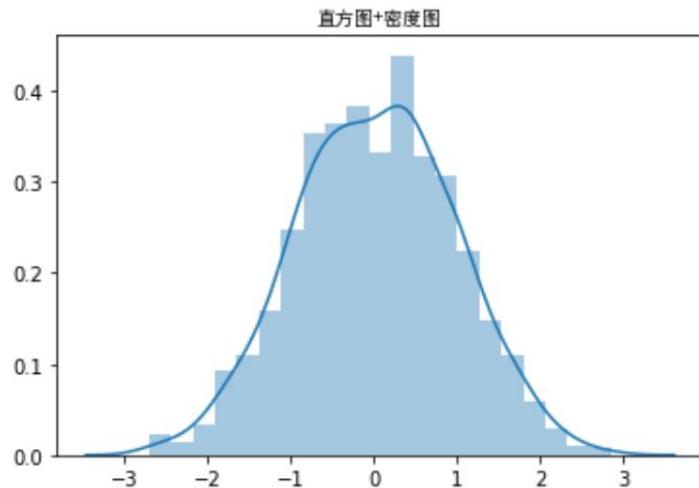
1、

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

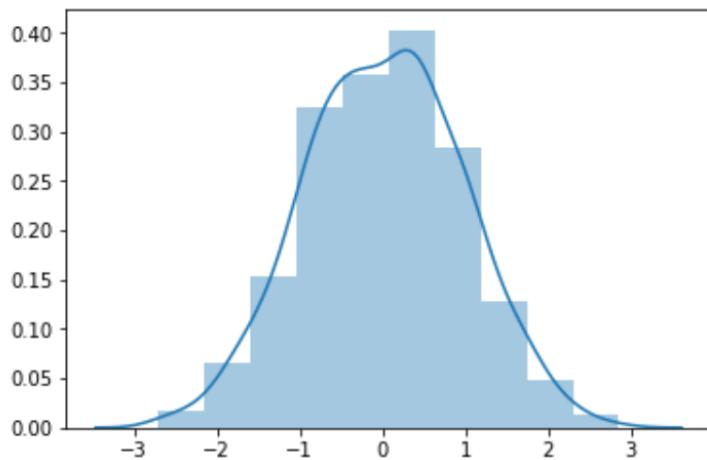
%matplotlib inline

n = 1024
x = np.random.normal(0, 1, n)
g = sns.distplot(x)
plt.title('直方图+密度图', fontproperties='SimHei')
```



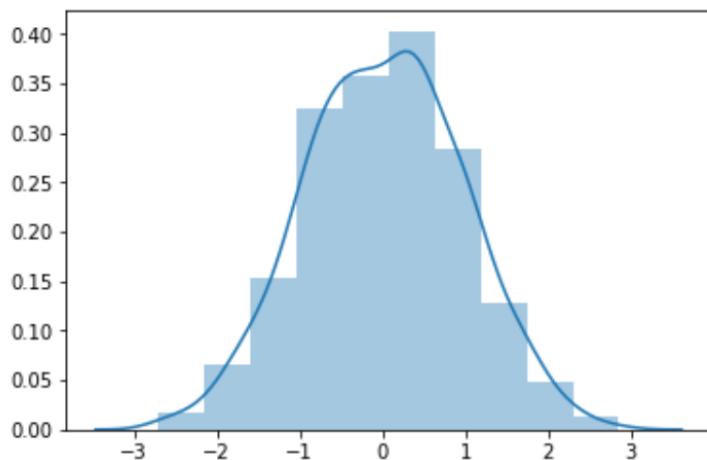
2、设置 bins 参数

```
g = sns.distplot(x,bins=10)
```



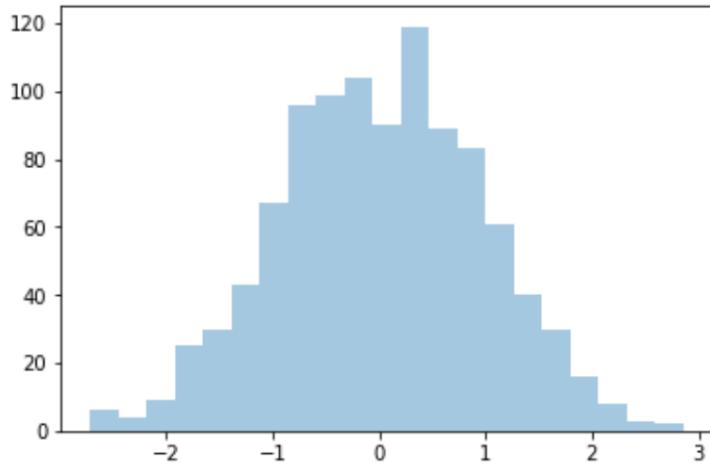
### 3、只显示密度图

```
g = sns.distplot(x, hist=False)
```



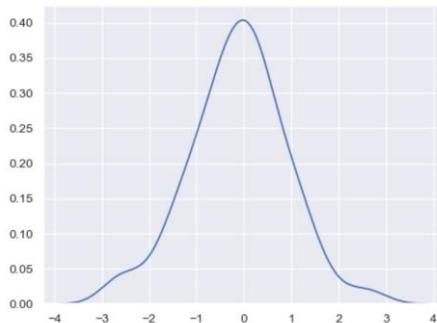
### 4、只显示直方图

```
g = sns.distplot(x, kde=False)
```

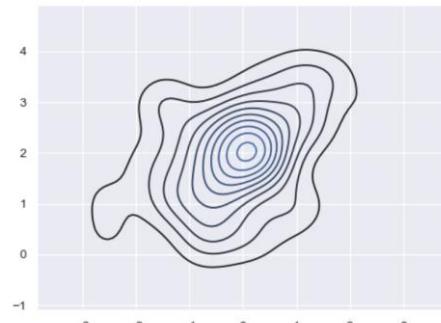


#### 4.6.2 密度 (KDE) 图

密度图和称为“核密度估计图”(KDE)，用来评估数据分布。使用 `kdeplot()` 函数可以绘制：单变量 KDE 图和双变量 KDE 图（等值线表示）。



单变量 KDE 图



等值线表示的双变量 KDE 图

`kdeplot` 函数的主要参数：

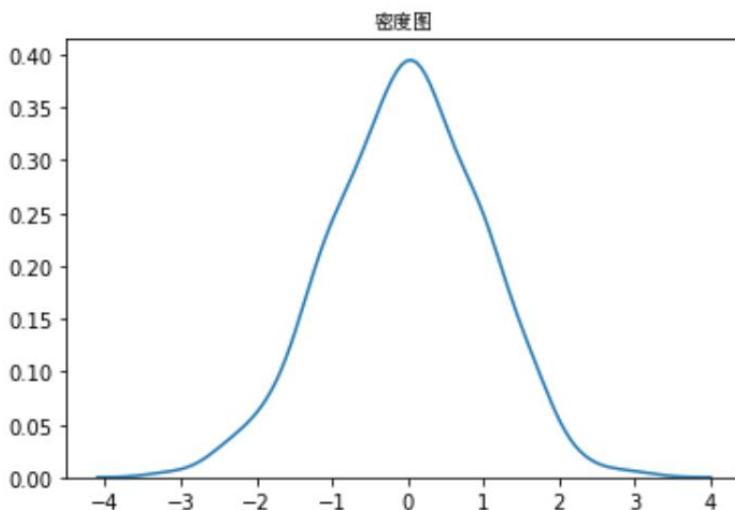
```
seaborn.kdeplot(data, data2=None, shade=False)
```

- `data` 参数是第 1 个变量数据，他可以是 Series、一维数组或列表。

- data2 参数是第 2 个变量数据，类型同 data 参数。
- shade 参数是否绘制阴影效果。

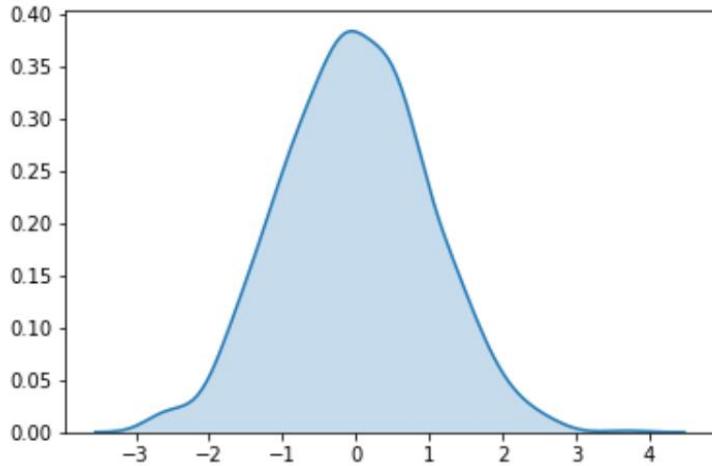
## 1、单变量密度图

```
n = 1024  
x = np.random.normal(0, 1, n)  
g = sns.kdeplot(x)  
plt.title('密度图', fontproperties='SimHei')
```



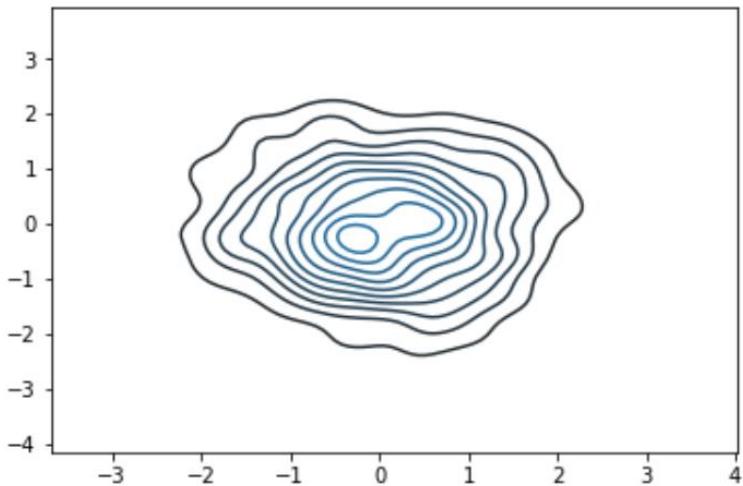
## 2、单变量密度图阴影效果

