

20

Visualizations in Pandas

Pandas 快速可视化

线图、子图、散点图、柱状图、箱型图...



善良一点，因为你遇到的每个人都在打一场更艰苦的战斗。

Be kind, for everyone you meet is fighting a harder battle.

—— 柏拉图 (Plato) | 古希腊哲学家 | 424/423 ~ 348/347 BC



- ▶ `pandas.DataFrame.plot()` 绘制线图
- ▶ `pandas.DataFrame.plot.area()` 绘制面积图
- ▶ `pandas.DataFrame.plot.bar()` 绘制柱状图
- ▶ `pandas.DataFrame.plot.barh()` 绘制水平柱状图
- ▶ `pandas.DataFrame.plot.box()` 绘制箱型图
- ▶ `pandas.DataFrame.plot.density()` 绘制 KDE 线图
- ▶ `pandas.DataFrame.plot.hexbin()` 绘制六边形图
- ▶ `pandas.DataFrame.plot.hist()` 绘制直方图
- ▶ `pandas.DataFrame.plot.kde()` 绘制 KDE 线图
- ▶ `pandas.DataFrame.plot.line()` 绘制线图
- ▶ `pandas.DataFrame.plot.pie()` 绘制饼图
- ▶ `pandas.DataFrame.plot.scatter()` 绘制散点图
- ▶ `pandas.plotting.scatter_matrix()` 成对散点图矩阵



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

20.1 Pandas 的可视化功能

Pandas 库本身虽然主要用于数据处理和分析，但也提供了一些基本的可视化功能。这一章介绍如何用 Pandas 绘制折线图、散点图、面积图、柱状图、箱型图等等。

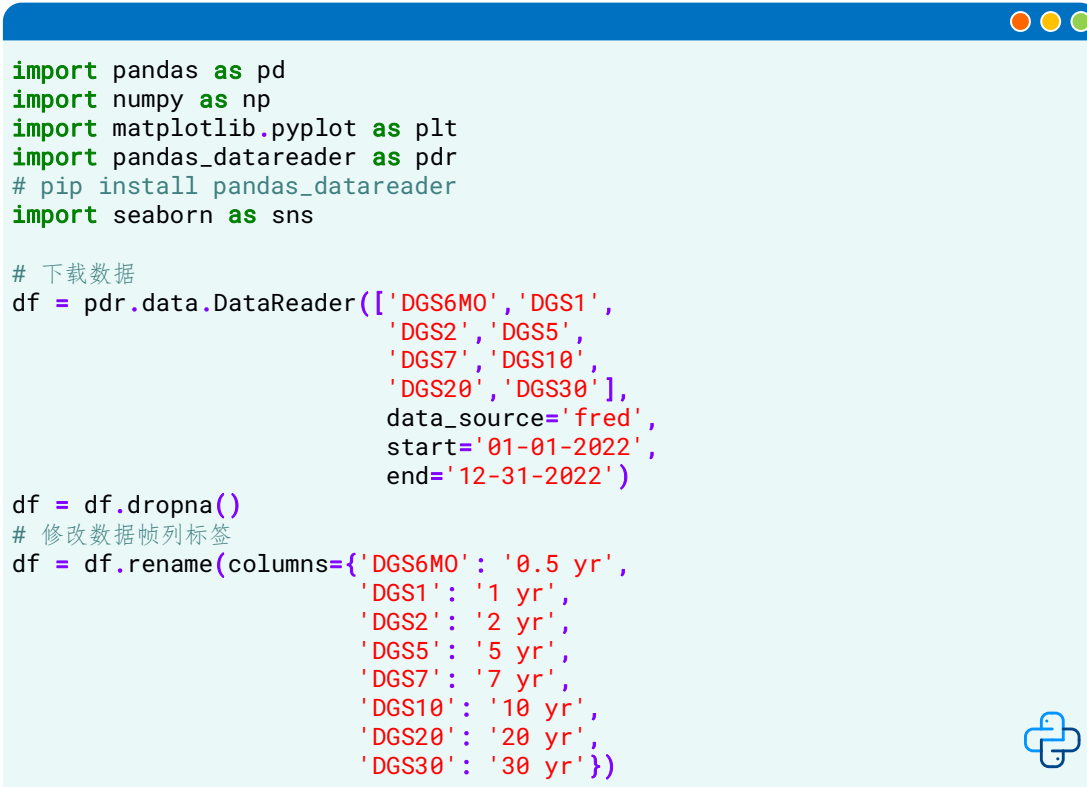
本书第 12 章介绍过如何用 Seaborn 完成各种统计描述可视化，请大家回顾。

本章用的是利率数据。^a 导入 pandas_datareader，简做 pdr。pandas_datareader 从多种数据源获取金融和经济数据，并将这些数据转换为 Pandas DataFrame 的形式。要想使用这个库，先需要安装。如^b所示，本书前文提过，在 Anaconda prompt 使用 `pip install pandas_datareader` 安装这个库。

^c 导入 seaborn，简做 sns。

^d 利用 pandas_datareader 从 FRED (Federal Reserve Economic Data) 下载利率数据，数据格式为 Pandas 数据帧。


