

19

Fundamentals of Pandas

聊聊 Pandas

Pandas DataFrame 类似 Excel 表格，有行列标签



数字是知识的终极形态；数字就是知识本身。

Numbers are the highest degree of knowledge. It is knowledge itself.

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



- ◀ `pandas.DataFrame()` 创建 Pandas 数据帧
- ◀ `pandas.DataFrame.add_prefix()` 给 DataFrame 的列标签添加前缀
- ◀ `pandas.DataFrame.add_suffix()` 给 DataFrame 的列标签添加后缀
- ◀ `pandas.DataFrame.axes` 同时获得数据帧的行标签、列标签
- ◀ `pandas.DataFrame.columns` 查询数据帧的列标签
- ◀ `pandas.DataFrame.corr()` 计算 DataFrame 中列之间 Pearson 相关系数 (样本)
- ◀ `pandas.DataFrame.count()` 返回数据帧每列 (默认 `axis=0`) 非缺失值数量
- ◀ `pandas.DataFrame.cov()` 计算 DataFrame 中列之间的协方差矩阵 (样本)
- ◀ `pandas.DataFrame.describe()` 计算 DataFrame 中数值列的基本描述统计信息，如平均值、标准差、分位数等
- ◀ `pandas.DataFrame.drop()` 用于从 DataFrame 中删除指定的行或列
- ◀ `pandas.DataFrame.head()` 用于查看数据帧的前几行数据，默认情况下，返回数据帧的前 5 行
- ◀ `pandas.DataFrame.iiterrows()` 遍历 DataFrame 的行
- ◀ `pandas.DataFrame.iloc()` 通过整数索引来选择 DataFrame 的行和列的索引器
- ◀ `pandas.DataFrame.index` 查询数据帧的行标签
- ◀ `pandas.DataFrame.info` 获取关于数据帧摘要信息
- ◀ `pandas.DataFrame.isnull()` 用于检查 DataFrame 中的每个元素是否为缺失值 NaN
- ◀ `pandas.DataFrame.iteritems()` 遍历 DataFrame 的列
- ◀ `pandas.DataFrame.kurt()` 计算 DataFrame 中列的峰度 (四阶矩)
- ◀ `pandas.DataFrame.kurtosis()` 计算 DataFrame 中列的峰度 (四阶矩)
- ◀ `pandas.DataFrame.loc()` 通过标签索引来选择 DataFrame 的行和列的索引器
- ◀ `pandas.DataFrame.max()` 计算 DataFrame 中每列的最大值
- ◀ `pandas.DataFrame.mean()` 计算 DataFrame 中每列的平均值
- ◀ `pandas.DataFrame.median()` 计算 DataFrame 中每列的中位数
- ◀ `pandas.DataFrame.min()` 计算 DataFrame 中每列的最小值
- ◀ `pandas.DataFrame.mode()` 计算 DataFrame 中每列的众数
- ◀ `pandas.DataFrame.nunique()` 计算数据帧中每一列的独特值数量
- ◀ `pandas.DataFrame.quantile()` 计算 DataFrame 中每列的指定分位数值，如四分位数、特定百分位等
- ◀ `pandas.DataFrame.rank()` 计算 DataFrame 中每列元素的排序排名
- ◀ `pandas.DataFrame.reindex()` 用于重新排序 DataFrame 的列标签
- ◀ `pandas.DataFrame.rename()` 对 DataFrame 的索引标签、列标签或者它们的组合进行重命名
- ◀ `pandas.DataFrame.reset_index()` 将 DataFrame 的行标签重置为默认的整数索引，默认并将原来的行标签转换为新的一列

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

- ◀ `pandas.DataFrame.set_axis()` 重新设置 DataFrame 的行或列标签
- ◀ `pandas.DataFrame.set_index()` 改变 DataFrame 的索引结构
- ◀ `pandas.DataFrame.shape` 返回一个元组，其中包含数据帧的行数、列数
- ◀ `pandas.DataFrame.size` 用于返回数据帧中元素，即数据单元格总数
- ◀ `pandas.DataFrame.skew()` 计算 DataFrame 中列的偏度（三阶矩）
- ◀ `pandas.DataFrame.sort_index()` 按照索引的升序或降序对 DataFrame 进行重新排序，默认 `axis = 0`
- ◀ `pandas.DataFrame.std()` 计算 DataFrame 中列的标准差（样本）
- ◀ `pandas.DataFrame.sum()` 计算 DataFrame 中每列元素的总和
- ◀ `pandas.DataFrame.tail()` 用于查看数据帧的后几行数据，默认情况下，返回数据帧的后 5 行
- ◀ `pandas.DataFrame.to_csv()` 将 DataFrame 数据保存为 CSV 格式文件
- ◀ `pandas.DataFrame.to_string()` 将 DataFrame 数据转换为字符串格式
- ◀ `pandas.DataFrame.values` 返回数据帧中的实际数据部分作为一个多维 NumPy 数组
- ◀ `pandas.DataFrame.var()` 计算 DataFrame 中列的方差（样本）
- ◀ `pandas.Series()` 创建 Pandas Series
- ◀ `seaborn.heatmap()` 绘制热图
- ◀ `seaborn.load_dataset()` 加载 Seaborn 示例数据集



19.1 什么是 Pandas?

Pandas 是一个开源的 Python 数据分析库，它提供了一种高效、灵活、易于使用的数据结构，可以完成数据操作、数据清洗、数据分析和数据可视化等任务。Pandas 最基本的数据结构是 Series 和 DataFrame。DataFrame 在本书中被叫做数据帧。

Series 是一种类似于一维数组的对象，相当于 NumPy 一维数组；而 DataFrame 是一种二维表格型的数据结构，可以容纳多种类型的数据，并且可以进行各种数据操作。本章主要介绍 DataFrame。

Pandas 还提供了大量的数据处理和操作函数，例如数据筛选、数据排序、数据聚合、数据合并等等。因此，Pandas 成为了 Python 数据科学和机器学习领域的重要工具之一。

注意，为了方便大家查看全英文技术文档，本书行文中会并用数据帧、Pandas DataFrame、DataFrame 这几个术语。此外，大家还会在书中看到 Panda Series、Series 等叫法。

比较 NumPy Array、Pandas DataFrame

NumPy Array 和 Pandas DataFrame 都是 Python 中重要的数据类型，但是两者存在区别。

NumPy Array 是多维数组对象，一般要求所有元素具有相同的数据类型，即本书前文提到的**同质性** (homogeneous)，从而保证高效存储运算。

Pandas DataFrame 是一个二维表格数据结构，类似于 Excel 表格，包含行标签和列标签。Pandas DataFrame 由多个列组成，每个列可以是不同的数据类型。

举个例子，鸢尾花数据集前 4 列都是**定量数据** (quantitative data)，而最后一列鸢尾花标签是**定性数据** (qualitative data)。

NumPy Array 使用整数索引，类似于 Python 列表。Pandas DataFrame 支持自定义行标签和列标签，可以使用标签而不仅仅是整数索引进行数据访问。

注意，本章中的行标签、列标签特指数据帧的标签；而对于数据帧，行索引、列索引则是指行列整数索引，这一点类似 NumPy 二维数组。默认情况下，数据帧行标签、列标签均为基于 0 的整数索引。

如图 1 所示，给一个 NumPy 二维数组加上行标签和列标签，我们便得到了一个 Pandas DataFrame。当然，Pandas DataFrame 也可以转化成 NumPy 数组。这是本章后续要介绍的内容。

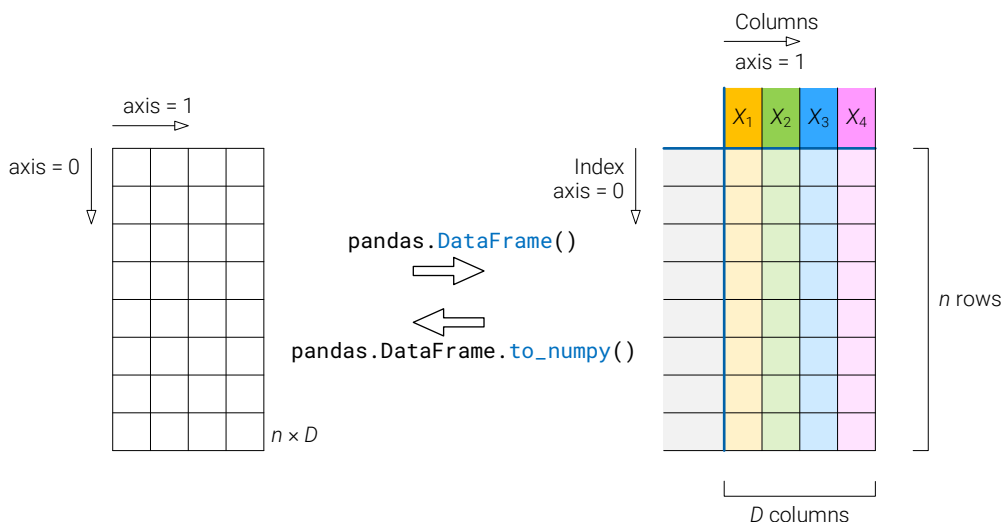


图 1. 比较 NumPy array 和 Pandas DataFrame，以及两者的相互转化

注意，图 1 中的 X_1 、 X_2 、 X_3 、 X_4 仅仅是示意，真实的列标签不会出现斜体、下标这些样式；在数据帧中，我们可以用 X_1 、 X_2 、 X_3 、 X_4 或者 X_1 、 X_2 、 X_3 、 X_4 。之所以写成 X_1 、 X_2 、 X_3 、 X_4 ，是为了帮助大家把数据帧的列和数学中的随机变量概念联系起来。