```
g = sns.kdeplot(x, shade=True)
```



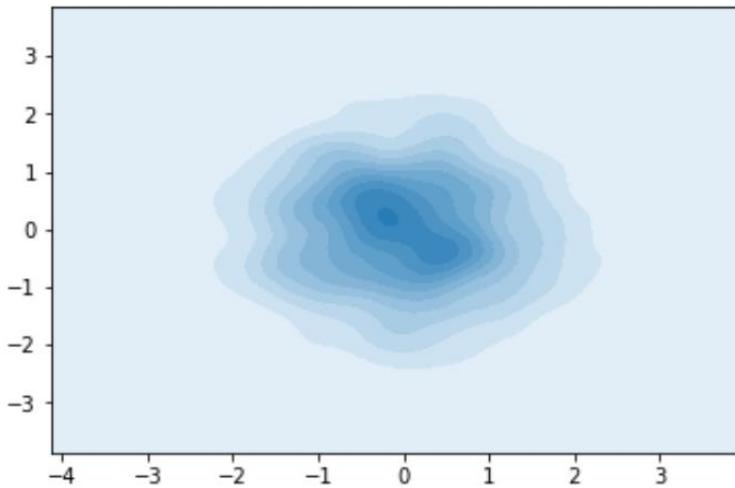
### 3、双变量密度图

```
n = 1024
x = np.random.normal(0, 1, n)
y = np.random.normal(0, 1, n)
g = sns.kdeplot(x, y)
```



### 4、双变量密度图阴影效果

```
g = sns.kdeplot(x, y, shade=True)
```



### 4.6.3 联合图

将单变量分布(Dist)图与双变量图绘制到一个图表中，可以使用 jointplot 函数， jointplot 函数的主要参数：

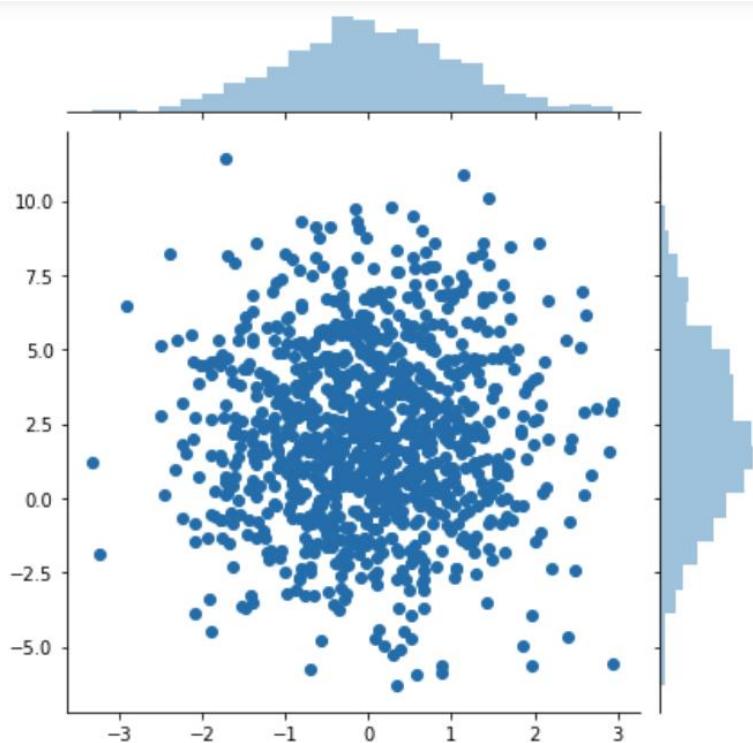
```
seaborn.jointplot (x, y, data = None, kind = 'scatter')
```

- kind 参数是绘制的双变量图形类型，取值主要有 scatter、reg、kde、hex， scatter 是散点， reg 是线性回归图， kde 是密度图， hex 是六角形。 scatter 是默认值。

#### 1、默认图

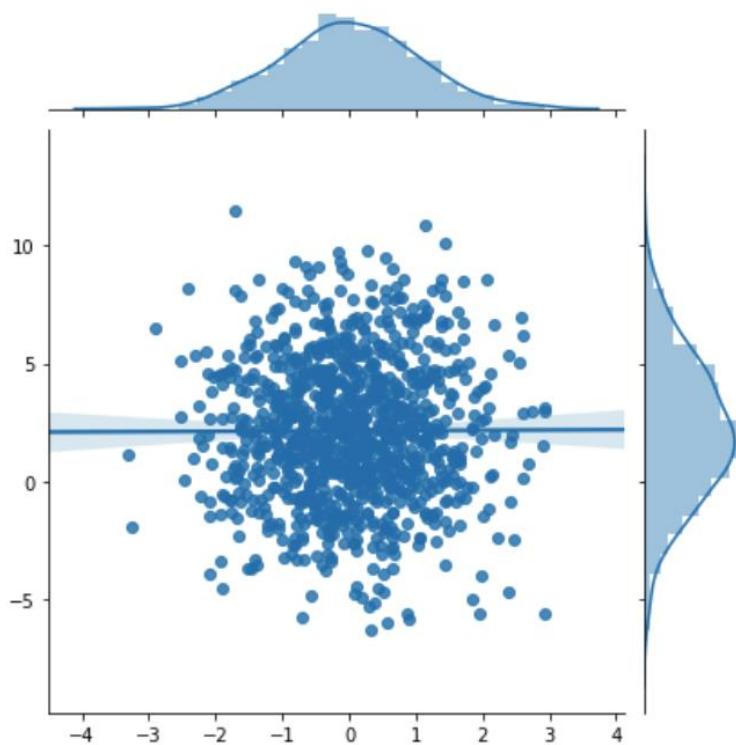
```
n = 1024
x = np.random.normal(0, 1, n)
y = np.random.normal(2, 3, n)

g = sns.jointplot(x, y)
```



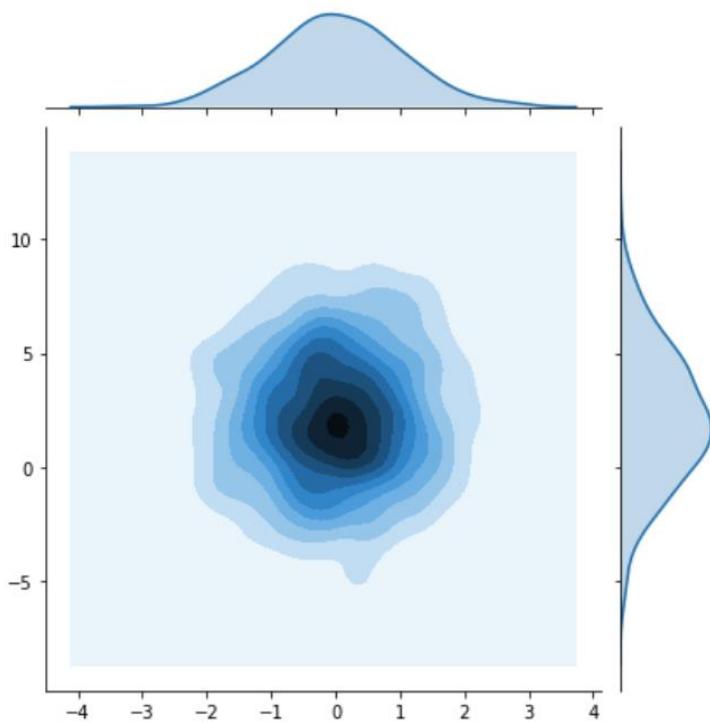
## 2、添加线性回归线

```
g = sns.jointplot(x, y, kind="reg")
```



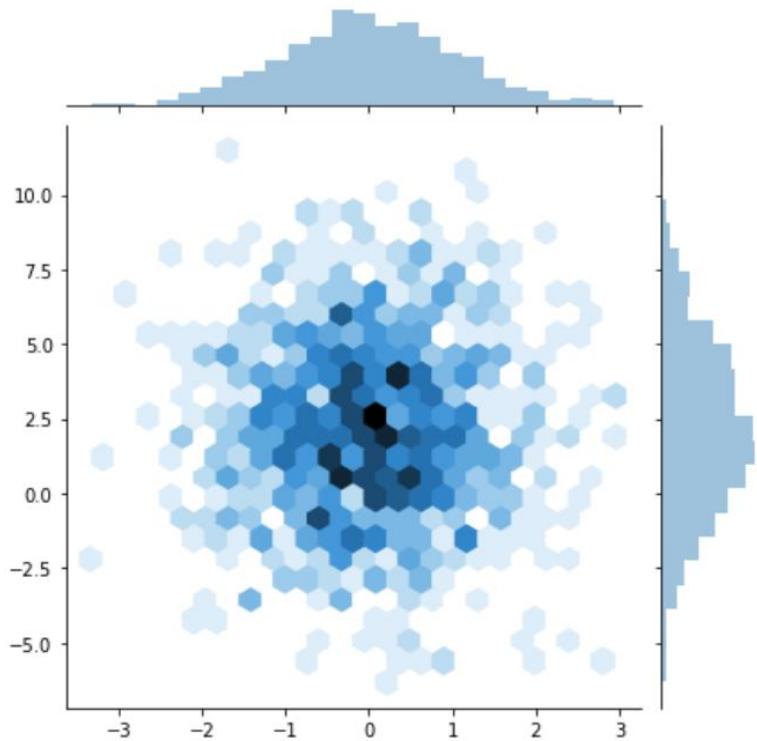
### 3、设置为密度图