^e 用 `dropna()` 删除数据帧中 NaN。^f 用 `rename()` 修改数据帧列标签。



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
a import pandas_datareader as pdr
b # pip install pandas_datareader
c import seaborn as sns

# 下载数据
d df = pdr.data.DataReader(['DGS6M0', 'DGS1',
                           'DGS2', 'DGS5',
                           'DGS7', 'DGS10',
                           'DGS20', 'DGS30'],
                           data_source='fred',
                           start='01-01-2022',
                           end='12-31-2022')

e df = df.dropna()
# 修改数据帧列标签
f df = df.rename(columns={'DGS6M0': '0.5 yr',
                           'DGS1': '1 yr',
                           'DGS2': '2 yr',
                           'DGS5': '5 yr',
                           'DGS7': '7 yr',
                           'DGS10': '10 yr',
                           'DGS20': '20 yr',
                           'DGS30': '30 yr'})
```

图 1. 下载分析利率数据，代码； Bk1_Ch20_01.ipynb

20.2 线图: `pandas.DataFrame.plot()`

图 2 所示为在一张图上展示利率数据的线图。我们还可以用如图 3 子图方案分别展示每条曲线。图 4 代码绘制图 2 和图 3，下面聊聊其中关键语句。

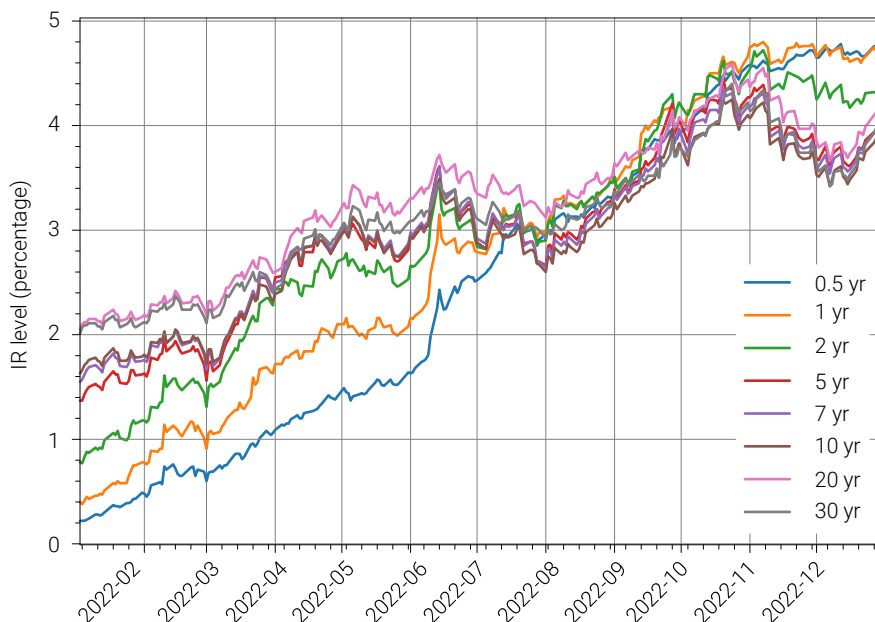


图 2. 利率时间数据线图

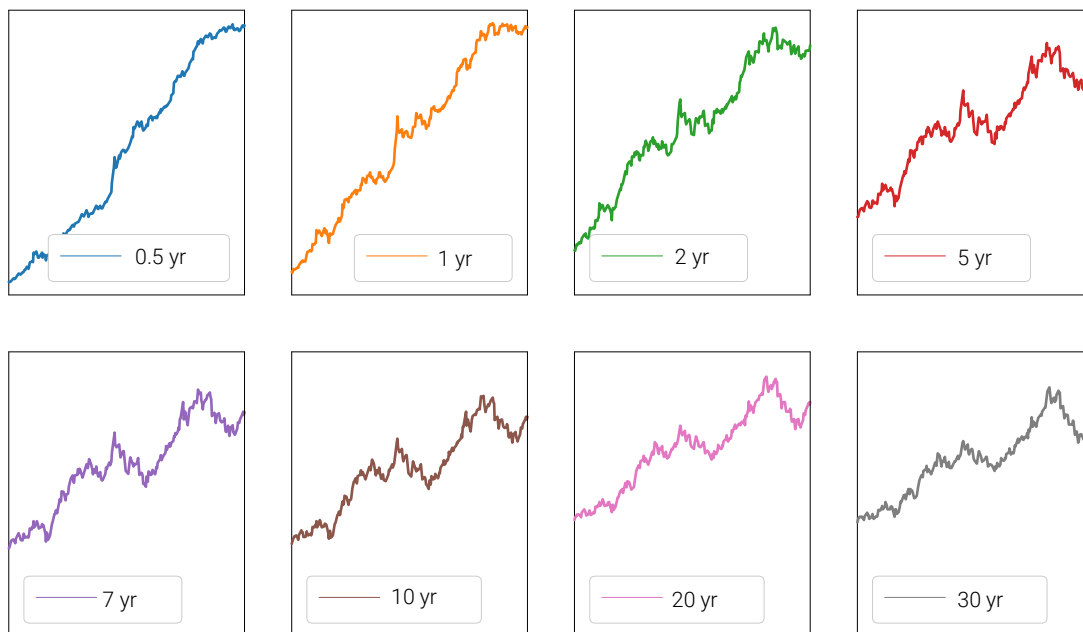


图 3. 利率时间数据线图，子图

图 4 代码中^a，对于 Pandas 数据帧，我们可以直接用 `.plot()` 方法绘制线图。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载: <https://github.com/Visualize-ML>

本书配套微视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱: jiang.visualize.ml@gmail.com

用 `xlabel`、`ylabel` 可以修改线图横纵坐标标签。

参数 `legend = True`（默认值），在图表显示图例；而 `legend = False` 则不显示图例。

b 将图片以 SVG 格式保存。


c 绘制线图时创建一个复杂的子图布局。其中，参数 `subplots=True`，表示我们要创建多个子图，每个子图将显示 `DataFrame` 中的不同列。

参数 `layout = (2, 4)` 指定子图的布局，表示总共有 2 行和 4 列子图，共计 8 个子图。这意味着 `DataFrame` 中的 8 个不同数据列将分别在这 8 个子图中显示，具体如图 3 所示。

参数 `sharex=True` 和 `sharey=True`，表示所有子图将共享相同的 x 轴和 y 轴。这意味着所有子图的 x 轴范围和 y 轴范围将相同，以便更容易比较子图之间的数据。

参数 `xticks=[]` 和 `yticks=[]` 设置 x 轴和 y 轴的刻度标签为空列表，意味着不显示。

参数 `xlim = (df.index.min(), df.index.max())` 指定了 x 轴的显示范围，即 x 轴的最小值和最大值。`df.index.min()` 取出时间序列数据帧横轴标签的最小值，`df.index.max()` 时间序列数据帧横轴标签的最大值。



