

23

Plotly Data Visualization and Storytelling

Plotly 统计可视化

用 Pandas + Plotly 讲故事：数据分析和可视化



别弄乱了我的圆！

Don't disturb my circles!

—— 阿基米德 (Archimedes) | 数学家、发明家、物理学家 | 287 ~ 212 BC



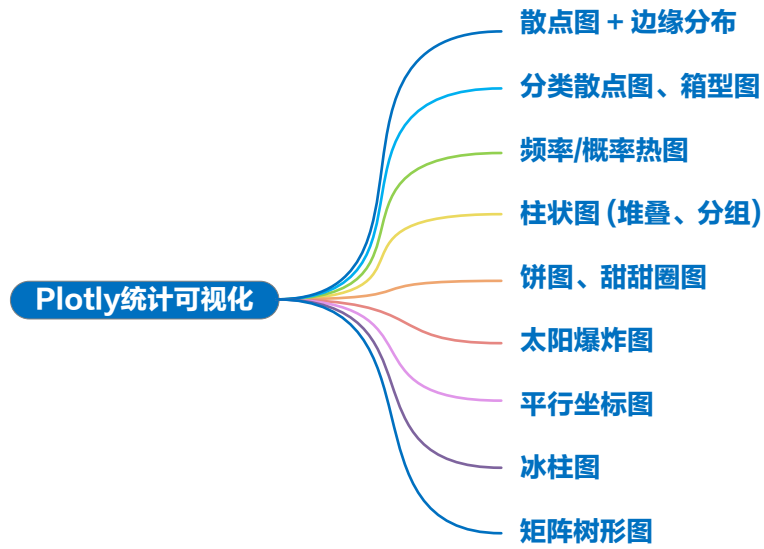
- ◀ `pandas.crosstab()` 创建交叉制表，根据两个或多个因素的组合统计数据的频数或其他聚合信息
- ◀ `pandas.cut()` 将数值列按照指定的区间划分为离散的分类，并进行标记
- ◀ `pandas.qcut()` 根据数据的分位数将数值列分成指定数量的离散区间
- ◀ `plotly.express.bar()` 创建交互式柱状图
- ◀ `plotly.express.box()` 创建交互式箱型图
- ◀ `plotly.express.density_heatmap()` 创建交互式频数/概率密度热图
- ◀ `plotly.express.icicle()` 绘制冰柱图
- ◀ `plotly.express.imshow()` 创建交互式热图
- ◀ `plotly.express.parallel_categories()` 创建交互式分类数据平行坐标图
- ◀ `plotly.express.pie()` 创建交互式饼图
- ◀ `plotly.express.scatter()` 创建交互式散点图
- ◀ `plotly.express.scatter_matrix()` 创建交互式成对散点图
- ◀ `plotly.express.sunburst()` 创建交互式太阳爆炸图
- ◀ `plotly.express.treemap()` 绘制矩形树形图



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



23.1 Plotly 常见可视化方案：以鸢尾花数据为例

自主探究学习时，我们通常一边完成运算，一边通常利用各种可视化方案完成数据分析和展示。本书第 12 章专门介绍过用 Seaborn 完成统计可视化操作，第 20 章则介绍了 Pandas 中“快速可视化”函数。

Plotly 库也有大量统计可视化方案，而且这些可视化方案具有交互化属性，特别适合探究式学习、结果演示。本章用鸢尾花数据举一个例子，帮大家看到“Pandas 运算 + Plotly 可视化”的力量。

散点图 + 边缘分布

图 1 所示为利用 `plotly.express.scatter()` 绘制的散点图，横轴为鸢尾花花萼长度，纵轴为鸢尾花花瓣长度，而且用不同颜色展示鸢尾花分类。

在这幅图上还绘制了**边缘箱型图** (marginal box plot)。在配套的 Jupyter Notebook 中大家会发现包括图 1 在内的本节所有图片都具有交互性，光标悬浮在图片具体对象上就会展示相关数值，很方便分析和展示。

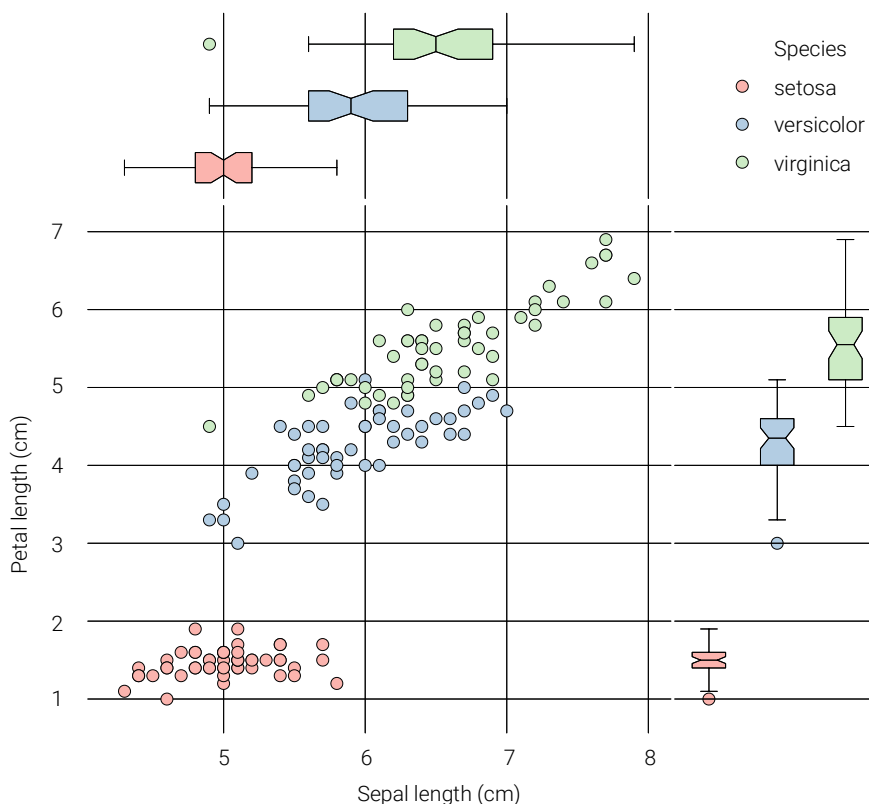


图 1. 用 Plotly 绘制散点图（横轴为花萼长度，纵轴为花瓣长度），边缘分布为箱型图，考虑鸢尾花分类

代码 1 绘制图 1，下面聊聊其中核心语句。

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

- a 从 Seaborn 库中导入鸢尾花数据集。
- b 利用 `plotly.express.scatter()`，简做 `px.scatter()`，创建了一个叫 `fig` 的散点图对象。

第一个参数 `df` 为鸢尾花数据集，数据格式为 Pandas DataFrame。

然后利用两个关键字参数指定横纵轴特征。数据集 `df` 中的 `sepal_length` 列作为 `x` 轴数据，`petal_length` 列作为 `y` 轴数据。

关键字参数 `color='species'` 指定了渲染编码的依据，即根据数据集中的 '`species`' 列的不同取值来区分不同种类的鸢尾花。

两个参数，`marginal_x='box'` 和 `marginal_y='box'`，分别表示在 `x` 轴和 `y` 轴的边缘添加一个箱型图，用于显示数据在每个轴上的分布情况。

Plotly 散点图还提供其他边缘分布的可视化方案，比如直方图 "`histogram`"、毛毯图 "`rug`"、小提琴图 "`violin`" 等，请大家练习使用。

参数 `template="plotly_white"` 设置了图片对象的主题风格，即使用 "`plotly_white`" 这种白色背景设计。

`width=600` 和 `height=500` 这两个参数分别设置了图表的宽度和高度，以像素为单位。

`color_discrete_sequence=px.colors.qualitative.Pastel1` 指定了颜色映射的调色板，即使用 Plotly Express 模块中提供的 "`Pastel1`" 调色板，以一组柔和的颜色来表示不同种类的花。

`labels={"sepal_length": "Sepal Length (cm)", "petal_length": "Petal length (cm)"}` 用于自定义图表的标签，将 `x` 轴标签设置为 "`Sepal Length (cm)`"，将 `y` 轴标签设置为 "`Petal length (cm)`"。

⚠ 注意，本节后文代码中遇到类似参数，将不再重复介绍。

- c 在 JupyterLab 中以交互形式展示图像对象 `fig`。

```

# 导入包
import seaborn as sns
import pandas as pd
import plotly.express as px

# 使用Seaborn加载鸢尾花数据集
df = sns.load_dataset("iris")

# 用plotly绘制散点图，边缘为箱型图，分类为 species
fig = px.scatter(df, x = 'sepal_length', y = 'petal_length',
                 color = 'species',
                 marginal_x = 'box', marginal_y = 'box',
                 width=600, height=500, template = "plotly_white",
                 color_discrete_sequence=px.colors.qualitative.Pastel1,
                 labels={"sepal_length": "Sepal Length (cm)",
                        "petal_length": "Petal length (cm)"})

fig.show()

```

代码 1. 用 Plotly 散点图可视化鸢尾花数据 | Bk1_Ch23_01.ipynb

代码 2 绘制成对散点图矩阵，请大家在本章配套 Jupyter Notebook 中查看结果。

a 调用 `plotly.express.scatter_matrix()`，参数 `dimensions` 用来指定散点图的维度。

注意，不同于 `seaborn.pairplot()`，`plotly.express.scatter_matrix()` 可以展示分类数据。

b 将对角线子图设为不可见。

```

# 绘制成对散点图
fig = px.scatter_matrix(df,
                       dimensions=["sepal_length", "sepal_width",
                                   "petal_length", "petal_width",
                                   "species"],
                       template = "plotly_white",
                       color = 'species', width = 600, height = 600)

fig.update_traces(diagonal_visible=False)
fig.show()

```

代码 2. 用 Plotly 绘制成对散点图 | Bk1_Ch23_01.ipynb

23.2 增加一组分类标签

为了增加数据分析的复杂度，我们引入了“花萼面积”这个新特征。

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

“花萼面积”用花萼长度和花萼宽度的乘积估计。然后，根据“花萼面积”的大小将 150 个样本数据几乎均匀地分成 5 个类别，并分别给它们新的标签 A、B、C、D、E，这一列列标签命名为 'Category'。

