12 Descriptive Statistics Using Seaborn

Seaborn可视化数据

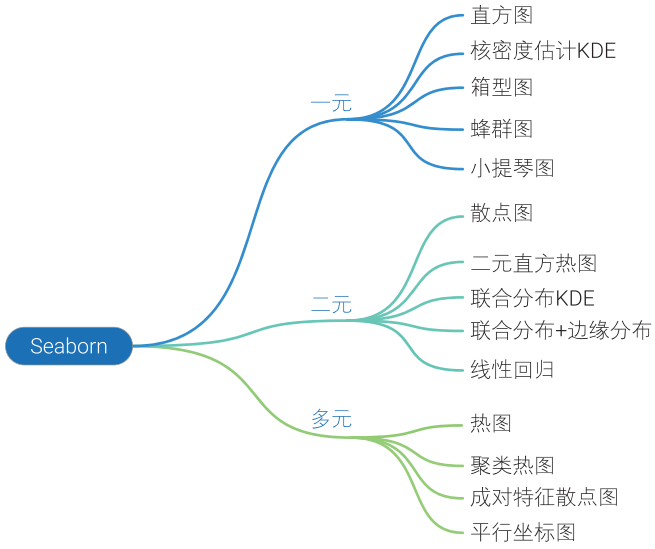
使用Seaborn完成样本数据统计描述

理性永恒，其他一切皆有终结之时。

Reason is immortal, all else mortal.

—— 毕达哥拉斯 (Pythagoras) | 古希腊哲学家、数学家 | 前570 — 495 年

- `pandas.plotting.parallel_coordinates()` 绘制平行坐标图
- `seaborn.boxplot()` 绘制箱型图
- `seaborn.heatmap()` 绘制热图
- `seaborn.histplot()` 绘制频数 / 概率 / 概率密度直方图
- `seaborn.jointplot()` 绘制联合分布和边缘分布
- `seaborn.kdeplot()` 绘制 KDE 核概率密度估计曲线
- `seaborn.lineplot()` 绘制线图
- `seaborn.lmplot()` 绘制线性回归图像
- `seaborn.pairplot()` 绘制成对分析图
- `seaborn.swarmplot()` 绘制蜂群图
- `seaborn.violinplot()` 绘制小提琴图



12.1 Seaborn：统计可视化利器

本书前文用Seaborn绘制了热图。实际上，Seaborn的真正价值在于统计可视化上。简单来说，Seaborn是一个用于数据可视化的Python库，它基于Matplotlib，并提供了一组高级的绘图函数和样式设置，可以轻松创建具有吸引力和专业外观的统计图表。

Seaborn提供了多种可视化方案，包括但不限于以下几种。

- **分布图：**包括直方图、核密度图、箱线图等，用于展示数据的分布情况。
- **散点图：**用于观察两个变量之间的关系，可以通过散点图添加颜色或大小编码第三个变量。
- **线性关系图：**通过绘制线性回归模型的置信区间，展示两个变量之间的线性关系。
- **分类图：**包括条形图、点图、计数图等，用于比较不同类别之间的数值关系。
- **矩阵图：**如热图和聚类图，用于显示数据的相似性和聚类结构。

本章以鸢尾花数据为例介绍如何用Seaborn可视化样本数据分布。

样本数据分布是指，在统计学中，对一组收集到的数据进行统计和描述的方式。

一元样本数据分布是指只包含一个随机变量的样本数据分布，如鸢尾花花萼长度。可视化一元样本分布的方法有：**直方图** (histogram)、**核密度估计** (Kernel Density Estimation，KDE)、**毛毯图** (rug plot)、**分散图** (strip plot)、**小提琴图** (violin plot)、**箱型图** (box plot)、**蜂群图** (swarm plot)等。

二元样本数据分布则涉及两个随机变量，如鸢尾花花萼长度和花萼宽度之间的关系。这种分布一般叫**联合分布** (joint distribution)。我们可以通过相关性系数量化联合分布。

边缘分布 (marginal distribution) 是指在多元数据分布中，对某一个或几个变量进行统计，而忽略其他变量的分布。例如，在花萼长度和花萼关系的二元数据分布中，对花萼长度的边缘分布就是仅考虑花萼长度变量的数据分布。

可视化二元样本分布的方法有**散点图** (scatter plot)、散点图 + 边缘直方图、散点图 + 毛毯图、散点图 + 回归图、频数热图、二元KDE等图形和图形组合。

多元样本数据分布则涉及两个以上随机变量，如鸢尾花花萼长度、花萼宽度、花瓣长度、花瓣宽度。多元样本数据的可视化方案有**热图** (heatmap)、**聚类热图** (cluster map)、**平行坐标图** (parallel plot)、成对特征散点图、Radviz等。特别地，我们还可以用协方差矩阵、相关系数矩阵来量化随机变量之间的关系。而热图可以用来可视化协方差矩阵、相关系数矩阵。

除此之外，我们在采用上述可视化方案时，还可以考虑分类，如鸢尾花种类。

下面我们来逐一展示这些统计可视化方案。

12.2 一元特征数据

直方图

直方图是一种常用的数据可视化图表，用于显示数值变量的分布情况。

如图12.1所示，将数据划分为不同的区间 (也称“柱子”)，一般计算每个区间内的数据频数 (样本数量)；简单来说，这个过程就是“查数”。然后，通过绘制每个区间的柱状条形来表示相应的频数。

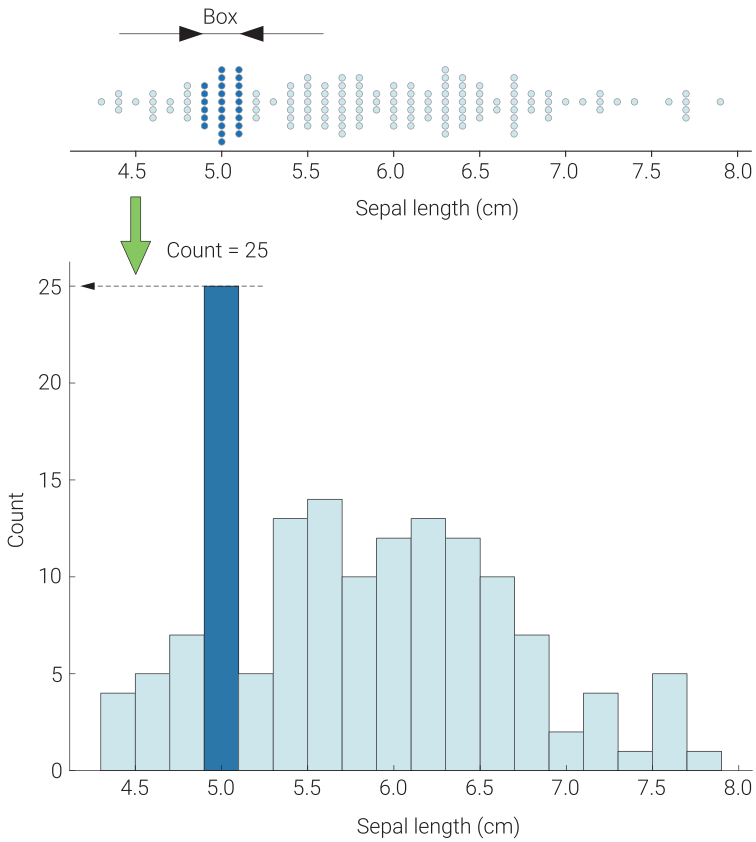


图12.1　直方图原理

比如，图12. 1中深蓝的“柱子”对应区间的样本数量为25，因此“柱子”的高度为25。

直方图的 x 轴表示变量的取值范围，而y 轴表示频数、概率、概率密度。图12.1中深蓝的“柱子”对应的频数为25，而样本总数为150，因此这个“柱子”对应的概率为25/150。“柱子”的宽度为0.2，因此这个深蓝色“柱子”的概率密度为 (25/150/0.2)。

图12.2所示为鸢尾花花萼长度样本数据的直方图，纵轴为频数。

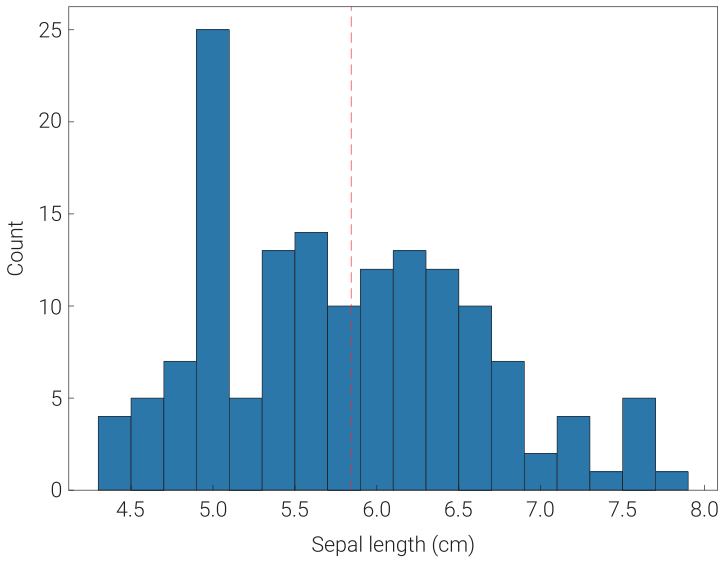


图12.2　鸢尾花花萼长度样本数据直方图 | Bk1_Ch12_01.ipynb

如果图12.2的纵轴为概率，图12.2的这些“柱子”的高度之和为1。如果图12.2的纵轴为概率密度，图12.2的这些“柱子”的面积之和为1。

标准差是方差的平方根。样本、样本均值、样本标准差，三者具有相同单位。

我们可以通过代码12.1绘制图12.2，下面讲解其中的关键语句。

ⓐ将seaborn导入，简写作sns。

ⓑ利用seaborn.load_dataset()，简写作sns.load_dataset()，导入鸢尾花数据。保存在Seaborn中的鸢尾花数据类型是Pandas DataFrame。

请大家自己解释ⓒ，简述fig和ax两个对象都有哪些用途。

ⓓ利用seaborn.histplot()绘制直方图。参数与data一般为Pandas数据帧，x为横轴对应的数据帧的列标签。

请大家在JupyterLab分别尝试绘制鸢尾花数据其他三个量化特征 (花萼宽度、花瓣长度、花瓣宽度) 的直方图。

此外，参数stat指定纵轴类型，比如'count'对应频数，'probability'对应概率，'density'对应概率密度。可以用bins指定直方图区间数量，binwidth定义区间宽度。

ⓔ利用axvline() 在轴对象ax绘制了花萼长度样本均值的位置。

这段代码中iris_sns.sepal_length可以取出数据帧的特定列，其中sepal_length是列标签。而iris_sns.sepal_length.mean() 则计算这一列的均值。

这是Pandas数据帧重要的计算方法——**链式法则** (method chaining)。简单来说，Pandas链式法则是一种编程风格，旨在通过将多个操作链接在一起，以更清晰、紧凑的方式执行数据处理任务。

请大家修改本章配套Jupyter Notebook，将“均值 ± 标准差”这两条直线也画上去。

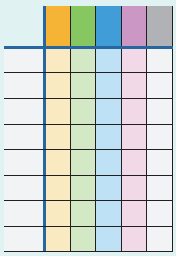
代码12.1 用Seaborn绘制直方图 | Bk1_Ch12_01.ipynb

```
# 导入包
import matplotlib.pyplot as plt
import pandas as pd
a import seaborn as sns

# 导入鸢尾花数据
b iris_sns = sns.load_dataset("iris")

# 绘制花萼长度样本数据直方图
c fig, ax = plt.subplots(figsize = (8, 6))

d sns.histplot(data=iris_sns, x="sepal_length",
               binwidth=0.2, ax = ax)
# 纵轴三个选择：频数、概率、概率密度
e ax.axvline(x = iris_sns.sepal_length.mean(),
             color = 'r', ls = '--')
# 增加均值位置竖直参考线
```