```
# 绘制利率走势图
a df.plot(xlabel="Time", ylabel="IR level",
          legend = True,
          xlim = (df.index.min(), df.index.max()))

b plt.savefig("利率走势图.svg")

# 绘制利率走势图，子图布置
c df.plot(subplots=True, layout=(2, 4),
          sharex = True, sharey = True,
          xticks = [], yticks = [],
          xlim = (df.index.min(), df.index.max()))

d plt.savefig("利率走势图, 子图.svg")
```

图 4. 绘制线图，使用时配合前文代码；  Bk1_Ch20_01.ipynb

为了更好地美化线图，我们还可以使用图 5 代码。**a** 用 `plt.subplots(figsize=(5, 5))` 创建一个图形对象和一个轴对象。

其中，`fig` 是一个图形对象，它代表整个图形窗口。我们可以在这个图形对象上添加一个或多个轴对象，以在图形窗口上绘制图表。

而 `ax` 是一个轴对象，它代表图形窗口中的一个子图或一个坐标系。我们可以在轴对象上绘制数据，并设置轴的属性，如图表大小、标题等。

b 在对数据帧使用 `.plot()` 方法时，用 `ax = ax` 指定了具体轴对象。然后，可以用各种方法美化轴对象 `ax`。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

```
# 美化线图
```

```
a fig, ax = plt.subplots(figsize = (5,5))
b df.plot(ax = ax, xlabel="Time", legend = True)
ax.set_xlim((df.index.min(), df.index.max()))
ax.set_ylim((0,5))
ax.set_xticks([])
ax.set_xlabel('Time')
ax.set_ylabel('IR level')
```

图 5. 绘制并美化线图, 使用时配合前文代码; Bk1_Ch20_01.ipynb

如图 6 所示, 我们还可以用 `df.plot.area()` 绘制面积图, 对应代码为图 7, 请大家自行分析注释代码。

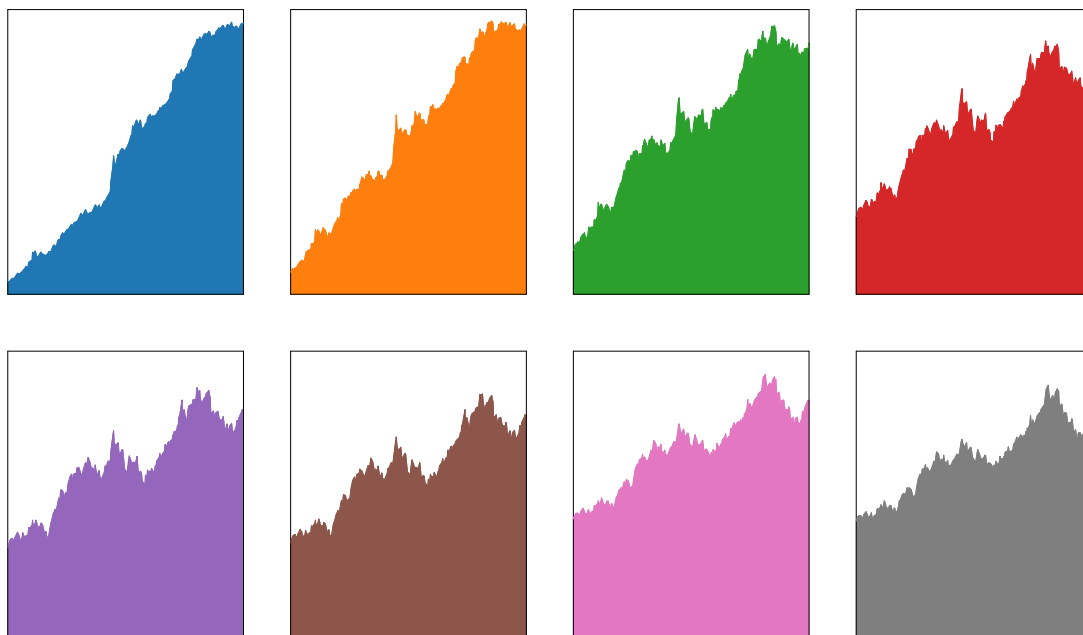


图 6. 利率时间数据面积图, 子图

