AI基础：数据可视化简易入门（Matplotlib 和 Seaborn）

原创：机器学习初学者 机器学习初学者 11月25日

0 导语

Matplotlib 是一个 Python 的 2D 绘图库，它以各种硬拷贝格式和跨平台的交互式环境生成出版质量级别的图形 。

通过 Matplotlib，开发者可以仅需要几行代码，便可以生成绘图，直方图，功率谱，条形图，错误图，散点图等。

Seaborn 是基于 Python 且非常受欢迎的图形可视化库，在 Matplotlib 的基础上，进行了更高级的封装，使得作图更加方便快捷。即便是没有什么基础的人，也能通过极简的代码，做出具有分析价值而又十分美观的图形。

在此之前，我已经写了一篇 **Numpy** 和 **Pandas** 的快速入门，本篇文章讲解数据可视化快速入门：

AI 基础：Numpy 简易入门

AI 基础：Pandas 简易入门

备注：本文代码可以在github下载

https://github.com/fengdu78/Data-Science-Notes/tree/master/5.data-visualization

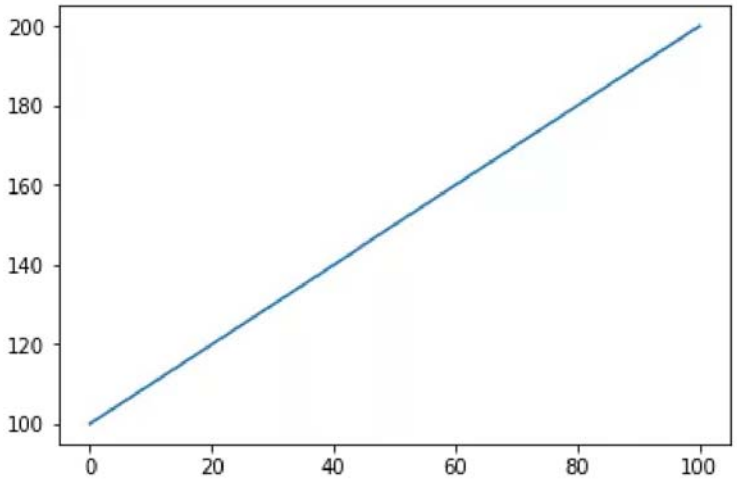
1.Matplotlib

1.1 通过 figure()函数创建画布

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

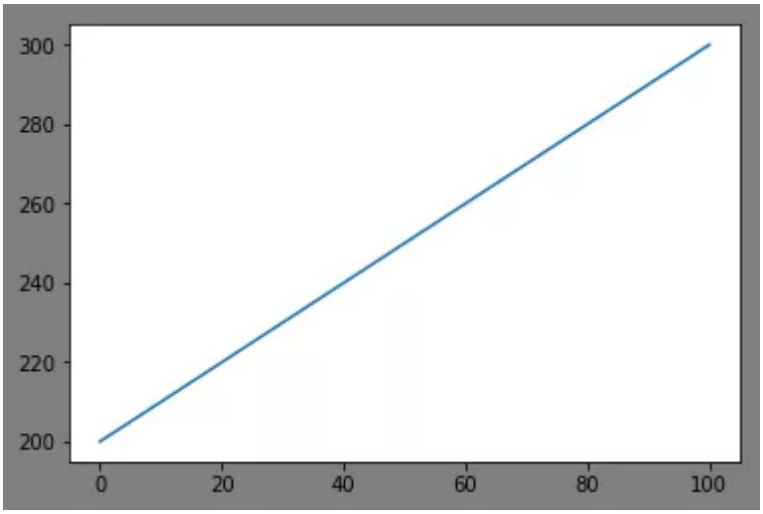
```
import numpy as np
data_one = np.arange(100, 201)  # 生成包含100~200的数组
```

```
plt.plot(data_one)                # 绘制data1折线图
plt.show()
```




```
# 创建新的空白画布，返回Figure实例
figure_obj = plt.figure()
```

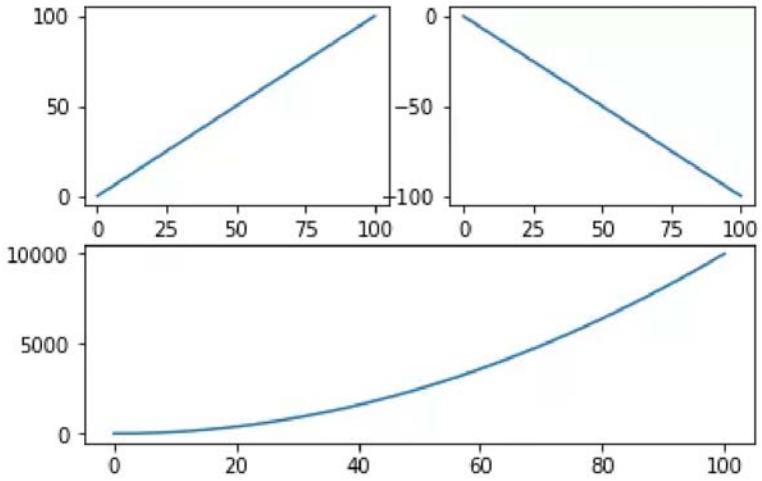
```
data_two = np.arange(200, 301)       # 生成包含200~300的数组
plt.figure(facecolor='gray')     # 创建背景为灰色的新画布
plt.plot(data_two)                      # 通过data2绘制折线图
plt.show()
```



1.2 通过 subplot()函数创建单个子图

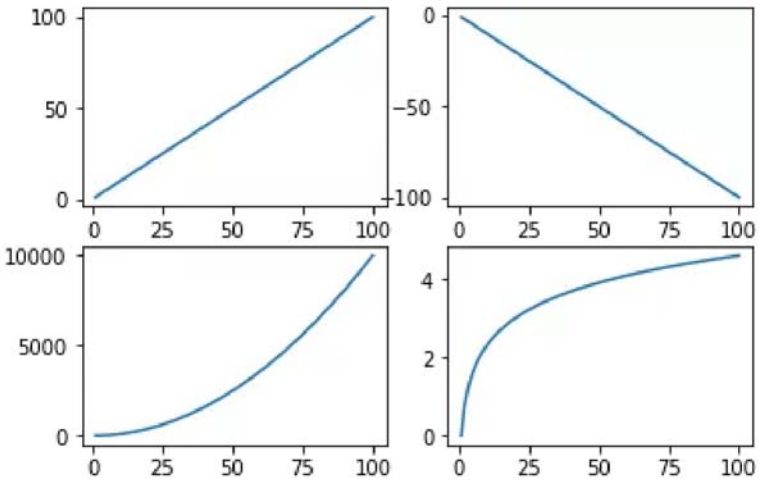
```
nums = np.arange(0, 101)     # 生成0~100的数组
# 分成2*2的矩阵区域，占用编号为1的区域，即第1行第1列的子图
plt.subplot(221)
# 在选中的子图上作图
```

```
plt.plot(nums, nums)
# 分成2*2的矩阵区域，占用编号为2的区域，即第1行第2列的子图
plt.subplot(222)
# 在选中的子图上作图
plt.plot(nums, -nums)
# 分成2*1的矩阵区域，占用编号为2的区域，即第2行的子图
plt.subplot(212)
# 在选中的子图上作图
plt.plot(nums, nums**2)
# 在本机上显示图形
plt.show()
```