```
g = sns.jointplot(x, y, kind="kde")
```



#### 4、设置为六角形图

```
g = sns.jointplot(x, y, kind="hex")
```



## 4.7 热力图

热力图 (heatmap) 是以矩阵形式表示数据的一种方式，数据值在图中表示为颜色。

Seaborn 中绘制热力图函数是 `seaborn. heatmap`，它的主要参数：

```
seaborn.heatmap(data, vmin=None, vmax=None, cmap=None, annot=None)
```

- `data` 参数，二维数据集。
- `vmin` 和 `vmax` 参数，图例中最大值和最小值的显示值。
- `cmap` 参数，设置颜色面板。
- `annot` 参数，设置热力图注解，如果 `True` 则在单元格中显示数据值。

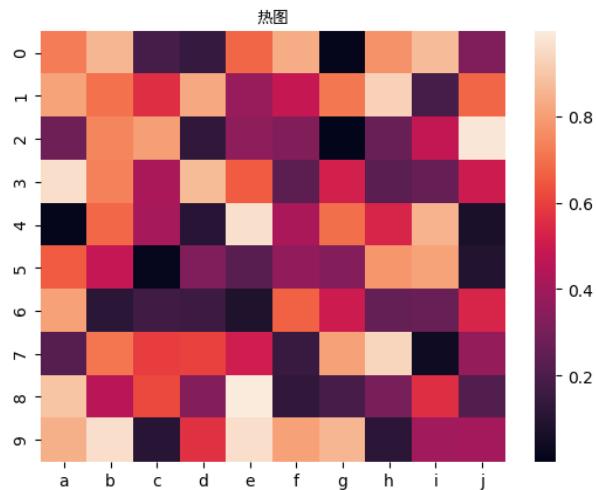
1、

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

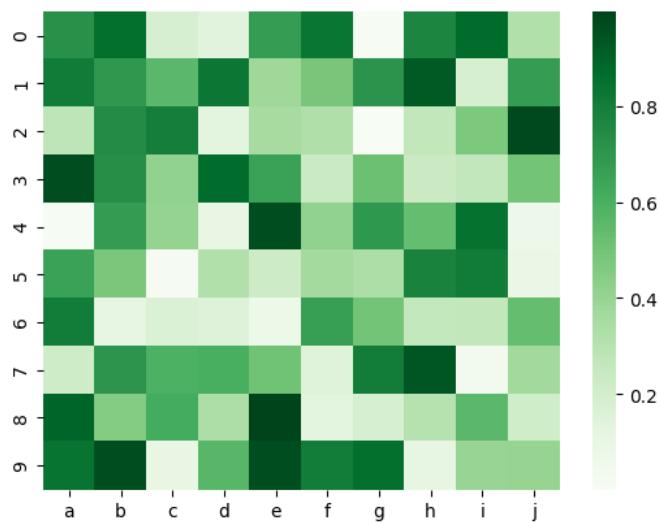
%matplotlib inline
# 设置中文
sns.set_style("darkgrid", {'font.family':'simhei'})

df = pd.DataFrame(np.random.rand(10, 10), columns=list('abcdefghijkl'))
sns.heatmap(df)
plt.title('热图', fontproperties='SimHei')
```



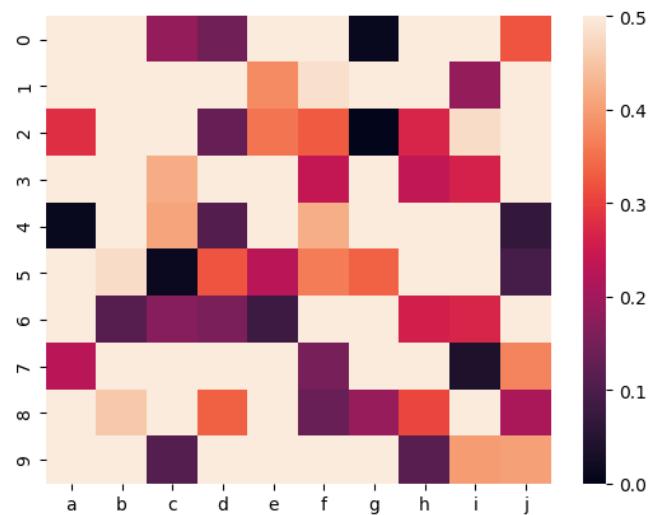
## 2、设置颜色面板

```
sns.heatmap(df, cmap="Greens")
```



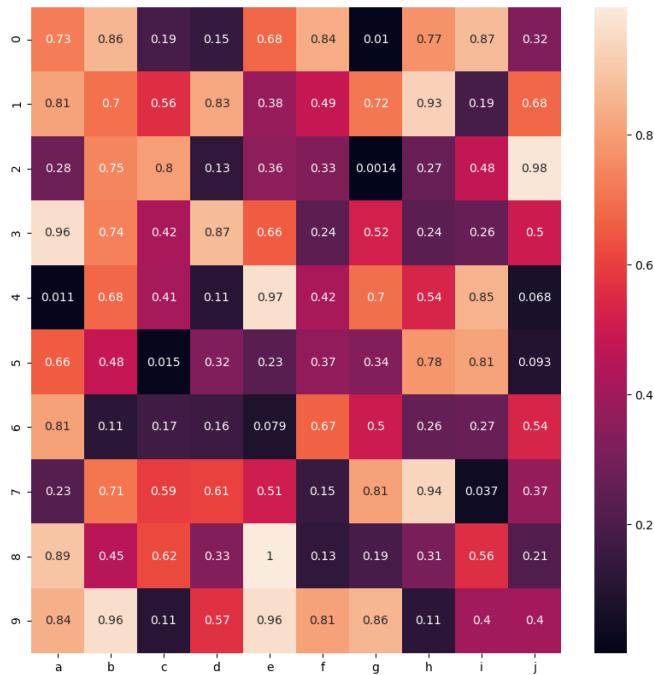
### 3、设置 vmin 和 vmax 参数

```
sns.heatmap(df, vmin=0, vmax=0.5)
```



### 4、设置 annot 参数

```
plt.figure(figsize=(10, 10))
sns.heatmap(df, annot=True)
```



## 4.8 线性回归图

线性回归图通过大量数据找到模型拟合的线性回归线，使用 Seaborn 的 regplot 函数。

回归图:线性回归图 regplot 和分面网格(FacetGrid)线性回归图 lmplot。

### 4.8.1 线性回归图 regplot

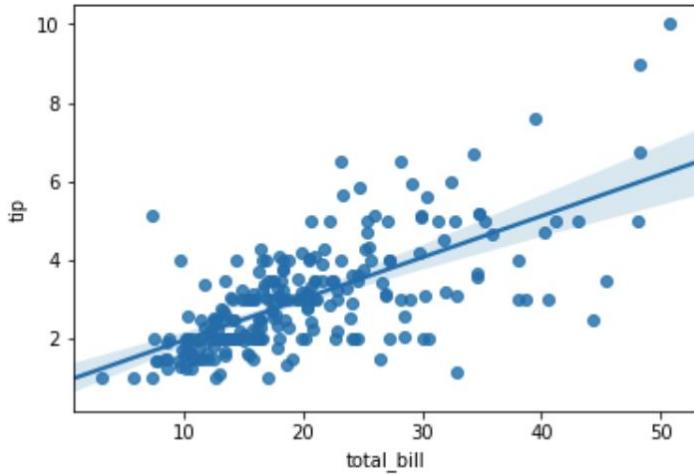
1、

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

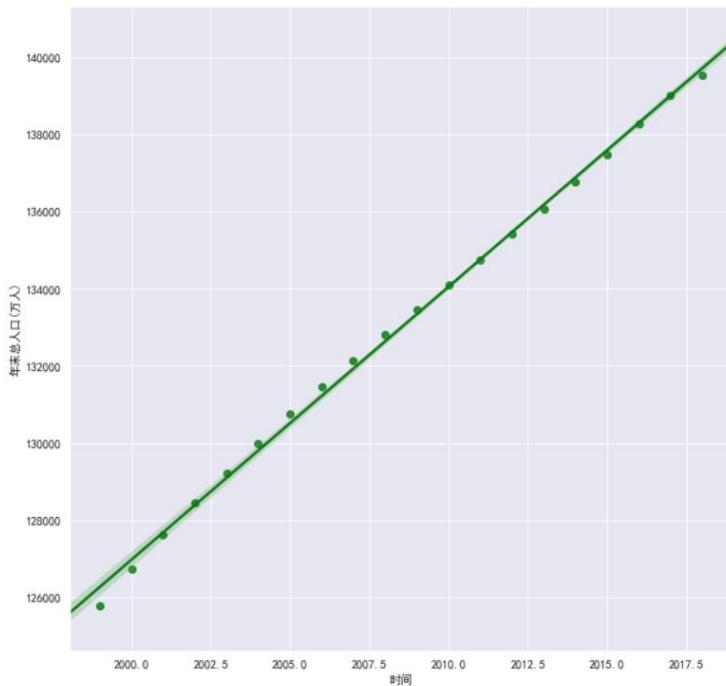
```
tips = sns.load_dataset('tips')
sns.regplot(x='total_bill', y='tip', data=tips)
```



2、

```
file_path='data\\'
df20 = pd.read_excel(file_path+'全国总人口数据.xls', sheet_name='20年
数据', skiprows=2, skipfooter=2)
df20["时间"] = df20["时间"].map(lambda x: int(x[0:-1]))

plt.figure(figsize=(10,10))
g=sns.regplot(x = "时间", y = "年末总人口(万人)", data = df20,
color="g")
plt.xlabel("时间", fontproperties='SimHei') # 添加x轴标题
plt.ylabel("年末总人口(万人)", fontproperties='SimHei') # 添加y轴标题
```