```
# 绘制利率走势线面积图, 子图布置
```

```
a df.plot.area(subplots=True, layout=(2,4),
               sharex = True, sharey = True,
               xticks = [], yticks = [],
               xlim = (df.index.min(), df.index.max()),
               ylim = (0,5), legend = False)

plt.savefig("利率走势面积图, 子图.svg")
```


图 7. 绘制面积图, 使用时配合前文代码;  Bk1_Ch20_01.ipynb

图 8 所示为利率日收益率折线图, 采用 2×4 子图布局。日收益率是用来衡量金融数据在一天内的水平变动幅度的指标。日收益率通常以百分比形式表示, 计算方法为: $\text{日收益率} = (\text{当日收盘价} - \text{前一日收盘价}) / \text{前一日收盘价} \times 100\%$ 。

图 9 代码计算日收益率, 并绘制图 8。[a](#) 用 `pct_change()` 计算日收益率。[b](#) 也是用 `.plot()` 方法绘制收益率线图八幅子图, 请大家自行完成注释。

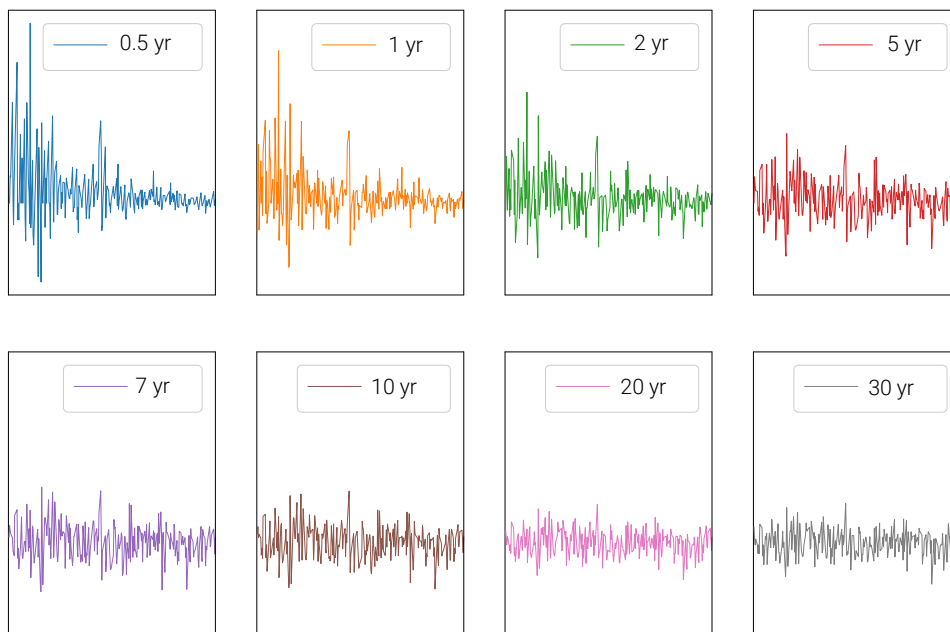


图 8. 利率日收益率折线图, 子图

```
# 计算日收益率
a r_df = df.pct_change()

# 绘制利率日收益率, 子图布置
b r_df.plot(subplots=True, layout=(2,4),
            sharex = True, sharey = True,
            xticks = [], yticks = [],
            xlim = (df.index.min(), df.index.max()))

plt.savefig("利率日收益率走势图, 子图.svg")
```

图 9. 绘制日收益率, 使用时配合前文代码;  Bk1_Ch20_01.ipynb

20.3 散点图

图 10 所示为利用 `pandas.DataFrame.plot.scatter()` 绘制散点图。图 11 代码^a 用参数 `x="1 yr"` 和 `y="2 yr"` 指定散点图中要使用的数据列。

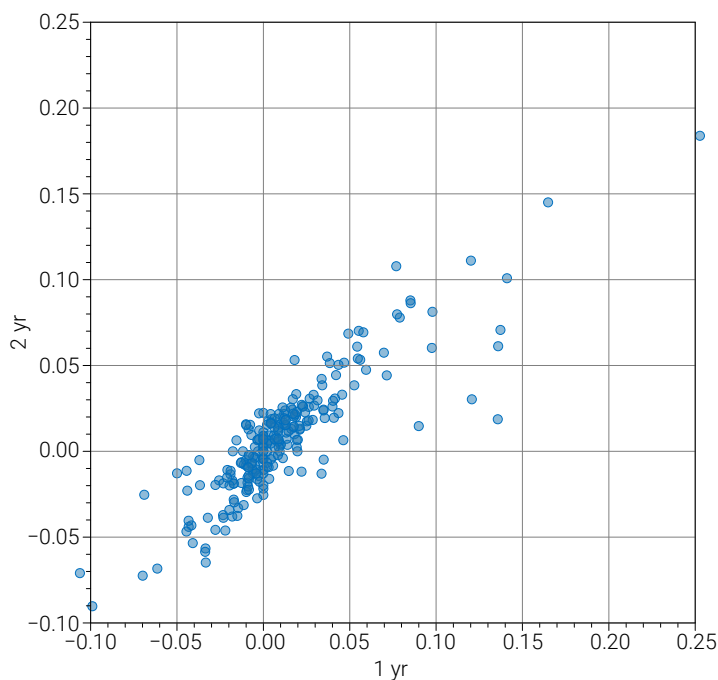


图 10. 日收益率散点图