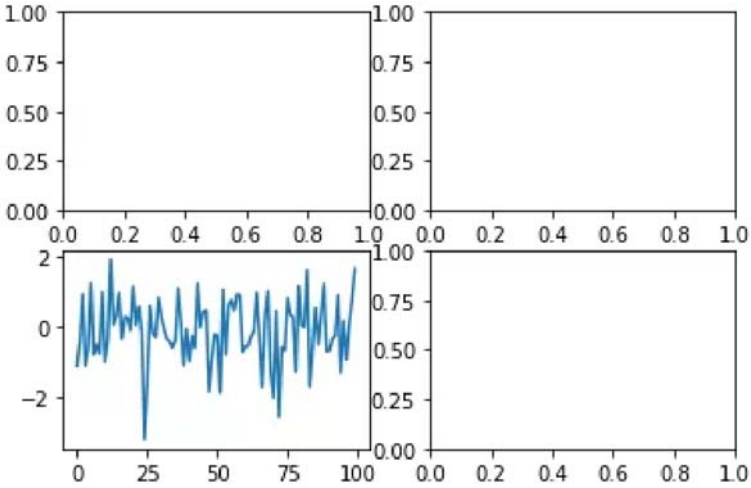
1.3 通过 subplots()函数创建多个子图

```
# 生成包含1～100之间所有整数的数组
nums = np.arange(1, 101)
# 分成2*2的矩阵区域，返回子图数组axes
fig, axes = plt.subplots(2, 2)
ax1 = axes[0, 0]  # 根据索引［0，0］从Axes对象数组中获取第1个子图
ax2 = axes[0, 1]   # 根据索引［0，1］从Axes对象数组中获取第2个子图
ax3 = axes[1, 0]   # 根据索引［1，0］从Axes对象数组中获取第3个子图
ax4 = axes[1, 1]   # 根据索引［1，1］从Axes对象数组中获取第4个子图
# 在选中的子图上作图
ax1.plot(nums, nums)
ax2.plot(nums, -nums)
ax3.plot(nums, nums**2)
ax4.plot(nums, np.log(nums))
plt.show()
```



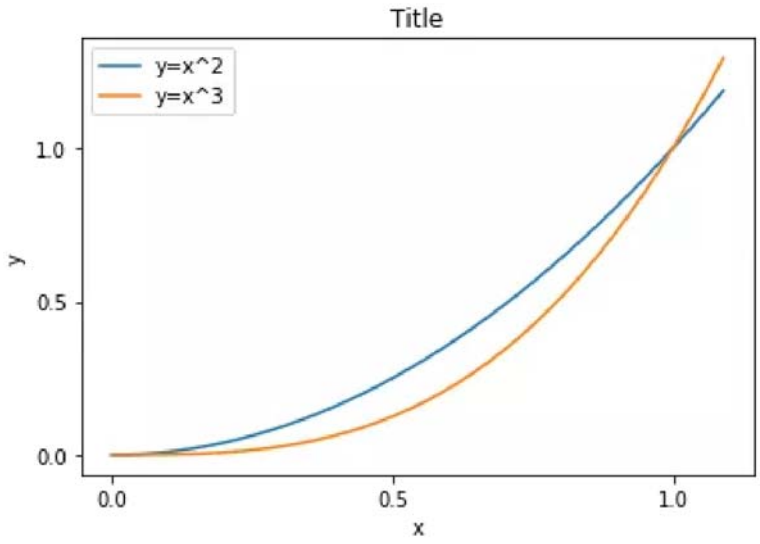
1.4 通过 add_subplot()方法添加和选中子图

```
# 引入matplotlib包
import matplotlib.pyplot as plt
import numpy as np
# 创建Figure实例
fig = plt.figure()
# 添加子图
fig.add_subplot(2, 2, 1)
fig.add_subplot(2, 2, 2)
fig.add_subplot(2, 2, 4)
fig.add_subplot(2, 2, 3)
# 在子图上作图
random_arr = np.random.randn(100)
# 默认是在最后一次使用subplot的位置上作图，即编号为3的位置
plt.plot(random_arr)
plt.show()
```



1.5 添加各类标签

```
import numpy as np
data = np.arange(0, 1.1, 0.01)
plt.title("Title")      # 添加标题
plt.xlabel("x")         # 添加x轴的名称
plt.ylabel("y")         # 添加y轴的名称
# 设置x和y轴的刻度
plt.xticks([0, 0.5, 1])
plt.yticks([0, 0.5, 1.0])
plt.plot(data, data**2)          # 绘制y=x^2曲线
plt.plot(data, data**3)          # 绘制y=x^3曲线
plt.legend(["y=x^2", "y=x^3"])   # 添加图例
plt.show()              # 在本机上显示图形
```