这样除了 'species' 之外，我们有了第 2 个类别标签。表 1 所示为“花萼面积”在不同 'Category' 的统计量总结。

表 1. “花萼面积”在不同 Category 分类下的统计量

Category	Area (cm ²)						
	min	max	mean	median	std	Range	Number
A	10.00	15.00	13.42	13.70	1.28	5.00	30
B	15.04	16.80	15.91	15.91	0.54	1.76	30
C	16.83	18.30	17.62	17.68	0.44	1.47	31
D	18.36	20.77	19.70	19.61	0.73	2.41	29
E	20.79	30.02	22.53	21.63	2.27	9.23	30

代码 3 完成上述运算分析，下面讲解其中关键语句。

- a 计算“花萼面积”，并将结果保存在原始数据帧中，列标签为 'area'。
- b 利用 `pandas.qcut()` 函数根据 'area' 列的大小将鸢尾花数据集大致均分为 5 个区间。同时，将生成的区间标签 'A'、'B'、'C'、'D'、'E' 分配给新创建的 'Category' 列。

有很多情况会造成“不完全均分”的情况，比如样本数量不能被区间数量整除，再比如某些样本量值重复出现。

- c 定义了一个名为 `list_stats` 的列表，其中包含了要计算的统计量的名称，其中包括 `min`（最小值）、`max`（最大值）、`mean`（均值）、`median`（中位数）、`std`（标准差）。
- d 利用 `pandas.DataFrame.groupby()` 进行分组计算统计量。这个方法利用 'Category' 对 `df` 进行分组，针对 'area'，归纳（agg）计算 `list_stats` 中列出的统计量。计算结果储存在新的 `DataFrame` 中，如所示，每一行代表不同的 'Category'，每一列代表不同统计量。

当然，在选择分组维度时，我们也可以选择不止一个标签，大家马上就会看到同时用 'Category'、'species' 进行分组的例子。

- e 计算极差（range），即最大值减去最小值。
- f 计算每个每组的计数。当然，大家也可以在 `list_stats` 加入 'count' 来完成计数。
- g 将分组统计结果存成 CSV 文件。可以在 JupyterLab 中打开查看，也可以用 Excel 打开查看。

```

# 用 花萼长度 * 花萼宽度 代表花萼面积
a df['area'] = df['sepal_length'] * df['sepal_width']

# 用花萼面积大小将样本等分为数量（大致）相等的5个区间
b df['Category'] = pd.qcut(df['area'], 5,
                           labels = ['A', 'B', 'C', 'D', 'E'])

# 按区间汇总（最小值，最大值，均值，标准差）
c list_stats = ['min', 'max', 'mean', 'median', 'std']
d stats_by_area = df.groupby('Category')['area'].agg(list_stats)

# 计算极差，最大值 - 最小值
e stats_by_area['Range'] = stats_by_area['max'] - stats_by_area['min']
# 每个区间的样本数量；还可以在list_stats中加 'count'
f stats_by_area['Number'] = df['Category'].value_counts()

# 将结果存为 CSV
g stats_by_area.to_csv('stats_by_area.csv')

```




代码 3. 增加“花萼面积”特征，并根据其大小对鸢尾花数据分类分析；使用时配合前文代码 |  Bk1_Ch23_01.ipynb

图 2 所示为不同 'Category' 条件下的“花萼面积”散点图，对应代码 4 中  a。图 3 不同 'Category' 条件下的“花萼面积”的“箱型图 + 散点图”，对应代码 4 中  b。

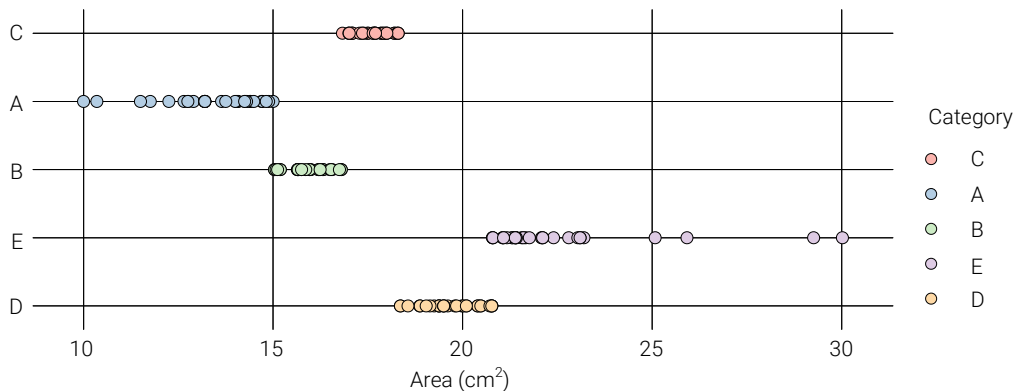


图 2. 用“花萼面积”对鸢尾花数据集再分割

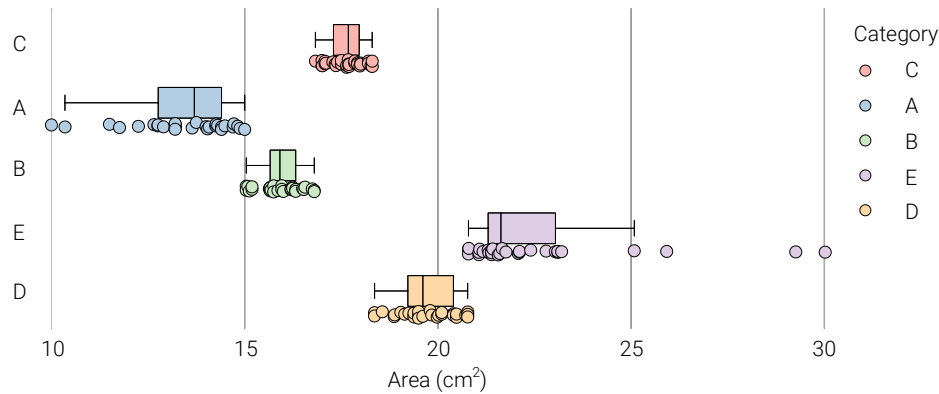


图 3. “花萼面积”的箱型图，考虑 'Category' 分类

```

# 用Plotly绘制散点图，维度为面积，分类为Category
fig = px.scatter(df, x = 'area', y = 'Category',
                 color = 'Category',
                 template = "plotly_white",
                 width=600, height=300,
                 color_discrete_sequence=px.colors.qualitative.Pastel1)
fig.show()

# 用Plotly绘制箱型图，维度为面积，分类为Category
fig = px.box(df, x = 'area', y = 'Category',
             color = 'Category', points="all",
             template = "plotly_white",
             width=600, height=300,
             color_discrete_sequence=px.colors.qualitative.Pastel1)
fig.show()

```

代码 4. 用 Plotly 绘制散点图和箱型图，分类展示“花萼面积”；使用时配合前文代码 | Bk1_Ch23_01.ipynb

新标签下的原始特征

类似 'species' 这个分类标签，我们也可以使用 'Category' 这个新标签分析原始特征数据，比如花萼长度。

图 4 所示为考虑 'Category' 分类情况下，鸢尾花长度的箱型图。

图 5 所示为花萼长度、花萼宽度的散点图，用不同颜色渲染 'Category' 分类。边缘分布还是箱型图。

代码 5 中 [a](#) 和 [b](#) 分别绘制图 4 和图 5，请大家根据前文讲解逐句注释。

值得一提的是，[a](#) 中 `category_orders={"Category": ["A", "B", "C", "D", "E"]}` 指定 'Category' 列的排序顺序。