Pandas DataFrame 更适用于处理结构化数据，如表格、CSV 文件、SQL 数据库查询结果等等。

此外，Pandas DataFrame 还支持时间序列数据。Pandas DataFrame 中的时间序列数据通常是指具有时间索引的数据，其中时间可以是一系列日期、时间戳或时间间隔，对应于数据的每个行或每个数据点。

Pandas DataFrame 提供大量数据操作、处理缺失值、数据过滤、数据合并、数据透视等更高级的数据分析功能。

实际应用中，Pandas 和 NumPy 常常一起使用，Pandas 负责数据的组织、清洗和分析，而 NumPy 负责底层数值计算。

如何学习 Pandas

学习 Pandas 需要从以下几个板块入手：

Pandas 基础知识：需要学习 Pandas 的数据结构，包括 Series 和 DataFrame，掌握如何创建、读取、修改、删除、索引和切片等操作，以及如何处理缺失值和重复值等数据清洗技巧。

数据操作：Pandas 提供了丰富的数据操作函数，例如数据筛选、排序、合并、聚合、透视等等。需要学习这些函数的用法和应用场景，以便在数据分析和处理中灵活运用。

数据可视化：Pandas 本身具备一些基本可视化工具；同时 Pandas 可以与 Matplotlib、Seaborn、Plotly 等库结合使用，进行数据可视化，大家需要学习如何使用这些库进行可视化和图表绘制。

时间序列：Pandas 中的时间序列是一种强大的数据结构，用于处理时间相关的数据，它能够轻松地对时间索引的数据进行清理、切片、聚合和频率转换等操作。同时，配合 Statsmodels 等 Python 库，可以进一步完成时间序列分析、建模模拟、机器学习等。

19.2 创建数据帧：从字典、列表、NumPy 数组 ...

在 Pandas 中，可以使用多种方法创建 DataFrame，下面介绍几种常用方法。

字典 dict

可以用 Python 中的字典 dict 来创建 Pandas DataFrame。字典的键 key 将成为 DataFrame 的列标签，而字典的值 value 将成为 DataFrame 的列数据。图 2 给出了一个示例。

a 将 pandas 导入，并定义别名 pd。运行后，Pandas 库将被导入，然后可以使用别名 pd 来调用 Pandas 的函数和类，例如 pd.DataFrame()、pd.Series() 等等。

b 构造一个字典。字典的键分别是 'Integer'、'Greek'，对应 DataFrame 的列标签。每个键对应的值是一个列表，这些列表将成为 DataFrame 中相应列的数据。

⚠ 注意，DataFrame 的 Index 和 Column 标签都区分大小写，也就是说，'Integer' 和 'integer' 代表两个不同标签。

请确保字典中的每个值（列表）的长度相同，以便正确创建 DataFrame。如果长度不一致，将会引发异常，异常信息为 'ValueError: All arrays must be of the same length'。

c 利用 pandas.DataFrame() 创建一个二维数据结构称为 DataFrame。

d 利用 pandas.DataFrame.set_index() 将数据帧的 'Integer' 这一列设置为行标签，原理如图 3 所示。此外，可以用 pandas.DataFrame.reset_index() 重置行标签，将行标签设置为从 0 开始的整数索引，同时加一个原来的行标签转换成一个新的列。使用 pandas.DataFrame.reset_index() 时，如果设置 drop=True，原来的行标签将会被删除。

```
a import pandas as pd
# 用字典 dict 创建数据帧
b dict_eg = {'Integer': [1, 2, 3, 4, 5],
             'Greek': ['alpha', 'beta', 'gamma',
                       'delta', 'epsilon']}
c df_from_dict = pd.DataFrame(data=dict_eg)
# 采用默认行索引, Zero-based numbering
# 将特定列设定为索引
d df_from_dict2 = df_from_dict.set_index('Integer')
```

图 2. 用字典创建 Pandas 数据帧; Bk1_Ch19_01.ipynb

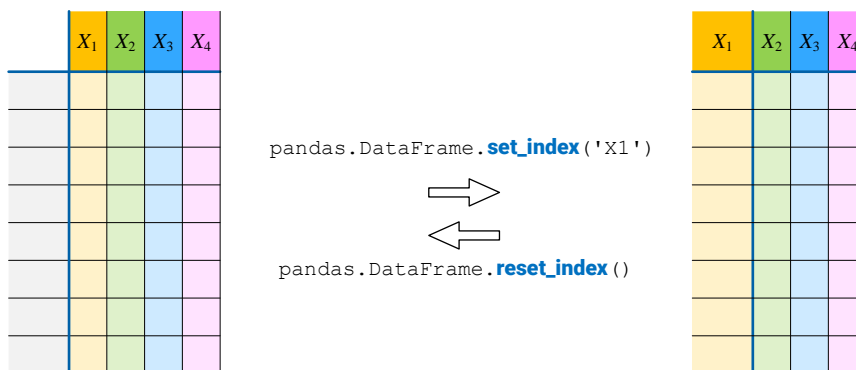


图 3. 设置 DataFrame 的索引

列表 list

还可以使用 Python 中的列表 `list` 来创建 Pandas `DataFrame`。列表 `list` 每个列代表 `DataFrame` 的一列数据，如图 4 所示。

```
import pandas as pd
# 用列表 list 创建数据帧
list_fruits = [
    ['apple', 11],
    ['banana', 22],
    ['cherry', 33],
    ['durian', 44]
]
df_list1 = pd.DataFrame(list_fruits)
# 采用默认行索引、列标签, Zero-based numbering
# 设定行索引
df_list1.set_axis(['a', 'b', 'c', 'd'], axis='index')
# 设定行标签
df_list1.set_axis(['Fruit', 'Number'], axis='columns')
# 设定行索引、列标签
df_list2 = pd.DataFrame(list_fruits,
                        columns=['Fruit', 'Number'],
                        index = ['a', 'b', 'c', 'd'])
```

图 4. 用列表创建 Pandas 数据帧; Bk1_Ch19_01.ipynb

图 4 中 **a** 构造了一个 4 行、2 列的列表。**b** 利用 `pandas.DataFrame()` 将列表转化为 Pandas 数据帧。

`pandas.DataFrame()` 这个函数的重要参数有 `pandas.DataFrame(data = ..., index = ..., columns = ...)`。

其中, `data` 可以是各种数据类型, 包括字典、列表、NumPy 数组、Pandas Series 等。这些数据将用于构建 `DataFrame` 的内容。

而 `index` 用于指定行标签的数据。

注意, `index` 是一个可选参数, 默认为从 0 开始的整数索引。

函数中 `columns` 参数用于指定列标签的数据。它也是一个可选参数，默认为从 0 开始的整数索引。^(b) 创建的数据帧的行标签、列标签均为默认从 0 开始的整数索引。

对于已经创建的数据帧，可以通过 `pandas.DataFrame.set_axis()` 修改行标签 (^(e))、列标签 (^(d))。

而 ^(e) 创建数据帧时设定了行标签、列标签。

NumPy 数组

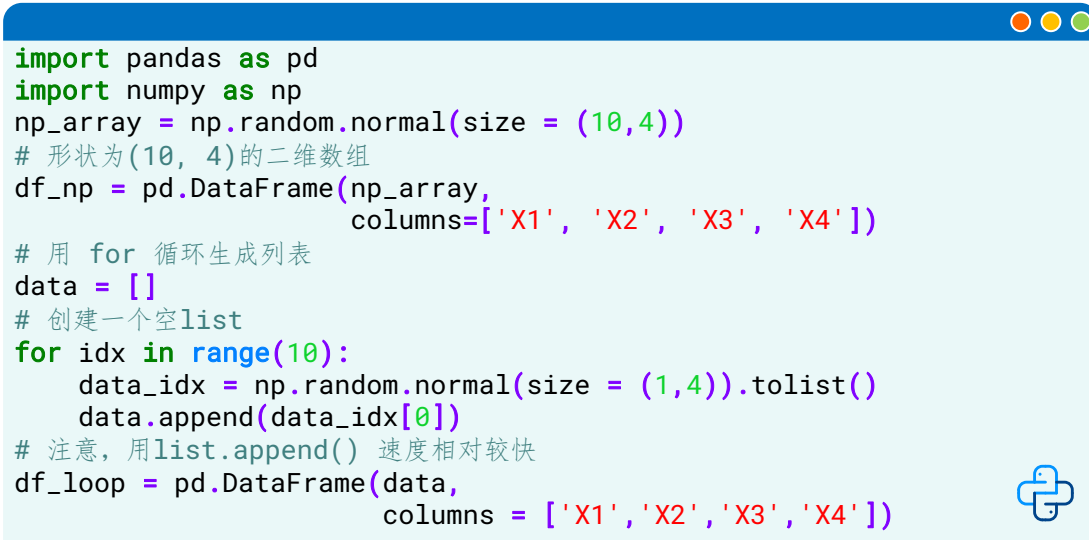
要使用二维 NumPy 数组创建 Pandas DataFrame，可以直接将二维 NumPy 数组作为参数传递给 `Pandas.DataFrame()` 函数。NumPy 数组每一行的元素将成为 DataFrame 的一行，而每一列的元素将成为 DataFrame 的一列。