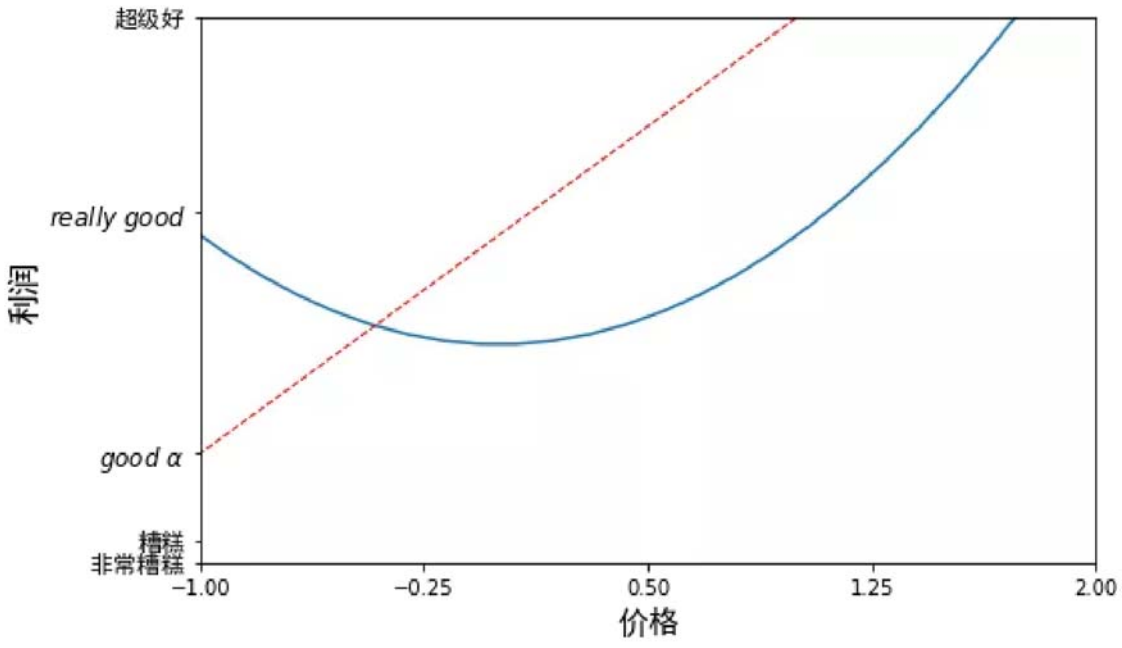



```
import numpy as np
x=np.linspace(-3,3,50)#产生-3到3之间50个点
y1=2*x+1#定义函数
y2=x**2
```

```
# num=3表示图片上方标题 变为figure3，figsize=(长，宽)设置figure大小
plt.figure(num=3, figsize=(8, 5))
plt.plot(x, y2)
# 红色虚线直线宽度默认1.0
plt.plot(x, y1, color='red', linewidth=1.0, linestyle='--')

plt.xlim((-1, 2))  #设置x轴范围
```

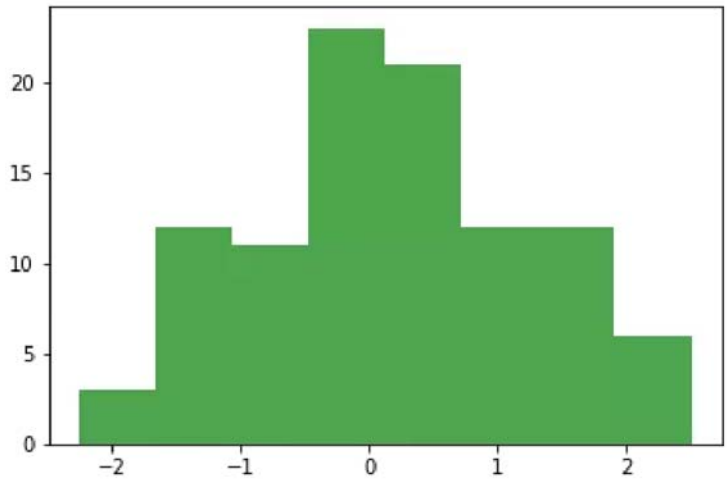
```
plt.ylim((-2, 3))  #设置轴y范围

#设置坐标轴含义，  注：英文直接写，中文需要后面加上fontproperties属性
plt.xlabel(u'价格', fontproperties='SimHei', fontsize=16)
plt.ylabel(u'利润', fontproperties='SimHei', fontsize=16)

# 设置x轴刻度
# -1到2区间，5个点，4个区间，平均分：[-1.,-0.25,0.5,1.25,2.]
new_ticks = np.linspace(-1, 2, 5)
print(new_ticks)
plt.xticks(new_ticks)

# 设置y轴刻度
'''
设置对应坐标用汉字或英文表示，后面的属性fontproperties表示中文可见，不乱码，
内部英文$$表示将英文括起来，r表示正则匹配，通过这个方式将其变为好看的字体
如果要显示特殊字符，比如阿尔法，则用转意符\alpha,前面的\ 表示空格转意
'''
plt.yticks([-2, -1.8, -1, 1.22, 3.],
           ['非常糟糕', '糟糕', r'$good\ \alpha$', r'$really\ good$', '超级好'],
           fontproperties='SimHei',
           fontsize=12)
plt.show()
```

```
[-1.   -0.25  0.5   1.25  2.  ]
```

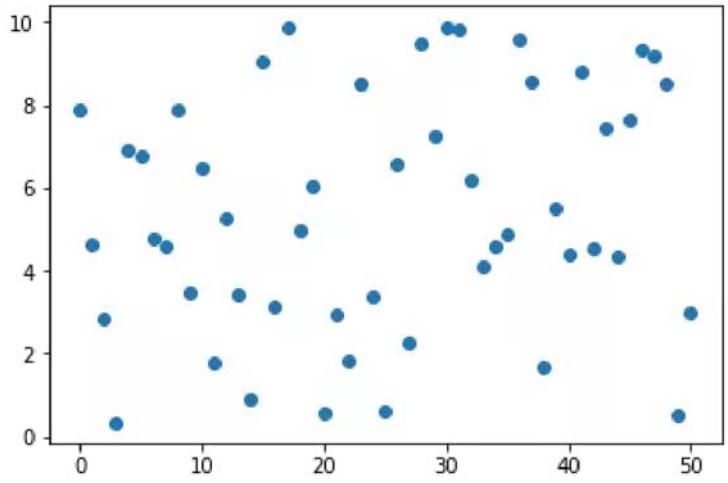


1.6 绘制常见类型图表

```
arr_random = np.random.randn(100)   # 创建随机数组
plt.hist(arr_random, bins=8, color='g', alpha=0.7) # 绘制直方图
plt.show()  # 显示图形
```

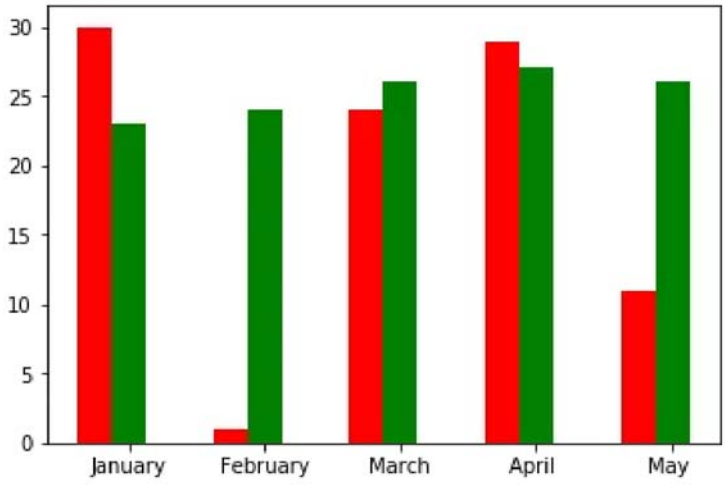



```
# 创建包含整数0~50的数组，用于表示x轴的数据
x = np.arange(51)
# 创建另一数组，用于表示y轴的数据
y = np.random.rand(51) * 10
plt.scatter(x, y)   # 绘制散点图
plt.show()
```