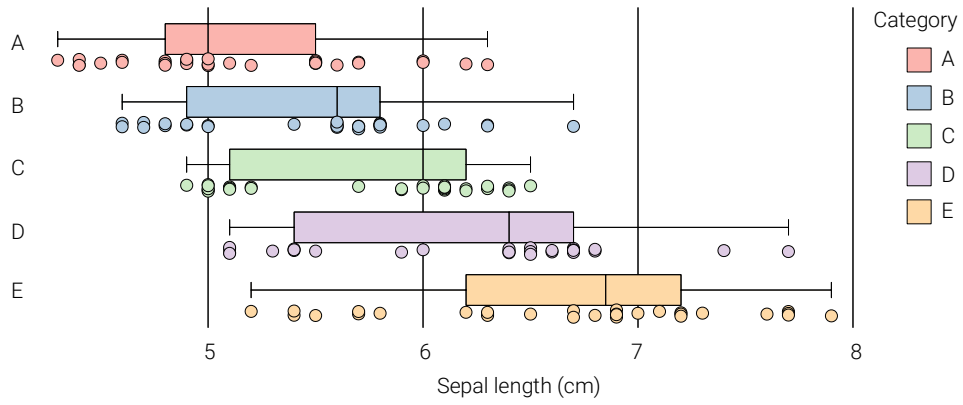


图 4. 花萼长度的箱型图，考虑'Category'分类

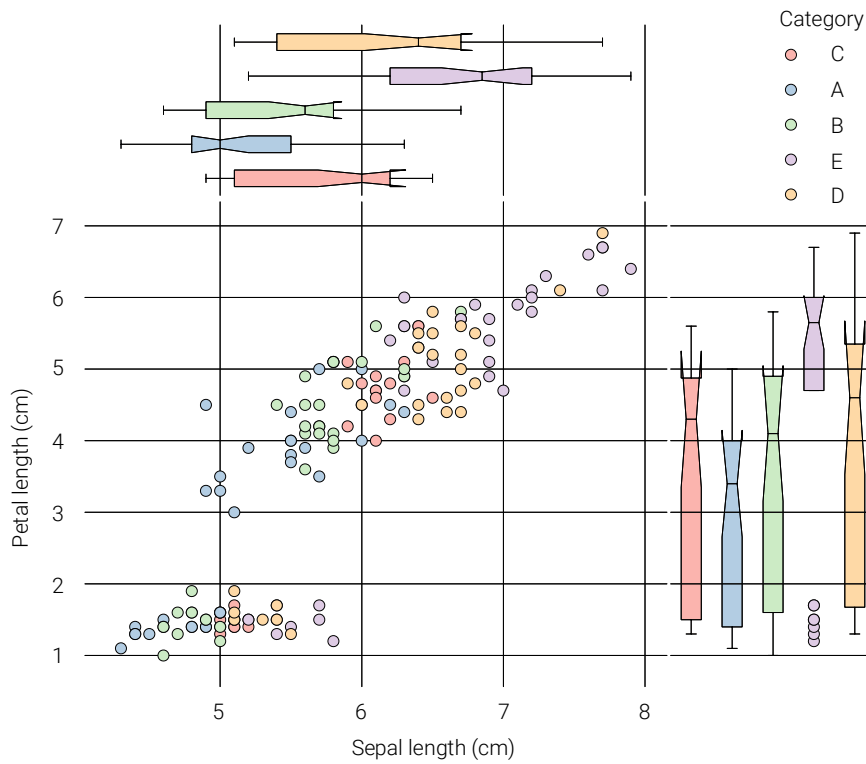


图 5. 用 Plotly 绘制散点图（横轴为花萼长度，纵轴为花瓣长度），边缘分布为箱型图，考虑“花萼面积”分类

```


# 花萼长度的箱型图，考虑 'Category' 分类
a fig = px.box(df, x = 'sepal_length', y = 'Category',
               color = 'Category', points="all",
               template = "plotly_white",
               width=600, height=300,
               category_orders={"Category": ["A", "B", "C", "D", "E"]},
               color_discrete_sequence=px.colors.qualitative.Pastel1,
               labels={"sepal_length": "Sepal Length (cm)"}))

fig.show()

# 用plotly绘制散点图，边缘为箱型图，分类为 Category
b fig = px.scatter(df, x = 'sepal_length', y = 'petal_length',
                  color = 'Category', marginal_x = 'box',
                  marginal_y = 'box', template = "plotly_white",
                  width=600, height=500,
                  color_discrete_sequence=px.colors.qualitative.Pastel1,
                  labels={"sepal_length": "Sepal Length (cm)",
                          "petal_length": "Petal length (cm)"}))

fig.show()

```

代码 5. 用 Plotly 绘制散点图和箱型图，分类展示“花萼长度”；使用时配合前文代码 |  Bk1_Ch23_01.ipynb

23.3 两组标签：两个维度

本节前文都是从单一维度分割 150 个样本数据，下面我们从两个维度，'species' 和 'Category'，分割数据。

如图 6 (a) 所示，从两个维度分割得到的结果为一个二维数组，即矩阵。

图 6 (a) 的数值为**频数** (frequency)，即**计数** (count)。也就是说，每个格子的数值代表满足分类条件的样本具体数量。

请大家用 Pandas 求和函数，计算图 6 (a) 所有值之和是否为 150。然后再分别计算图 6 (a) 沿行方向、沿列方向的求和，并用 `df['Category'].value_counts().sort_index()` 和 `df['species'].value_counts().sort_index()` 验证结果。

图 6 (b) 的每个格子代表满足特定分类条件的样本概率，即计数除以样本总数。

也请大家分别计算图 6 (b) 所有数值总和，以及沿行方向、沿列方向的求和，并想办法验证结果。

下面聊聊代码 6 代码中关键语句。

a 利用 `pandas.crosstab()` 创建交叉制表，对 `df` 指定两个列进行交叉分析。这个函数默认计算频率，也可以设置一个或多个聚合运算。

参数 `index = df['Category']` 指定了要用作交叉制表的行索引的列。当然，我们也可以指定两个或更多列，本章后文会介绍。'Category' 列的值将用作行索引，每个不同的值都将成为交叉制表中的一行。

参数 `columns = df['species']` 指定了要用作交叉制表的列索引的列。'species' 列的值将用作列索引，每个不同的值都将成为交叉表中的一列。

总结来说，这句代码生成一个交叉制表，其中的行表示 'Category' 列的不同值，列表示 'species' 列的不同值，而表格中的每个单元格则表示在这个维度组合下鸢尾花数据样本出现的频率或计数。本章后续还会介绍用 `pandas.crosstab()` 完成其他统计聚合运算。

b 利用 `plotly.express.imshow()` 创建热图对象。

参数 `text_auto=True` 用于控制是否自动在图中显示文本标签。将其设置为 `True`，表示在每个单元格中显示数值文本标签，以显示交叉制表的频率或计数值。

❷ 类似 ❶；不同的是参数 `normalize = 'all'` 指定标准化的方法，`'all'` 表示对整个交叉制表进行标准化，将每个频数除以样本数总和，得到概率值。这就是为什么图 6 (b) 中热图所有格子值的总和为 1。

❸ 类似；不同的是 `text_auto='.3f'`，表示数值以浮点数的格式保留三位小数显示。

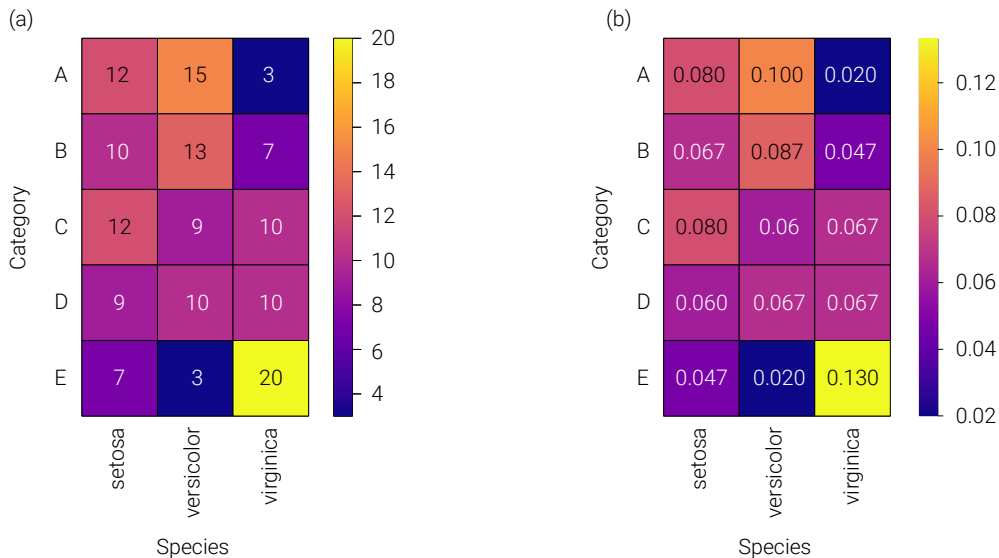
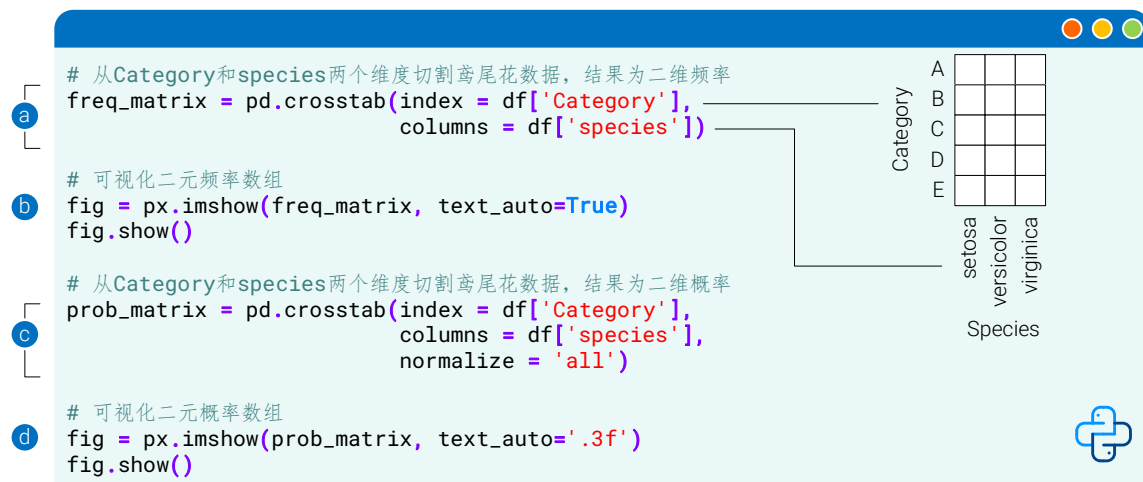


图 6. 用 `plotly.express.imshow()` 绘制频率和概率热图，两个分割维度



代码 6. 用 `plotly.express.imshow()` 绘制频率和概率热图；使用时配合前文代码 | Bk1_Ch23_01.ipynb

图 6 两幅子图的结果是利用 `pandas.crosstab()` 计算得到的。当然我们也可以使用 `plotly.express.density_heatmap()` 跳过计算直接绘制类似图像，具体如图 7 所示。

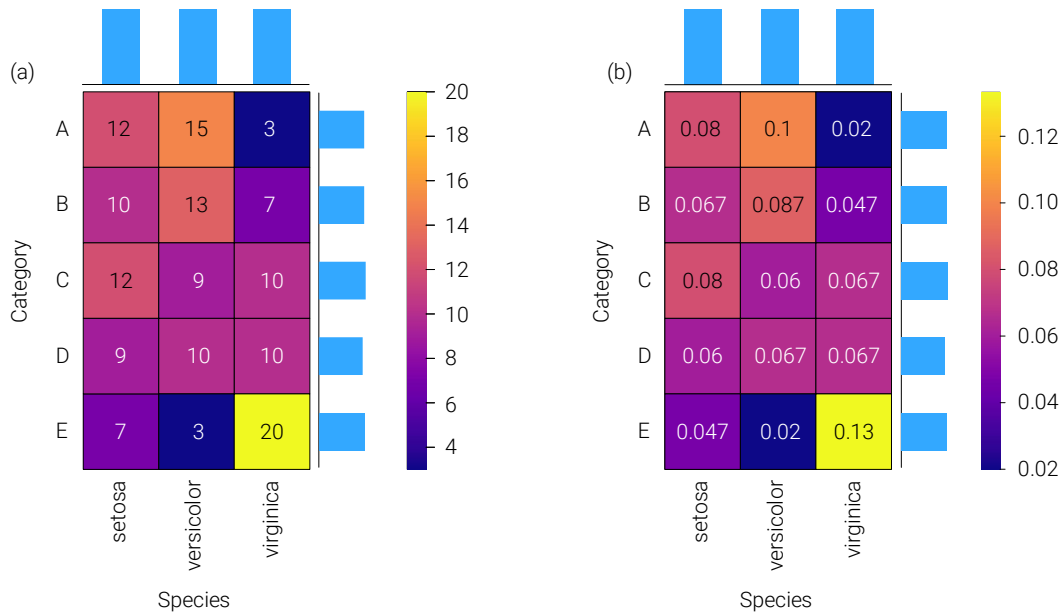


图 7. 用 `plotly.express.density_heatmap()` 绘制频率和概率热图，边缘分布为直方图，两个分割维度

代码 7 绘制图 7。 ^a 利用 `plotly.express.density_heatmap()` 绘制频数/概率热图。注意，函数默认计算频数，即计数。

参数 `x = 'species'` 和 `y = 'Category'` 指定了在热图中要显示的数据的 `x` 轴和 `y` 轴变量。


再次强调，`category_orders={"Category": ["A", "B", "C", "D", "E"]}` 采用字典指定分类变量 'Category' 的顺序。大家也可以试着同时指定 `species` 的顺序。