图 5 中 ^(a) 利用 `numpy.random.normal()` 函数生成一个形状为 (10, 4) 的二维数组，数组中的元素是从高斯分布中随机抽取的样本数据。

^(b) 利用 `pandas.DataFrame()` 创建数据帧，并设置列标签。

^(c) 则是在 for 循环中生成列表，然后再将其转化成数据帧。

Pandas 还支持从 Excel 文件、SQL 数据库、JSON、HTML 等数据来源中读取数据来创建 DataFrame。



```
import pandas as pd
import numpy as np
(a) np_array = np.random.normal(size = (10,4))
# 形状为(10, 4)的二维数组
(b) df_np = pd.DataFrame(np_array,
                        columns=['X1', 'X2', 'X3', 'X4'])
# 用 for 循环生成列表
data = []
# 创建一个空list
(c) for idx in range(10):
    data_idx = np.random.normal(size = (1,4)).tolist()
    data.append(data_idx[0])
# 注意，用list.append() 速度相对较快
df_loop = pd.DataFrame(data,
                        columns = ['X1', 'X2', 'X3', 'X4'])
```

图 5. 用 NumPy 数组创建 Pandas 数据帧;  Bk1_Ch19_01.ipynb

19.3 数据帧操作：以鸢尾花数据为例

本书前文介绍过鸢尾花数据集 (Fisher's Iris data set)。这一节我们利用鸢尾花数据集介绍常用数据帧操作。

导入鸢尾花数据

图 6 所示为从 Seaborn 库中导入鸢尾花数据集。

- a 导入 Seaborn 库时使用的 `as sns` 是给 Seaborn 库起了一个别名，以方便在代码中使用。
- b 利用 `seaborn.load_dataset()` 函数导入鸢尾花数据集，格式为数据帧。在 Seaborn 中，"iris" 数据集通常是以 Pandas DataFrame 的形式加载的，它包含了 150 行和 5 列，具体如表 1 所示。每个鸢尾花样本在 DataFrame 中都有一个唯一的行标签（也是默认行整数索引），通常从 0 到 149。

鸢尾花样本 DataFrame 列标签有 5 个：(第 0 列) 'sepal_length' 萼片长度，浮点数类型；(第 1 列) 'sepal_width' 萼片宽度，浮点数类型；(第 2 列) 'petal_length'：花瓣长度，浮点数类型；(第 3 列) 'petal_width' 花瓣宽度，浮点数类型；(第 4 列) 'species'：鸢尾花的品种，字符串类型。

- c 利用 `seaborn.heatmap()` 可视化鸢尾花数据集前四列，具体如图 7 所示。
- c 代码中 `iris_df.iloc[:, 0:4]` 利用 `pandas.dataframe.iloc[]` 对 Pandas DataFrame 进行切片操作，用于从 DataFrame 中选择特定的行和列。`[:, 0:4]`：这是对 DataFrame 进行切片的部分。

在 `iloc` 中，第一个冒号 `:` 表示选择所有的行，而 `0:4` 表示选择列的范围，即列索引位置从 0 到 3，不包括 4。

Python 的切片操作通常是左闭右开区间，所以 `0:4` 选择了索引位置 0、1、2 和 3 的列。



下一章专门介绍 Pandas 数据帧的索引和切片。

```
import pandas as pd
a import seaborn as sns
import matplotlib.pyplot as plt
b iris_df = sns.load_dataset("iris")
# 从Seaborn中导入鸢尾花数据帧
# 用热图可视化鸢尾花数据
fig, ax = plt.subplots(figsize = (5,9))
sns.heatmap(iris_df.iloc[:, 0:4],
             cmap = 'RdYlBu_r',
             ax = ax,
             vmax = 0, vmin = 8,
             cbar_kws = {'orientation':'vertical'},
             annot=False)
c # 将热图以SVG格式保存
d fig.savefig('鸢尾花数据dataframe.svg', format='svg')
```

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 6. 从 Seaborn 中导入鸢尾花数据集，格式为数据帧；Bk1_Ch19_01.ipynb

表 1. 鸢尾花样本数据构成的数据帧

Index	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5	3.6	1.4	0.2	setosa
...
145	6.7	3	5.2	2.3	virginica
146	6.3	2.5	5	1.9	virginica
147	6.5	3	5.2	2	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3	5.1	1.8	virginica

`pandas.DataFrame.to_csv()` 将 `DataFrame` 数据保存为 CSV（逗号分隔值，comma-separated values）文件。CSV 是一种常见的文本文件格式，用于存储表格数据，每行代表一条记录，每个字段由逗号或其他特定字符分隔。

`pandas.DataFrame.to_string()` 将 `DataFrame` 数据转换为字符串格式。

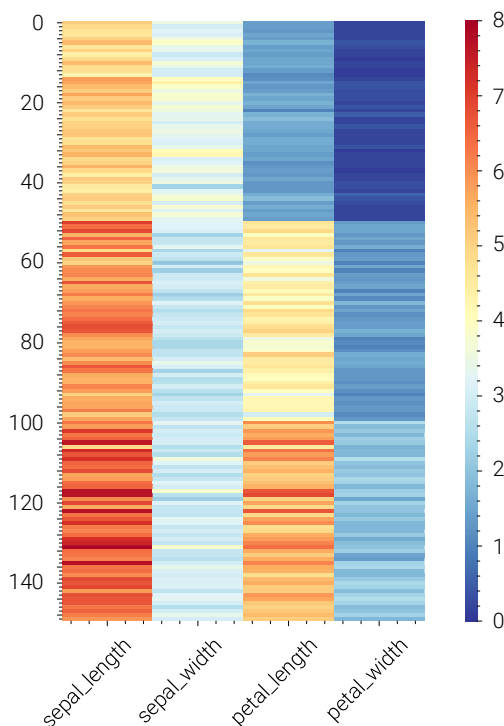


图 7. 热图可视化鸢尾花数据集数据帧

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>

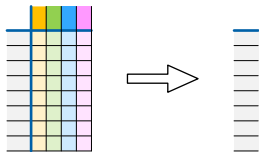


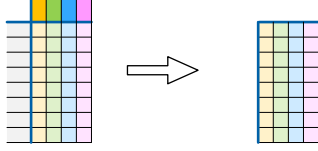
本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

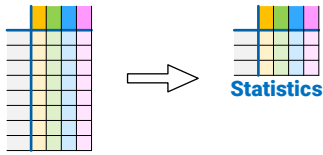
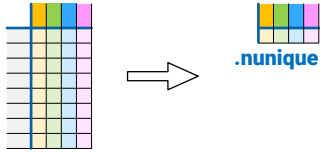
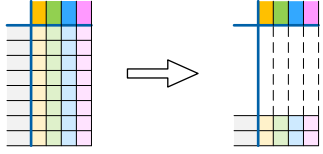
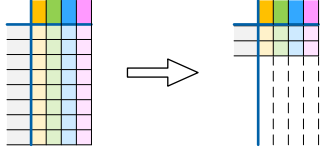
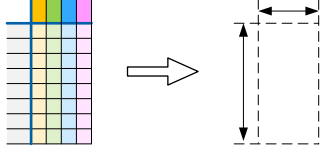
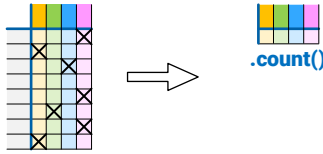
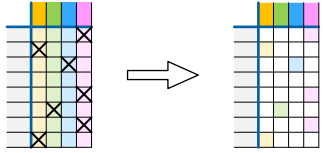
欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