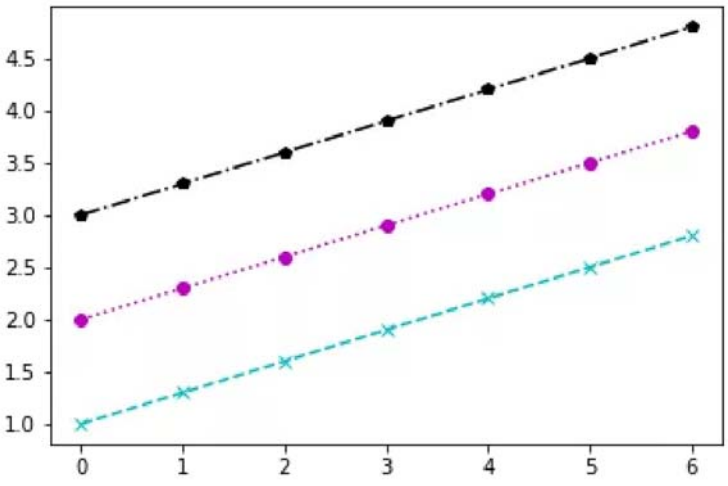



```
# 创建包含0~4的一维数组
x = np.arange(5)
# 从上下限范围内随机选取整数，创建两个2行5列的数组
y1, y2 = np.random.randint(1, 31, size=(2, 5))
width = 0.25                                # 条形的宽度
ax = plt.subplot(1, 1, 1)                # 创建一个子图
ax.bar(x, y1, width, color='r')          # 绘制红色的柱形图
```

```
ax.bar(x+width, y2, width, color='g')  # 绘制另一个绿色的柱形图
ax.set_xticks(x+width)                 # 设置x轴的刻度
# 设置x轴的刻度标签
ax.set_xticklabels(['January', 'February', 'March', 'April ', 'May '])
plt.show()                             # 显示图形
```




```
data = np.arange(1, 3, 0.3)
# 绘制直线，颜色为青色，标记为“x”，线型为长虚线
plt.plot(data, color="c", marker="x", linestyle="--")
# 绘制直线，颜色为品红，标记为实心圆圈，线型为短虚线
plt.plot(data+1, color="m", marker="o", linestyle=":")
# 绘制直线，颜色为黑色，标记为五边形，线型为短点相间线
plt.plot(data+2, color="k", marker="p", linestyle="-.")
# 也可采用下面的方式绘制三条不同颜色、标记和线型的直线
# plt.plot(data, 'cx--', data+1, 'mo:', data+2, 'kp-.')
plt.show()
```



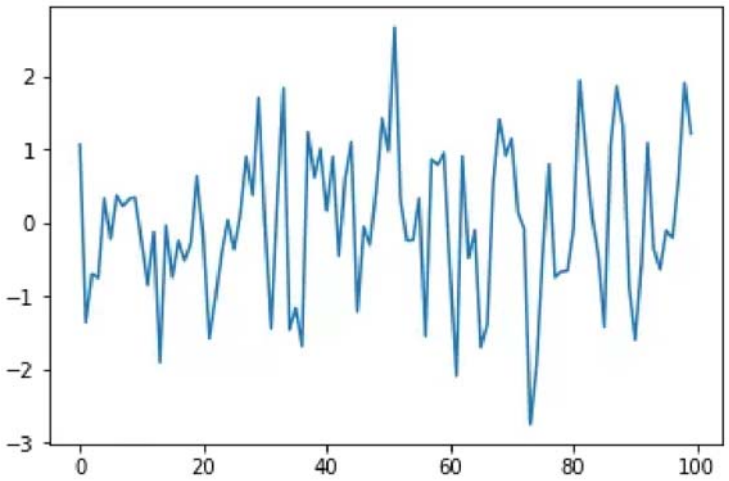
1.7 本地保存图形

```
# 创建包含100个数值的随机数组
import numpy as np
random_arr = np.random.randn(100)
```

```
random_arr
```

```
array([-2.02009735, -1.21035005,  0.57679581, -0.00584516,  0.59612158,
       -0.31118333, -0.67245832, -0.56589637,  0.25570972,  0.68256563,
       -0.45816656,  0.34956566,  0.51020863, -0.75272333,  1.42433863,
       -0.05658573,  0.35273745, -0.35519388,  0.15499307,  0.39895018,
       -1.86154032, -1.23949979, -0.63471999,  1.09811855,  0.02552633,
       -0.16804823,  0.34956809,  0.93485716,  0.37747537, -0.16523647,
       -1.04335227, -0.01702448,  1.60924259,  1.15294223, -0.15174045,
       -0.03772519,  1.090792  ,  0.65279282,  0.38186503, -1.3393988 ,
        0.10098444, -0.67411024, -2.39433996, -0.43594683, -0.155494  ,
        0.54676898, -0.97705035, -1.34799225,  1.64568965, -1.30594202,
       -0.30704745, -0.61612604,  1.09322798,  0.88921527, -0.22512233,
       -1.10477607, -0.61717627,  0.73952416,  0.30252205,  0.60808863,
       -0.3400892 , -2.01174842, -0.46480751,  1.54980369,  1.74610516,
       -0.53146867, -0.70904096,  1.73856111, -0.09254733,  0.43490467,
       -0.87201768, -0.73685075, -0.65868507, -0.18305015,  0.62559549,
        0.30743734, -0.78680136, -0.05808801, -0.23935035, -1.14580197,
        0.99154585,  0.07974613,  0.61315198,  0.93667393,  0.76542518,
        1.90500996,  0.0306359 , -2.53801425,  0.17371482,  1.75721226,
        0.25076371, -1.00032227,  0.20617839,  0.81751139,  0.64920089,
        1.3145223 ,  1.05360644,  2.06404062,  1.7208791 , -0.09375516])
```