`marginal_x="histogram"` 和 `marginal_y="histogram"` 这两个参数指定了在 `x` 轴和 `y` 轴上显示边际直方图。

^b 和 ^a 类似，不同的是参数 `histnorm='probability'` 指定直方图的标准化的方法。设置为 'probability' 表示热图中的数值标准化为概率密度。

```
# 绘制二维直方热图 + 边缘直方图，计数
fig = px.density_heatmap(df, x = 'species', y = 'Category',
                        category_orders={"Category":
                                         ["A", "B", "C", "D", "E"]},
                        marginal_x="histogram", marginal_y="histogram",
                        text_auto = True, width = 400, height = 500)
fig.show()

# 绘制二维直方热图 + 边缘直方图，概率
fig = px.density_heatmap(df, x = 'species', y = 'Category',
                        category_orders={"Category":
                                         ["A", "B", "C", "D", "E"]},
                        marginal_x="histogram", marginal_y="histogram",
                        histnorm = 'probability',
                        text_auto='.3f', width = 400, height = 500)
fig.show()
```

代码 7. 用 `plotly.express.density_heatmap()` 绘制二维频率和概率热图；使用时配合前文代码 | 

Bk1_Ch23_01.ipynb

图 8 这幅图也是用 `plotly.express.density_heatmap()` 绘制。这幅图明显的特点是采用 5×3 子图布局。这样便于展示在不同分类组合条件下，其他量化特征的分布情况。

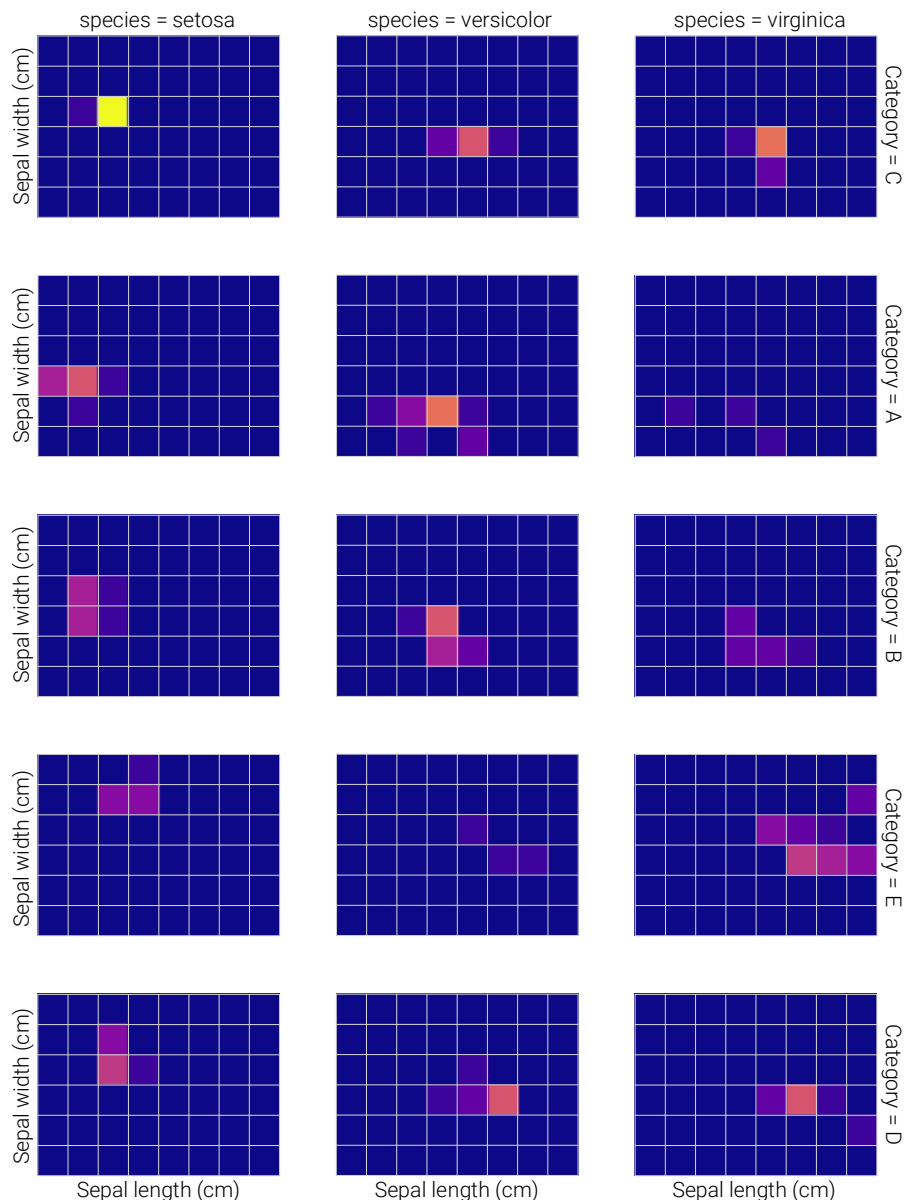
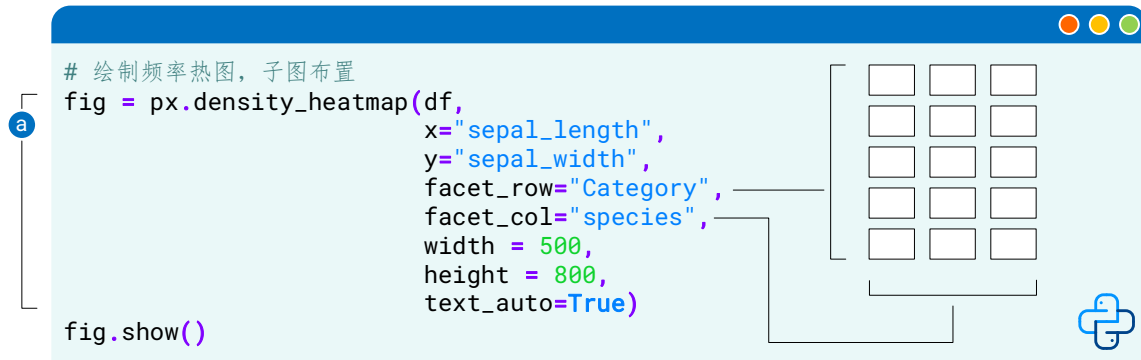



图 8. 频率热图，子图布局

代码 8 绘制图 8。其中，`x="sepal_length"` 和 `y="sepal_width"` 这两个参数指定了子图热图中要显示的数据的 `x` 轴和 `y` 轴变量。

这两个参数，`facet_row="Category"` 和 `facet_col="species"` 用于进行子图布局。

`facet_row` 将数据分成多行子图，每行对应于不同的 'Category' 值；`facet_col` 将数据分成多列子图，每列对应于不同的 'species' 值。



代码 8. 用 `plotly.express.density_heatmap()` 绘制频率热图，子图布置；使用时配合前文代码 | 
Bk1_Ch23_01.ipynb

23.4 可视化比例：柱状图、饼图

图 9 分别采用水平柱状图和饼图展示 'Category' 分类下样本计数的比例。这种中间掏空的饼图还有一个“可爱”的名字——甜甜圈图（donut chart）。

直方图（histogram）和**柱状图**（bar chart）都可以用来可视化数据分布，但是它们存在一些区别。

直方图常用来可视化连续数据的分布，比如鸢尾花花萼长度样本数据分布。垂直直方图的纵轴可以是频数、概率、概率密度。

而柱状图一般用来展示类别、离散数据、区间的频数\概率。

⚠ 注意，不管是直方图还是柱状图，如果采用频数，则图中频数总和为样本总数。这个样本总数可以对应样本全集，也可以对应特定子集，比如某个条件分类。

从外观上来看，直方图一般情况下柱子之间是连续的，如果不连续则说明特定区间没有样本点。而柱状图的柱子通常是分离的，有明显间隔。

但是，直方图和柱状图之间也可以存在联系。还是以鸢尾花花萼长度为例，用直方图展示样本数据在花萼长度上的分布很容易。如果根据长度数值将花萼数据分为 5 个区间，并用 A、B、C、D、E 命名，这种情况下，我们可以用柱状图可视化频数。本章下文将用这个角度切割鸢尾花样本数据。

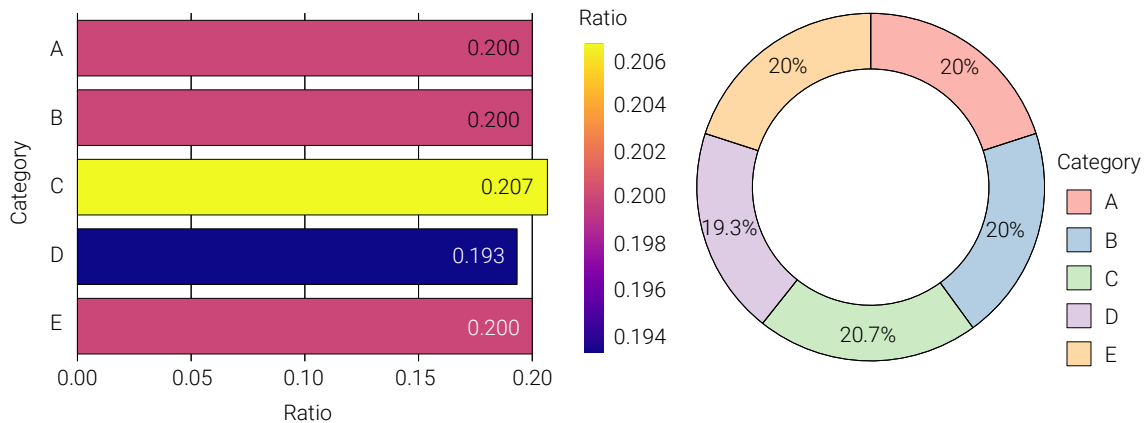


图 9. 用 `plotly.express.bar()` 和 `plotly.express.pie()` 可视化 'Category' 分类比例

代码 9 绘制图 9，下面聊聊其中关键语句。

a 用 `.value_counts(normalize=True)` 方法计算概率。其中，`normalize=True` 参数的作用是计算概率，即频率比例，而不是频数。因此图 9 左图中柱状图五个柱子数值之和为 1。