如代码12.2所示，利用seaborn.histplot()绘制鸢尾花数据直方图时，如果指定hue = 'species'，我们便得到每个类别鸢尾花单独的直方图，具体如图12.3所示。

seaborn.histplot() 还可以用来绘制二维直方热图，本章后文将介绍。此外，本章配套的Jupyter Notebook还给出了函数的其他用法。

图12.3中直方图纵轴为概率密度值。

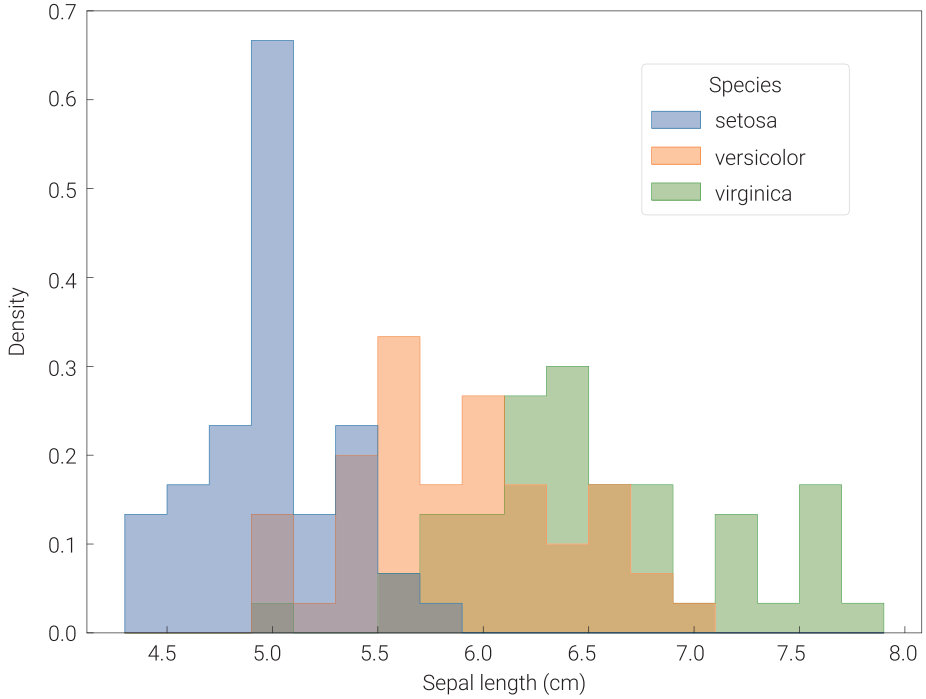


图12.3 鸢尾花花萼长度样本数据直方图 (考虑鸢尾花分类，纵轴为概率密度) | Bk1_Ch12_01.ipynb

代码12.2　用Seaborn绘制直方图（考虑鸢尾花分类，使用时须配合前文代码）| Bk1_Ch12_01.ipynb

```
# 绘制花萼长度样本数据直方图，考虑鸢尾花分类
fig, ax = plt.subplots(figsize=(8,6))

a sns.histplot(data=iris_sns,  x="sepal_length",
               hue='species', binwidth=0.2, ax=ax,
               element="step", stat='density')
# 纵轴为概率密度
```

在直方图中，以下是频数、概率和概率密度的确切定义。

直方图中每个区间内的样本数量被称为**频数** (frequency)，表示数据落入该区间的次数或计数。

概率 (probability) 是指某个事件发生的可能性。在直方图中，可以将频数除以总观测值的数量，得到每个区间的概率。这样计算得到的概率表示该区间中的观测值出现的相对概率。

概率密度 (probability density)：是指在概率分布函数中某一点附近单位自变量取值范围内的概率。在直方图中，概率密度可以通过将每个区间的概率除以该区间的宽度得到。概率密度函数描述了变量的分布形状，而不是具体的概率值。

《统计至简》第1章专门讲解直方图、偏态、峰度等概念。

直方图可以显示数据的分布形状，如**对称** (symmetry)、**偏态** (skewness)、**峰度** (kurtosis) 等，以及数据的中心趋势和离散程度。通过观察直方图，我们可以直观地了解数据的分布特征，如数据的集中程度、范围和**异常值** (outlier) 等。

核密度估计KDE

核密度估计 (Kernel Density Estimation，KDE) 是一种非参数方法，用于估计连续变量的**概率密度函数** (Probability Density Function，PDF)。它通过将每个数据点视为一个核函数 (通常是高斯核函数)，在整个变量范围内生成一系列核函数，然后将这些核函数进行平滑和叠加，从而得到连续的概率密度估计曲线。具体原理如图12.4所示。

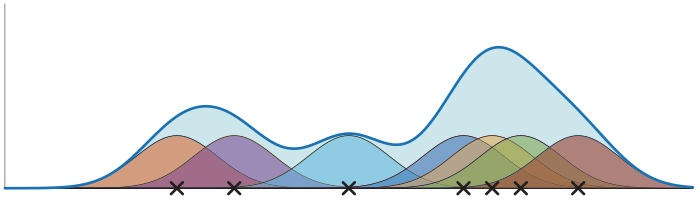


图12.4　高斯核密度估计原理

核密度估计的目标是通过在数据点附近生成高斯分布的核函数，捕捉数据的分布特征和结构。具体地说，每个数据点的核函数会在其附近产生一个小的高斯分布，然后将所有核函数叠加在一起。通过调整核函数的带宽参数，可以控制估计曲线的平滑程度和敏感度。

图12.5所示为利用seaborn.kdeplot() 绘制的鸢尾花花萼长度数据高斯核密度估计PDF。可以这样理解，图12.5是图12.2中直方图的“平滑”处理结果。

本书第27章介绍如何使用Statsmodels中的核密度估计函数；《统计至简》第17章专门讲解核密度估计原理。

图12.5的横轴还有用seaborn.rugplot() 绘制的毛毯图。毛毯图常用于展示数据在一维空间上的分布。它通过在坐标轴上绘制短线，或称为“毛毯”，表示数据点的位置和密度。这种图形通常用于辅助其他类型的图表，如直方图或密度图，以更清晰地显示数据的分布特征。

在用seaborn.kdeplot() 绘制花萼长度样本数据核密度估计曲线时，我们还可以用hue来绘制三类鸢尾花种类各自的分布，具体如图12.6所示。

换个角度理解图12.6，图12.6中三条曲线叠加便得到图12.5。图12.7更好地解释了这一点。用seaborn.kdeplot() 绘制这幅图时，需要设置multiple="stack"。大家可能已经发现，图12.5、图12.7曲线并不完全相同，这是因为高斯核密度估计曲线和带宽参数有关。《统计至简》会讲到这一点。

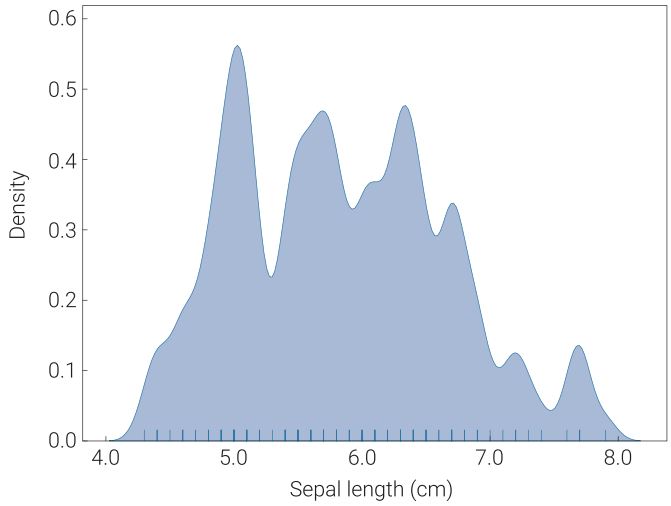


图12.5　鸢尾花花萼长度样本数据核密度估计 | Bk1_Ch12_01.ipynb

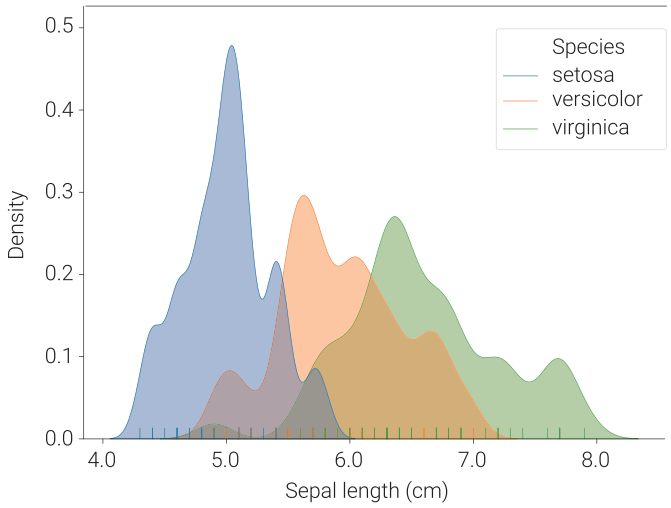


图12.6　鸢尾花花萼长度样本数据核密度估计 (考虑鸢尾花分类) | Bk1_Ch12_01.ipynb

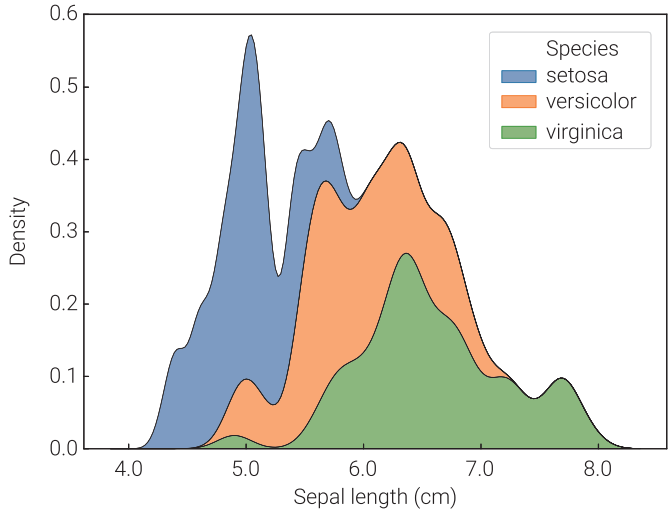


图12.7　三条KDE曲线叠加 | Bk1_Ch12_01.ipynb

特别地，在利用绘制核密度估计曲线时，如果设置multiple = 'fill'，我们便获得图12.8。图中每条曲线准确来说，都是“**后验概率** (posterior)”。而这个后验概率值可以用来完成分类。

想要理解后验概率这个概念，需要大家深入理解贝叶斯定理，《统计至简》第18、19章专门介绍利用贝叶斯定理完成分类。

也就是说，给定具体花萼长度，比较该点处红蓝绿三条曲线对应的宽度，最宽的曲线对应的鸢尾花种类可以作为该点的鸢尾花分类预测值。因此，这个后验概率值也叫“**成员值** (membership score)”。