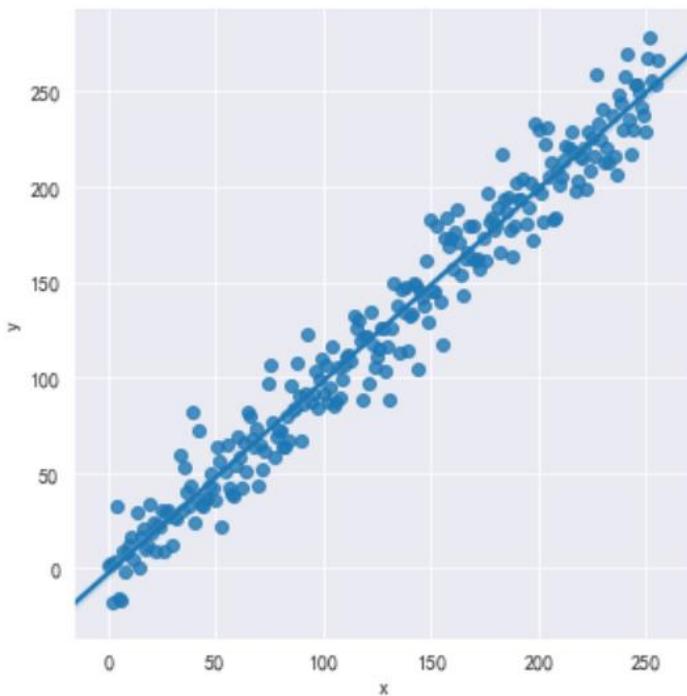


## 4.8.2 分面网格线性回归图 lmplot

### 1、默认绘制

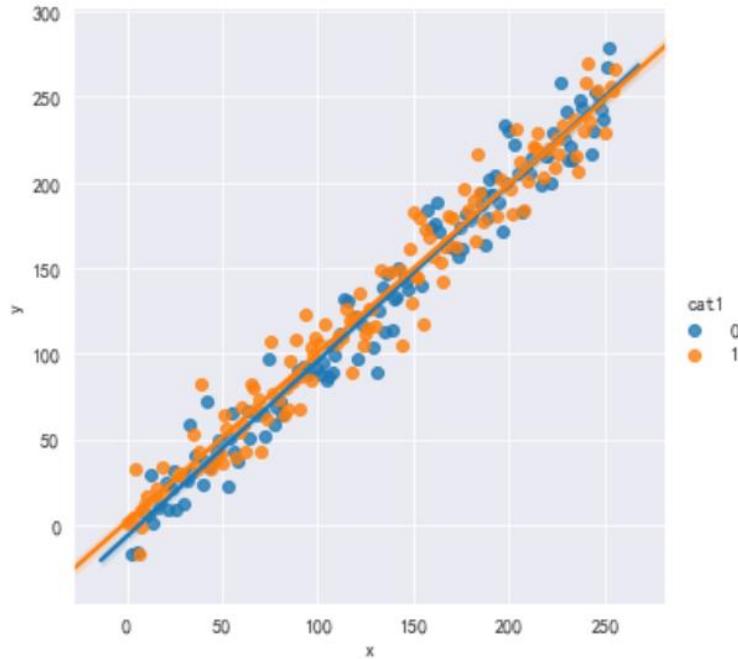
```
n = 256
sigma = 15
x = np.arange(n)
y = x + sigma*np.random.randn(n)
cat1 = np.random.randint(2, size=(n,))    # 分类1 取值0, 1
cat2 = np.random.randint(1, 4, size=(n,)) # 分类2 取值1, 2, 3
cat3 = np.random.randint(5, 7, size=(n,)) # 分类3 取值5, 6

df = pd.DataFrame({'x':x, 'y':y, 'cat1':cat1, 'cat2':cat2,
'cat3':cat3})
sns.lmplot(data = df, x='x', y='y')
```



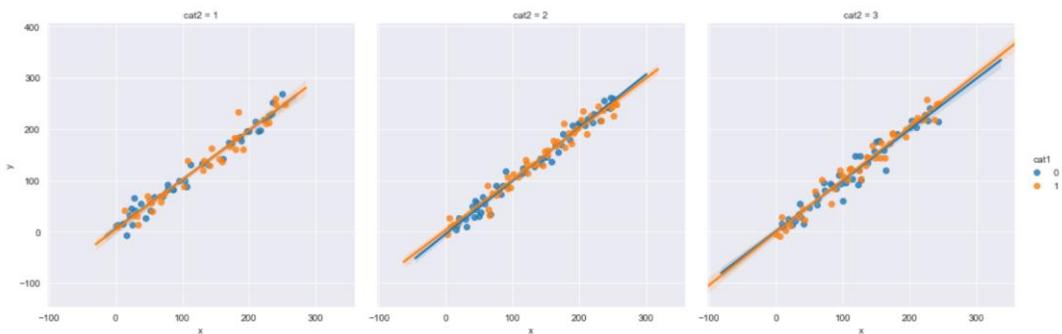
## 2、设置 hue 参数

```
g=sns.lmplot(data=df, x='x', y='y', hue='cat1')
```



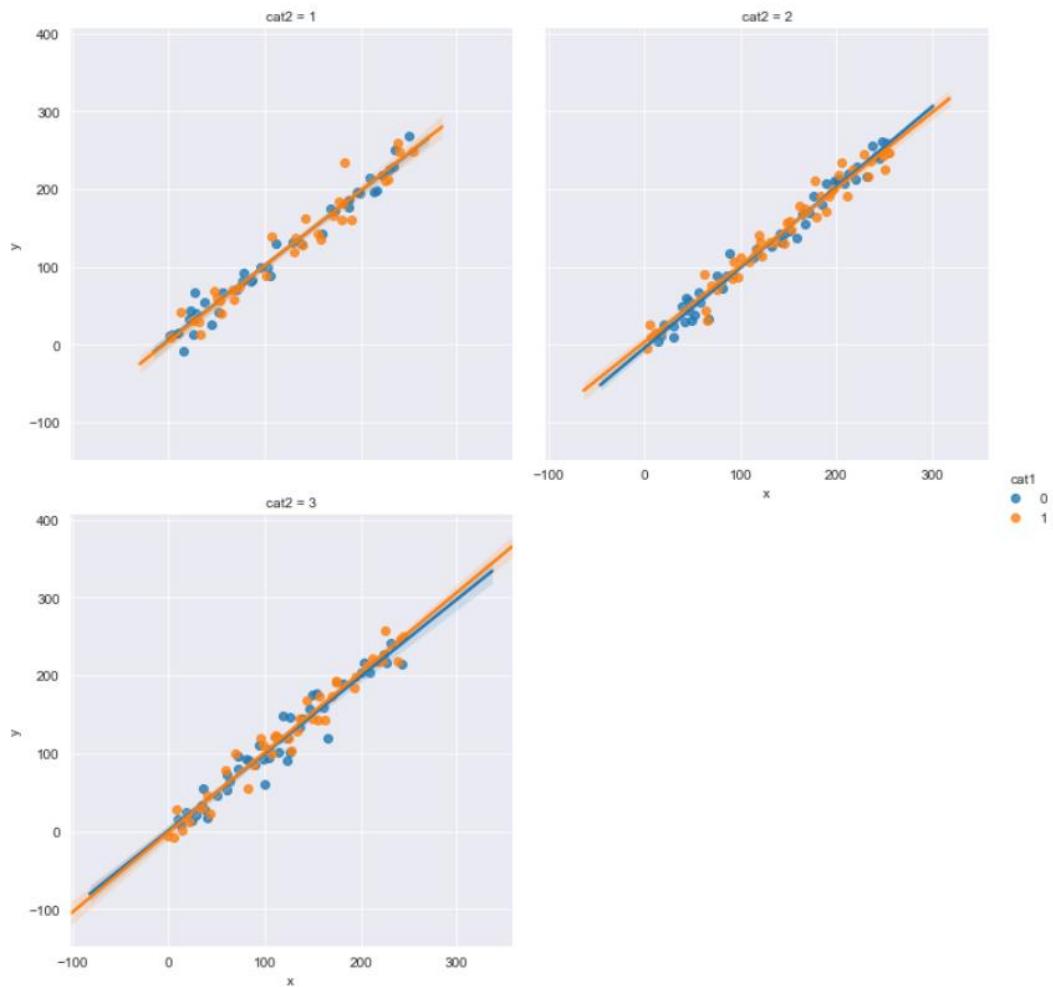
### 3、设置 col 参数

```
g = sns.lmplot(data=df, x='x', y='y', hue='cat1', col='cat2')
```