大家前文见过，如果 `normalize` 参数为 `False`（默认值），则 `value_counts()` 方法将计算频数（计数）。

注意，**a** 的结果 `ctg_percent` 数据类型为 `pandas series`。

b 用 `pandas.DataFrame()` 将 `Pandas Series` 转换为一个新的 `DataFrame`，其中包含两列——'Category' 和 'Percent'。

具体来说，`{'Category': ctg_percent.index, 'Percent': ctg_percent.values}`：这是传递给 `pandas.DataFrame()` 的字典，其中包含两个键值对。

'Category' 是新生成 `DataFrame` 中的第一列的名称，即 'Category' 列。
`ctg_percent.index` 是 `Pandas Series` 的索引，即不同类别的标签。它被用作新 `DataFrame` 的 'Category' 列的数据。

'Percent' 则为 `DataFrame` 中的第二列的名称，即 'Percent' 列。

`ctg_percent.values` 是 `Pandas Series` 中的值，即频率百分比。它被用作新 `DataFrame` 的 'Percent' 列的数据。

c 用 `plotly.express.bar()` 绘制水平柱状图。请大家对参数的意义进行注释。

d 利用 `plotly.express.pie()` 绘制饼图（甜甜圈图）。

参数 `category_orders={"Category": ["A", "B", "C", "D", "E"]}` 指定了饼图中类别的顺序。

参数 `color_discrete_sequence=px.colors.qualitative.Pastel1` 设置了饼图中各个类别的颜色。

参数 `values='Ratio'` 指定了饼图中每个扇形（环形）区域的数值应该来自数据集中的哪一列。

参数 `names='Category'` 指定了饼图中每个扇形区域的标签应该来自数据集中的哪一列。

e 用 `fig.update_traces(hole=.68)` 将饼图中 68% 区域挖空，将饼图变成环形图。

```


# 计算 Category 分类比例
a ctg_percent = df['Category'].value_counts(normalize=True)
b ctg_percent = pd.DataFrame({'Category':ctg_percent.index,
                             'Percent':ctg_percent.values})

# 用柱状图展示 Category 分类比例
c fig = px.bar(ctg_percent,
               x="Percent", y="Category",
               category_orders={"Category": ["A", "B", "C", "D", "E"]},
               color = "Percent", orientation='h',
               text_auto = '.3f')

fig.show()

# 用饼图可视化 Category 百分比
d fig = px.pie(ctg_percent,
               category_orders={"Category": ["A", "B", "C", "D", "E"]},
               color_discrete_sequence=px.colors.qualitative.Pastel1,
               values='Ratio', names='Category')
e fig.update_traces(hole=.68)
fig.show()

```

代码 9. 可视化 'Category' 分类比例，柱状图和饼图；使用时配合前文代码 |  Bk1_Ch23_01.ipynb

类似图 9，图 10 也用水平柱状图和饼图可视化 'species' 分类条件下样本数据的频数比例。代码 10 绘制图 10，请大家逐行注释。

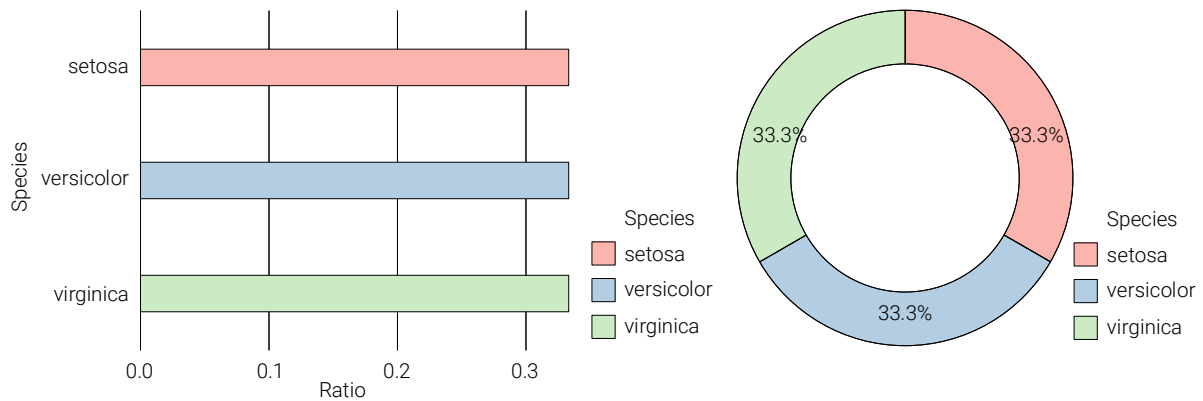


图 10. 用 `plotly.express.bar()` 和 `plotly.express.pie()` 可视化 'species' 分类比例


```

# 计算 species 分类比例
a species_percent = df['species'].value_counts(normalize=True)
b species_percent = pd.DataFrame({'species':species_percent.index,
                                'Ratio':species_percent.values})


# 用柱状图可视化 species 分类比例
c fig = px.bar(species_percent,
               x="Ratio", y="species",
               category_orders={"species":
                               ["setosa", "versicolor", "virginica"]},
               color_discrete_sequence=px.colors.qualitative.Pastel1,
               color = "species", orientation='h',
               text_auto = '.3f')

fig.show()

# 用饼图可视化 species 分类百分比
d fig = px.pie(species_percent,
               category_orders={"species":
                               ["setosa", "versicolor", "virginica"]},
               color_discrete_sequence=px.colors.qualitative.Pastel1,
               values='Ratio', names='species')

fig.update_traces(hole=.68)
fig.show()

```

代码 10. 可视化 'species' 分类比例，柱状图和饼图；使用时配合前文代码 |  Bk1_Ch23_01.ipynb

23.5 钻取：多个层次之间的导航和探索

既然我们有两个不同分类维度，那么问题来了，我们能否量化并可视化不同 'Category' 中 'species' 的比例？

同理，我们能否量化并可视化不同 'species' 中 'Category' 的比例？

类似这种操作都叫做**钻取** (drill down)。钻取常用于在数据的多个层次之间进行导航和探索，以深入了解数据的细节。简单来说，钻取就是不断细分；用大白话来说就是，不断切片切块、切丝切条。

下面，我们就利用 Pandas 和 Plotly 试着完成不同类别之间样本数据比例值钻取运算和可视化。

图 11 采用**堆叠柱状图** (stacked bar chart) 可视化 Category 中 species 的绝对比例；钻取顺序为 Category → species。所谓绝对比例就是指，图 11 所有分段柱子之和均为 1。

观察图 11，我们可以发现 E 类中 virginica 的比例更高等有趣现象，值得进一步分析。

图 12 也是采用堆叠柱状图，可视化 species 中 Category 的绝对比例，即钻取顺序为 species → Category。

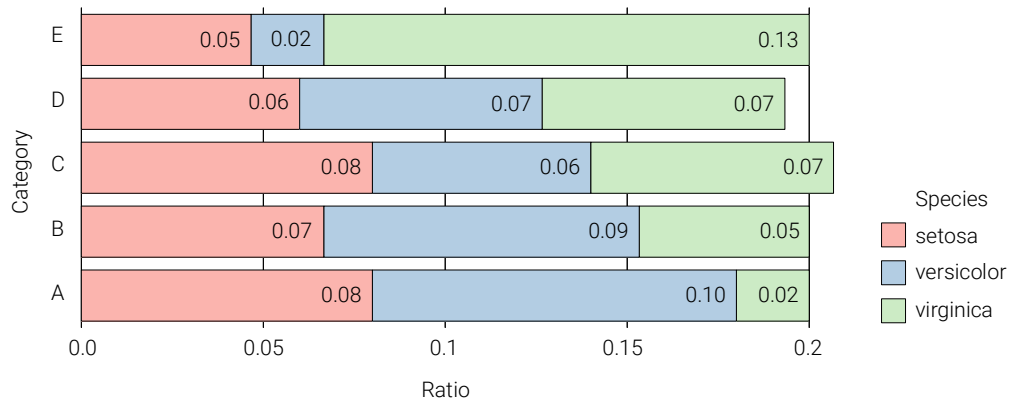


图 11. 用堆叠柱状图可视化 Category 中 species 的绝对比例；钻取顺序 Category → species；所有分段柱子之和为 1

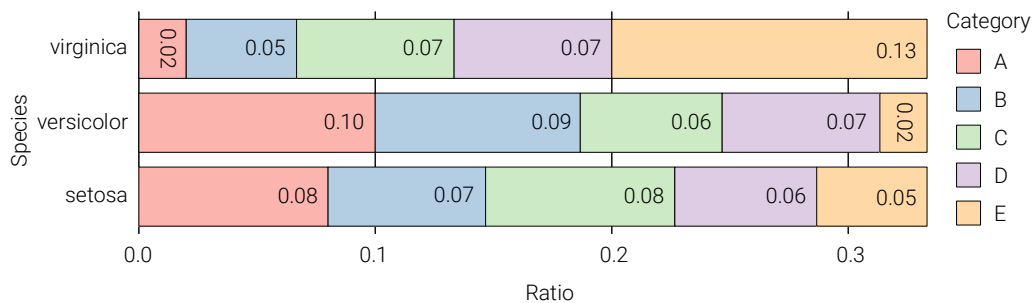


图 12. 用堆叠柱状图可视化 species 中 Category 的绝对比例；钻取顺序 species → Category；所有分段柱子之和为 1

代码 11 绘制图 11 和图 12，下面解释其中关键语句。

首先，用到了代码 6 中计算得到的 prob_matrix，对应表 2。

a 利用 plotly.express.bar() 可视化 prob_matrix 时，先后顺序为先行后列，即 Category → species。先在 Category 维度上切条，后在 species 维度上切块。

b 则可可视化 prob_matrix 的转置，对应表 3。用 plotly.express.bar() 这个转置后的数据帧时，顺序为 species → Category。