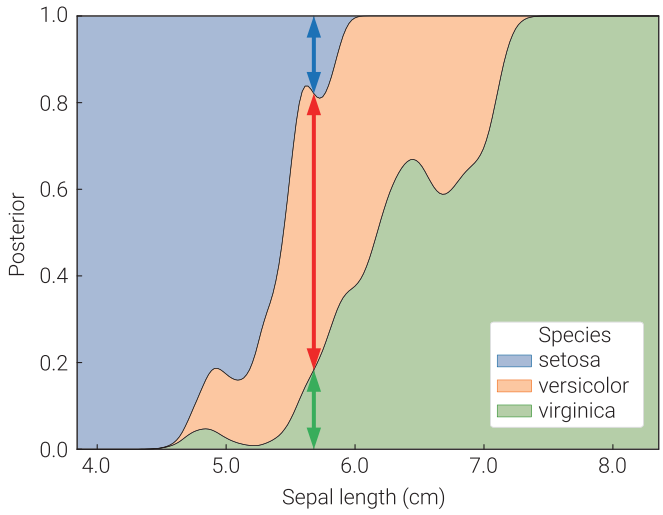


图12.8　后验概率曲线

我们可以通过代码12. 3绘制图12.5 ~ 图12.8，请大家自行分析这段代码，并逐行注释。

代码12.3　用Seaborn绘制高斯核密度估计（使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
# 绘制花萼长度样本数据，高斯核密度估计
fig, ax = plt.subplots(figsize=(8,6))
sns.kdeplot(data=iris_sns, x='sepal_length',
            bw_adjust=0.3, fill=True)
sns.rugplot(data=iris_sns, x='sepal_length')

# 绘制花萼长度样本数据，高斯核密度估计，考虑鸢尾花类别
fig, ax = plt.subplots(figsize=(8,6))
sns.kdeplot(data=iris_sns, x='sepal_length', hue='species',
            bw_adjust=0.5, fill=True)
sns.rugplot(data=iris_sns, x='sepal_length', hue='species')

# 绘制花萼长度样本数据，高斯核密度估计，考虑鸢尾花类别，堆叠
fig, ax = plt.subplots(figsize=(8,6))
sns.kdeplot(data=iris_sns, x='sepal_length', hue='species',
            multiple='stack', bw_adjust=0.5)

# 绘制后验概率(成员值)
fig, ax = plt.subplots(figsize=(8,6))
sns.kdeplot(data=iris_sns, x='sepal_length',
            hue='species', bw_adjust=0.5, multiple='fill')
```

什么是贝叶斯定理?

贝叶斯定理是一种用于更新概率推断的数学公式。它描述了在获得新信息后如何更新我们对某个事件发生概率的信念。贝叶斯定理基于先验概率（我们对事件发生的初始信念）和条件概率（给定新信息的情况下事件发生的概率），通过计算后验概率（在获得新信息后事件发生的概率）来实现更新。贝叶斯定理在统计学、机器学习和人工智能等领域具有广泛应用。

分散点图

分散点图 (strip plot) 一般用来可视化一组分类变量与连续变量的关系。在分散图中，每个数据点通过垂直于分类变量的轴上的一个点表示，连续变量的取值则沿着水平轴展示。

这种图形通常用于可视化分类变量和数值变量之间的关系，以观察数据的分布、聚集和离散程度，同时也可以用于比较不同分类变量水平下的数值变量。

代码12.4中的seaborn.stripplot() 是 Seaborn 库中用于绘制分散点图的函数。需要注意的是，分散点图适用于较小的数据集，当数据点重叠较多时，可考虑使用 seaborn.swarmplot() 函数来避免重叠点问题。我们可以通过代码12.4绘制图12.9。

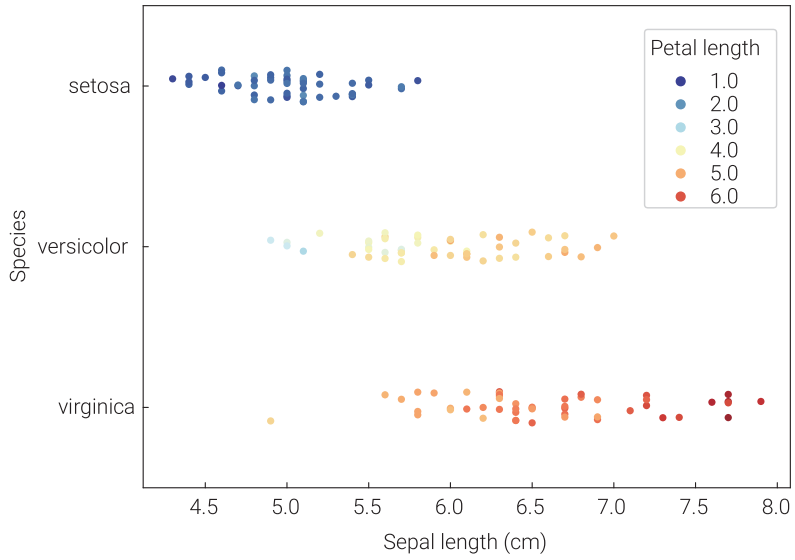


图12.9　分散点图 | Bk1_Ch12_01.ipynb

代码12.4　用Seaborn绘制分散点图（考虑鸢尾花分类，使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
# 绘制鸢尾花花萼长度分散点图
fig, ax = plt.subplots(figsize=(8,6))
sns.stripplot(data=iris_sns, x='sepal_length', y='species',
              hue='petal_length', palette='RdYlBu_r', ax=ax)
```

蜂群图

蜂群图 (swarm plot) 是一种用于可视化分类变量和数值变量关系的图表类型。它通过在分类轴上对数据进行分散排列，避免数据点的重叠，以展示数值变量在不同类别下的分布情况。每个数据点在分类轴上的位置表示其对应的数值大小，从而呈现出数据的密度和分布趋势。

蜂群图可以帮助我们比较不同类别之间的数值差异和趋势，适用于数据探索、特征分析和可视化报告等场景。

图 12.10所示为利用seaborn.swarmplot() 绘制的蜂群图。图12.11所示为考虑鸢尾花分类的蜂群图。请大家自行分析代码12.5。

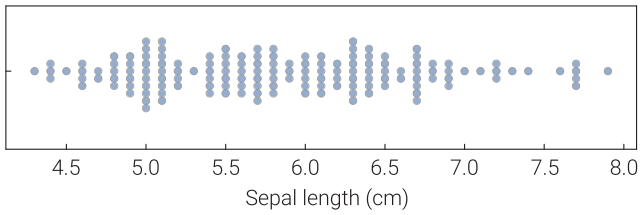


图12.10　蜂群图 | Bk1_Ch12_01.ipynb

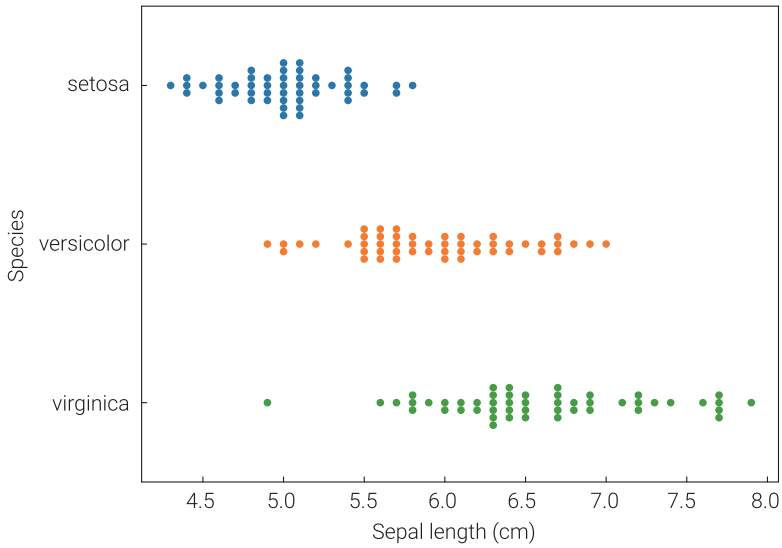


图12.11　蜂群图 (考虑鸢尾花分类) | Bk1_Ch12_01.ipynb

代码12.5　用Seaborn绘制蜂群图（使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
# 绘制花萼长度样本数据，蜂群图
fig, ax = plt.subplots(figsize=(8,4))
sns.swarmplot(data=iris_sns, x="sepal_length", ax=ax)

# 绘制花萼长度样本数据，蜂群图，考虑分类
fig, ax = plt.subplots(figsize=(8,4))
sns.swarmplot(data=iris_sns, x="sepal_length", y='species',
              hue='species', ax=ax)
```

箱型图

箱型图 (box plot) 是一种常用的统计图表，用于展示数值变量的分布情况和异常值检测。

箱型图通过绘制数据的五个关键统计量，即**最小值** (minimum)、**第一四分位数** (first quartile) Q_1、**中位数** (median, second quartile) Q_2、**第三四分位数** (third quartile) Q_3、**最大值** (maximum) 以及可能存在的异常值来提供对数据的直观概览，如图12.12所示。

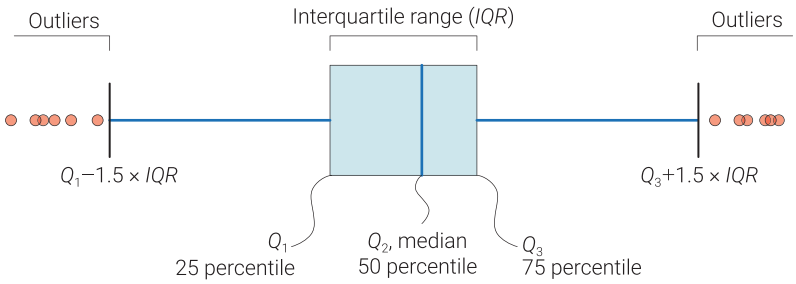


图12.12　箱型图原理

什么是四分位?

四分位是统计学中用于描述数据集分布的概念，将数据按大小顺序分成四等份。第一个四分位数 Q_1 表示25%的数据小于或等于它，第二个四分位数 Q_2 是中位数，表示50%的数据小于或等于它，第三个四分位数 Q_3 表示75%的数据小于或等于它。四分位可以帮助了解数据的中心趋势、分散程度和异常值。四分位与盒须图、离群值检测等统计分析方法密切相关。

图12.13所示为利用seaborn.boxplot() 绘制的鸢尾花花萼长度样本数据的箱型图。图12.14所示为考虑鸢尾花分类的箱型图。

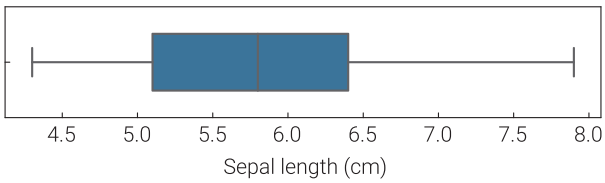


图12.13　箱型图 | Bk1_Ch12_01.ipynb

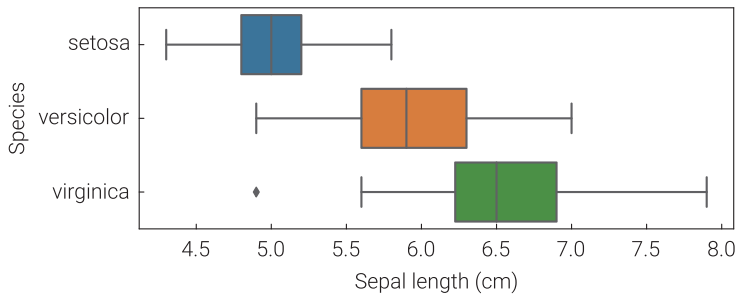


图12.14　箱型图 (考虑鸢尾花分类)

箱型图的主要元素包括以下几个。

- **箱体** (box)：由第一四分位数Q_1和第三四分位数Q_3之间的数据范围组成。箱体的高度表示数据的四分位距$IQR = Q_3 - Q_1$，箱体的中线表示数据的中位数。
- **须** (whisker)：延伸自箱体的线段，表示数据的整体分布范围。通常，须的长度为 1.5 倍的四分位距。但是，仔细观察图12.13，我们会发现用Seaborn绘制的箱型图左须距离Q_1、右须距离Q_3宽度并不相同。根据Seaborn的技术文档，左须、右须延伸至该范围 $[Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR]$ 内最远的样本点，具体如图12.15所示。更为极端的样本会被标记为异常值。
- **异常值** (outliers)：范围 $[Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR]$ 之外的数据点，被认为是异常值，可能表示数据中的极端值或异常观测。

通过观察箱型图，可以快速了解数据的中心趋势、离散程度以及是否存在异常值等关键信息。

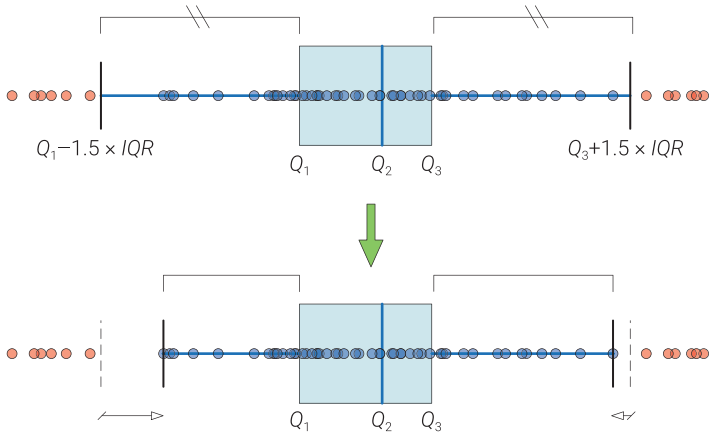


图12.15　Seaborn绘制箱型图左须、右须位置

请大家自行分析代码12.6。

代码12.6　用Seaborn绘制箱型图（使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
# 绘制鸢尾花花萼长度箱型图
fig, ax = plt.subplots(figsize = (8,2))
sns.boxplot(data = iris_sns, x='sepal_length', ax = ax)

# 绘制鸢尾花花萼长度箱型图，考虑鸢尾花分类
fig, ax = plt.subplots(figsize = (8,3))
sns.boxplot(data = iris_sns, x = 'sepal_length',
            y = 'species', ax = ax)
```