```
# 绘制散点图
fig, ax = plt.subplots(figsize = (5,5))
a r_df.plot.scatter(x="1 yr", y="2 yr",
                  ax = ax)

a ax.set_xlim(-0.1, 0.25)
    ax.set_ylim(-0.1, 0.25)
    plt.savefig("散点图.svg")
```


图 11. 绘制散点图，使用时配合前文代码；  Bk1_Ch20_01.ipynb

图 12 所示为用 `pandas.DataFrame.plot.hexbin()` 绘制的六边形图，用来可视化两个变量之间的分布情况。这幅图的功能类似于直方热图。

图 12 代码^a 中参数 `cmap="RdYlBu_r"` 指定了用于着色六边形的色谱/颜色映射。"RdYlBu_r" 是一种颜色映射的名称，它表示一种颜色渐变，从红色到黄色到蓝色，"_r" 表示颜色映射的反转。

参数 `gridsize=15` 指定六边形数量。

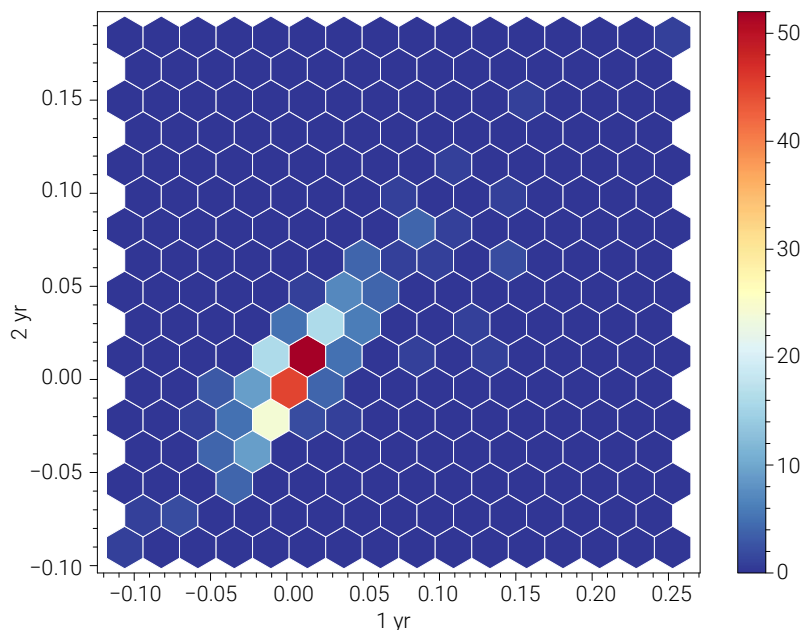


图 12. 六边形图