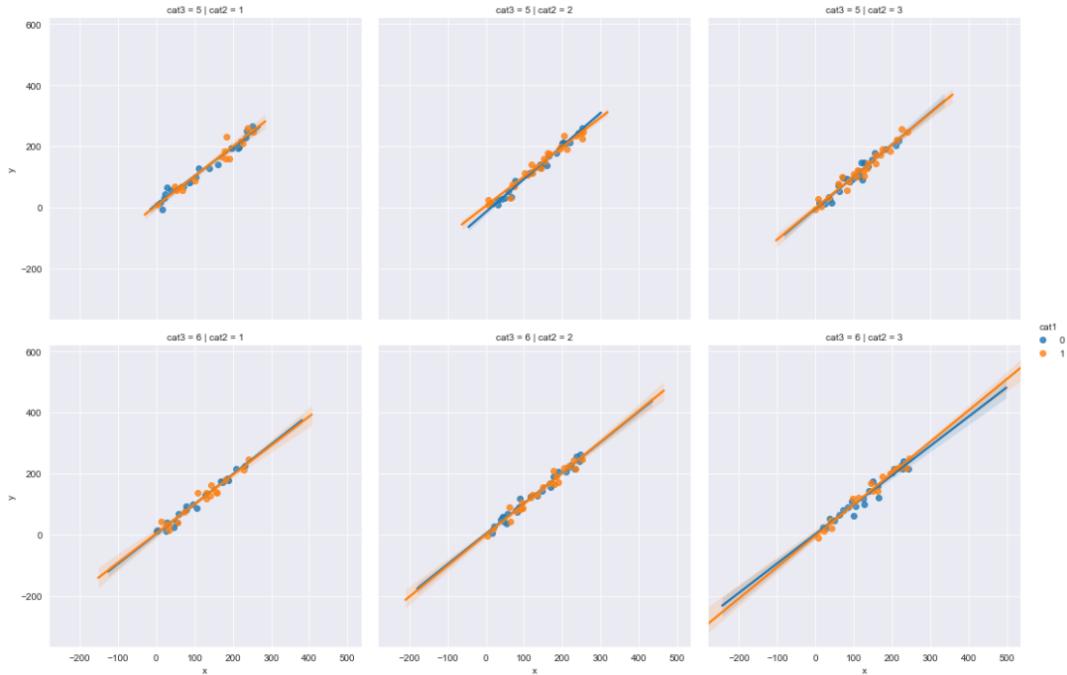
### 4、设置 col 和 col\_wrap 参数

```
g = sns.lmplot(data=df, x='x', y='y', hue='cat1', col='cat2',  
col_wrap=2)
```



## 5、设置 col 和 row 参数

```
g = sns.lmplot(data=df, x='x', y='y', hue='cat1', col='cat2',  
row='cat3')
```



## 4.9 分面网格绘图

可以通过 `seaborn.FacetGrid` 类绘图分面网格，然后再通过 `seaborn.FacetGrid.map` 方法将一个绘图函数作用于一个网格，并绘制子图。

`seaborn.FacetGrid` 构造函数：

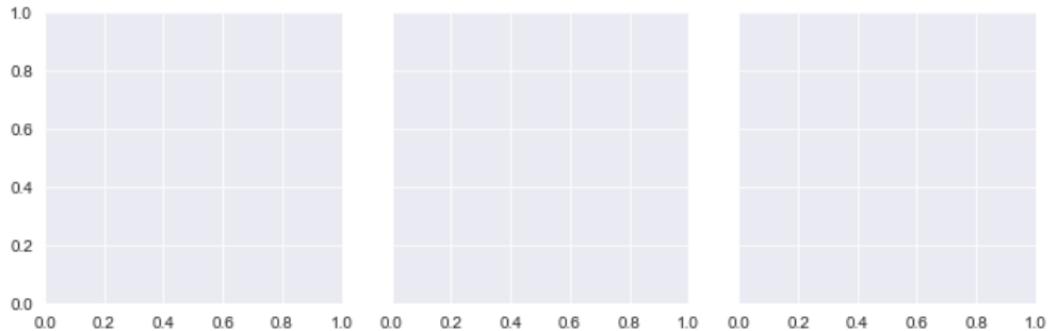
```
seaborn.FacetGrid(data, row=None, col=None, hue=None, col_wrap=None)
```

`FacetGrid.map` 方法：

```
FacetGrid.map(func, *args, **kwargs)
```

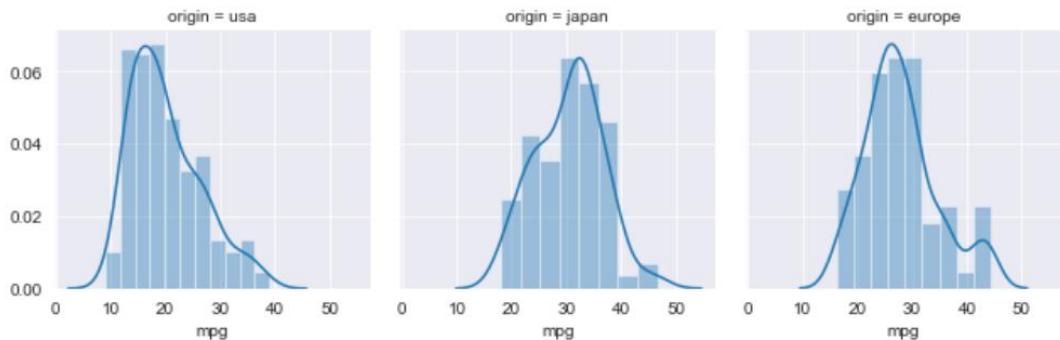
### 1、绘图单变量分面网格

```
mpg_df = sns.load_dataset("mpg")
g = sns.FacetGrid(mpg_df, col="origin")
```



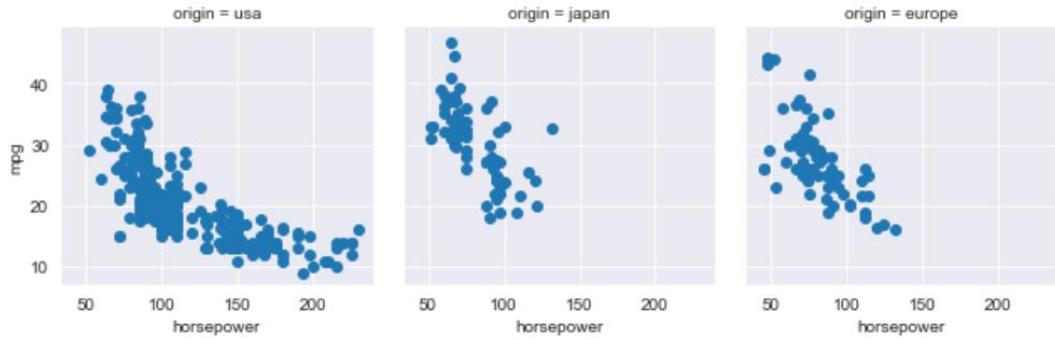
## 2、映射一个绘图函数

```
g = sns.FacetGrid(mpg_df, col="origin")
g.map(sns.distplot, "mpg")
```



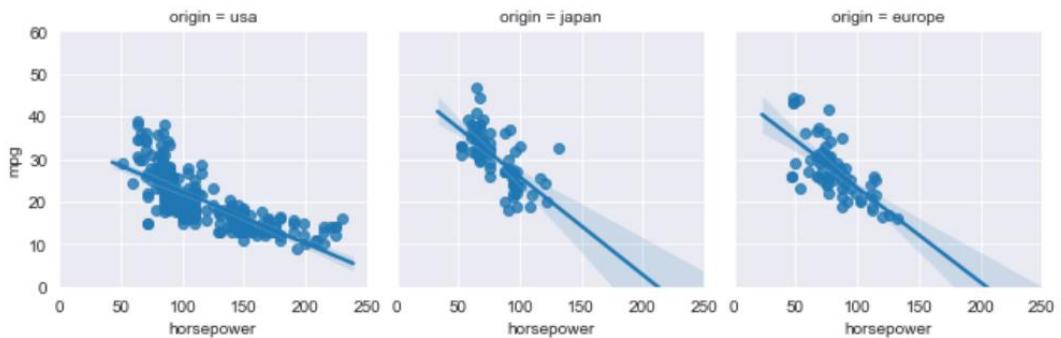
## 3、绘制 mpg 与马力的散点图

```
g = sns.FacetGrid(mpg_df, col="origin")
g.map(sns.scatterplot, "horsepower", "mpg")
```



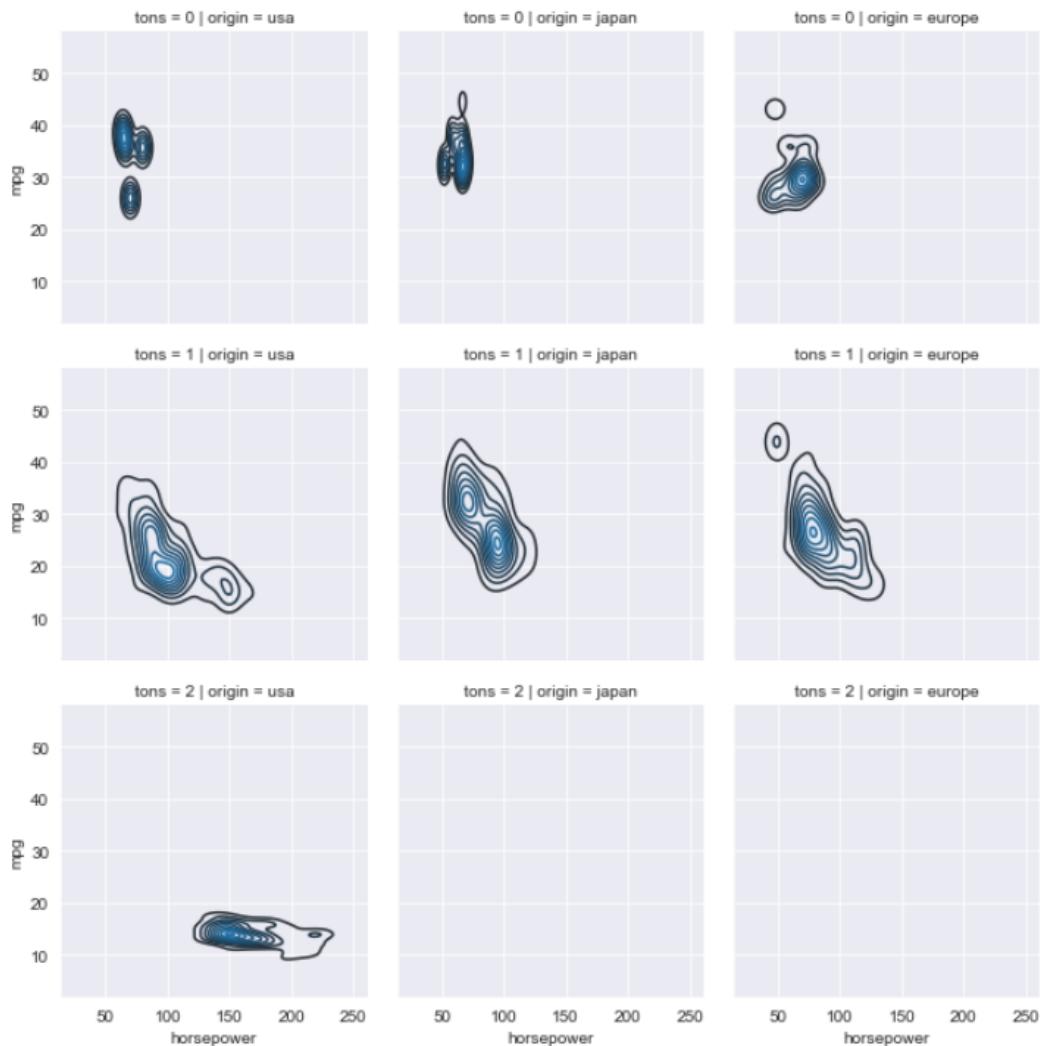
## 4、绘制回归图

```
g = sns.FacetGrid(mpg_df, col="origin")
g.map(sns.regplot, "horsepower", "mpg")
plt.xlim(0, 250)
plt.ylim(0, 60)
```