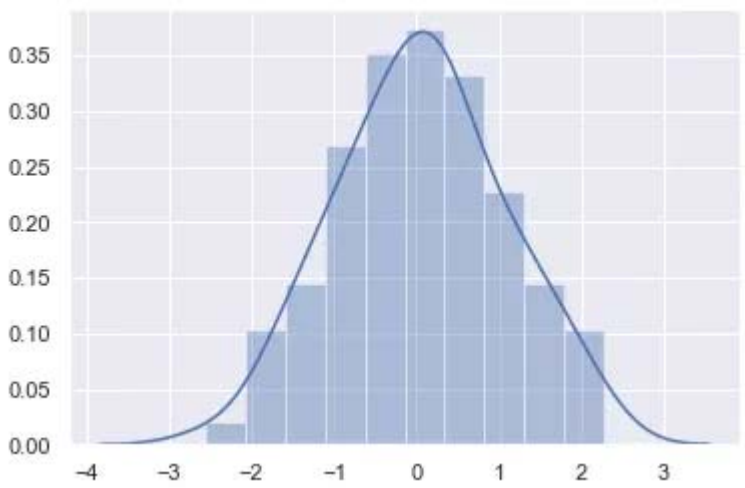
```
# 将随机数组的数据绘制线形图
plt.plot(random_arr)
plt.show()
```



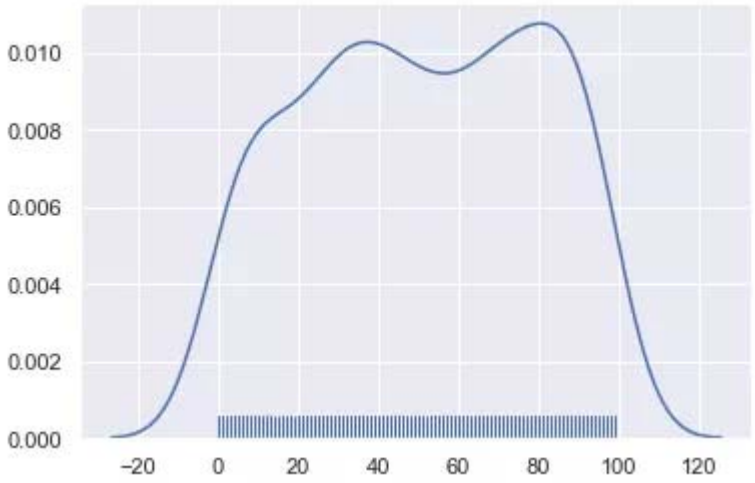
2 seaborn—绘制统计图形

2.1 可视化数据的分布

```
import seaborn as sns
%matplotlib inline
import numpy as np
sns.set()                          # 显式调用set()获取默认绘图
np.random.seed(0)                  # 确定随机数生成器的种子
arr = np.random.randn(100)         # 生成随机数组
ax = sns.distplot(arr, bins=10)    # 绘制直方图
```




```
# 创建包含500个位于[0，100]之间整数的随机数组
array_random = np.random.randint(0, 100, 500)
# 绘制核密度估计曲线
sns.distplot(array_random, hist=False, rug=True)
```



```
# 创建DataFrame对象
import pandas as pd
dataframe_obj = pd.DataFrame({"x": np.random.randn(500),"y": np.random.randn(500)})
dataframe_obj
```

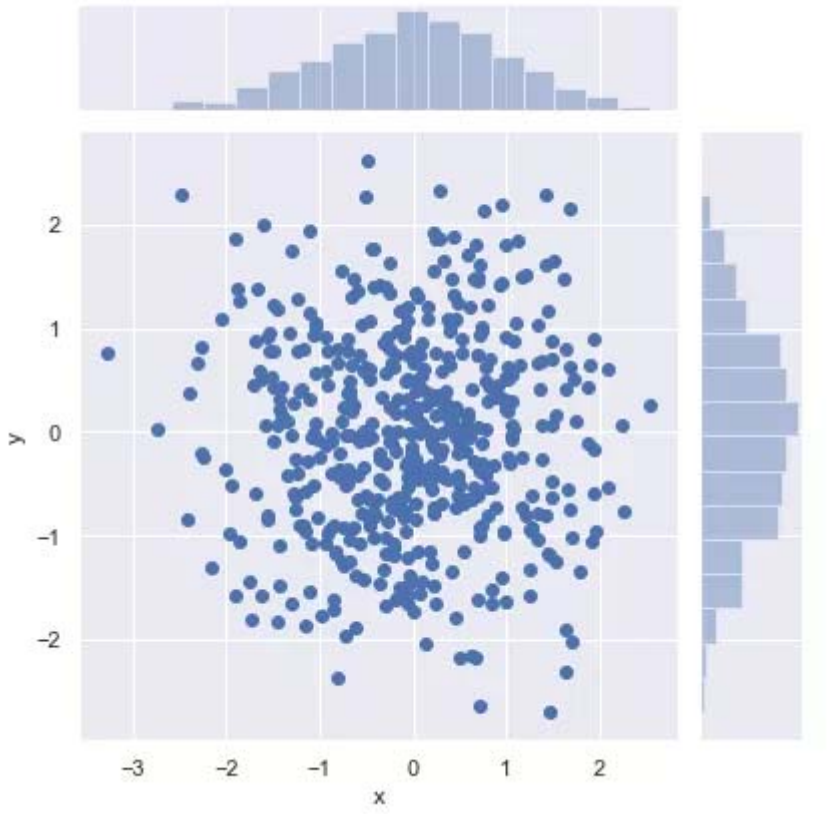
	x	y
0	0.478215	1.246931
1	-0.053906	0.187860
2	-1.241901	1.281412
3	-1.658495	1.375265
4	-0.353372	1.420608
5	1.656508	-0.557275
6	1.511913	1.657975
7	-0.906804	0.452821
8	-0.777217	-0.368433
9	-0.739228	-1.286740
10	0.987989	-1.634521
11	-0.026473	-0.010277
12	-1.262669	-0.256035
13	-1.561165	0.918040
14	-0.939354	-0.127256
15	0.335453	0.217671
16	-1.489752	0.432434
17	-1.066911	-0.515731
18	1.035863	-0.297603
19	0.631313	-0.653702
20	-1.894367	1.868757
21	0.036571	0.237410
22	-0.312502	-1.319956
23	0.814248	-0.811489

	x	y
24	0.382404	-0.449499
25	1.646666	0.410724
26	0.227553	0.313078
27	-1.399875	0.431041
28	-2.161313	-1.314429
29	0.280750	2.321291
...	...	...
470	-1.266559	-0.595866
471	-0.766566	0.096873
472	0.205730	-1.270893
473	-0.608373	-1.875642
474	-0.323170	0.336776
475	-1.615268	-1.565554
476	0.433679	1.887319
477	-0.217975	-0.728759
478	1.023324	0.201026
479	-0.134135	-0.746496
480	0.046724	1.299394
481	-0.595088	-0.641203
482	-1.949716	-0.520380
483	-0.530026	-0.348830
484	-1.060356	-0.013075
485	-0.908488	-0.981377
486	-0.034975	-1.450624
487	-1.426397	0.320157
488	-1.302537	1.746811
489	-1.190758	0.407325
490	-0.170543	0.311181