表 2. 绝对比例，数据帧 prob_matrix 的形状；表格概率值之和为 1

species	setosa	versicolor	virginica
Category			
A	0.080	0.100	0.020
B	0.067	0.087	0.047
C	0.080	0.060	0.067
D	0.060	0.067	0.067
E	0.047	0.020	0.133

表 3. 数据帧 prob_matrix 转置后的形状


Category	A	B	C	D	E
species					
setosa	0.080	0.067	0.080	0.060	0.047
versicolor	0.100	0.087	0.060	0.067	0.020
virginica	0.020	0.047	0.067	0.067	0.133

```

# 对 Category 比例值在 species 维度上钻取
a fig = px.bar(prob_matrix,
               template = "plotly_white", orientation = 'h',
               color_discrete_sequence=px.colors.qualitative.Pastel1,
               width=600, height=300, text_auto = '.2f')
fig.show()

# 对 species 比例值在 Category 维度上钻取
b fig = px.bar(prob_matrix.T,
               template = "plotly_white", orientation = 'h',
               color_discrete_sequence=px.colors.qualitative.Pastel1,
               width=600, height=300, text_auto = '.2f')
fig.show()

```

代码 11. 用 plotly.express.bar() 绘制堆叠柱状图；使用时配合前文代码 |  Bk1_Ch23_01.ipynb

相对比例钻取


图 11 有个缺陷，它不能直接回答“Category 为 A 时，setosa 在其中占比为多少？”

也就是说，问这个问题时，我们不再关心“绝对比例值”，而是关心“相对比例值”。

想要回答这个问题，需要计算 $0.08 / (0.08 + 0.10 + 0.02)$ ，结果为 0.4。这个值在概率统计中也叫**条件概率**（conditional probability）。

而图 13 和图 14 可以帮助我们回答类似上述问题。

以图 13 为例，大家很容易发现水平方向三个分段堆叠柱子对应数值之和为 1，这就是相对比例，即概率统计中的条件概率值。这个条件就是 Category 分别为 A、B、C、D、E。

 注意，图 13 和图 14 中所有水平方向分段柱子之和均为 1。图 13 所有柱子之和为 5，图 14 所有柱子之和为 3。



鸢尾花书《统计至简》将专门讲解条件概率。

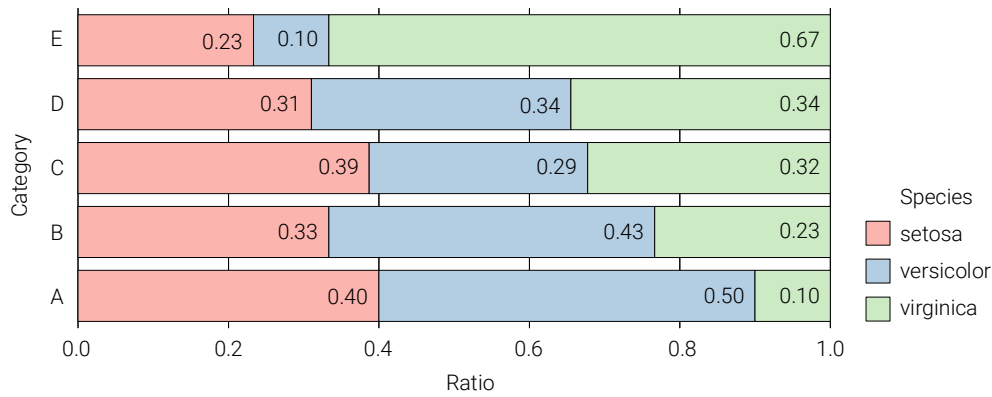


图 13. 用堆叠柱状图可视化 Category 中 species 的相对比例；钻取顺序 Category → species

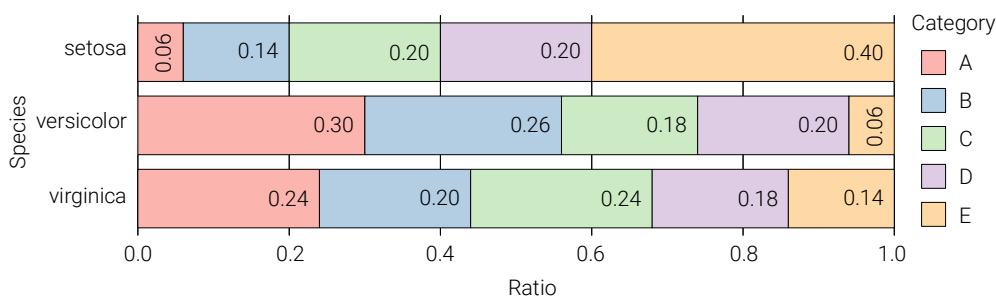


图 14. 用堆叠柱状图可视化 species 中 Category 的相对比例；钻取顺序 species → Category

代码 12 绘制图 13 和图 14，下面聊聊其中关键语句。

a 也是采用 `pandas.crosstab()` 创建交叉制表，不同以往这里使用了 `normalize='index'` 参数进行行归一化 (normalization)，结果如表 4 所示。这意味着，在计算每一行中的每个单元格的值占该行总和的比例，即条件概率。这将使每一行的值之和等于 1。

b 用堆叠柱状图可视表 4 相对比例值。请大家逐行注释。

c 类似 **a**，不同的是行列分类对调，结果如表 5 所示，每一行的值之和还是为 1。

d 也是用堆叠柱状图可视表 5 相对比例值。

表 4. 相对比例，钻取顺序 Category → species；归一化方向为 index，即表格每行概率值之和为 1

species	setosa	versicolor	virginica
Category			
A	0.400	0.500	0.100
B	0.333	0.433	0.233
C	0.387	0.290	0.323
D	0.310	0.345	0.345
E	0.233	0.100	0.667

表 5. 相对比例，钻取顺序 species → Category；归一化方向为 index，即表格每行概率值之和为 1

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

Category	A	B	C	D	E
species					
setosa	0.240	0.200	0.240	0.180	0.140
versicolor	0.300	0.260	0.180	0.200	0.060
virginica	0.060	0.140	0.200	0.200	0.400

 请大家思考，如果将归一化方向改为 `columns`，即 `normalize = 'columns'`，结果会怎样？

```

# 计算Category中species的相对比例
a ratio_species_in_category = pd.crosstab(index = df['Category'],
                                          columns = df['species'],
                                          normalize = 'index')

# 可视化
b fig = px.bar(ratio_species_in_category,
               template = "plotly_white", orientation = 'h',
               color_discrete_sequence=px.colors.qualitative.Pastel1,
               width=600, height=300, text_auto = '.2f')
fig.show()

# 计算species中Category的相对比例
c ratio_category_in_species = pd.crosstab(index = df['species'],
                                          columns = df['Category'],
                                          normalize = 'index')

# 可视化
d fig = px.bar(ratio_category_in_species,
               template = "plotly_white", orientation = 'h',
               color_discrete_sequence=px.colors.qualitative.Pastel1,
               width=600, height=300, text_auto = '.2f')
fig.show()

```

代码 12. 计算并可视化相对比例；使用时配合前文代码 |  Bk1_Ch23_01.ipynb

23.6 太阳爆炸图：展示层次结构

Plotly Express 库中 `plotly.express.sunburst()` 是可以用于创建**太阳爆炸图** (sunburst chart) 的函数。

太阳爆炸图一般呈太阳状或环状，通常用于展示层次结构或树状数据的分布情况，以及不同级别之间的关系。

可以这么理解，太阳爆炸图相当于多层饼图，每一层代表一个钻取维度。从集合角度，每个圆弧都代表特定子集。

下面，我们用太阳爆炸图可视化上一节介绍的样本比例钻取。

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 15 所示为钻取顺序 `species` → `Category` 的太阳爆炸图。这幅图中，我们还用颜色映射渲染比例值。图 16 这幅太阳爆炸图对应的钻取顺序为 `Category` → `species`。