```
# 六边形图
a r_df.plot.hexbin(x="1 yr", y="2 yr",
                  gridsize = 15,
                  cmap="RdYlBu_r")
plt.savefig("六边形图.svg")
```

图 13. 绘制六边形图，使用时配合前代码； Bk1_Ch20_01.ipynb

图 11 这幅平面散点图展示的一对特征的关系。当然利用 `matplotlib.pyplot.scatter()`、`seaborn.scatterplot()`、`plotly.express.scatter()`，我们可以通过散点的颜色、大小、`marker` 展示几个其他特征。问题是，当数据特征（维度）进一步增大，这种一对一平面散点图的能力很局限了。

图 14 而所示的成对散点图，或成对散点图矩阵，则可以展示多特征数据的任意一对特征的关系。

代码图 15 中，^a 从 `pandas.plotting` 模块导入 `scatter_matrix()`。

^b 用 `scatter_matrix()` 绘制成对散点图矩阵。

参数 `alpha` 可以控制散点填充颜色的透明度。

参数 `figsize` 可以用来控制图片大小。

参数 `diagonal` 则可以用来选择对角线图像的类型，'kde' 代表核密度估计（Kernel Density Estimation, KDE）。

实践中，`seaborn.pairplot()` 和 `plotly.express.scatter_matrix()` 在绘制成对散点图矩阵的效果更好。特别是，`plotly.express.scatter_matrix()` 可视化的结果是交互式的。

为了更好地量化特征之间的相关性，我们可以计算相关性系数矩阵，并且用热图可视化。

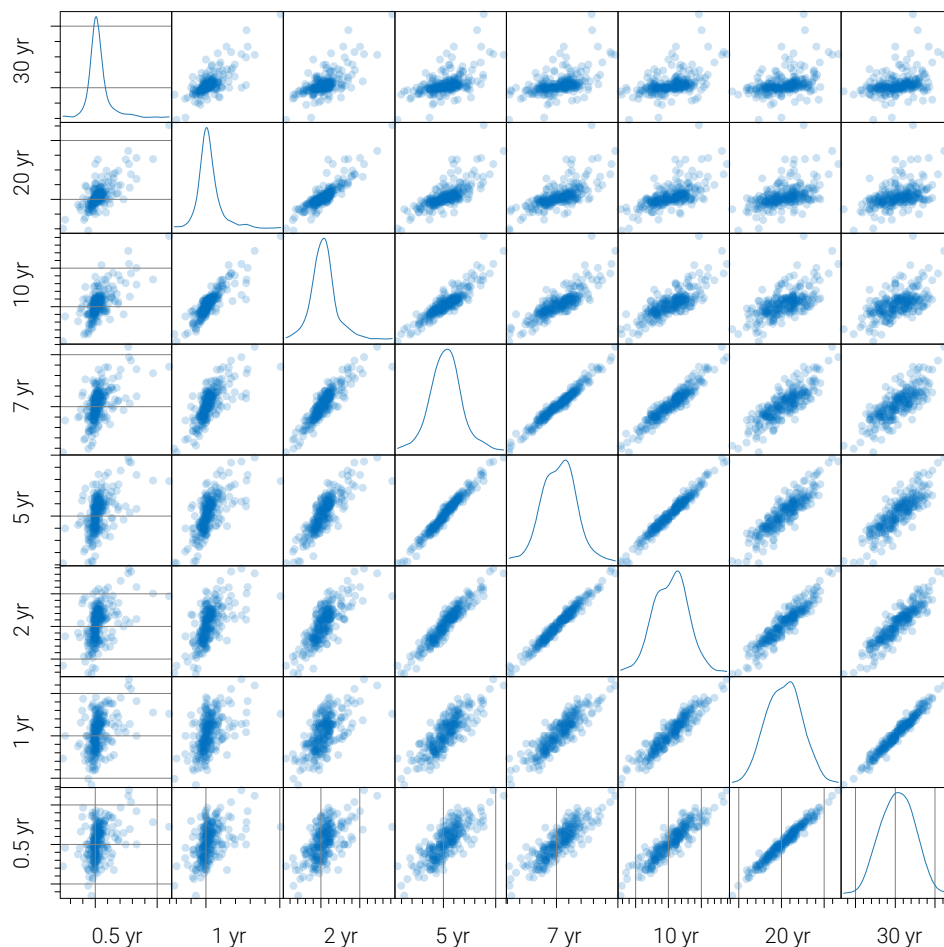


图 14. 成对特征散点图矩阵

```
# 绘制成对特征散点图
a from pandas.plotting import scatter_matrix
b scatter_matrix(r_df, alpha=0.2,
                figsize=(6, 6),
                diagonal="kde")
plt.savefig("成对特征散点图.svg")
```

图 15. 绘制成对特征散点图，使用时配合前文代码；  Bk1_Ch20_01.ipynb

20.4 柱状图

图 16 (a) 所示为用 `pandas.DataFrame.plot.bar()` 绘制的竖直柱状图 (vertical bar chart)。图 16 (b) 所示为用 `pandas.DataFrame.plot.barh()` 绘制的水平柱状图 (horizontal bar chart)。实践时，我们常用 `matplotlib.pyplot.bar()`、`seaborn.barplot()`、`plotly.express.bar()` 绘制柱状图。

请大家自行分析图 17 代码，此外请大家自行绘制标准差的竖直、水平柱状图。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

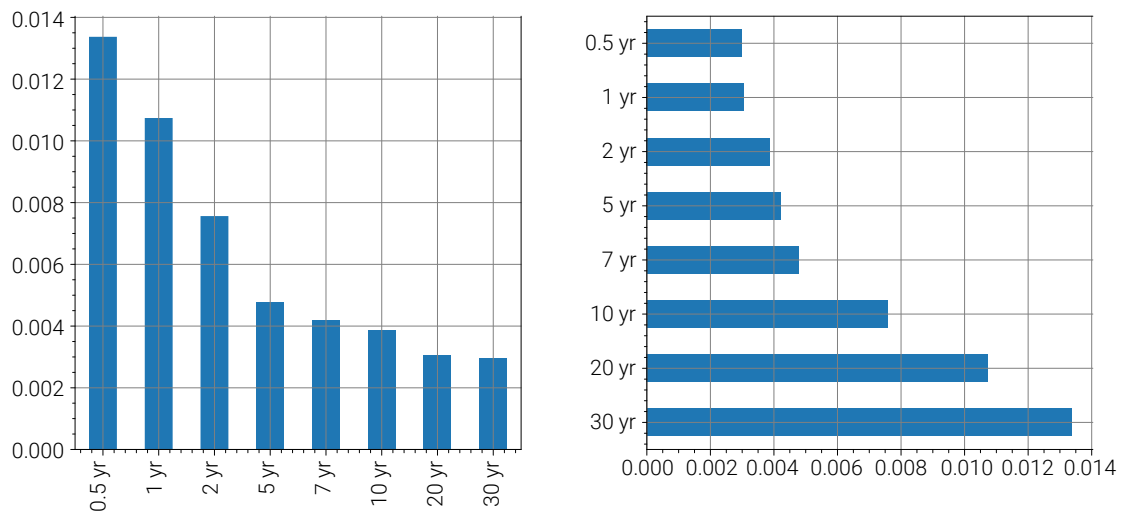


图 16. 柱状图