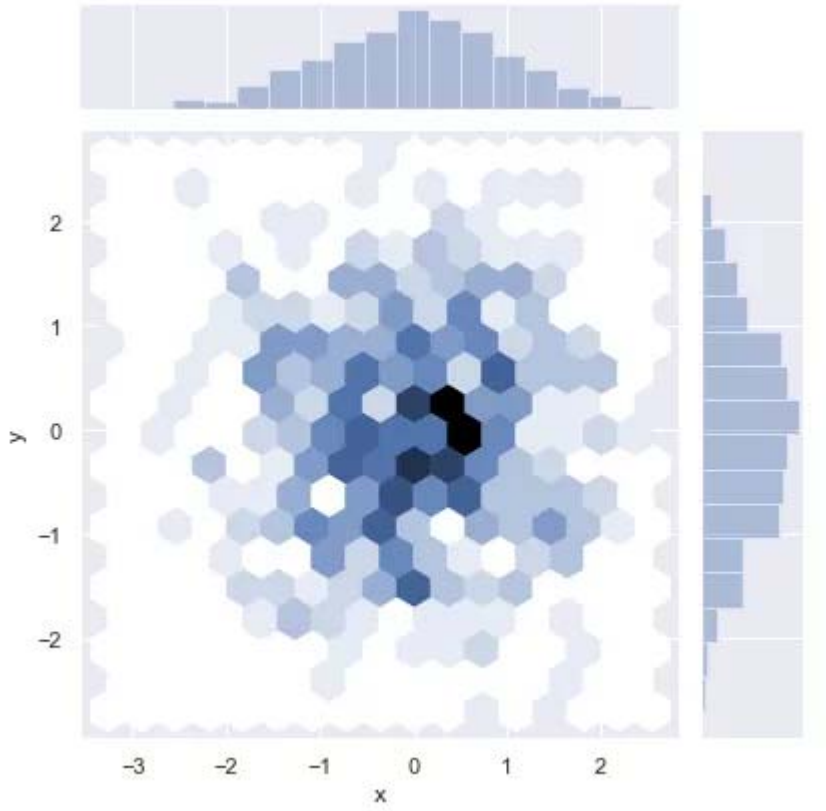
	x	y
491	0.814052	0.299761
492	-0.520146	0.591630
493	1.934602	-0.165131
494	-0.052196	-0.524848
495	-1.057486	0.939177
496	-0.158090	-1.588747
497	-0.238412	1.627092
498	0.279500	-0.218554
499	1.962078	-0.956771

500 rows × 2 columns

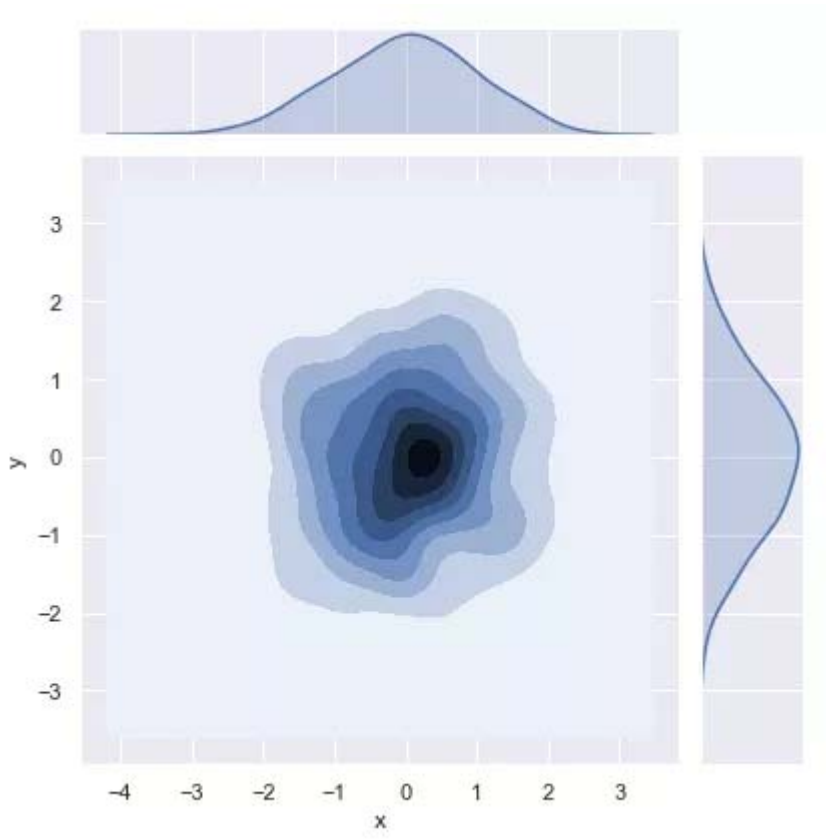
```
# 绘制散布图
sns.jointplot(x="x", y="y", data=dataframe_obj)
```