数据帧基本信息

Pandas 提供很多函数查询数据帧信息，表 2 介绍几个常用函数。

表 2. 获取数据帧基本信息的几个常用函数（属性、方法）

函数	用法
<p><code>pandas.DataFrame.index</code></p> 	<p>查询数据帧的行标签。</p> <p>比如 <code>iris_df.index</code> 的结果为 <code>'RangeIndex(start=0, stop=150, step=1)'</code>。</p> <p>如果想要知道行标签的具体值，则用 <code>list(iris_df.index)</code>。</p> <p>以下是获取数据帧行数的几种不同方法：</p> <pre>iris_df.shape[0] len(iris_df) len(iris_df.index) len(iris_df.axes[0])</pre>
<p><code>pandas.DataFrame.columns</code></p> 	<p>查询数据帧的列标签。</p> <p>比如 <code>iris_df.columns</code> 的结果为 <code>'Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'], dtype='object')'</code>。同样 <code>list(iris_df.columns)</code> 可以得到列标签的列表。</p> <p>以下是获取数据帧列数的几种不同方法：</p> <pre>iris_df.shape[1] len(iris_df.T) # T len(iris_df.columns) len(iris_df.axes[1])</pre>
<p><code>pandas.DataFrame.axes</code></p> 	<p>同时获得数据帧的行标签、列标签。</p> <p>比如 <code>iris_df.axes</code> 的结果为 <code>[RangeIndex(start=0, stop=150, step=1), Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'], dtype='object')]</code>。</p>
<p><code>pandas.DataFrame.values</code></p> 	<p>用于返回数据帧中的实际数据部分作为一个多维 NumPy 数组。返回的数组可以用于进行数值计算、传递给其他库或以其他方式处理数据。</p> <p>比如，<code>iris_df.values</code> 返回的是二维 NumPy 数组。</p>
<p><code>pandas.DataFrame.info</code></p>	<p>获取关于数据帧摘要信息，比如数据帧的结构、数据类型、缺失值情况、内存占用等基本信息，对于数据的初步探索和诊断非常有用。</p>

<p><code>pandas.DataFrame.describe()</code></p> 	<p>用于生成关于数据帧统计摘要信息。它提供了数据的基本统计信息，如计数、均值、标准差、最小值、最大值和分位数等。本书后文将专门介绍数据帧运算，其中包括统计运算。</p> <p>比如，<code>iris_df.describe()</code> 计算鸢尾花列数据统计值。</p> <p>如果想要打印小数点后一位，可以用 <code>iris_df.describe().round(1)</code>。</p>
<p><code>pandas.DataFrame.nunique()</code></p> 	<p>用于计算数据帧中每一列的唯一值/独特值 (unique value) 数量。</p> <p>比如，对于鸢尾花数据来说，最后一列 (species) 的独特值个数为 3。</p> <p>类似地，<code>pandas.unique()</code> 可以计算得到数据帧某一列的具体独特值。</p> <p>比如，<code>iris_df['species'].unique()</code> 的结果为 <code>array(['setosa', 'versicolor', 'virginica'], dtype=object)</code>。</p>
<p><code>pandas.DataFrame.head()</code></p> 	<p>用于查看数据帧的前几行数据，默认情况下，返回数据帧的前 5 行。</p> <p>比如，<code>iris_df.head(2)</code> 返回数据帧前 2 行。</p>
<p><code>pandas.DataFrame.tail()</code></p> 	<p>用于查看数据帧的后几行数据，默认情况下，返回数据帧的后 5 行。</p> <p>比如，<code>iris_df.tail(2)</code> 返回数据帧后 2 行。</p>
<p><code>pandas.DataFrame.shape</code></p> 	<p>用于获取数据帧的维度信息。函数返回一个元组，其中包含数据帧的行数、列数。</p> <p>比如，<code>iris_df.shape</code> 返回的结果为 <code>(150, 5)</code>。</p>
<p><code>pandas.DataFrame.size</code></p>	<p>用于返回数据帧中元素，即数据单元格总数，就是数据帧行数乘以列数的结果。</p> <p>比如，<code>iris_df.size</code> 返回的结果为 750。</p>
<p><code>pandas.DataFrame.count()</code></p> 	<p>返回数据帧每列（默认 <code>axis=0</code>）非缺失值数量。这个函数可以快速了解每列中有多少个有效的非缺失数据，这对于数据清洗和数据质量的检查非常有用。将参数设置为 <code>axis=1</code>，可以查询每行的非缺失值数量。</p> <p>比如，<code>iris_df.count() * 100 / len(iris_df)</code> 计算每一列非缺失值的百分比。</p>
<p><code>pandas.DataFrame.isnull()</code></p> 	<p>用于检查 DataFrame 中的每个元素是否为缺失值 NaN。函数返回一个与原始 DataFrame 结构相同的布尔值 DataFrame，其中的每个元素都对应于原始 DataFrame 中的一个元素，并且其值为 <code>True</code> 表示该元素是缺失值，<code>False</code> 表示该元素不是缺失值。</p> <p>比如，<code>iris_df.isnull().sum() * 100 / len(iris_df)</code> 计算每一列缺失值百分比。</p>


循环

如图 8 所示，在 Pandas 中可以使用 `iterrows()` 方法来遍历 `DataFrame` 的行，或者使用 `iteritems()` 或 `items()` 方法来循环 `DataFrame` 的列。另外，还可以直接使用 `for` 循环来遍历 `DataFrame` 的列。

```
import pandas as pd
import seaborn as sns

iris_df = sns.load_dataset("iris")
# 从Seaborn中导入鸢尾花数据帧
# 遍历数据帧的行
a for idx, row_idx in iris_df.iterrows():
    print('=====')
    print('Row index =',str(idx))
    print(row_idx['sepal_length'],
            row_idx['sepal_width'])

# 遍历数据帧的列
b for column_idx in iris_df.iteritems():
    print(column_idx)
```

图 8. 遍历数据帧行、列;  Bk1_Ch19_01.ipynb

修改数据帧

表 3 总结了 Pandas 中常用的各种修改数据帧行标签、列标签函数。

表 3. 修改数据帧行标签、列标签

函数	用法
<code>pandas.DataFrame.rename()</code>	<p>对 <code>DataFrame</code> 的索引标签、列标签或者它们的组合进行重命名。</p> <p>需要注意的是，<code>rename()</code>方法默认返回新的 <code>DataFrame</code>，如果想要在原地修改 <code>DataFrame</code>，可以将 <code>inplace=True</code> 参数设置为 <code>True</code>。</p> <p>比如，对列标签重命名：</p> <pre>iris_df.rename(columns={'sepal_length': 'X1', 'sepal_width': 'X2', 'petal_length': 'X3', 'petal_width': 'X4', 'species': 'Y'})</pre> <p>比如，对行标签重命名，给每个行标签前面加前缀 <code>idx_</code>：</p> <pre>iris_df.rename(lambda x: f'idx_{x}')</pre>

	每个行标签后面加后缀_idx: iris_df.rename(lambda x: f'{x}_idx')
pandas.DataFrame.add_suffix()	给 DataFrame 的列标签添加后缀，并返回一个新的 DataFrame，原始 DataFrame 保持不变。这个方法对于在合并多个 DataFrame 时，避免列名冲突很有用。通过添加后缀，可以清楚地区分来自不同 DataFrame 的列。 比如，iris_df_suffix = iris_df.add_suffix('_col') 以上数据帧要想除去列标签后缀_col，可以用： iris_df_suffix.rename(columns = lambda x: x.strip('_col'))
pandas.DataFrame.add_prefix()	给 DataFrame 的列标签添加前缀，并返回一个新的 DataFrame，原始 DataFrame 保持不变。这个方法对于在合并多个 DataFrame 时，避免列名冲突很有用。通过添加前缀，可以清楚地区分来自不同 DataFrame 的列。 比如，iris_df_prefix = iris_df.add_prefix('col_').head() 以上数据帧要想除去列标签前缀 col_，可以用： iris_df_prefix.rename(columns = lambda x: x.strip('col_'))

更改列标签顺序

如图 9 所示，数据帧创建后，列标签的顺序可以根据需要进一步修改。

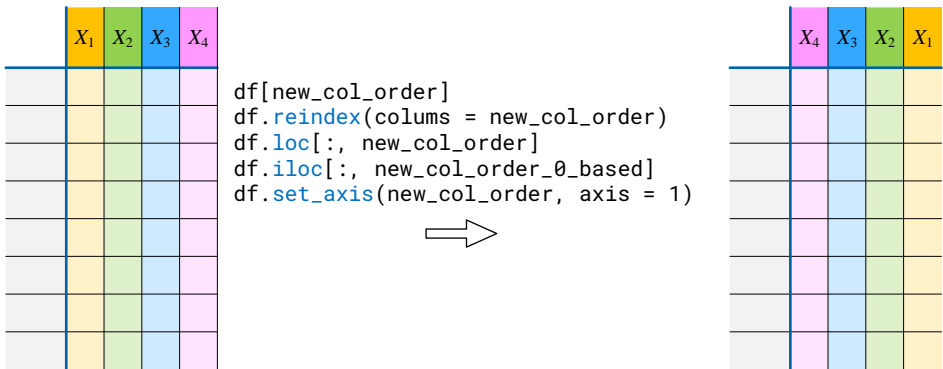


图 9. 修改列标签顺序

图 10 代码介绍不同修改列标签顺序的代码。

pandas.DataFrame.reindex() 方法用于重新排序 DataFrame 的列标签。

一般来讲，pandas.DataFrame.loc() 可以用来索引、切片数据帧；当然这个方法也可以用来重新排序列标签。下一章将专门介绍数据帧索引和切片。

pandas.DataFrame.iloc() 是 pandas 中用于通过整数索引来选择 DataFrame 的行和列的索引器。与 pandas.DataFrame.loc 不同，iloc 使用整数索引而不是标签索引。

```

import pandas as pd
import seaborn as sns

iris_df = sns.load_dataset("iris")
# 从Seaborn中导入鸢尾花数据帧

# 自定义列标签顺序
new_col_order = ['species',
                  'sepal_length', 'petal_length',
                  'sepal_width', 'petal_width']
a df_1 = iris_df[new_col_order]
b df_2 = iris_df.reindex(columns=new_col_order)
c df_3 = iris_df.loc[:, new_col_order]
d df_4 = iris_df.iloc[:, [4,0,2,1,3]]
e df_5 = iris_df.set_axis(new_col_order, axis=1)