小提琴图

小提琴图 (violin plot) 是一种用于可视化数值变量分布的图表类型。它结合了核密度估计曲线和箱型图的特点，可以同时展示数据的分布形状、**中位数** (median)、**四分位数** (quartile) 和**离群值** (outlier) 等信息。seaborn.violinplot() 是 Seaborn 库中用于绘制小提琴图的函数。

小提琴图的主要组成部分包括以下几个。

- **背景形状：**由核密度估计曲线组成，表示数据在不同值上的概率密度。
- **中位数线：**位于核密度估计曲线的中间位置，表示数据的中位数。
- **四分位线：**分别位于核密度估计曲线的 25% 和 75% 位置，表示数据的四分位范围。
- **离群值点：**位于核密度估计曲线之外的离群值数据点。

图12.16所示为用seaborn.violinplot() 绘制的鸢尾花花萼长度样本数据的小提琴图。图12.17为考虑鸢尾花分类的小提琴图。图12.18所示为“蜂群图 + 小提琴图”的可视化方案。

请大家自行分析代码12.7。

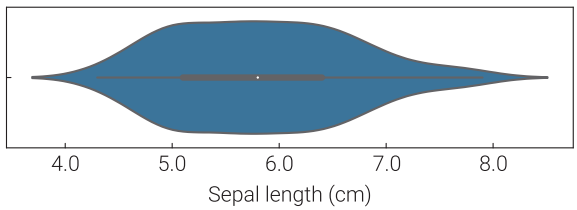


图12.16　小提琴图 | Bk1_Ch12_01.ipynb

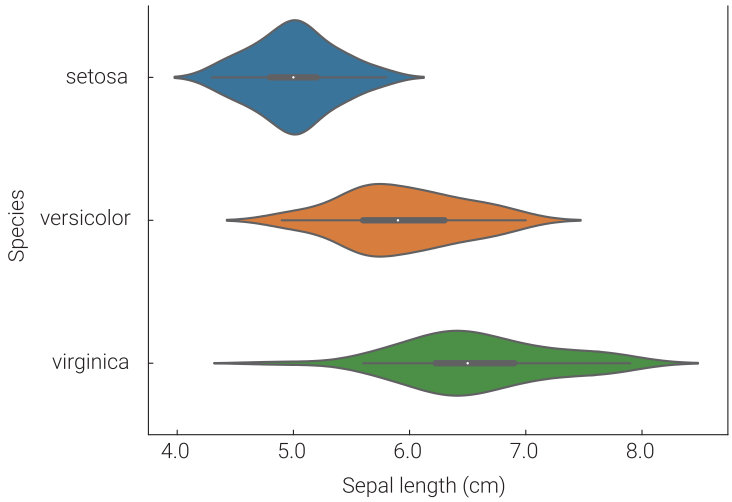


图12.17　小提琴图 (考虑鸢尾花分类) | Bk1_Ch12_01.ipynb

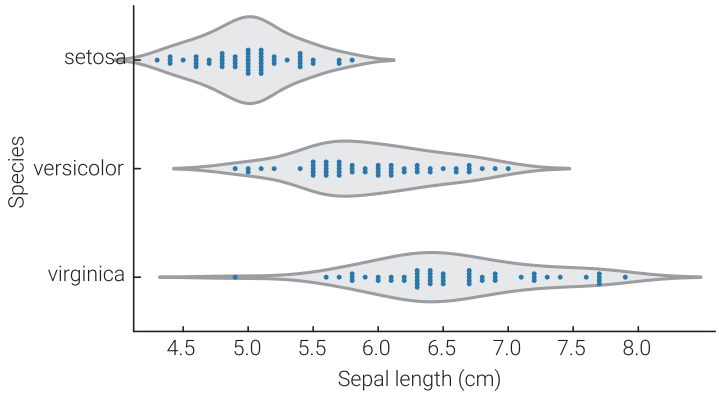


图12.18　蜂群图 + 小提琴图 (考虑鸢尾花分类) | Bk1_Ch12_01.ipynb

代码12.7　用Seaborn绘制小提琴图（使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
# 绘制花萼长度样本数据，小提琴图
fig, ax = plt.subplots(figsize=(8,2))
sns.violinplot(data=iris_sns, x='sepal_length', ax=ax)

# 绘制花萼长度样本数据，小提琴图，考虑鸢尾花分类
fig, ax = plt.subplots(figsize=(8,4))
sns.violinplot(data=iris_sns, x='sepal_length',
               y='species', ax=ax)

# 蜂群图 + 小提琴图，考虑鸢尾花分类
sns.catplot(data=iris_sns, x='sepal_length', y='species',
            kind='violin', color='.9', inner=None)

sns.swarmplot(data=iris_sns, x='sepal_length',
              y='species', size=3)
```

12.3 二元特征数据

散点图

散点图是一种数据可视化图表，用于展示两个变量之间的关系。在坐标系中它以点的形式表示每个数据点，横轴代表一个变量，纵轴代表另一个变量。

散点图可以帮助我们观察和分析数据点之间的趋势、分布和相关性。通过观察点的聚集程度和分布形状，我们可以推断两个变量之间的关系类型，如线性正相关、线性负相关、线性无关，甚至是非线性关系。

图12.19所示为利用seaborn.scatterplot() 绘制的散点图，散点图的横轴为花萼长度，纵轴为花萼宽度。通过观察散点趋势，可以发现花萼长度、花萼宽度似乎存在线性正相关。但是实际情况可能并非如此。本章最后将通过线性相关性系数进行量化确认。

图12.19中，我们还用毛毯图分别可视化花萼长度、花萼宽度的分布情况。

用不同颜色散点代表鸢尾花分类，我们便得到图12.20所示散点图。观察这幅图中蓝色点，即setosa类，我们可以发现更强的线性正相关性。

请大家自行分析代码12.8，并逐行注释。

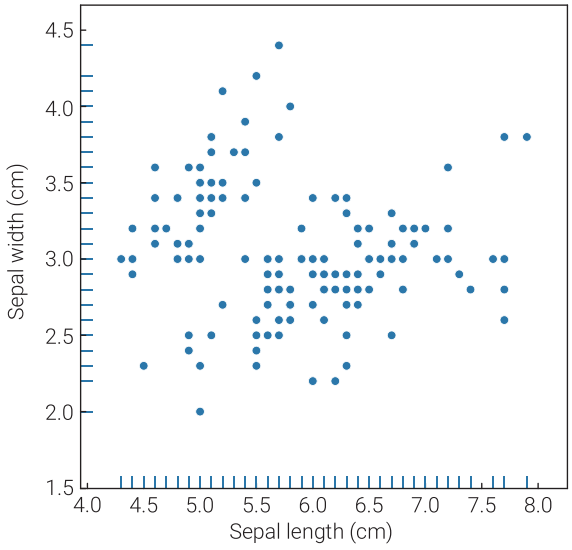


图12.19　散点图 + 毛毯图 | Bk1_Ch12_01.ipynb

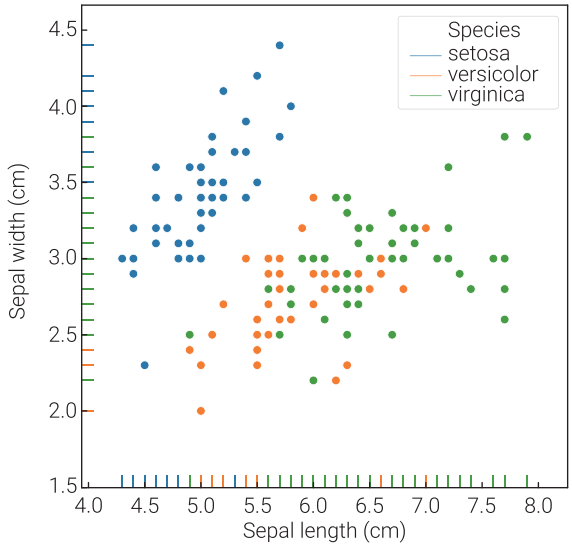


图12.20　散点图 + 毛毯图 (考虑鸢尾花分类) | Bk1_Ch12_01.ipynb

图12.32所示为几种用seaborn.scatterplot() 绘制的鸢尾花数据集散点图。

图12.32 (a) 的横轴是花萼长度，纵轴是花萼宽度。调转横纵轴特征便得到图12.32 (b)。

在图12.32 (a) 的基础上，可以用色调代表花瓣长度。这样一幅二维散点图上，我们便可视化了三个量化特征。

图12.32 (d) 在图12.32 (c) 基础上又进一步，用散点大小代表花瓣宽度。

图12.32 (e) 则用颜色可视化鸢尾花的分类标签。在此基础上，我们还可以用散点大小可视化花瓣宽度。

图12.32 (g) 则集合前几幅散点图，并且用不同标识符号代表鸢尾花分类标签。这种散点图显然“信息过载”，并不推荐。

代码12.8　用Seaborn绘制二元散点图 + 毛毯图（使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
# 鸢尾花散点图 + 毛毯图
fig, ax = plt.subplots(figsize=(4,4))

a sns.scatterplot(data=iris_sns, x='sepal_length', y='sepal_width')
b sns.rugplot(data=iris_sns, x='sepal_length', y='sepal_width')

# 鸢尾花散点图 + 毛毯图，考虑鸢尾花分类
fig, ax = plt.subplots(figsize=(4,4))

c sns.scatterplot(data=iris_sns, x='sepal_length',
                  y='sepal_width', hue='species')
d sns.rugplot(data=iris_sns, x='sepal_length',
              y='sepal_width', hue='species')
```