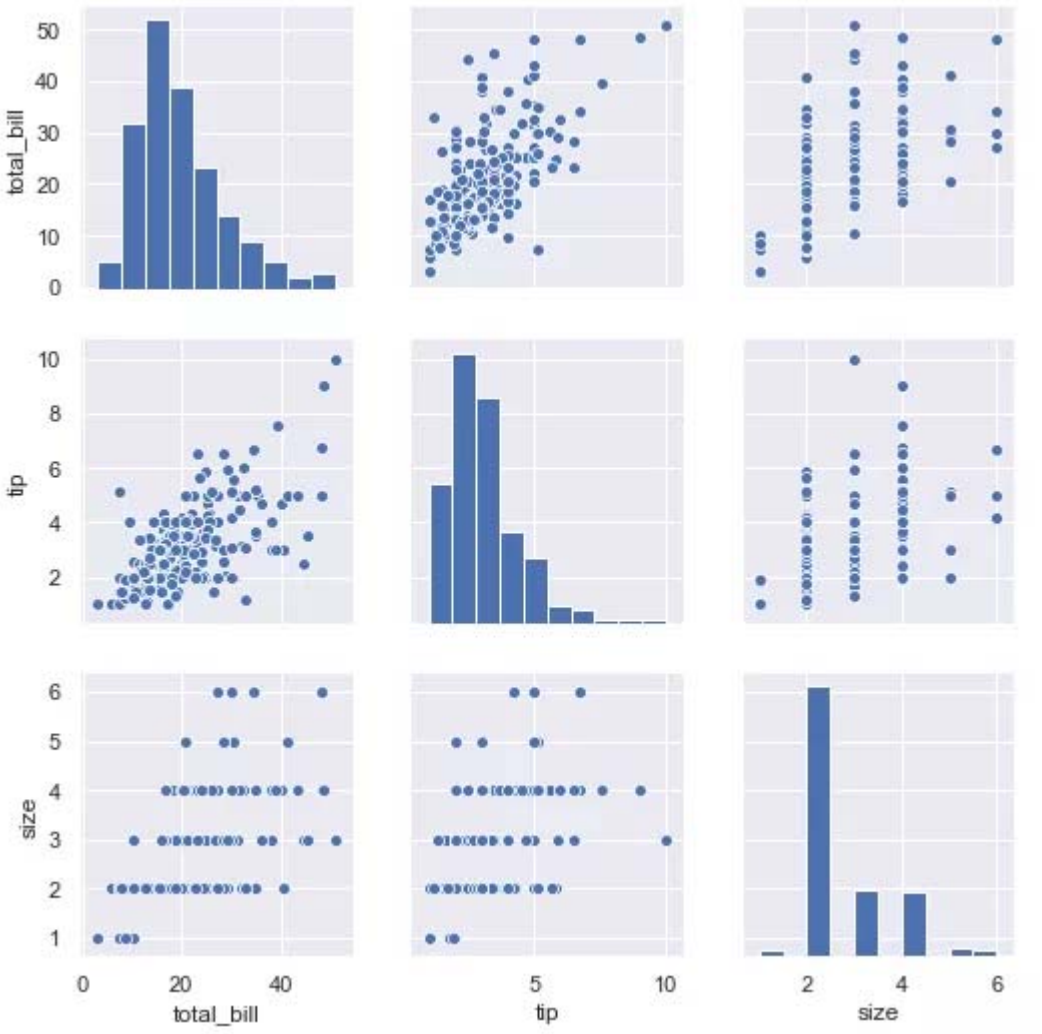

```
# 绘制二维直方图
sns.jointplot(x="x", y="y", data=dataframe_obj, kind="hex")
```




```
# 核密度估计
sns.jointplot(x="x", y="y", data=dataframe_obj, kind="kde")
```

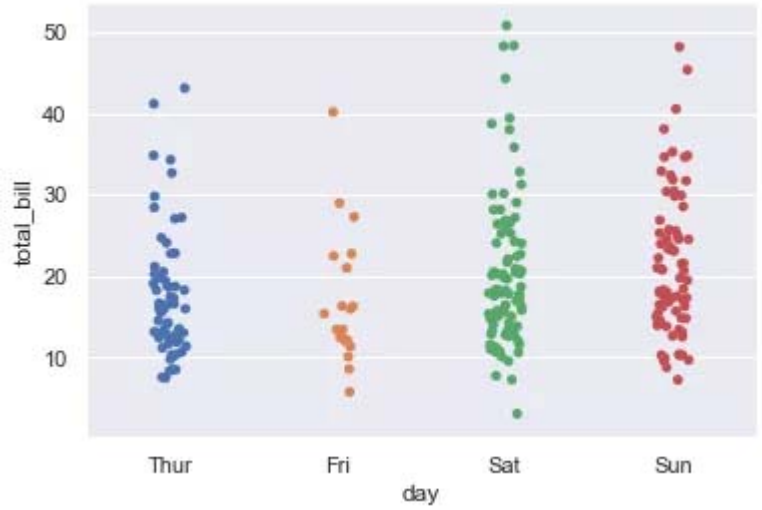


```
# 加载seaborn中的数据集
dataset = sns.load_dataset("tips")
# 绘制多个成对的双变量分布
sns.pairplot(dataset)
```

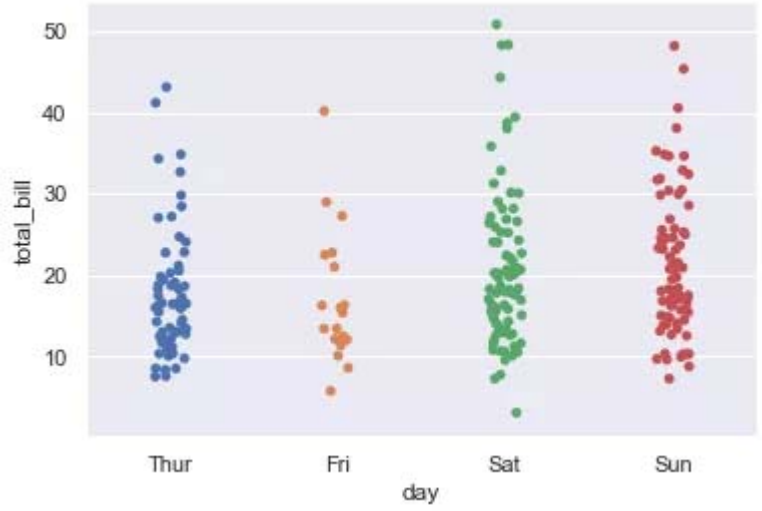


2.2 用分类数据绘图

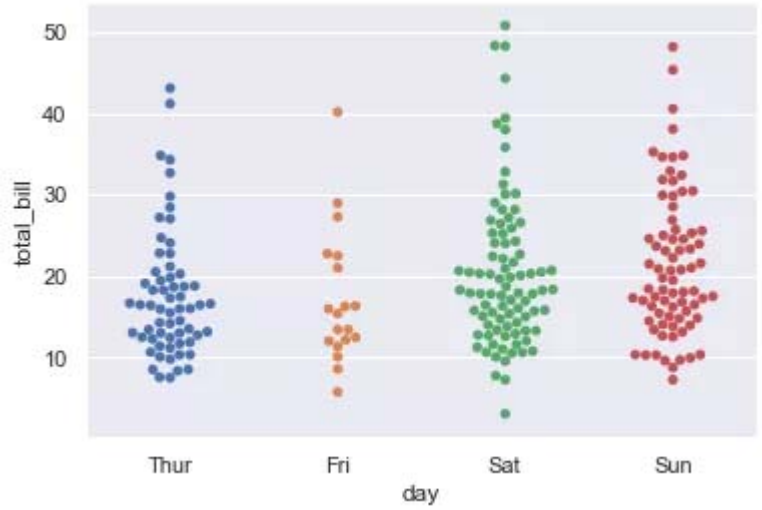
```
tips = sns.load_dataset("tips")
sns.stripplot(x="day", y="total_bill", data=tips)
```




```
tips = sns.load_dataset("tips")
sns.stripplot(x="day", y="total_bill", data=tips, jitter=True)
```