## 5、绘图双变量分面网格

```
mpg_df['tons'] = (mpg_df.weight/2000).astype(int)
g = sns.FacetGrid(mpg_df, col="origin", row="tons")
g.map(sns.kdeplot, "horsepower", "mpg")
```

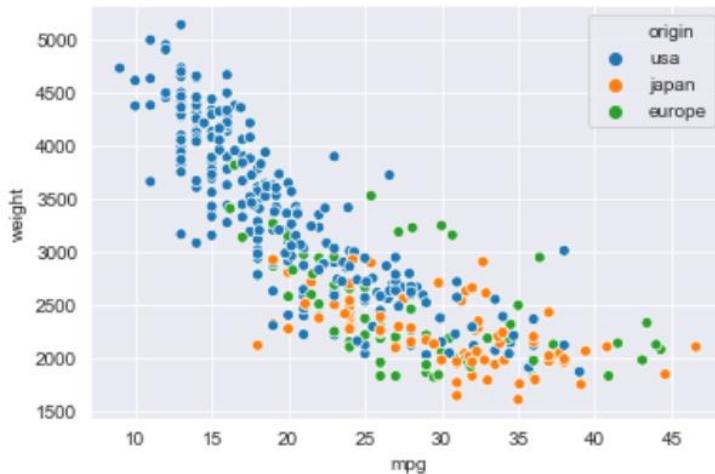


## 4.10 课后练习

1、使用 Seaborn 库内置数据集 mpg 绘制关联散点图。

参考答案：

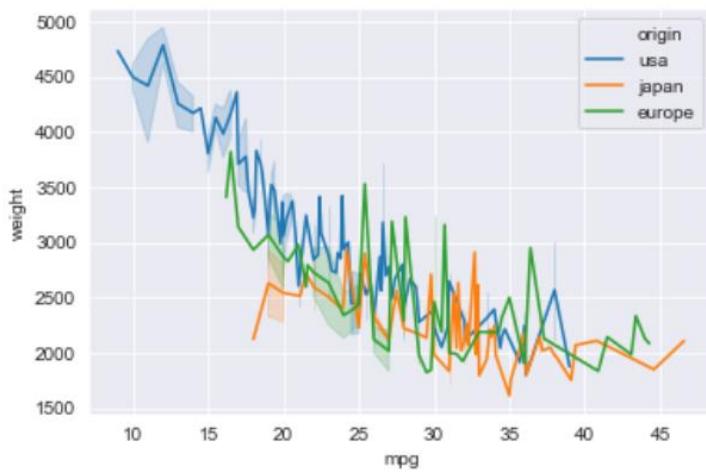
```
mpg_df = sns.load_dataset('mpg')
sns.scatterplot(data=mpg_df, x='mpg', y='weight', hue='origin')
```



2、使用 Seaborn 库内置数据集 mpg 绘制关联线图。

参考答案：

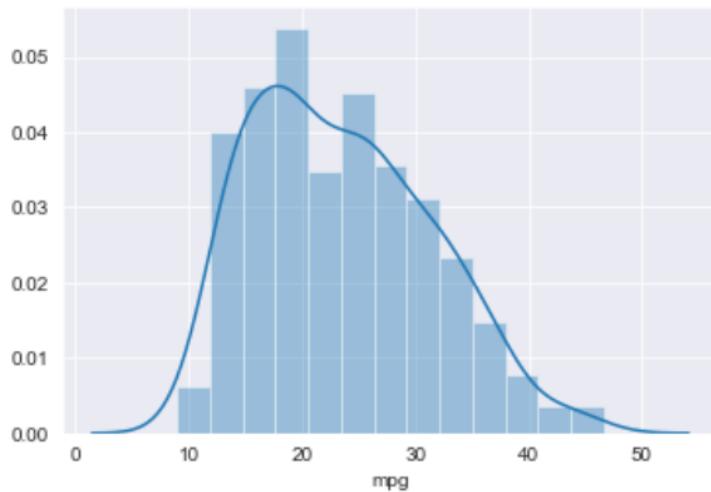
```
sns.lineplot(data=mpg_df, x='mpg', y='weight', hue='origin')
```



3、使用 Seaborn 库内置数据集 mpg 绘制 Dist 分布图。

参考答案：

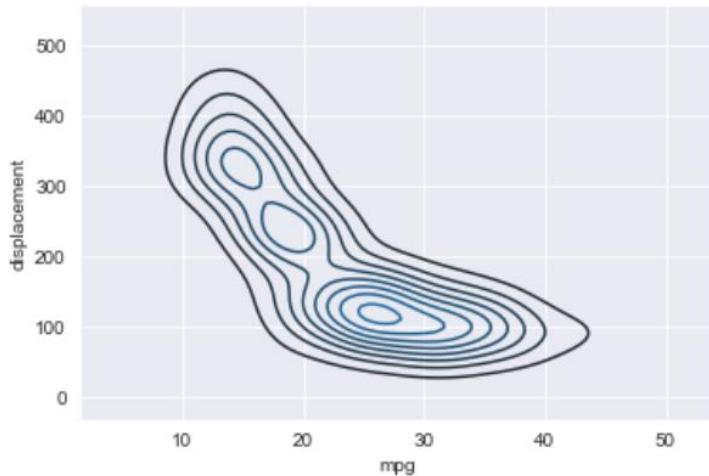
```
sns.distplot(mpg_df['mpg'])
```



4、使用 Seaborn 库内置数据集 mpg 绘制 KDE 密度图。

参考答案：

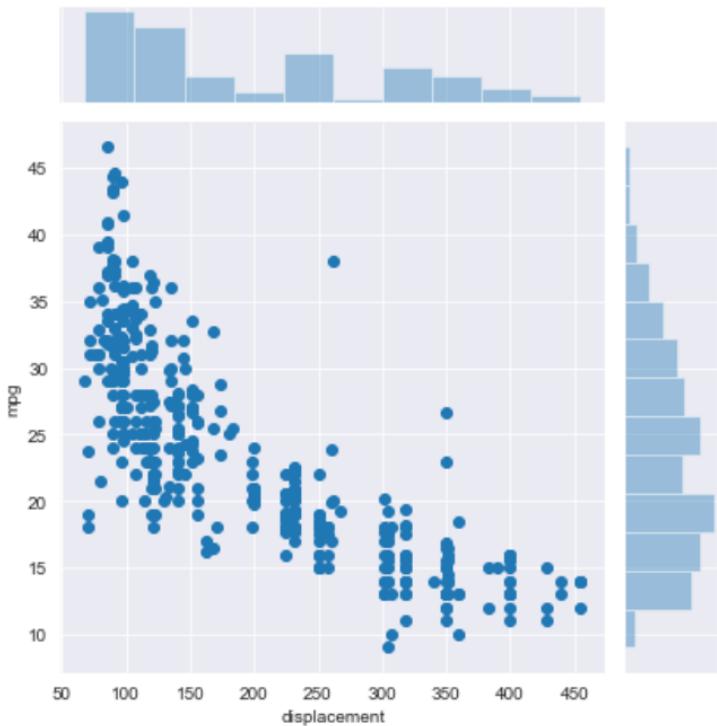
```
sns.kdeplot(mpg_df['mpg'], mpg_df['displacement'])
```



5、使用 Seaborn 库内置数据集 mpg 绘制联合图。

参考答案：

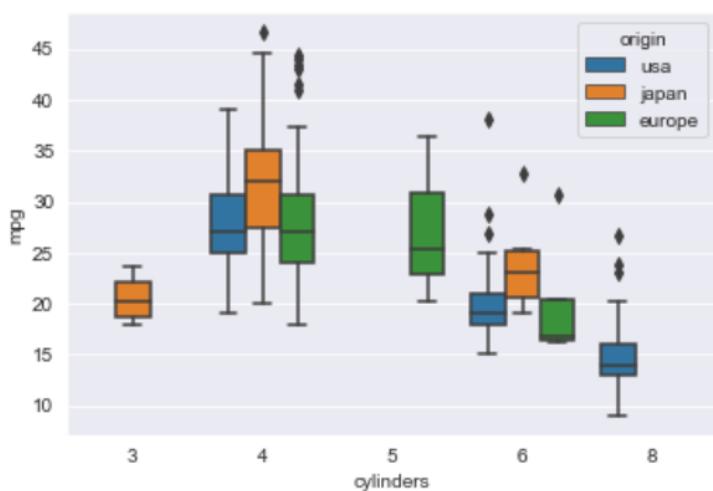
```
sns.jointplot(x='displacement', y='mpg', data=mpg_df)
```