二元直方热图

本章前文，我们将一元样本数据划分成不同区间便绘制了一元直方图。

类似地，如果我们把图12.19所示平面划分成如图12.21所示一系列格子，计算每个格子中的样本数，我们便可以绘制类似图12.22的二元直方图。

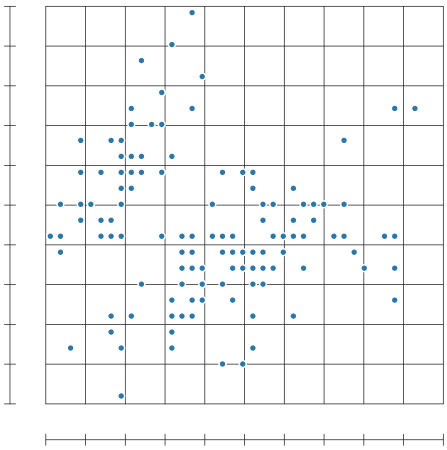


图12.21　二元直方图原理

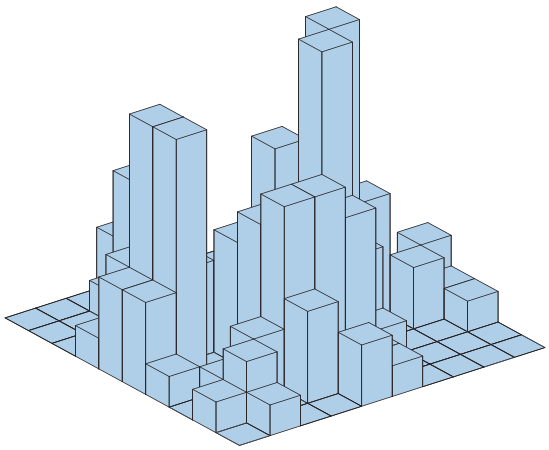


图12.22　二元直方图 (柱状图可视化方案)

显然，这种可视化方案并不理想。一方面“柱子”的高度很难确定，而且固定某个特定视角之后，一些较矮的“柱子”必定会被遮挡。因此，在实践中我们常常使用二元直方热图作为可视化方案。

二元直方热图由一个矩形网格组成，其中每个单元格的颜色代表了对应的数据频数、概率、概率密度。通常，行和列代表两个不同的随机变量，而单元格中的颜色强度表示频数、概率、概率密度。

二元直方热图可以帮助我们观察两个变量之间的关系以及它们的分布模式。通过观察颜色的变化和集中区域，我们可以得出关于两个变量之间的相关性、联合分布和潜在模式的初步结论。

图12.23所示为利用seaborn.displot() 绘制的二元直方热图，横轴为鸢尾花花萼长度，纵轴为花萼宽度。如图12.24所示，二元直方热图沿着某个方向压缩便得到一元直方图；反过来看，直方图沿着特定方向展开便得到二元直方热图。

请大家自行分析代码12.9。

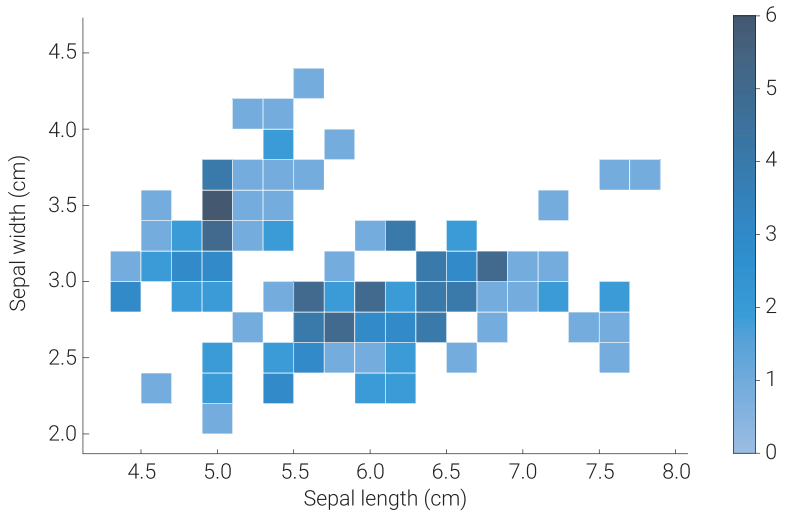


图12.23　鸢尾花花萼长度、花萼宽度的二元直方热图 | Bk1_Ch12_01.ipynb

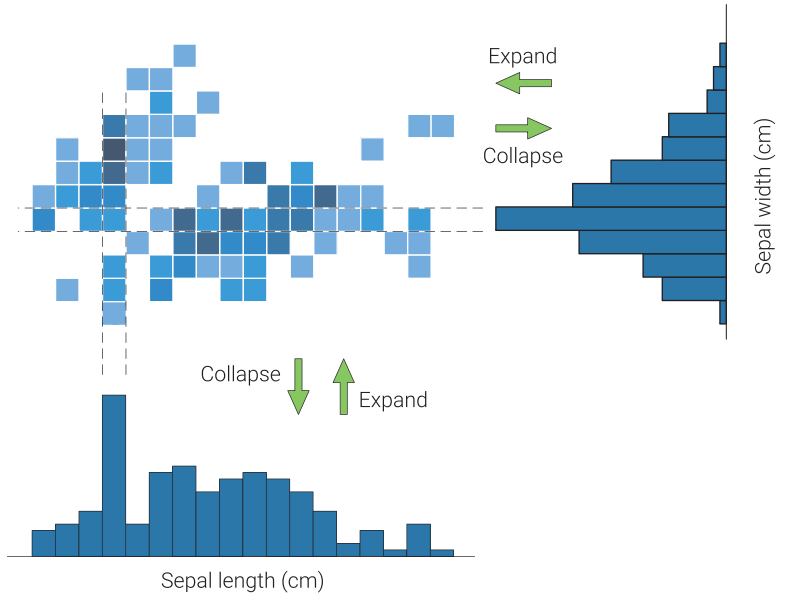


图12.24　一元直方图和二元直方热图之间的关系

代码12.9　用Seaborn绘制二元直方热图（使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
# 鸢尾花二元频数直方热图

sns.displot(data=iris_sns, x="sepal_length", y="sepal_width",
            binwidth=(0.2, 0.2), cbar=True)
```

联合分布KDE

前文的高斯核函数KDE也可以用在估算二元联合分布。图12.25所示为用seaborn.kdeplot() 绘制的鸢尾花花萼长度、花萼宽度联合分布概率密度估计等高线。图12.25 (b) 还考虑了鸢尾花三个不同类别。请大家自行分析代码12.10。

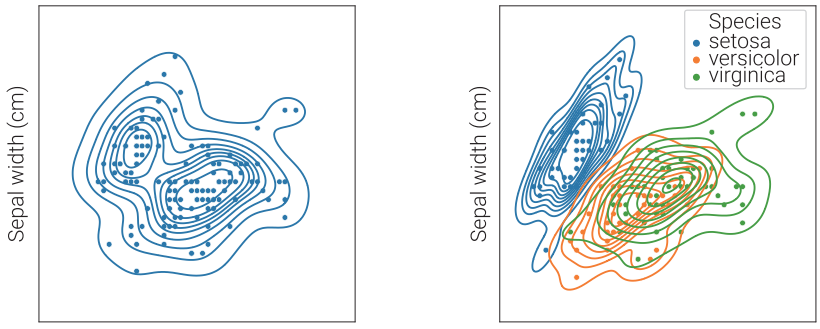


图12.25　鸢尾花花萼长度、花萼宽度的联合分布 (高斯核密度估计) | Bk1_Ch12_01.ipynb

什么是联合分布？

联合分布是统计学中用于描述两个或多个随机变量同时取值的概率分布。它提供了关于多个变量之间关系的信息，包括它们的联合概率、相互依赖程度以及共同变化的模式。联合分布可以以多种形式呈现，如概率质量函数（离散变量）或概率密度函数（连续变量）。通过分析联合分布，我们可以洞察变量之间的相关性、条件概率以及预测和推断未来事件的可能性。联合分布在概率论、统计建模、数据分析和机器学习等领域具有广泛应用。

代码12.10　用Seaborn绘制联合分布概率密度等高线（使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
# 联合分布概率密度等高线
sns.displot(data=iris_sns, x='sepal_length',
            y='sepal_width', kind='kde')

# 联合分布概率密度等高线，考虑分布
sns.kdeplot(data=iris_sns, x='sepal_length',
            y='sepal_width', hue = 'species')
```

联合分布 + 边缘分布

图12.26所示为利用seaborn.jointplot() 可视化的“联合分布 + 边缘分布”。

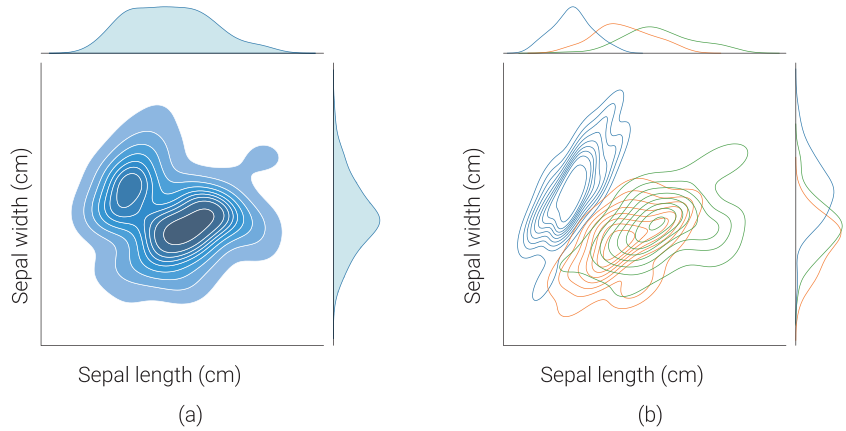


图12.26　鸢尾花花萼长度、花萼宽度的联合分布和边缘分布 | Bk1_Ch12_01.ipynb

seaborn.jointplot() 函数用于创建联合图，其结合了两个变量的散点图和各自的边缘分布图。它可以帮助我们同时可视化两个变量之间的关系以及它们的边缘分布。

seaborn.jointplot() 函数默认情况下会绘制散点图和边缘直方图。其中，散点图展示了两个变量之间的关系，而边缘直方图则分别显示了每个变量的边缘分布情况。

请大家自行分析代码12.11。

本章配套的Jupyter Notebook还提供seaborn.jointplot() 其他几种可视化方案，请大家自行学习。

什么是边缘分布？

边缘分布是指在多变量数据集中，针对单个变量的分布情况。它表示了某个特定变量在与其他变量无关时的概率分布。边缘分布可以通过将多变量数据集投影到某个特定变量的轴上来获得。通过分析边缘分布，我们可以了解每个变量单独的分布特征，包括均值、方差、偏度、峰度等统计量，以及分布的形状和模式。边缘分布对于探索数据集的特征、进行单变量分析和了解数据的单个方面非常有用。

代码12.11　用Seaborn绘制联合分布和边缘分布（使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
# 联合分布、边缘分布
sns.jointplot(data=iris_sns, x='sepal_length', y='sepal_width',
              kind='kde', fill=True)