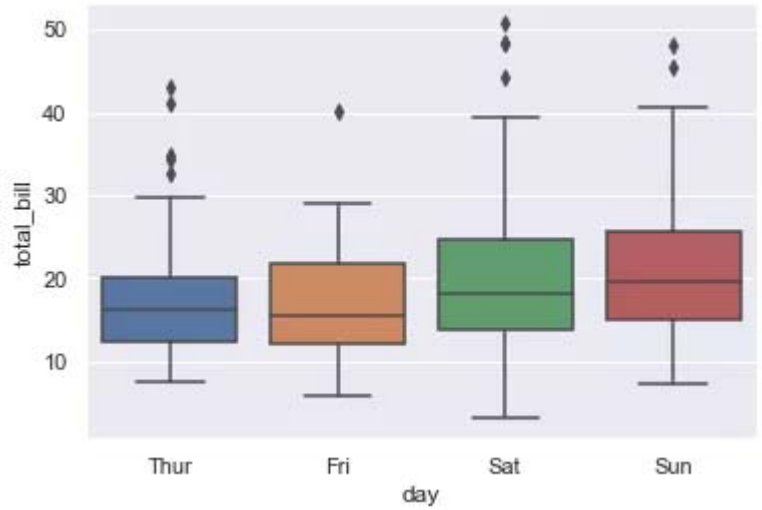



```
sns.swarmplot(x="day", y="total_bill", data=tips)
```

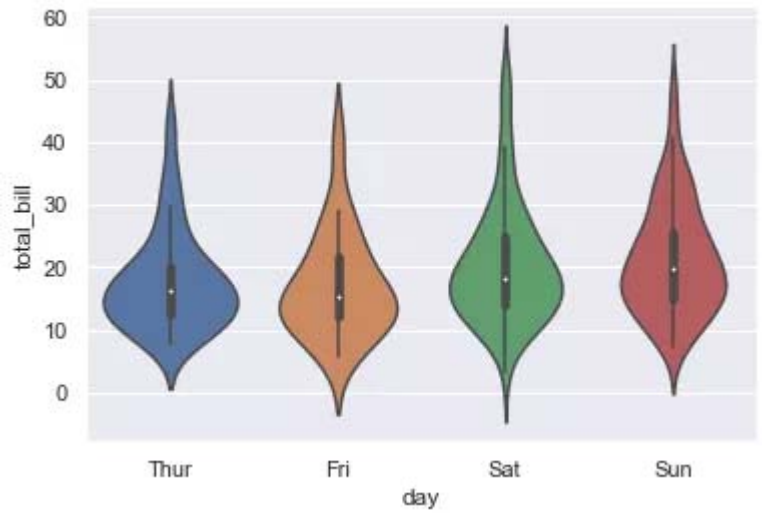



```
```

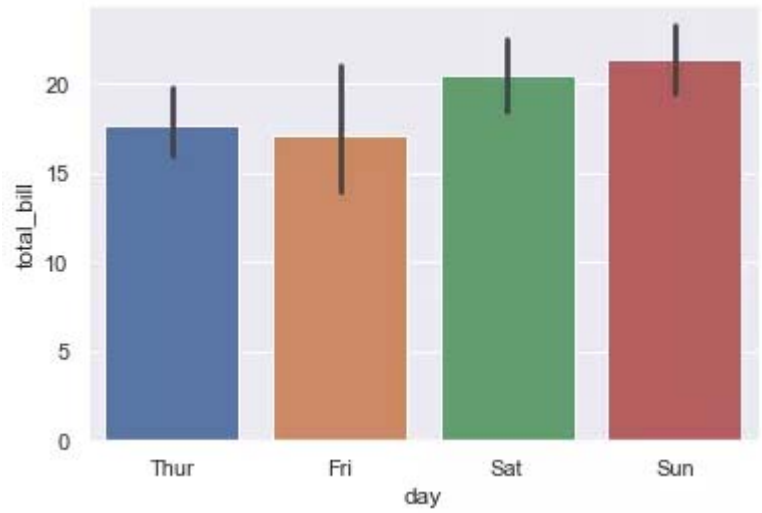
```
sns.boxplot(x="day", y="total_bill", data=tips)
```



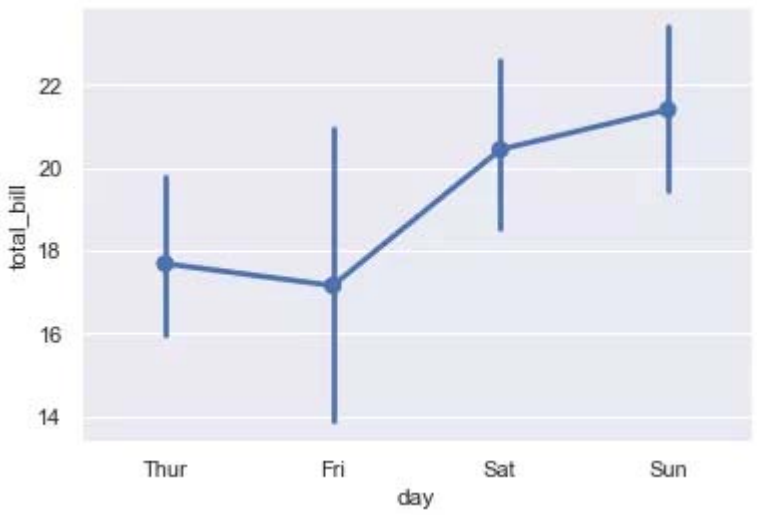

```
sns.violinplot(x="day", y="total_bill", data=tips)
```




```
sns.barplot(x="day", y="total_bill", data=tips)
```



```
sns.pointplot(x="day", y="total_bill", data=tips)
```



备注：本文代码可以在github下载

https://github.com/fengdu78/Data-Science-Notes/tree/master/5.data-visualization

往期精彩回顾

- 那些年做的学术公益-你不是一个人在战斗
- 适合初学者入门人工智能的路线及资料下载
- 机器学习在线手册
- 深度学习在线手册

备注：加入本站微信群或者qq群，请回复“加群”

加入知识星球（4500+用户，ID：92416895），请回复“知识星球”