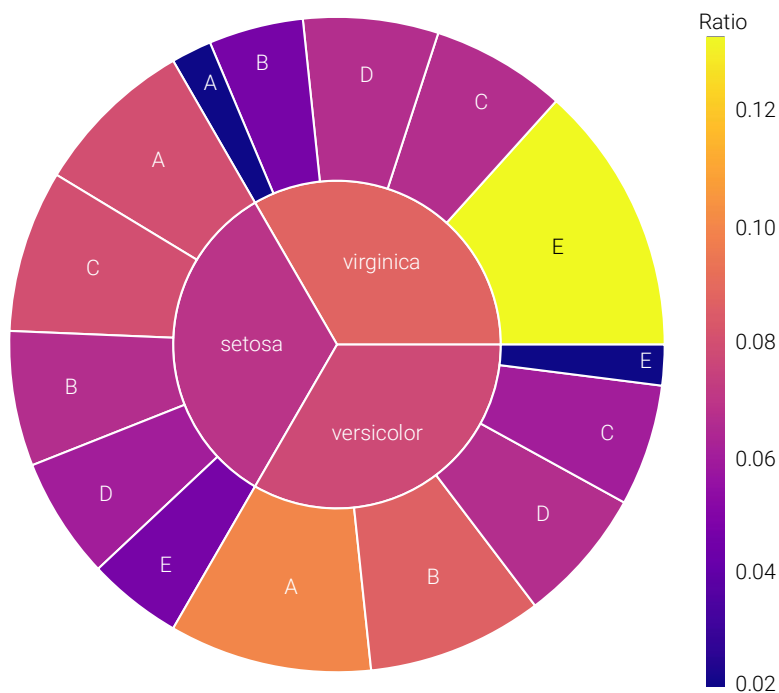


图 15. 太阳爆炸图可视化绝对比例钻取，钻取顺序 `species` → `Category`

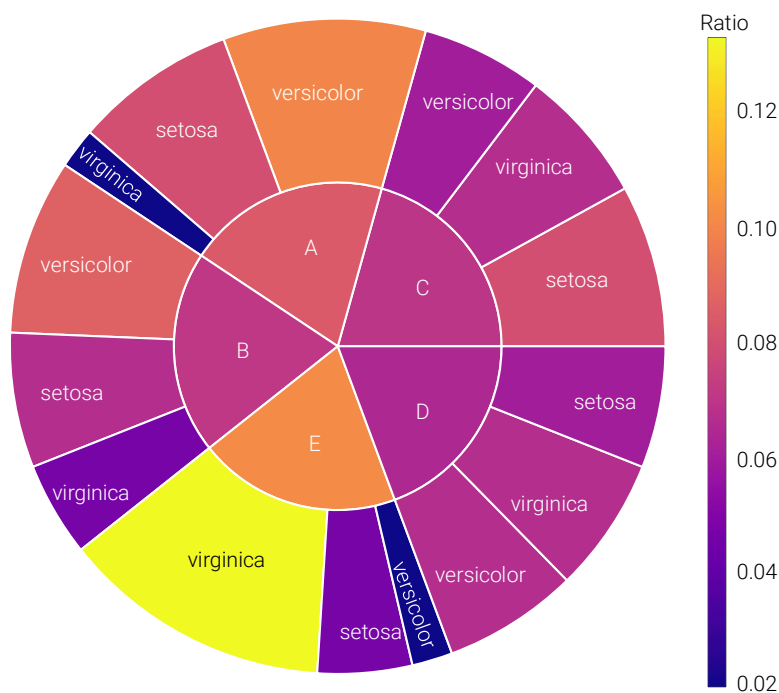


图 16. 太阳爆炸图可视化绝对比例钻取，钻取顺序 `Category` → `species`

代码 13 绘制图 15 和图 16，下面聊聊其中关键语句。

a 是一连串链式操作，先用 `stack()` 堆叠将数据帧从宽格式转为长格式，然后使用 `reset_index()` 方法，将之前堆叠的索引还原为 `DataFrame` 的标准整数索引。

最后，使用 `rename()` 方法，将 `DataFrame` 的列名从默认的 "0" 重命名为 "Ratio"。

b 在利用 `plotly.express.sunburst()` 绘制太阳爆炸图时，用 `path=['species', 'Category']` 指定了太阳爆炸图的路径，即钻取顺序。

如图 15 所示，太阳爆炸图的内层为 "species"，外层为 "Category"。

参数 `values='Ratio'` 指定了太阳爆炸图中每个扇形区域的值应该来自数据帧中的哪一列。

参数 `color='Ratio'` 指定了太阳爆炸图中每个扇形区域的颜色对应数据帧中的哪一列。

c 类似 **b**，但钻取顺序不同。

? 请大家查询 Plotly 技术文档想办法修改太阳爆炸图的颜色映射。

表 6. 绝对比例数据帧，长格式

	Category	species	Ratio
0	A	setosa	0.0800
1	A	versicolor	0.1000
2	A	virginica	0.0200
3	B	setosa	0.0667
4	B	versicolor	0.0867
5	B	virginica	0.0467
6	C	setosa	0.0800
7	C	versicolor	0.0600
8	C	virginica	0.0667
9	D	setosa	0.0600
10	D	versicolor	0.0667
11	D	virginica	0.0667
12	E	setosa	0.0467
13	E	versicolor	0.0200
14	E	virginica	0.1333

```

# 将概率值（比例值）stack 起来
a prob_matrix_stacked = prob_matrix.stack().reset_index().rename(
    columns={0: "Ratio"})

# 用太阳爆炸图进行钻取，先 species, 再 Category
b fig = px.sunburst(prob_matrix_stacked,
    path=['species', 'Category'],
    values='Ratio', color='Ratio',
    width = 600, height = 600)

fig.show()

# 用太阳爆炸图进行钻取，先 Category, 再 species
c fig = px.sunburst(prob_matrix_stacked,
    path=['Category', 'species'],
    values='Ratio', color='Ratio',
    width = 600, height = 600)

fig.show()

```

代码 13. 用 plotly.express.sunburst() 绘制太阳爆炸图；使用时配合前文代码 | Bk1_Ch23_01.ipynb

23.7 增加第三切割维度

下面，我们进一步提升分析的复杂度！

将 DataFrame 中的花萼长度数据根据指定区间分组，每一组添加一个标签。比如，“4 - 5 cm”表示鸢尾花样本中花萼长度 4 厘米到 5 厘米的样本点，区间左闭右开 [4, 5)。

图 17 所示为用水平柱状图和饼图可视化 sepal_length_bins 维度鸢尾花数据样本计数。可以发现这个维度有 4 个分类，其中花萼长度 5 厘米到 6 厘米的样本点最多，为 61 个，占总数的 40.7%。

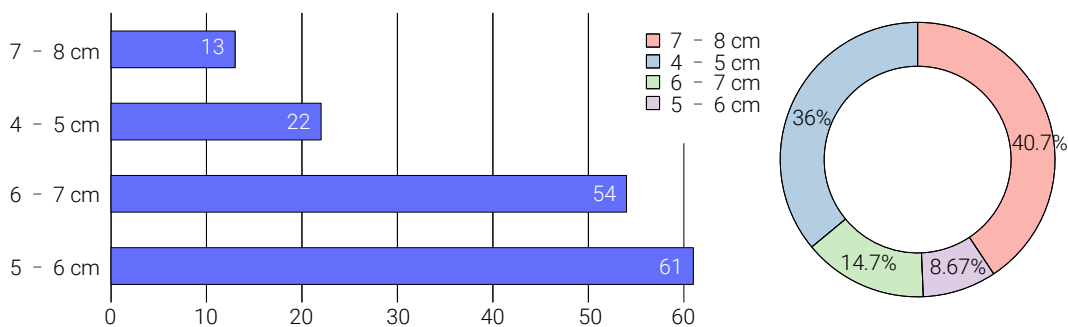


图 17. 可视化 sepal_length_bins 维度的样本计数

代码 14 绘制图 17，下面聊聊其中核心语句。

a 利用列表生成式创建了一个标签列表 labels。这个列表中的每个标签都是一个字符串，通过格式化字符串方法 format() 创建。

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

这些标签将用于表示花萼长度不同的区间，例如 "4 - 5 cm"、"5 - 6 cm" 等。其中循环 `for i in range(4, 8)` 循环遍历从 4 到 7 的整数。

b 用 `pandas.cut()` 划分数据。其中，`df.sepal_length` 是要划分数据的依据，即数据帧的花萼长度的列。

`range(4, 9)` 用于定义区间的范围，即 `[4, 5)`、`[5, 6)`、`[6, 7)`、`[7, 8]`。

参数 `right=False`，意味着右边界是开放的，即区间的右边界不包括在内。如果想让区间左开右闭可以设置参数 `right == True`。

`labels=labels` 指定了区间的标签，即每个区间对应的标签。这里使用了之前创建的 `labels` 列表。注意，区间数量和标签数量相等。

c 使用 `value_counts()` 方法统计了 `df` 中的 `sepal_length_bins` 列中每个不同区间样本出现的次数。

d 用 `pandas.DataFrame()` 将 Pandas Series 转化为一个数据帧。

e 用 `plotly.express.bar()` 绘制水平柱状图，表示 `sepal_length_bins` 这个维度上的频数。**f** 用 `plotly.express.pie()` 绘制饼图。

```
# 再增加一层钻取维度
# 设置标签
a labels = ["{0} - {1} cm".format(i, i+1) for i in range(4, 8)]
# 用pandas.cut() 划分区间
b df["sepal_length_bins"] = pd.cut(df.sepal_length, range(4, 9),
                                   right=False, labels=labels)

# 计算频数
c sepal_length_bins_counts = df["sepal_length_bins"].value_counts()
d sepal_length_bins_counts = pd.DataFrame({
    'sepal_length_bins': sepal_length_bins_counts.index,
    'Count': sepal_length_bins_counts.values})

# 可视化第三维度样本计数
e fig = px.bar(sepal_length_bins_counts,
               x = 'Count', y = 'sepal_length_bins',
               orientation = 'h', text_auto=True)
fig.show()

# 可视化第三维度样本百分比
f fig = px.pie(sepal_length_bins_counts,
               color_discrete_sequence=px.colors.qualitative.Pastel1,
               values='Count', names='sepal_length_bins')
fig.update_traces(hole=.68)
fig.show()
```


代码 14. 根据花萼长度再增加一个分类维度；使用时配合前文代码 |  Bk1_Ch23_01.ipynb

图 18 也是用太阳爆炸图展示三个维度探索鸢尾花数据，钻取顺序 `species` → `Category` → `sepal_length_bins`。

代码 15 中 **a** 用列表定义了钻取的顺序。

b 用 `.groupby()` 完成分组聚合操作。

方法 `.groupby(dims)` 按 `dims` 进行分组。['sepal_length'] 给出计算的对象。

方法 `.apply(lambda x: x.count()/len(df))` 利用 `lambda` 函数计算每个分组内的数据数量 (`count()` 方法)，除以样本总数 (`len(df)`) 得到每个分组内数据的占比。

- c 用 `.reset_index()` 方法重新设置索引，把多级行索引变成了数据帧的列。
- d 修改数据帧列名。
- e 还是用 `plotly.express.sunburst()` 绘制太阳爆炸图。相比图 15 和图 16，图 18 这幅太阳爆炸图层次更丰富。请大家修改钻取顺序，重新绘制图 18。