6、使用 Seaborn 库内置数据集 mpg 绘制箱线图。

参考答案：

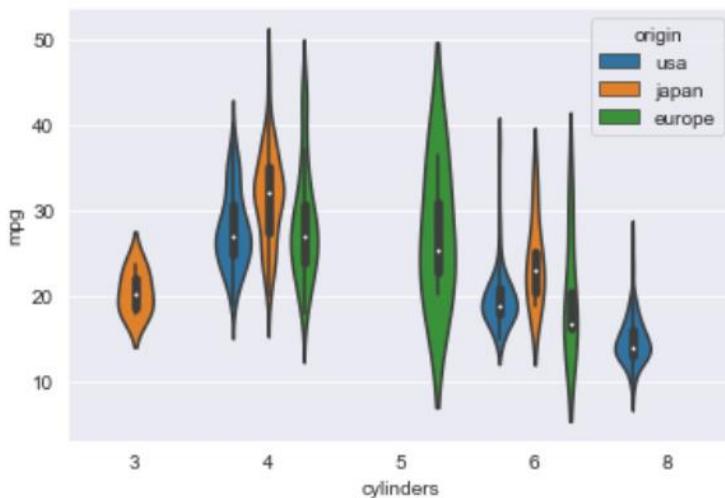
```
sns.boxplot(x='cylinders', y='mpg', hue='origin', data=mpg_df)
```



7、使用 Seaborn 库内置数据集 mpg 绘制小提琴图。

参考答案：

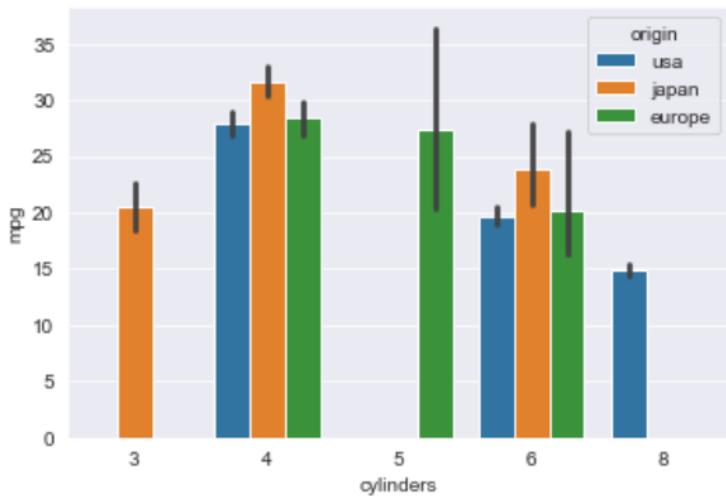
```
sns.violinplot(x='cylinders', y='mpg', hue='origin', data=mpg_df)
```



8、使用 Seaborn 库内置数据集 mpg 绘制柱状图。

参考答案：

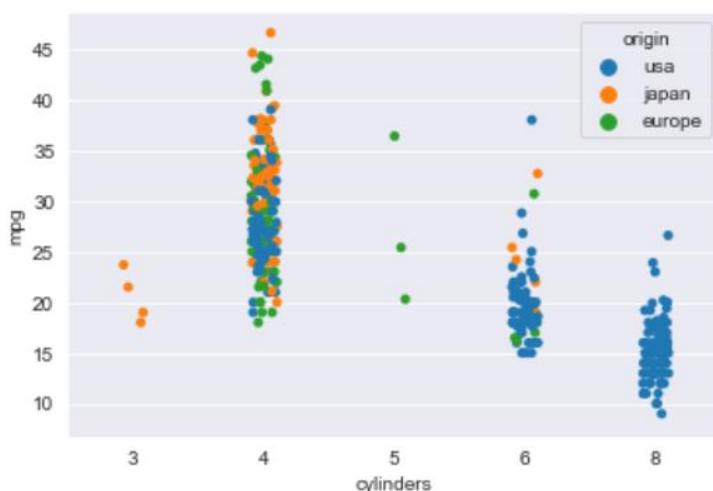
```
sns.barplot(x='cylinders', y='mpg', hue='origin', data=mpg_df)
```



9、使用 Seaborn 库内置数据集 mpg 绘制 Strip 散点图。

参考答案：

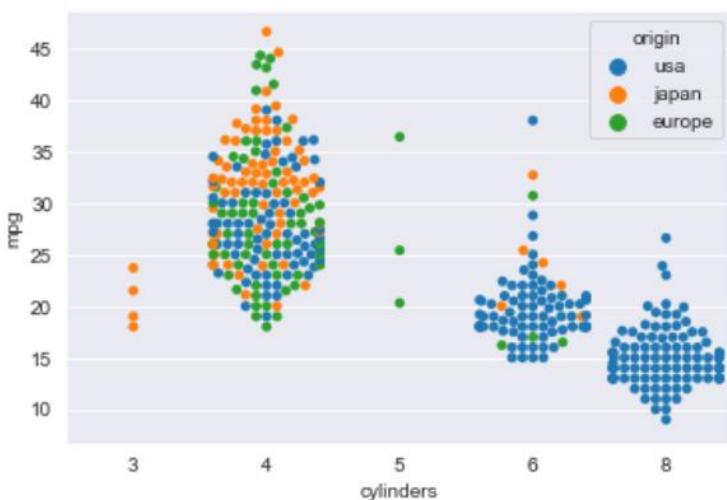
```
sns.stripplot(x="cylinders", y="mpg", hue="origin", data=mpg_df)
```



10、使用 Seaborn 库内置数据集 mpg 绘制 Swarm 散点图。

参考答案：

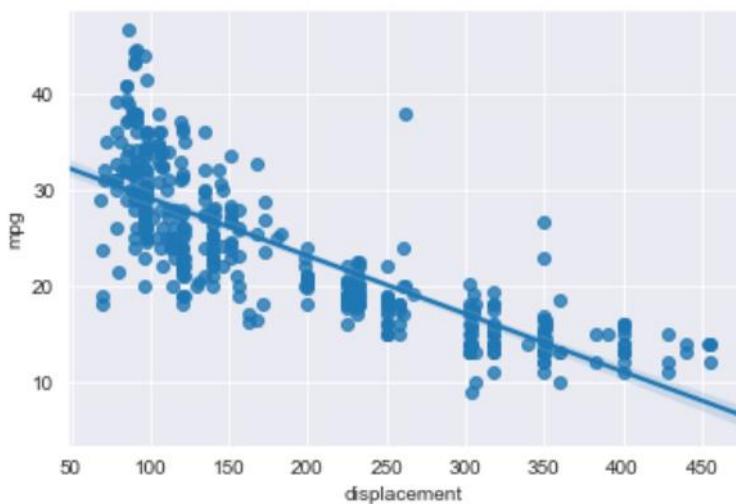
```
sns.swarmplot(x='cylinders', y='mpg', hue='origin', data=mpg_df)
```



11、使用 Seaborn 库内置数据集 mpg 绘制线性回归图。

参考答案：

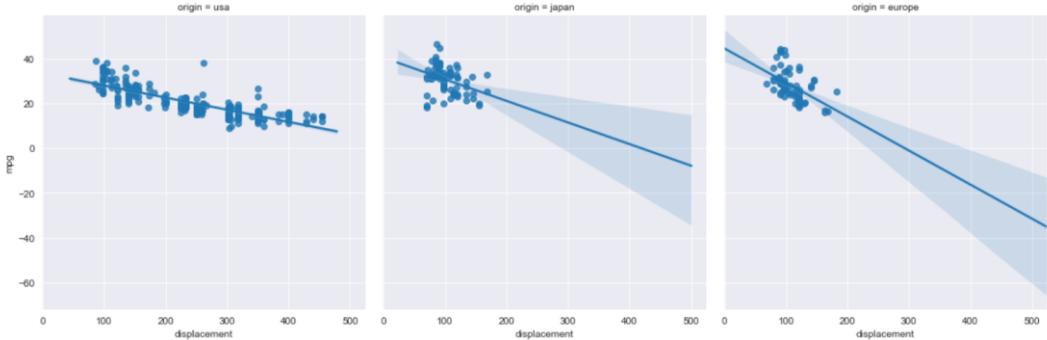
```
sns.regplot(x='displacement', y='mpg', data=mpg_df)
```



12、使用 Seaborn 库内置数据集 mpg 绘制分面网格线性回归图。

参考答案：

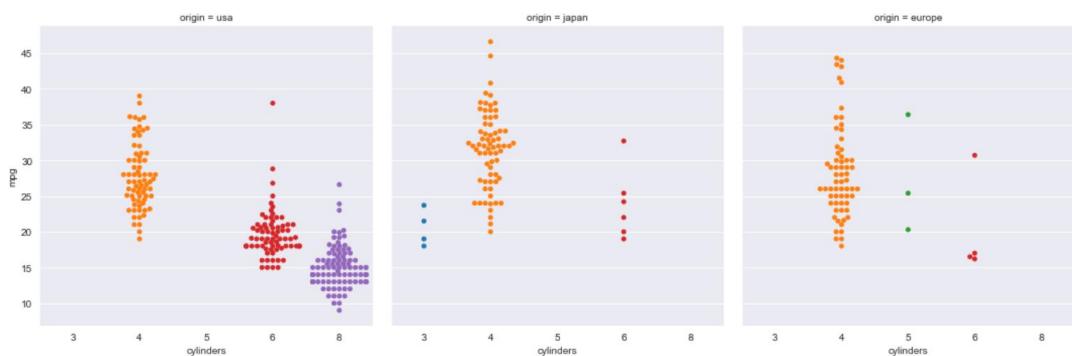
```
sns.lmplot(x='displacement', y='mpg', col='origin', data=mpg_df)
```