# 联合分布、边缘分布，考虑鸢尾花分类
sns.jointplot(data=iris_sns, x='sepal_length', y='sepal_width',
              hue='species', kind='kde')
```

线性回归

图12.27所示为利用seaborn.lmplot() 绘制的鸢尾花花萼长度、花萼宽度之间的线性回归关系图。seaborn.lmplot() 函数默认情况下会绘制散点图和拟合的线性回归线。其中，散点图展示了两个变量之间的关系，而线性回归线表示了拟合的线性关系。

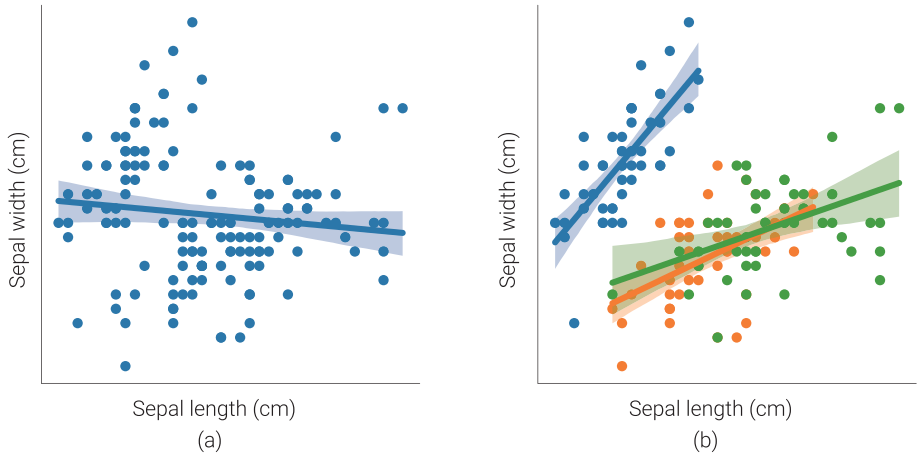


图12.27　鸢尾花花萼长度、花萼宽度的线性回归关系 | Bk1_Ch12_01.ipynb

《数据有道》第9、10章专门介绍线性回归。

除了基本语法外，seaborn.lmplot() 还支持其他参数，例如hue参数用于指定一个额外的分类变量，可以通过不同的颜色展示不同类别的数据点和回归线。请大家自行分析代码12.12。

代码12.12　用Seaborn可视化线性回归关系（使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
# 可视化线性回归关系
sns.lmplot(data=iris_sns, x='sepal_length', y='sepal_width')

# 可视化线性回归关系，考虑鸢尾花分类
sns.lmplot(data=iris_sns, x='sepal_length', y='sepal_width',
           hue = 'species')
```

12.4 多元特征数据

分散点图、小提琴图

我们当然可以使用一元可视化方案展示多元数据的特征，如图12.28所示。

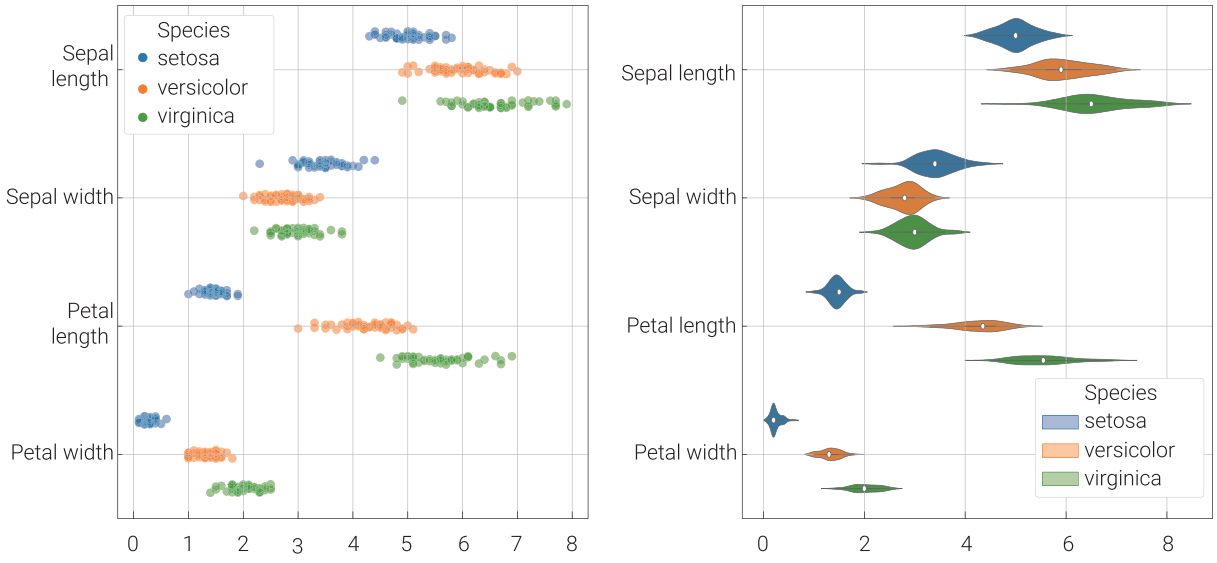


图12.28　分散点图、小提琴图 (多特征) | Bk1_Ch12_01.ipynb

我们可以通过代码12.13绘制图12.28，下面咱们讲解其中关键语句。

a利用pandas.melt()，简写作pd.melt()，将鸢尾花数据集从**宽格式** (wide format) 转换为**长格式** (long format)。

宽格式数据帧如表12.1所示，长格式数据帧如表12.2所示。函数输入'species'是要保留的标识变量，也就是不进行融合。参数var_name='measurement'指定了在融合过程中生成的新列的名称。

请大家自行分析b和c，并逐行注释。

但是图12.28中的这两幅图最致命的缺陷是仅仅展示了单个特征分布，并没有展示特征之间的联系。下面我们聊聊其他能够可视化多元特征之间关系的可视化方案。

本书第22章介绍包括pandas.melt()在内的各种常用数据帧规整方法。

代码12.13　用Seaborn绘制多特征分散点图、小提琴图（使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
a iris_melt  = pd.melt(iris_sns , 'species' , var_name ='measurement' )
  # 数据从宽格式(wide format) 转换为长格式(long format)

  # 绘制多特征分散图
b sns.stripplot (data =iris_melt , x='value' , y='measurement' ,
                 hue ='species' , dodge =True, alpha =.25,
                 zorder =1, legend =True)
  plt.grid()

  # 绘制多特征小提琴图
c sns.violinplot (data =iris_melt , x='value' , y='measurement' ,
                 hue ='species' , dodge =True, alpha =.25,
                 zorder =1, legend =True)
  plt.grid()
```

表12.1　宽格式

	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
…	…	…	…	…	…
149	5.9	3	5.1	1.8	virginica

表12.2　长格式

	species	measurement	value
0	setosa	sepal_length	5.1
1	setosa	sepal_length	4.9
2	setosa	sepal_length	4.7
…	…	…	…
599	virginica	petal_width	1.8

聚类热图

seaborn.clustermap()函数用于创建聚类热图，它能够可视化数据集中的聚类结构和相似性。聚类热图使用层次聚类算法对数据进行聚类，并以热图的形式展示聚类结果。

聚类热图的原理是通过计算数据点之间的相似性（例如欧几里得距离或相关系数），然后使用**层次聚类** (hierarchical clustering) 算法将相似的数据点分组为聚类簇。层次聚类将数据点逐步合并形成聚类树状结构，根据相似性的距离进行聚类的层次化过程。聚类热图将聚类树状结构可视化为热图，同时显示数据点的排序和聚类关系。

代码12.14利用.iloc[:,:-1] 方法索引和切片数据帧。简单来说，方法iloc是Pandas DataFrame的索引器之一，用于按照整数位置进行选择。第一个冒号代表所有行，:-1表示选择除了最后一列之外的所有列。

本书第21章专门介绍Pandas数据帧索引和切片。

《机器学习》专门介绍各种聚类算法。

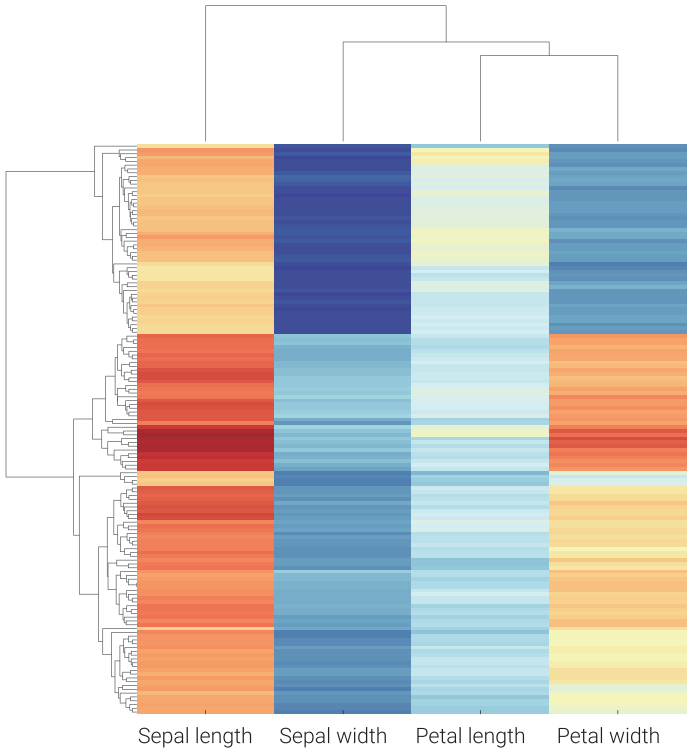


图12.29　鸢尾花数据集，聚类热图 | Bk1_Ch12_01.ipynb

代码12.14　用Seaborn绘制聚类热图（使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
# 聚类热图
sns.clustermap (iris_sns .iloc[:,:-1], cmap = 'RdYlBu_r' ,
                vmin = 0, vmax = 8)
```

什么是聚类?

机器学习中的聚类是一种无监督学习方法，用于将数据集中的样本按照相似性进行分组或聚集。聚类算法通过自动发现数据的内在结构和模式，将相似的样本归为一类，从而实现数据的分组和分类。聚类的目标是使得同一类别内的样本相似度高，而不同类别之间的样本相似度低。聚类算法通常基于样本之间的距离或相似性度量进行操作，如欧几里得距离、余弦相似度等。常见的聚类算法包括K均值聚类、层次聚类、DBSCAN、高斯混合模型等。

成对特征散点图

seaborn.pairplot() 函数用于创建成对特征散点图矩阵，可视化多个变量之间的关系和分布。它会将数据集中的每对特征绘制为散点图，并展示变量之间的散点关系和单变量的分布。

代码12.15中，seaborn.pairplot() 函数会根据数据集中的每对特征生成散点图，并以网格矩阵的形式展示。对角线上的图形通常是单变量的直方图或核密度估计图，表示每个变量的分布情况。非对角线上的图形是两个变量之间的散点图，展示它们之间的关系。