```

图 10. 修改列标签顺序; Bk1_Ch19_01.ipynb

更改行标签顺序

图 11 介绍几种修改行标签顺序的方法。

- a 用 `pandas.DataFrame.reindex()` 重新排序 `DataFrame` 的行标签。
- b 用 `pandas.DataFrame.loc()` 通过定义行标签来重新排序 `DataFrame` 行顺序。下一章还会用这个函数在 `axis = 0` 方向进行索引、切片。
- c 用 `pandas.DataFrame.loc()` 通过定义整数行标签来重新排序 `DataFrame` 行顺序。
- d `pandas.DataFrame.sort_index()` 按照索引的升序或降序对 `DataFrame` 进行重新排序，默认 `axis = 0`。

```

import pandas as pd
import seaborn as sns

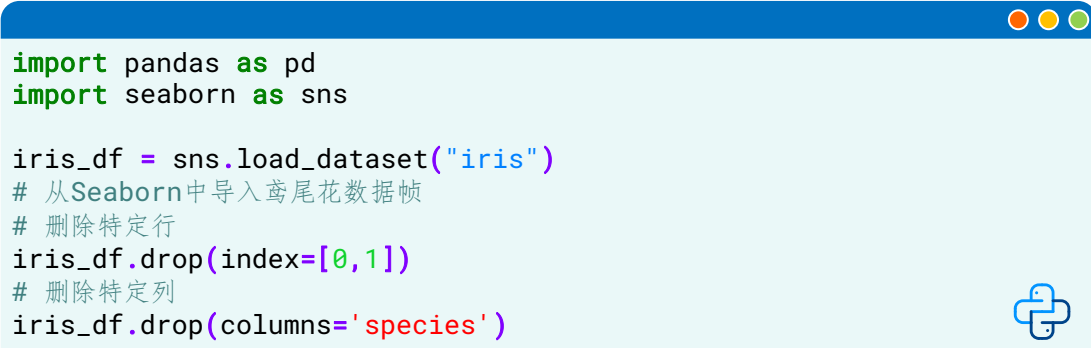
iris_df = sns.load_dataset("iris")
# 从Seaborn中导入鸢尾花数据帧
# 取出前5行，并修改行索引
iris_df_ = iris_df.iloc[:5,:].rename(lambda x:
                                     f'idx_{x}')
# 重新排序索引
new_order = ['idx_4', 'idx_2', 'idx_0', 'idx_3', 'idx_1']
a df_1 = iris_df_.reindex(new_order)
b df_2 = iris_df_.loc[new_order]
new_order_int = [4, 2, 0, 3, 1]
c iris_df_.iloc[new_order_int]
d iris_df_.sort_index(ascending=False)

```

图 11. 修改行标签顺序; Bk1_Ch19_01.ipynb

删除

`pandas.DataFrame.drop()` 方法用于从 `DataFrame` 中删除指定的行或列。默认情况下，`drop()` 方法不对原始 `DataFrame` 做修改，而是返回一个修改后的副本。将 `inplace` 参数设置为 `True`，`inplace = True`，可以在原地修改 `DataFrame`，而不返回一个新的 `DataFrame`。



```
import pandas as pd
import seaborn as sns

iris_df = sns.load_dataset("iris")
# 从Seaborn中导入鸢尾花数据帧
# 删除特定行
a iris_df.drop(index=[0,1])
# 删除特定列
b iris_df.drop(columns='species')
```

图 12. 删除特定行、列;  Bk1_Ch19_01.ipynb

19.4 四则运算：各列之间

在 `Pandas` 中，可以通过简单的语法实现各列之间的四则运算。以鸢尾花数据帧为例，图 13 中代码所示为鸢尾花数据帧花萼长度 (X_1)、花萼宽度 (X_2) 两列之间的运算。

- a 对花萼长度去均值 (demean)，即 $X_1 - E(X_1)$ 。其中，`X_df['X1'].mean()` 计算列均值。也可以用 `pandas.DataFrame.sub()` 完成减法运算。
- b 对花萼宽度去均值，即 $X_2 - E(X_2)$ 。
- c 计算花萼长度、宽度之和，即 $X_1 + X_2$ 。也可以用 `pandas.DataFrame.add()` 完成加法运算。
- d 计算花萼长度、宽度之差，即 $X_1 - X_2$ 。
- e 计算花萼长度、宽度乘积，即 $X_1 X_2$ 。也可以用 `pandas.DataFrame.mul()` 完成乘法运算。
- f 计算花萼长度、宽度比例，即 X_1 / X_2 。也可以用 `pandas.DataFrame.div()` 完成除法运算。


```

import seaborn as sns
import pandas as pd

iris_df = sns.load_dataset("iris")
# 从Seaborn中导入鸢尾花数据帧

X_df = iris_df.copy()
X_df.rename(columns = {'sepal_length': 'X1',
                      'sepal_width': 'X2'},
            inplace = True)
X_df_ = X_df[['X1', 'X2', 'species']]

# 数据转换
a X_df_['X1 - E(X1)'] = X_df_['X1'] - X_df_['X1'].mean()
b X_df_['X2 - E(X2)'] = X_df_['X2'] - X_df_['X2'].mean()
c X_df_['X1 + X2'] = X_df_['X1'] + X_df_['X2']
d X_df_['X1 - X2'] = X_df_['X1'] - X_df_['X2']
e X_df_['X1 * X2'] = X_df_['X1'] * X_df_['X2']
f X_df_['X1 / X2'] = X_df_['X1'] / X_df_['X2']
X_df_.drop(['X1', 'X2'], axis=1, inplace=True)

# 可视化
sns.pairplot(X_df_, corner=True, hue="species")