```

## 柱状图，竖直
a r_df.mean().plot.bar()
  plt.savefig("柱状图.svg")

## 柱状图，水平
b r_df.mean().plot.barh()
  plt.savefig("水平柱状图.svg")

```

图 17. 绘制柱状图，使用时配合前文代码； Bk1_Ch20_01.ipynb

20.5 箱型图

图 18 所示为利用 `pandas.DataFrame.boxplot()` 绘制的箱型图。简单来说，箱型图 (box plot) 是一种用于可视化数据分布的统计图表。它提供了一些关于数据集的重要统计信息 (四分位)，并帮助观察数据的离散程度以及潜在的异常值。请大家自行分析图 19 代码。

本书第 12 章介绍过如何用 `seaborn.boxplot()` 绘制箱型图，还介绍了箱型图的基本原理，请大家回顾。

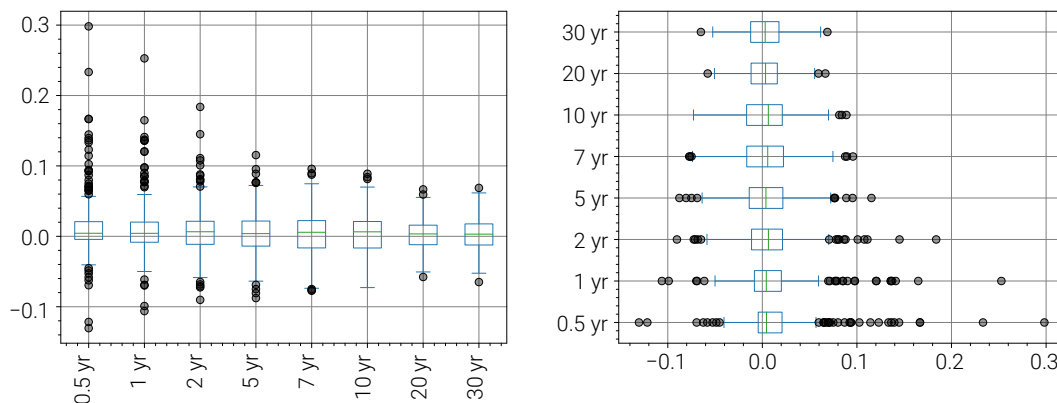


图 18. 箱型图

```
# 绘制箱型图
a r_df.plot.box()
plt.savefig("利率日收益率箱型图.svg")