此外，seaborn.pairplot()函数还支持其他参数，例如hue参数用于根据一个分类变量对散点图进行颜色编码，使不同类别的数据点具有不同的颜色。

通过使用seaborn.pairplot()函数，我们可以轻松地可视化多个变量之间的关系和分布。这对于探索变量之间的相关性、识别数据中的模式和异常值等非常有用。

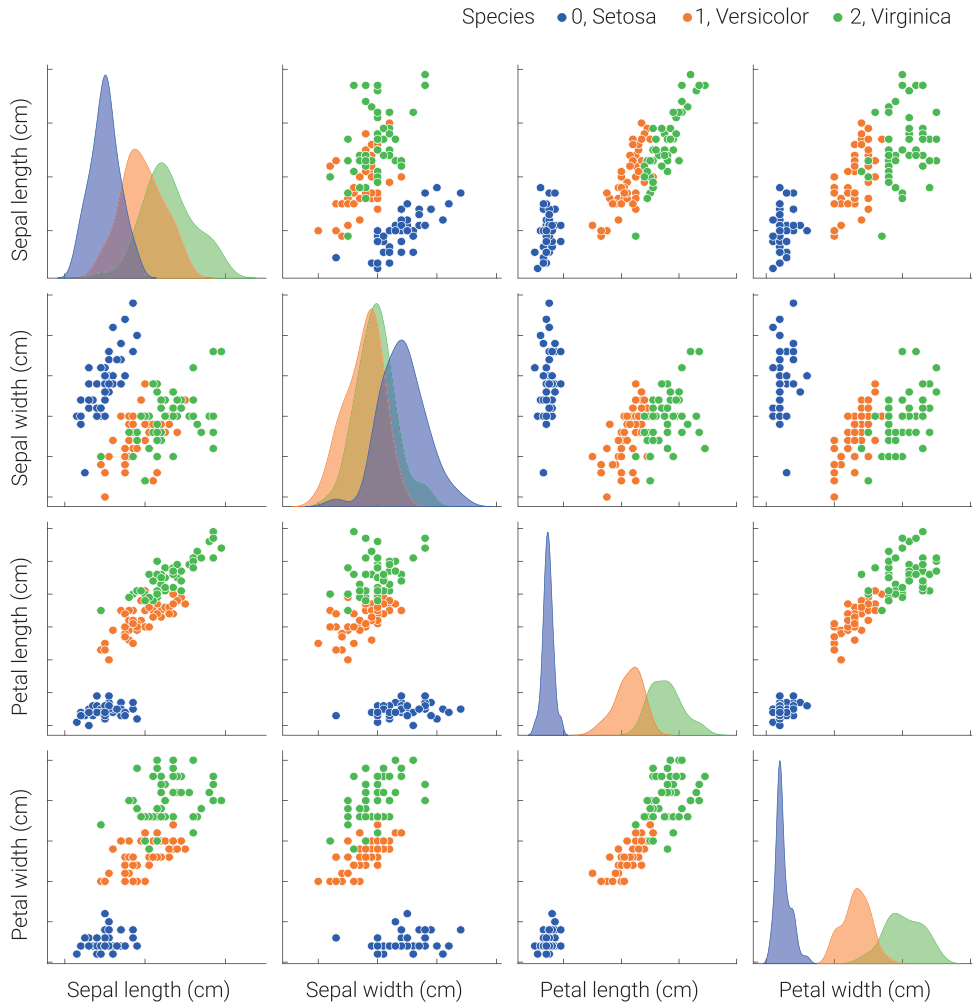


图12.30　鸢尾花数据成对特征散点图 (考虑分类标签) | Bk1_Ch12_01.ipynb

代码12.15　用Seaborn绘制成对特征散点图（使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
# 绘制成对特征散点图
sns.pairplot(iris_sns, hue = 'species')
```

平行坐标图

平行坐标图 (parallel coordinates plot) 是一种可视化多个连续变量之间关系的图形方法。它使用平行的垂直线段来表示每个变量，这些线段相互平行并沿着水平轴排列。每个变量的值通过垂直线段在对应的轴上进行表示。

在平行坐标图中，每个数据样本由一条连接不同垂直线段的折线表示。这条折线的形状和走势反映了数据样本在不同变量之间的关系。通过观察折线的走势，我们可以识别出变量之间的相对关系，如正相关、负相关或无关系。同时，我们也可以通过折线的位置和形状来比较不同样本之间的差异。

平行坐标图常用于数据探索、特征分析和模式识别等任务。它能够帮助我们发现多个变量之间的关系、观察变量的分布模式，并对数据样本进行可视化比较。此外，通过添加颜色映射或其他可视化元素，还可以在平行坐标图中显示附加信息，如类别标签或异常值指示。

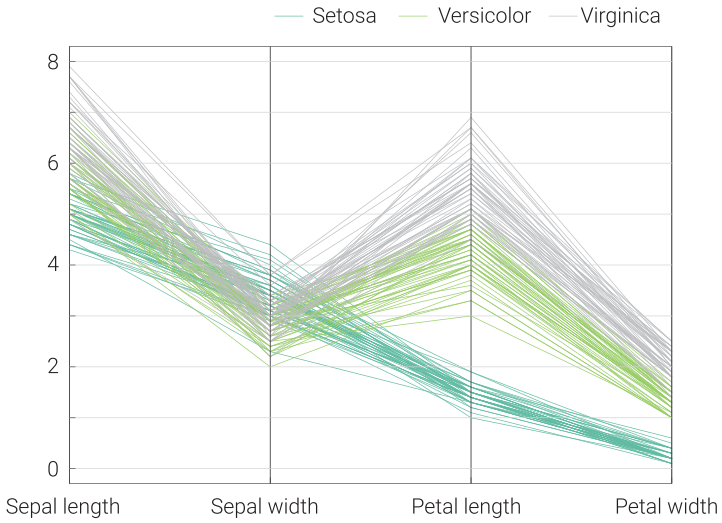


图12.31　鸢尾花数据，平行坐标图 | Bk1_Ch12_01.ipynb

目前Seaborn并没有绘制平行坐标图的工具，本章配套的Jupyter Notebook中采用的是pandas.plotting.parallel_coordinates() 函数。

代码12.16　用Pandas绘制平行坐标图（使用时配合前文代码）| Bk1_Ch12_01.ipynb