```


图 13. 鸢尾花数据帧花萼长度 (X_1)、花萼宽度 (X_2) 两列之间的运算;  Bk1_Ch19_02.ipynb

图 14 所示为经过上述转换后用 `seaborn.pairplot()` 绘制的成对特征散点图。我们在鸢尾花书《统计至简》还会用到这幅图来介绍随机变量函数。

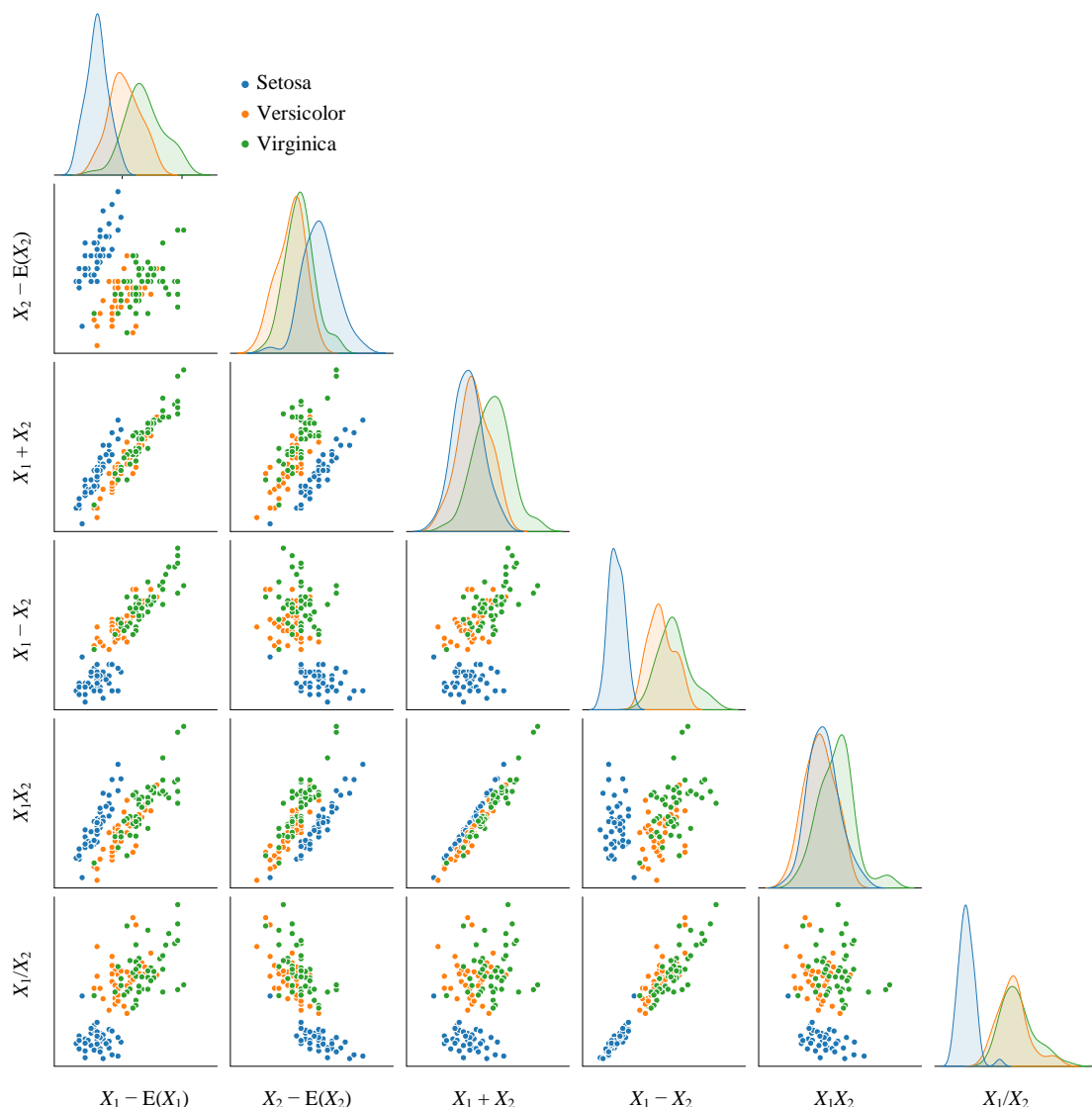


图 14. 鸢尾花花萼长度、宽度特征完成转换后的成对特征散点图

19.5 统计运算：聚合、降维、压缩、折叠 ...

拿到一组样本数据，如果数据量很大，我们不可能一个个观察样本值；这时，我们就需要各种统计量来描述数据集的不同方面，包括中心趋势、离散度和分布形状。

本书前文提过，从样本数据到某个统计量的过程，从数据角度来看，可以视作一种降维，也可以看成是折叠、压缩。

这些统计量可以帮助我们更好地了解和描述数据集的特征，从而支持数据分析和决策制定过程。在实际应用中，这些描述统计量通常与可视化工具结合使用，以更全面地理解数据的性质。

以下是常见的单一特征统计量化描述。

► 均值 (average, mean) 是数据集中所有值的总和除以数据点的数量。

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

- ▶ 众数 (mode) 是数据集中出现频率最高的值。一个数据集可以有一个或多个众数。
- ▶ 中位 (median) 数是将数据集中的所有值按大小排序后位于中间位置的值。它不受异常值的影响，用于度量数据的中心趋势。当数据点数量为奇数时，中位数就是中间的值；当数据点数量为偶数时，中位数是中间两个值的平均值。
- ▶ 最大值 (maximum) 是数据集中的最大数值，而最小值 (minimum) 是数据集中的最小数值，用于表示数据的范围。
- ▶ 方差 (variance) 度量了数据点与均值之间的离散程度。较高的方差表示数据点更分散，较低的方差表示数据点更接近均值。
- ▶ 标准差 (standard deviation) 是方差的平方根，用于衡量数据的离散程度。与方差不同，标准差的单位与数据集的单位相同，因此更容易理解。
- ▶ 分位点 (percentile) 是将数据集划分成若干部分的值，通常以百分比形式表示。例如，第 25 百分位数是将数据集划分成四分之一的值，第 50 百分位数就是中位数。
- ▶ 偏度 (skewness) 度量了数据分布的偏斜程度。如果数据分布偏向左侧 (负偏)，偏度为负数；如果数据分布偏向右侧 (正偏)，偏度为正数。偏度为零表示数据分布大致对称。
- ▶ 峰度 (kurtosis) 度量了数据分布的尖锐程度。峰度值通常与正态分布的峰度值相比较。正峰度表示数据分布具有比正态分布更尖锐的峰值，负峰度表示数据分布的峰值较平缓。

如图 15 所示，我们可以用直方图和核密度估计 KDE 来展示数据分布。KDE 相当于对“平滑”直方图之后的结果。注意，这两幅图的纵轴都是概率密度。也就是说它们和横轴围成的面积都为 1。

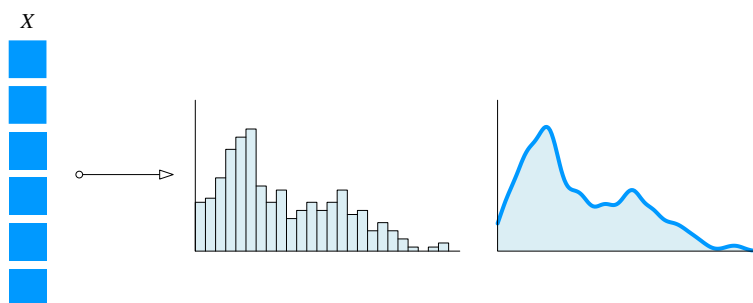


图 15. 单一特征可视化，直方图和 KDE

如图 16 所示，如果仅仅考虑单一特征样本数据的均值和样本标准差，我们相当于利用一元高斯分布“近似”数据分布。

有些时候，这种近似可能很糟糕。图 15 的“双峰”显然不能被一元高斯分布捕捉到。

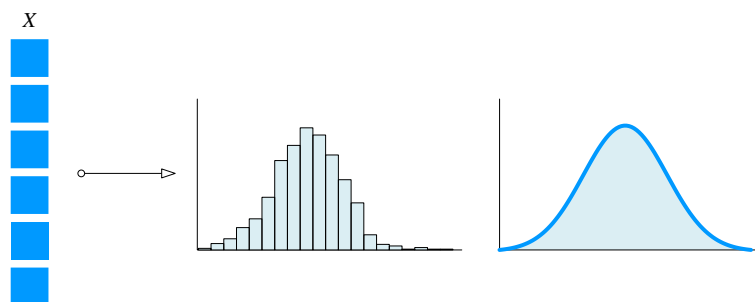


图 16. 单一特征可视化，一元高斯分布近似

当涉及到多个特征时，我们还需要两个或多个特征的常见统计描述，比如。

- ▶ 质心 (centroid) 是多个特征的平均值，通常用于表示多维数据的中心点。对更高维数据，对每个特征分别求均值的结果就是质心。
- ▶ 协方差 (covariance) 度量了两个特征之间的线性关系，它可以为正数、负数或零。正协方差表示两个特征具有正相关关系，负协方差表示它们具有负相关关系，而零协方差表示它们之间没有线性关系。
- ▶ 皮尔逊相关系数 (Pearson Correlation Coefficient, PCC)，简称相关性系数，是协方差的标准化版本，用于度量两个特征之间的线性关系的强度和方向。相关性系数在衡量线性关系时更常用，取值范围为-1 到 1，其中 1 表示完全正相关，-1 表示完全负相关，0 表示无线性关系。
- ▶ 协方差矩阵 (covariance matrix) 是一个对称方阵，其中对角线元素为方差，其余元素表示不同特征之间的协方差。
- ▶ 相关系数矩阵 (correlation matrix) 相当于是协方差矩阵的标准化版本。它的对角线元素为 1，其余元素为成对特征之间的相关性系数。