# 绘制箱型图，水平
b r_df.plot.box(vert=False)
plt.savefig("利率日收益率箱型图，水平.svg")
```

图 19. 绘制箱型图，使用时配合前文代码； Bk1_Ch20_01.ipynb

20.6 直方图和核密度估计曲线

直方图 (histogram) 是一种用于可视化数据分布的图表，它将数据集分成不同的区间 (柱子)，并用柱子的高度表示每个区间中样本点频数 (frequency)、概率、概率密度。直方图有助于理解数据的分布形状、中心趋势和离散程度，以及检测数据中的模式和异常值。

图 20 所示直方图的纵轴为频数，也叫计数 (count)。图 20 每个子图中柱子高度之和为样本数据样本数。实践中，我们常用 `seaborn.histplot()`、`plotly.express.histogram()` 绘制直方图。

图 21 所示为高斯核密度估计曲线，每个子图的纵轴为概率密度，曲线和横轴围成面积为 1。从图像上来看，图 20 柱子是离散的，“平滑”后的结果就是图 21。我们常用 `seaborn.kdeplot()` 绘制 KDE 曲线。

鸢尾花书《统计至简》将会专门介绍高斯核密度估计。本书第 27 章会介绍如何使用 `Statsmodels` 中的核密度估计工具。

图 22 代码绘制图 20 和图 21，请大家逐行注释。

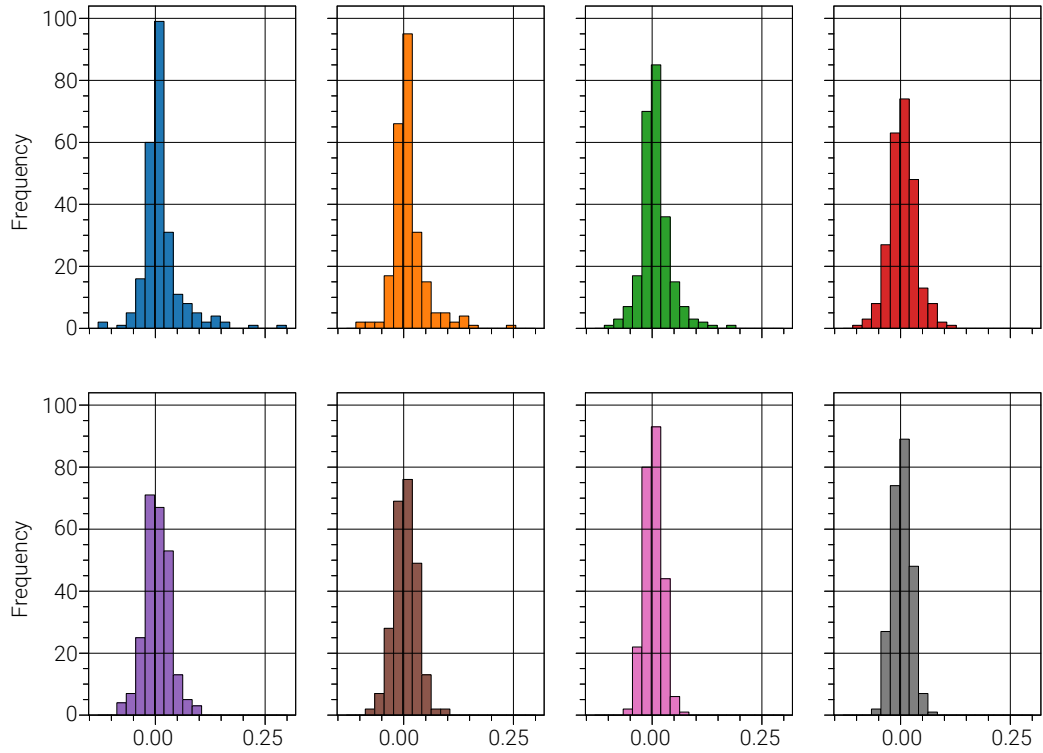


图 20. 直方图，子图布局

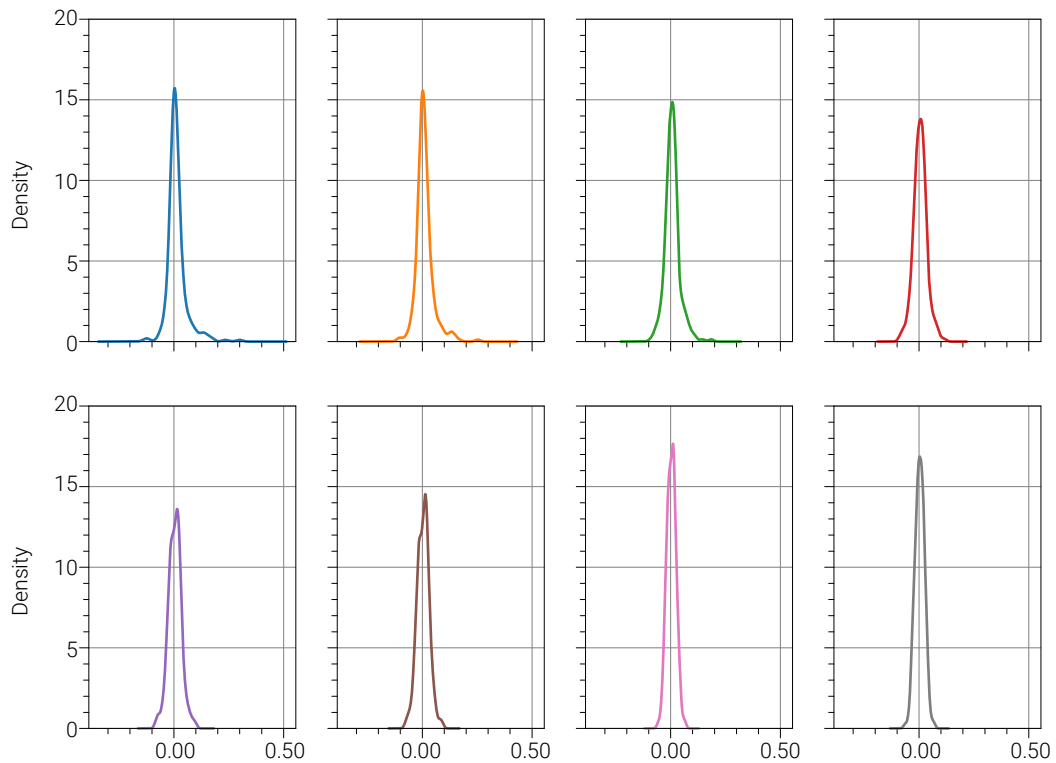


图 21. 高斯核密度估计，子图布局

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

```


# 直方图, 子图布置
a r_df.plot.hist(subplots=True, layout=(2,4),
                 sharex = True, sharey = True,
                 bins = 20,
                 legend = False)

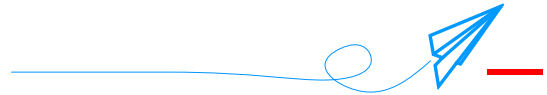
plt.savefig("利率日收益率直方图, 子图.svg")

# KDE, 子图布置
b r_df.plot.kde(subplots=True, layout=(2,4),
               sharex = True, sharey = True,
               ylim = (0,20),
               legend = False)

plt.savefig("利率日收益率KDE, 子图.svg")

```

图 22. 绘制直方图和高斯核密度估计, 使用时配合前文代码;  Bk1_Ch20_01.ipynb



本章介绍的是一些 Pandas 库中常用的可视化函数, 通过这些函数, 我们可以在数据分析过程中快速生成各种类型的图表以更好地理解数据。如果需要更复杂的可视化, 通常还需要使用 Matplotlib、Seaborn、Plotly 或其他专门的可视化库。