```
a from pandas.plotting import parallel_coordinates
  # 可视化函数来自Pandas
  # 绘制平行坐标图
b parallel_coordinates(iris_sns, 'species',
                       colormap=plt.get_cmap("Set2"))
  plt.show()
```

类似平行坐标图的可视化方案还有**安德鲁斯曲线** (Andrews curves)。在安德鲁斯曲线中，每个特征被映射为一个三角函数（通常是正弦函数和余弦函数），并按照给定的顺序排列。本章配套的Jupyter Notebook也用pandas.plotting.andrews_curves() 绘制了鸢尾花样本数据的安德鲁斯曲线。

量化多特征样本数据任意两个随机变量关系的最方便的工具莫过于协方差矩阵、相关系数矩阵。这是上一章已经介绍过的内容，本章不再赘述。

请大家完成以下题目。

Q1. 分别绘制鸢尾花花萼宽度、花瓣长度、花瓣宽度的直方图、KDE概率密度估计。

Q2. 绘制鸢尾花花萼长度、花瓣长度的散点图、二元直方热图、联合分布KDE等高线。

Q3. 自行学习本章配套代码Bk1_Ch12_02.ipynb。

* 本章题目不提供答案。

本章介绍了Seaborn库，这个库特别适合统计可视化。和Matplotlib一样，Seaborn也是提供静态可视化方案。

此外，Plotly也有大量统计可视化方案，而且都具有交互属性。本书第23、24章将结合Pandas介绍Plotly的统计可视化工具。

本书专门介绍可视化的板块到此结束。《可视之美》将介绍更多可视化方案。

下一板块将用6章内容专门介绍NumPy。

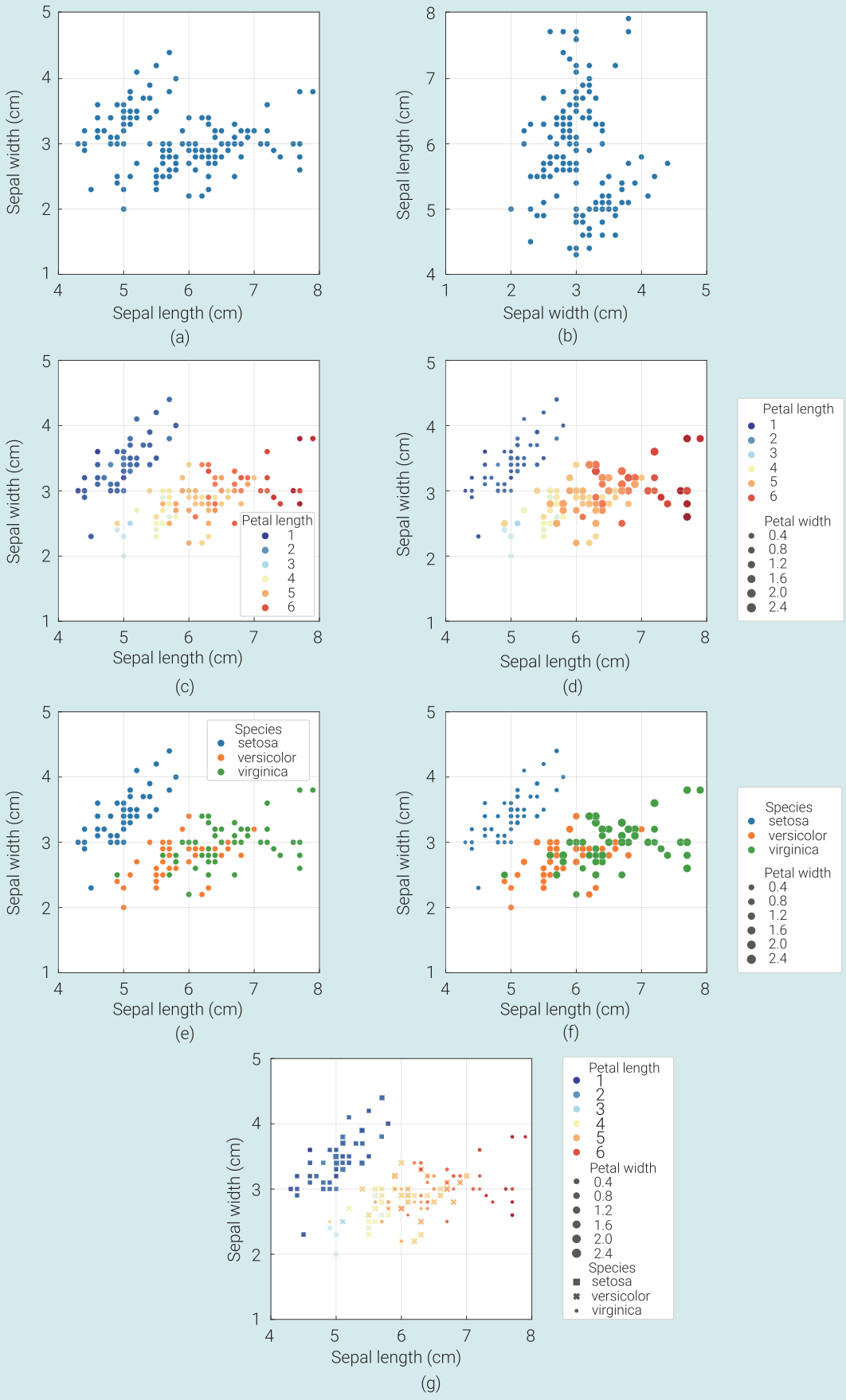
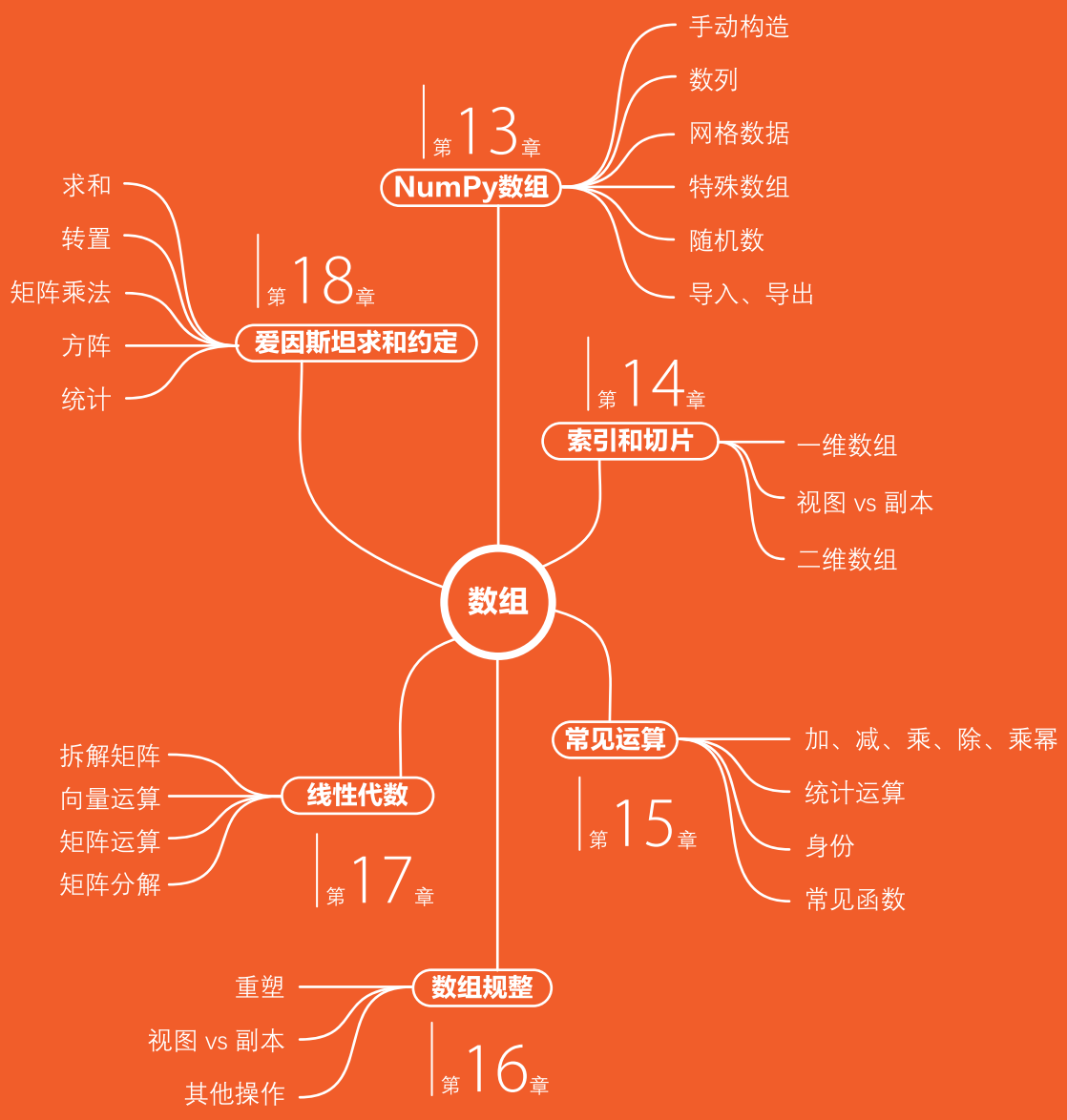


图12.32　几种用seaborn绘制的二元散点图 | Bk1_Ch12_02.ipynb

04 Section 04 数 组



学习地图 | 第4板块

13 Fundamentals of NumPy

聊聊NumPy

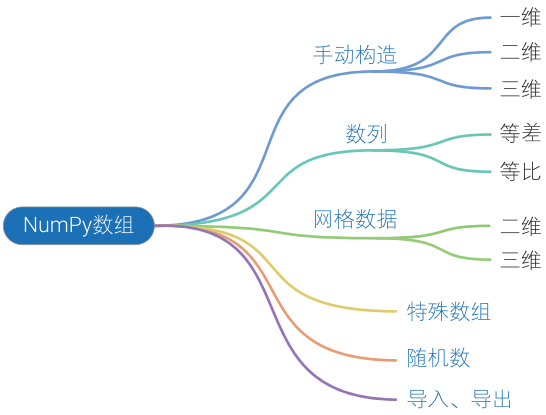
本节的核心是用NumPy产生不同类型数组

重要的不是生命的长度，而是深度。

It is not the length of life, but the depth.

—— 拉尔夫·沃尔多·爱默生 (Ralph Waldo Emerson) | 美国思想家、文学家 | 1803 — 1882年

- `math.ceil()` 向上取整
- `matplotlib.cm` 是Matplotlib中的一个模块，用于颜色映射
- `matplotlib.patches.Circle()` 创建正圆图形
- `matplotlib.pyplot.contour()` 绘制等高线图
- `matplotlib.pyplot.contourf()` 绘制填充等高线图
- `matplotlib.pyplot.scatter()` 绘制散点图
- `numpy.arange()` 根据指定的范围以及步长，生成一个等差数组
- `numpy.array()` 创建array数据类型
- `numpy.empty()` 创建指定形状的NumPy空（未初始化）数组
- `numpy.empty_like()` 创建一个与给定输入数组具有相同形状的未初始化数组
- `numpy.exp()` 计算括号中元素的自然指数
- `numpy.eye()` 用于创建单位矩阵
- `numpy.full()` 创建一个指定形状且所有元素值相同的数组
- `numpy.full_like()` 创建一个与给定输入数组具有相同形状且所有元素值相同的数组
- `numpy.linspace()` 在指定的间隔内，返回固定步长等差数列
- `numpy.logspace()` 创建在对数尺度上均匀分布的数组
- `numpy.meshgrid()` 创建网格化坐标数据
- `numpy.ones_like()` 用来生成和输入矩阵形状相同的全1矩阵
- `numpy.random.multivariate_normal()` 用于生成多元正态分布的随机样本
- `numpy.random.uniform()` 产生满足连续均匀分布的随机数
- `numpy.zeros()` 返回给定形状和类型的新数组，用零填充
- `numpy.zeros_like()` 用来生成和输入矩阵形状相同的零矩阵
- `seaborn.heatmap()` 绘制热图



13.1 什么是NumPy?

简单来说，NumPy是Python科学计算中非常重要的一个库，它提供了快速、高效的多维数组对象及其操作方法，是众多其他科学计算库的基础。下面展开聊聊NumPy的主要功能。

NumPy最重要的功能之一是提供了高效的多维数组对象ndarray，可以用来表示向量、矩阵和更高维的数组。它是Python中最重要的科学计算数据结构，支持广泛的数值运算和数学函数操作。

此外，如果大家需要处理有标签、多维数组数据的话，推荐使用Xarray。Xarray可以看作是在ndarray的基础上，增加了标签和元数据的功能。Xarray可以对多个数组进行向量化计算，避免了循环操作，提高了计算效率。此外，Xarray提供了多种统计分析函数，可以方便地对多维数组数据进行统计分析。本书不会展开讲解Xarray。

NumPy提供了多种数组操作方法，包括数组索引、切片、迭代、转置、变形、合并等，以及广播 (broadcasting) 机制，这些使得数组操作更加方便、高效。这些话题是本书后续要展开讲解的内容。

《数学要素》一册大量使用这些函数库来可视化常见函数。

本书中会简要介绍这些常见线性代数操作，详细讲解请大家参考《矩阵力量》一册。

NumPy提供了丰富的数学函数库，包括三角函数、指数函数、对数函数、逻辑函数、统计函数、随机函数等，这些能够满足大多数科学计算需要。

NumPy支持多种文件格式的读写操作，包括文本文件、二进制文件、CSV文件等。NumPy基于C语言实现，因此可以利用底层硬件优化计算速度，同时还支持多线程、并行计算和向量化操作，这些使得计算更加高效。

NumPy提供了丰富的线性代数操作方法，包括矩阵乘法、求逆矩阵、特征值分解、奇异值分解等，因此可以方便地解决线性代数问题。

NumPy可以与Matplotlib库集成使用，方便地生成各种图表，如线图、散点图、柱状图等。相信大家在本书前文已经看到基于NumPy数据绘制的二维、三维图像。

NumPy提供了一些常用的数据处理方法，如排序、去重、聚合、统计等，以方便对数据进行预处理。即便如此，“鸢尾花书”中我们更常用Pandas处理数据，本书后续将专门介绍Pandas。

Python中许多数据分析和机器学习的库都是基于NumPy创建的。Scikit-Learn是一个流行的机器学习库，它基于NumPy、SciPy和Matplotlib创建，提供了各种机器学习算法和工具，如分类、回归、聚类、降维等。

PyTorch是一个开源的机器学习框架，它基于NumPy创建，提供了张量计算和动态计算图等功能，可以用于构建神经网络和其他机器学习算法。

TensorFlow是一个深度学习框架，它基于NumPy创建，提供了各种神经网络算法和工具，包括卷积神经网络、循环神经网络等。

《数据有道》专门讲解回归、降维这两类机器学习算法，而《机器学习》一册则侧重于分类、聚类。

本节配套的Jupyter Notebook文件主要是Bk1_Ch13_01.ipynb，请大家边读正文边在JupyterLab中探究学习。

13.2 手动构造数组

从numpy.array() 说起

我们可以利用numpy.array() 手动生成一维、二维、三维等数组。下面首先介绍如何使用numpy.array() 这个函数。如图13.1所示。

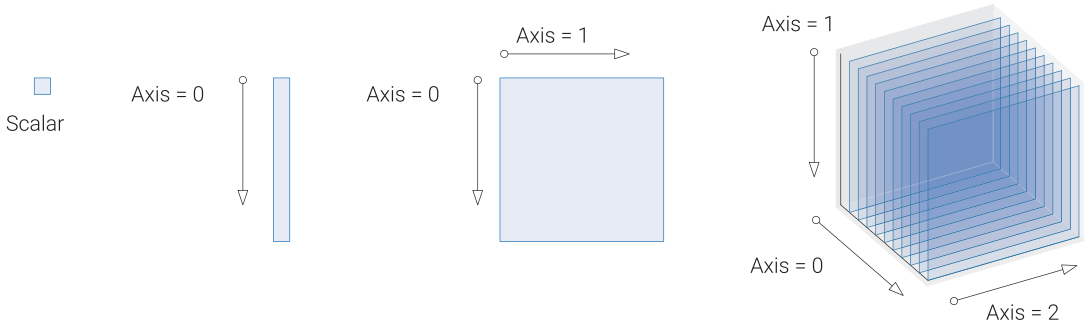


图13.1　标量、一维数组、二维数组、三维数组

numpy.array(object, dtype)

这个函数的重要输入参数：

- object 会转换为数组的输入数据，object可以是列表、元组、其他数组或类似序列的对象。
- dtype参数用于指定数组的数据类型。如果不指定dtype参数，则NumPy会自动推断数组的数据类型。

请大家在JupyterLab中自行学习下例。

```
import numpy as np

# 从列表中创建一维数组
arr1=np.array([1, 2, 3, 4])

# 指定数组的数据类型
arr2=np.array([1, 2, 3, 4], dtype=float)

# 从元组中创建二维数组
arr3=np.array([(1, 2, 3), (4, 5, 6)])

# 指定最小维度
arr4=np.array([1, 2, 3, 4], ndmin=2)
```

NumPy中的array是什么？

在NumPy中，array是一种多维数组对象，它可以用于表示和操作向量、矩阵和张量等数据结构。array是NumPy中最重要的数据结构之一，它支持高效的数值计算和广播操作，可以用于处理大规模数据集和科学计算。与Python中的列表不同，array是一个固定类型、固定大小的数据结构，它可以支持多维数组操作和高性能数值计算。array的每个元素都是相同类型的，通常是浮点数、整数或布尔值等基本数据类型。在创建array时，用户需要指定数组的维度和类型。例如，可以使用numpy.array() 函数创建一个一维数组或二维数组，也可以使用numpy.zeros() 函数或numpy.ones() 函数创建指定大小的全0或全1数组，还可以使用numpy.random模块生成随机数组等。除了基本操作之外，NumPy还提供了许多高级的数组操作，如数组切片、数组索引、数组重塑、数组转置、数组拼接和分裂等。`

代码13.1和代码13.2定义了两个可视化函数。

下面，让我们首先讲解代码13.1。

ⓐ从Matplotlib中导入cm模块。cm模块提供了许多预定义的颜色映射和相关方法。

ⓑ自定义可视化函数用来展示二维数组。

ⓒ中array.shape[1] 返回数组的列数，然后用math.ceil() 向上取整，确保结果是整数。这个结果用来作为图像宽度。

类似地，ⓓ结果用来作为图像高度。

ⓔ用matplotlib.pyplot.subplots()，简写作plt.subplots()，创建图形对象fig和轴对象ax。

ⓕ调用seaborn.heatmap()，简写作sns.heatmap()，绘制热图可视化二维数组。

seaborn.heatmap()函数输入参数的含义请参考代码13.1注释。

代码13.1　自定义函数，可视化二维数组 | Bk1_Ch13_01.ipynb

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import math
ⓐ from matplotlib import cm

# 定义二维数组可视化函数
ⓑ def visualize_2D(array, title, vmax, vmin):
```