图 17 用三维直方图展示数据分布，它的纵轴可以是某个区间样本数据的频数、概率或概率密度。

图 18 所示为利用散点图和直方热图可视化两特征样本数据分布。注意，实践时我们用的更多是直方热图，很少用三维直方图。

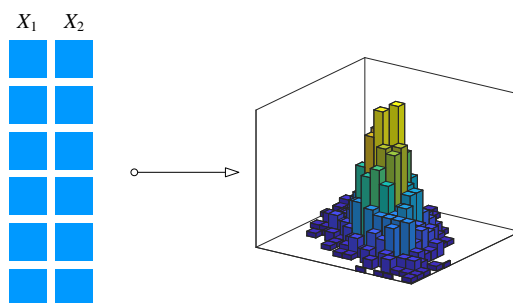


图 17. 两个特征可视化，三维直方图

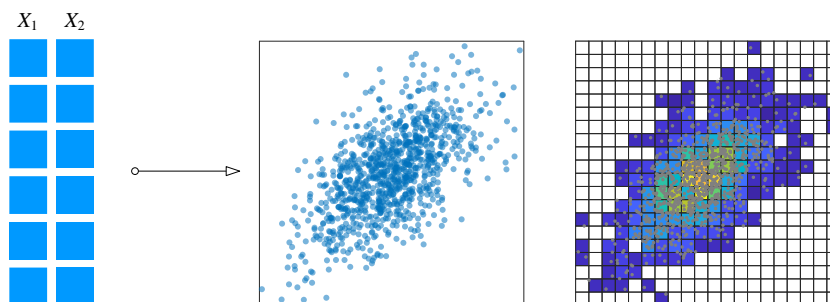


图 18. 两个特征可视化，散点图和直方热图

和前文类似，图 19 和图 20 告诉我们二元高斯分布也可以用来“近似”两特征样本数据分布，前提是样本数据足够“正态”。

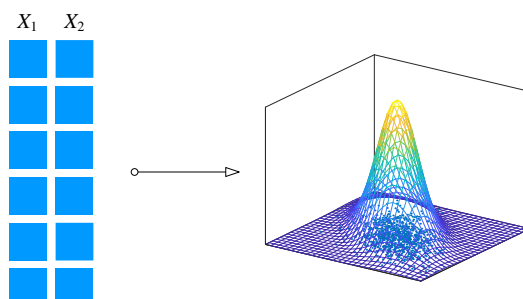


图 19. 两个特征可视化，近似二元高斯分布

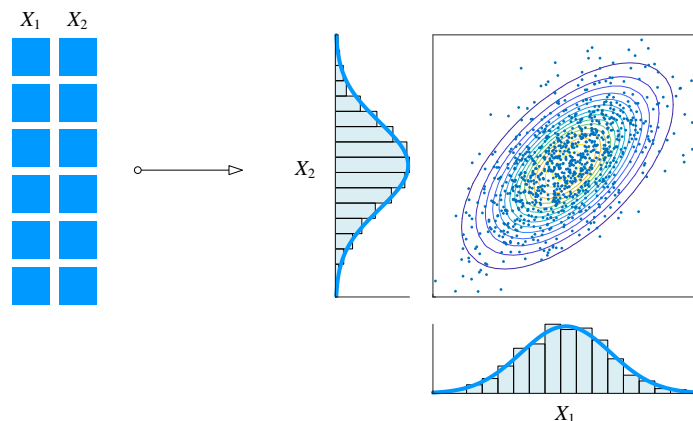


图 20. 两个特征可视化，近似二元高斯分布，边缘分布

对于多特征数据，如图 21 所示，协方差矩阵、相关性系数矩阵时量化成对特征关系的重要工具。很多机器学习算法的起点也是协方差矩阵，比如主成分分析、多输入-多输出线性回归等等。

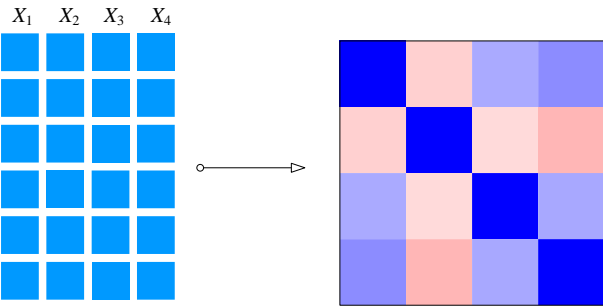



图 21. 多个特征可视化，方差协方差矩阵，相关性矩阵

Pandas 给出大量用于统计运算（也叫聚合操作）的方法，表 4 总结常用的几种方法。

表 4. Pandas 中常用统计运算方法；  Bk1_Ch19_02.ipynb

函数名称	描述
<code>pandas.DataFrame.corr()</code>	计算 DataFrame 中列之间 Pearson 相关系数（样本）
<code>pandas.DataFrame.count()</code>	计算 DataFrame 每列的非缺失值的数量
<code>pandas.DataFrame.cov()</code>	计算 DataFrame 中列之间的协方差矩阵（样本）
<code>pandas.DataFrame.describe()</code>	计算 DataFrame 中数值列的基本描述统计信息，如平均值、标准差、分位数等
<code>pandas.DataFrame.kurt()</code>	计算 DataFrame 中列的峰度（四阶矩）
<code>pandas.DataFrame.kurtosis()</code>	计算 DataFrame 中列的峰度（四阶矩）
<code>pandas.DataFrame.max()</code>	计算 DataFrame 中每列的最大值
<code>pandas.DataFrame.mean()</code>	计算 DataFrame 中每列的平均值
<code>pandas.DataFrame.median()</code>	计算 DataFrame 中每列的中位数
<code>pandas.DataFrame.min()</code>	计算 DataFrame 中每列的最小值
<code>pandas.DataFrame.mode()</code>	计算 DataFrame 中每列的众数
<code>pandas.DataFrame.quantile()</code>	计算 DataFrame 中每列的指定分位数值，如四分位数、特定百分位等
<code>pandas.DataFrame.rank()</code>	计算 DataFrame 中每列元素的排序排名
<code>pandas.DataFrame.skew()</code>	计算 DataFrame 中列的偏度（三阶矩）
<code>pandas.DataFrame.sum()</code>	计算 DataFrame 中每列元素的总和
<code>pandas.DataFrame.std()</code>	计算 DataFrame 中列的标准差（样本）
<code>pandas.DataFrame.var()</code>	计算 DataFrame 中列的方差（样本）
<code>pandas.DataFrame.nunique()</code>	计算 DataFrame 中每列中的独特值数量

在数据分析中，聚合操作（aggregation）通常用于从大量数据中提取出有意义的摘要信息，以便更好地理解数据的特征和行为。

常见的聚合操作包括计算平均值、求和、计数、标准差、方差、相关性等。这些操作可以帮助我们了解数据的集中趋势、离散程度、相关性等特征，从而做出更准确的分析和决策。

图 22 所示为 `pandas.DataFrame.cov()` 和 `pandas.DataFrame.corr()` 计算得到的鸢尾花前四列协方差矩阵、相关系数热图。

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

当然，在计算协方差时，我们也可以考虑到数据标签，本书第 22 章将利用 `groupby()` 完成分组聚合计算。

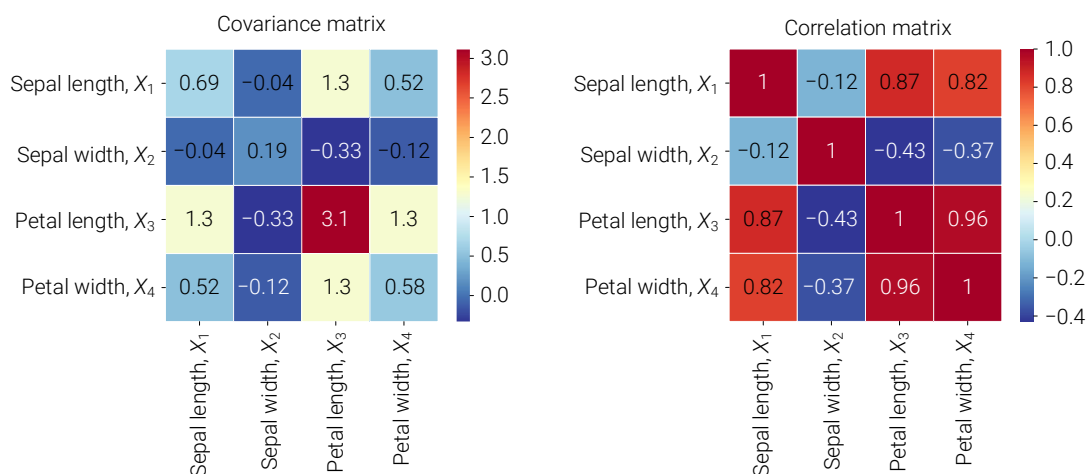


图 22. 鸢尾花数据协方差矩阵、相关性系数矩阵热图; Bk1_Ch19_02.ipynb

此外，`pandas.DataFrame.agg()` 方法用于对 `DataFrame` 中的数据进行自定义聚合操作。该方法按照指定的函数对数据进行聚合，可以是内置的统计函数，也可以是自定义的函数。

比如，`iris_df.iloc[:,0:4].agg(['sum', 'min', 'max', 'std', 'var', 'mean'])` 对鸢尾花数据帧前四列进行各种统计计算。

19.6 时间序列：按时间顺序排列的数据

时间序列 (timeseries) 是指按照时间顺序排列的一系列数据点或观测值，通常是等时间间隔下的测量值，如每天、每小时、每分钟等。时间序列数据通常用于研究时间相关的现象和趋势，例如股票价格、气象数据、经济指标等。本节主要介绍如何获得时间序列数据帧。

本地 CSV 数据

图 24 所示为利用 `pandas.csv_read()` 读入 CSV 时间序列数据。Pandas 还提供快速可视化的各种函数，比如，图 23 所示为用 `pandas.DataFrame.plot()` 绘制的时间序列线图。

a 定义了 CSV 文件的名称。大家可以在本章配套代码中找到这个文件。本书前文提过，CSV 是 Comma-Separated Values 的缩写，代表逗号分隔值，它是一种用于存储表格数据的文本文件格式。我们可以用 Excel 或 Textbook 打开，CSV 文件也可以在 JupyterLab 中查看，十分便捷。

b 利用 `pandas.read_csv()` 读取 CSV 文件。表 5 总结 Pandas 中读取不同格式数据常用的函数。

c 利用 `pandas.to_datetime()` 将指定列转换为日期时间对象。

d 用 `pandas.DataFrame.set_index()` 设置一个或多个列作为 `DataFrame` 的索引。
`inplace` 设置为 `True`，意味着将在原始 `DataFrame` 上进行修改，而不是创建一个新的 `DataFrame`。

e 这一句介绍了一种更快捷读入并处理数据的用法。`pandas.read_csv()` 有很多参数设置可以很方便地帮助我们读入、处理数据。

比如，`parse_dates` 用于指定是否解析日期列。当设置为 `True` 时，Pandas 将尝试解析 CSV 文件中的日期数据，并将其转换为日期时间对象。