13、使用 Seaborn 库内置数据集 mpg 绘制分面网格分类图。

参考答案：

```
sns.catplot(x='cylinders', y='mpg', col='origin', data=mpg_df,  
kind='swarm')
```

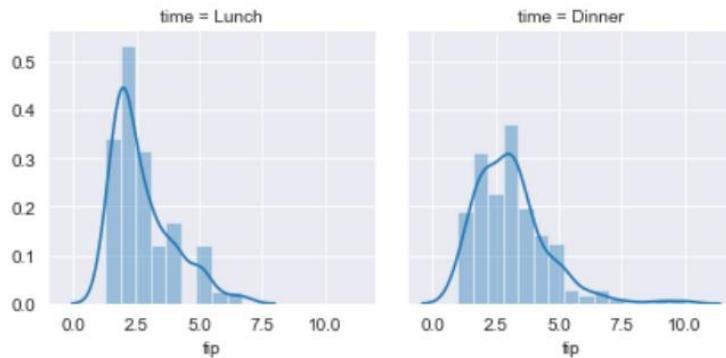


14、使用 Seaborn 库内置数据集 tips 绘制，通过 FacetGrid 类绘制分面网格密度图。

参考答案：

```
tips = sns.load_dataset('tips')  
g = sns.FacetGrid(tips, col='time')
```

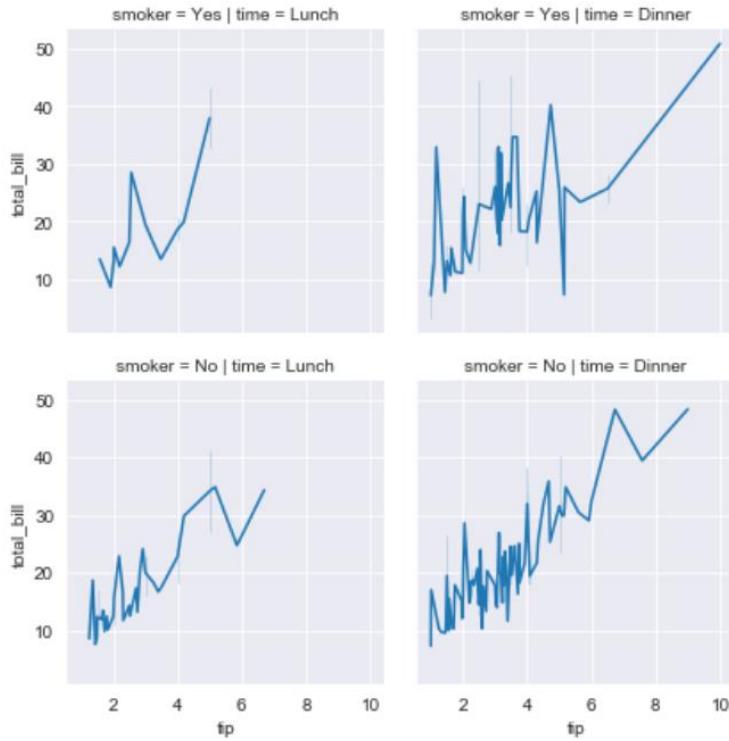
```
g.map(sns.distplot, 'tip')
```



- 15、使用 Seaborn 库内置数据集 tips 绘制，通过 FacetGrid 类绘制分面网格线图。

参考答案：

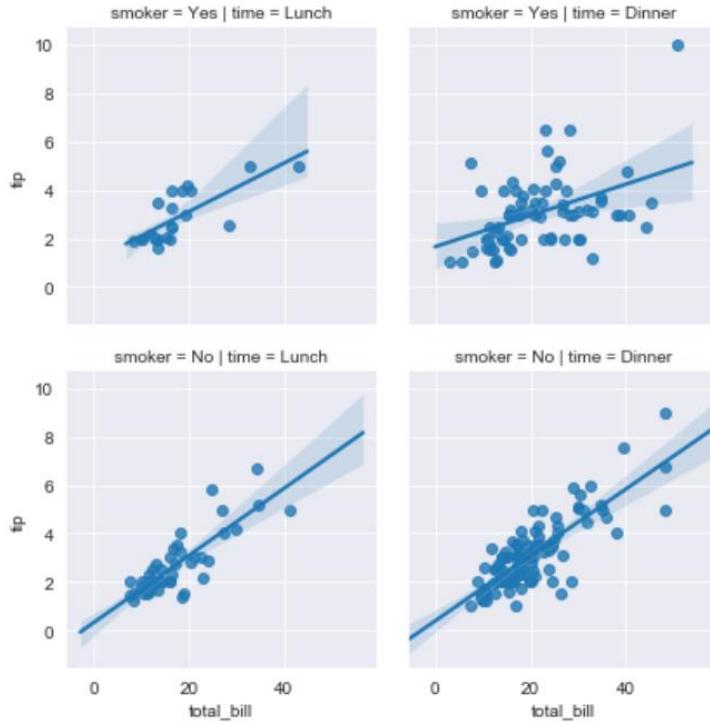
```
g = sns.FacetGrid(tips, col='time', row='smoker')
g.map(sns.lineplot, 'total_bill', 'tip')
```



16、使用 Seaborn 库内置数据集 tips 绘制，通过 FacetGrid 类绘制分面网格线性回归图。

参考答案：

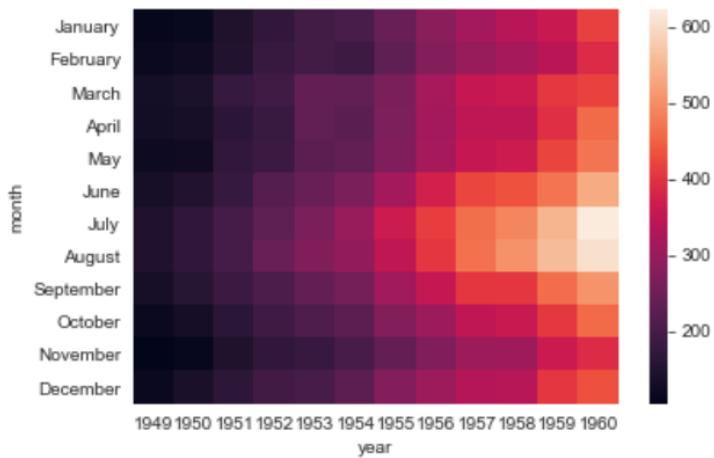
```
g = sns.FacetGrid(tips, col='time', row='smoker')
g.map(sns.regplot, 'total_bill', 'tip')
```



17、使用 Seaborn 库内置数据集 flights 绘制热点图。

参考答案：

```
flights_long = sns.load_dataset('flights')
flights = flights_long.pivot('month', 'year', 'passengers')
sns.heatmap(flights)
```



# 第5章 项目实战：股票数据可视化

## 5.1 获取贵州茅台股票历史数据

<http://q.stock.sohu.com/cn/600519/lshq.shtml>

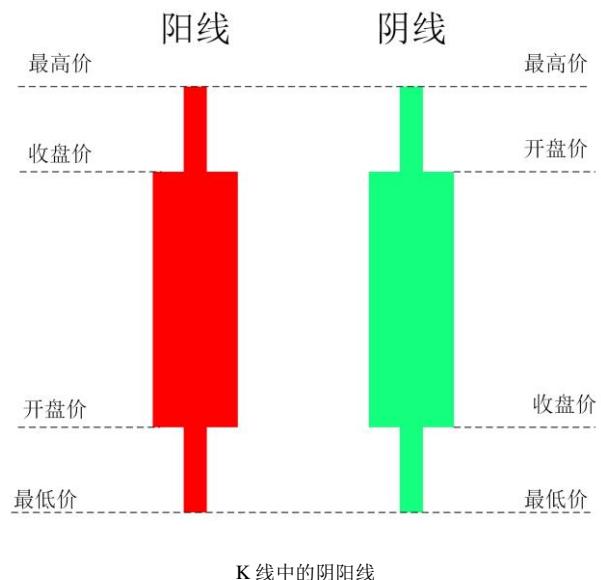
- 1、通过网络爬虫，爬取数据。
- 2、通过 pandas.read\_html()方法。
- 3、通过 pandas.read\_clipboard()方法。

## 5.2 绘制股票 OHLC 线图

它将 OHLC（开盘价、最高价、最低价、收盘价）信息绘制在一张图表上，宏观上可以反映出价格走势，微观上可以看出每天涨跌等信息。

## 5.3 绘制股票成交量线图

## 5.4 绘制股票 K 线图



mpl\_finance 安装可以使用 pip 工具，安装指令如下：

```
pip install mpl_finance
```

绘制股票 K 线图是 mpl\_finance 库中的 candlestick\_ohlc 函数：

```
candlestick_ohlc(ax, quotes, width=0.2,  
                  colorup='k', colordown='r', alpha=1.0)
```

- ❑ ax: 绘制图表的轴对象
- ❑ quotes: 股票数据, 是一个包括(time, open, high, low, close)元组的 zip 对象
- ❑ width: 线宽度
- ❑ colorup: 阳线颜色
- ❑ colordown: 阴线颜色
- ❑ alpha: 透明度