参数 `index_col=0` 指定第 1 列被用作 `DataFrame` 的索引。

有了这两个参数设定，我们就可以省去 **c** 和 **d** 两句代码，请大家自行练习。

更多的参数设置，请大家参考如下官方技术文档。

https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

f 调用 `pandas` 中快速绘制线图函数，绘制图 23。

此外，请大家尝试使用 `pandas.DataFrame.to_pickle()` 将 `DataFrame` 写成 `.pkl` 文件，然后再用 `pandas.read_pickle()` 读入。

表 5. Pandas 中读取不同格式数据常用函数

函数名称	数据类型介绍
<code>pandas.read_excel()</code>	用于从 Microsoft Excel 文件（.xls 或 .xlsx 格式）中读取数据
<code>pandas.read_json()</code>	用于从 JSON (JavaScript Object Notation) 格式的数据中读取数据
<code>pandas.read_html()</code>	用于从 HTML 网页中提取表格数据
<code>pandas.read_xml()</code>	用于从 XML (Extensible Markup Language) 格式的数据中读取数据
<code>pandas.read_sql_query()</code>	用于执行 SQL (Structured Query Language) 查询并将查询结果读取到 Pandas DataFrame 中
<code>pandas.read_sas()</code>	用于从 SAS (Statistical Analysis System) 数据文件中读取数据
<code>pandas.read_pickle()</code>	用于从 Pickle 文件中读取数据。Pickle 是 Python 的一种序列化格式，可以用于保存和加载 Python 对象，包括 DataFrame



本书第 20 章将专门介绍 Pandas 中常用的快速可视化函数。



图 23. 可视化时间序列

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

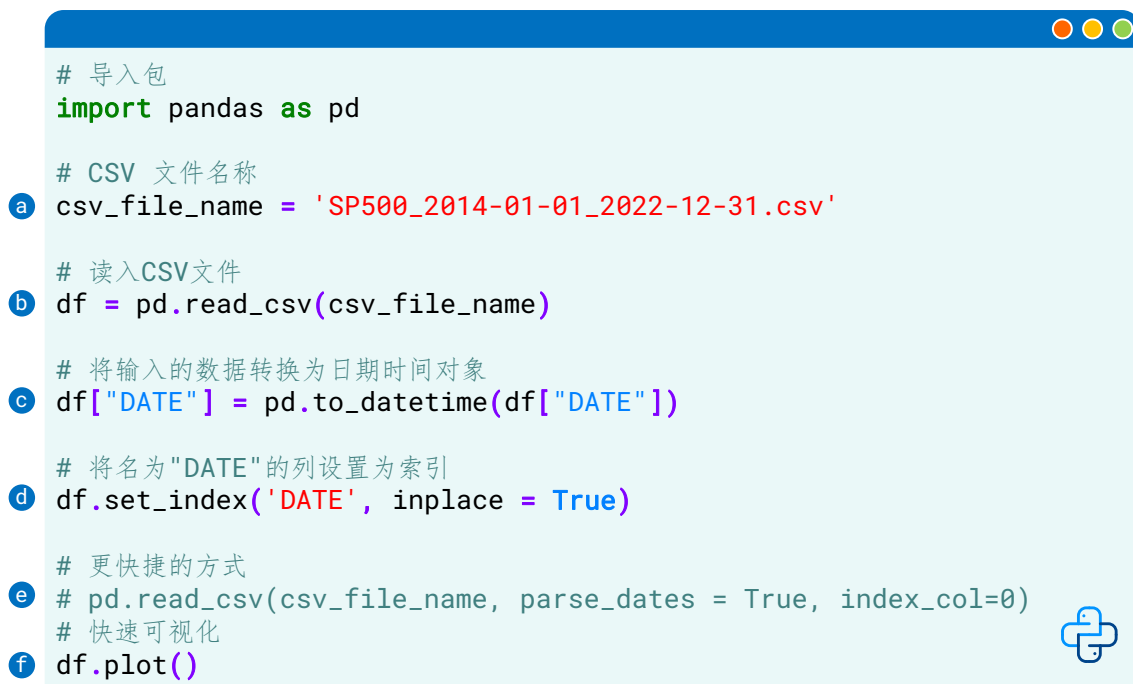


图 24. 从 CSV 读入时间序列数据; Bk1_Ch19_03.ipynb

网页下载数据

图 25 所示代码为直接从 FRED (Federal Reserve Economic Data) 官网下载数据。

a 导入 Python 标准库中的 `requests` 模块。`requests` 模块提供了一种用于发出 HTTP 请求的简单而强大的方法，使 Python 程序能够与 Web 服务器进行通信，并获取 Web 上的数据。

本书不会展开讲解网页访问和爬虫相关内容，对 `requests` 模块感兴趣的读者可以参考。

<https://pypi.org/project/requests/>

b 变量指向一个包含标准普尔 500 指数数据的文本文件的 URL (Uniform Resource Locator)。

c 使用 `requests` 库中的 `get()` 函数向上述 URL 发送 HTTP GET 请求，并将服务器的响应存储在一个名为 `response` 的变量中。

d 是一个条件语句，它检查服务器的响应状态码是否等于 200。HTTP 状态码 200 表示请求成功，服务器已成功处理了请求并返回了所请求的数据。

e 利用 `pandas.read_csv()` 读取数据并将其转换为 `DataFrame`。

参数 `skiprows=44` 指示在解析 (parse) 数据时跳过文件的前 44 行。这通常用于跳过文件的头部信息或注释行，以便读取实际的数据部分。

参数 `sep='\s+'` 字段分隔符，即制表符或多个连续的空格字符。这是因为数据文件可能是以制表符或多个空格字符作为字段分隔符。

f 利用 `pandas.to_datetime()` 将指定列转换为日期时间对象。**g** 设置索引。

如果大家无法访问上述 URL，可以使用保存在配套文件中 CSV 文件。本书后续还会利用 FRED 金融数据设计案例介绍各种 Python 库功能。

```

import pandas as pd
import requests

# 设置数据源的URL
url = 'https://fred.stlouisfed.org/data/SP500.txt'

# 发送GET请求并获取数据
response = requests.get(url)

# 检查是否成功获取数据
if response.status_code == 200:
    # 数据以制表符分隔
    df = pd.read_csv(url, skiprows=44, sep='\s+')
else:
    print("Failed to fetch data from the source")

df['DATE'] = pd.to_datetime(df['DATE'])
df.set_index('DATE', inplace=True)

```

图 25. 从网页下载数据; Bk1_Ch19_04.ipynb

用第三方库下载数据

图 26 所示代码所示为利用第三方库 `pandas_datareader` 从 FRED 下载数据。

a 将名为 `pandas_datareader` 的 Python 库导入当前的代码环境中，简做 `pdr`。这个库通常用于从各种金融数据源中获取数据，并将其整合到 Pandas 数据结构中，以便进行数据分析和处理。本书第 2 章介绍过如何安装这个库。

b 导入 Python 标准库中的 `datetime` 模块。`datetime` 模块提供了处理日期和时间的功能，包括日期和时间的创建、解析、格式化以及各种日期和时间操作等等。

c 利用 `datetime.datetime` 创建表示日期和时间的对象。(2014, 1, 1) 表示要创建的日期的年、月 and 日，这是下载数据起始日期。

本书不展开介绍 `datetime` 库，感兴趣的读者可以参考官网技术文档。

<https://docs.python.org/3/library/datetime.html>

d 用 `datetime.datetime` 创建下载数据的结束日期。

e 创建了一个名为 `ticker_list` 的 Python 列表，其中包含一个字符串 'SP500'。这个列表用于指定我们要获取数据标识符 (identification, ID)。列表中可以放置不止一个数据 ID。

大家可以到 FRED 官网查看不同数据的标识符 ID。

<https://fred.stlouisfed.org/>

f 调用了 `pandas_datareader` 库中的 `DataReader` 函数，以获取数据。

参数 `ticker_list` 包含了我们要获取数据标识符 ID 列表。

'fred' 是数据源的名称，表示我们将从 FRED 数据源获取数据。

两个日期对象 `start_date` 和 `end_date` 用于指定数据的时间范围。

很多在线数据库都提供了下载数据的 API，比如大家可以参考如下网页找到下载 FRED 数据的不同方式：

<https://fred.stlouisfed.org/docs/api/fred/>

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

```

# 导入包
a import pandas_datareader as pdr
# 需要安装 pip install pandas-datareader
import pandas as pd
b import datetime

# 从FRED下载标普500 (S&P 500)
c start_date = datetime.datetime(2014, 1, 1)
d end_date = datetime.datetime(2022, 12, 31)

# 下载数据
e ticker_list = ['SP500']
f df = pdr.DataReader(ticker_list, 'fred',
                      start_date, end_date)

```

图 26. 利用 pandas_datareader 从 FRED 下载数据;  Bk1_Ch19_05.ipynb



本书第 24 章将专门介绍常见时间序列数据帧操作。



Pandas 库最佳参考资料莫过于“Pandas 之父”Wes McKinney 创作的 *Python for Data Analysis*, 全书开源, 地址为:

<https://wesmckinney.com/>