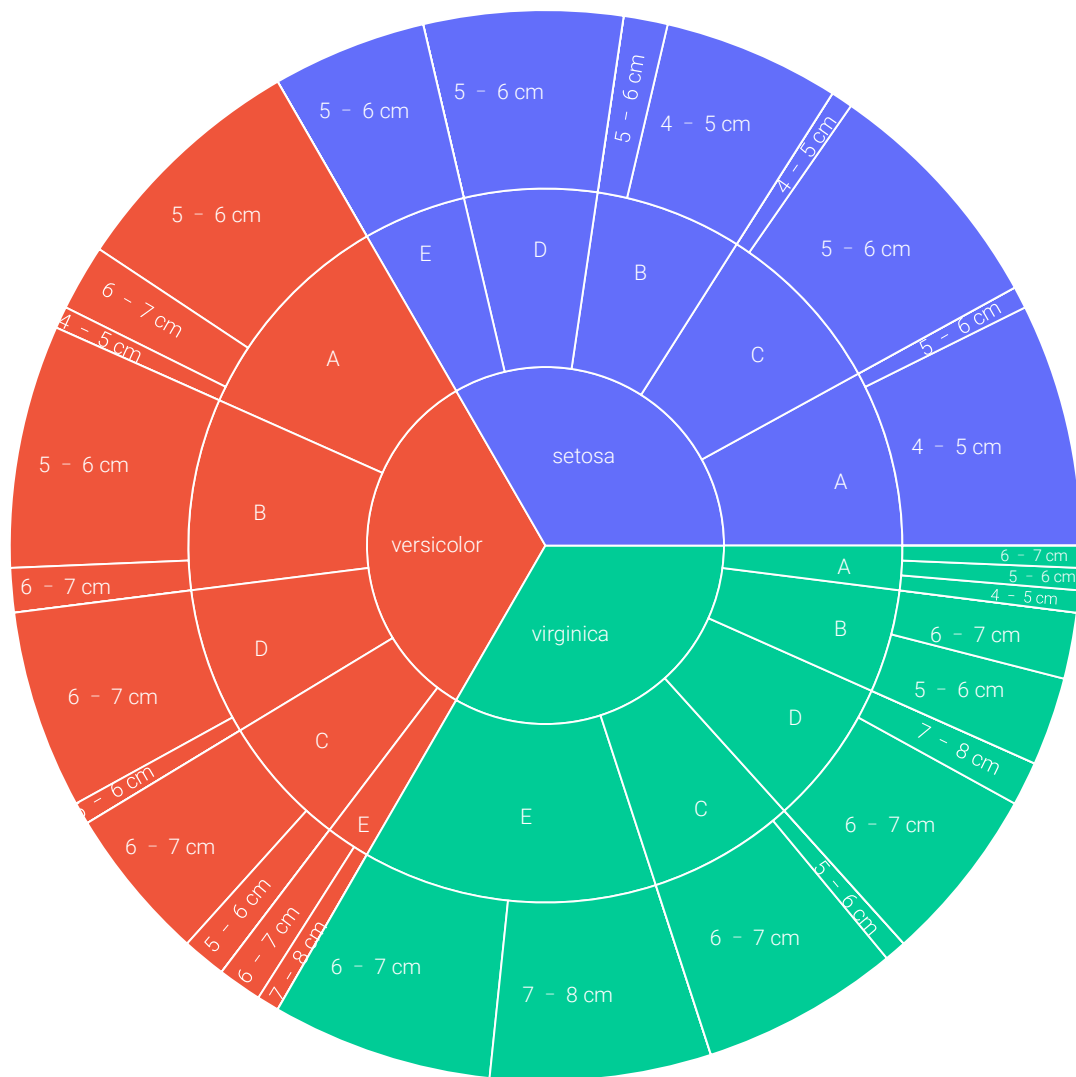


图 18. 太阳爆炸图可视化绝对比例，钻取顺序 `species` → `Category` → `sepal_length_bins`

```

# 计算三个维度转取的比例（概率）值
a dims = ['species', 'Category', 'sepal_length_bins']

b prob_matrix_by_3 = df.groupby(dims)['sepal_length'].apply(
    lambda x: x.count()/len(df))
c prob_matrix_by_3 = prob_matrix_by_3.reset_index()
d prob_matrix_by_3.rename(columns = {'sepal_length': 'Ratio'},
    inplace = True)

# 用太阳爆炸图进行钻取，先 Category，再 species，最后sepal_length_bins
e fig = px.sunburst(prob_matrix_by_3,
    path=dims,
    values='Ratio',
    width = 600, height = 600)

fig.show()

```

代码 15. 太阳爆炸图，三个维度；使用时配合前文代码 | Bk1_Ch23_01.ipynb

除了本章前文介绍的几种可以用来可视化“钻取”的方案，本节最后还要再介绍其他几种方法。

图 19 所示为利用 `plotly.express.parallel_categories()` 绘制分类数据平行坐标图。

平行坐标图 (parallel coordinate plot) 是一种用于可视化多维数据的图表类型，它通过在平行的坐标轴上显示各个特征来展示不同特征之间的关系。每个数据点在图中的位置由其特征值决定，从而可以观察到特征之间的模式、关联和分布情况。

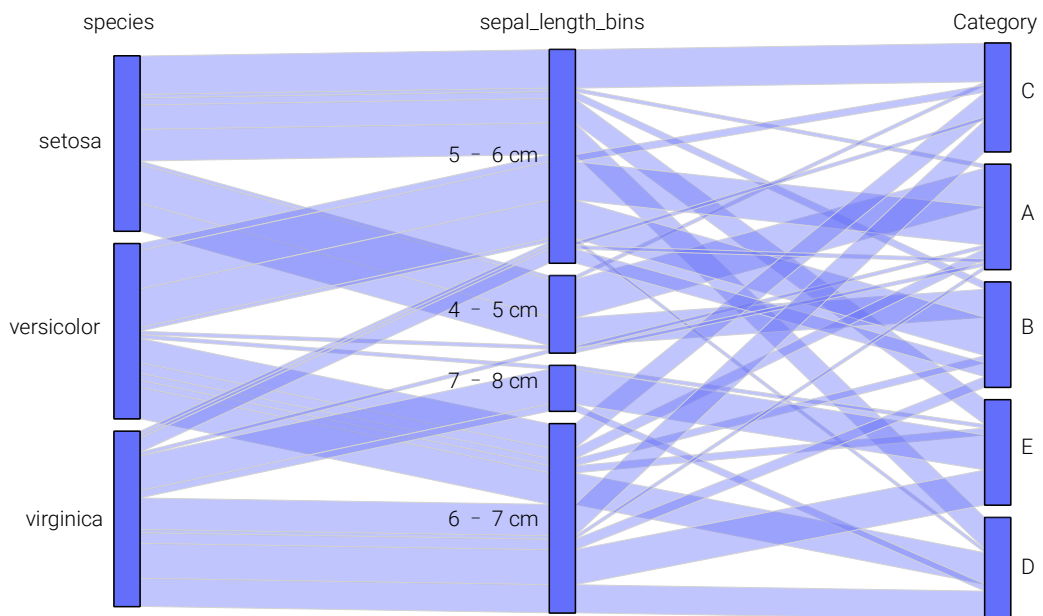


图 19. 分类数据平行坐标图，钻取顺序 species → sepal_length_bins → Category；单一颜色

代码 16 中 ^a 在用 `plotly.express.parallel_categories()` 绘图时，利用 `dimensions` 设定了钻取顺序。

```

# 平行坐标图，分类数据关系图
dims_2 = ['species', 'sepal_length_bins', 'Category']
fig = px.parallel_categories(df,
                           dimensions = dims_2,
                           width = 800, height = 500)
fig.show()

```

代码 16. 分类数据平行坐标图；使用时配合前文代码 | Bk1_Ch23_01.ipynb

不同于图 19，图 20 在绘制平行坐标图时，我们用色谱渲染选定特定（图 20 选定的是 `species` 分类）。

目前，`plotly.express.parallel_categories()` 函数不能接受分类字符串作为颜色映射的输入值，我们需要做一次映射，把鸢尾花分类字符串（`'setosa'`、`'versicolor'`、`'virginica'`）转化成数值。

这就是代码 17^a 要完成的任务。

^b 调取了 Plotly Express 库的 `colors.sequential` 模块中预定义的颜色映射。

^c 在绘制平行坐标图时，用参数 `color="species_numerical"` 指定了用于着色的列。这意味着不同的 `"species_numerical"` 值将被映射到不同的颜色，以区分不同的鸢尾花分类。

`color_continuous_scale=cmap` 这个参数用于指定颜色映射。

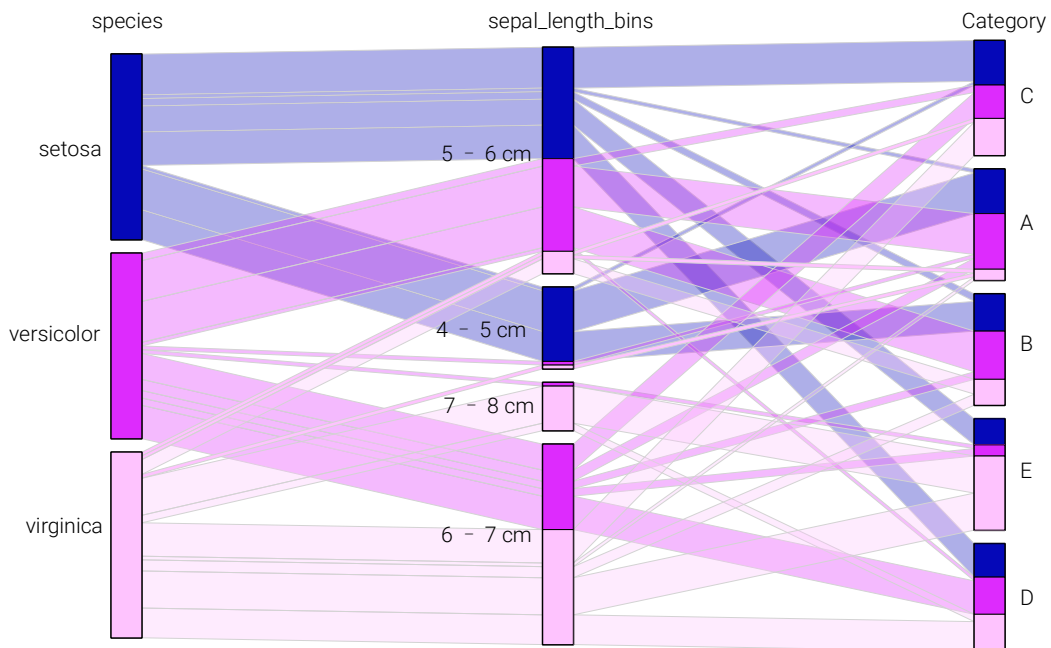


图 20. 分类数据平行坐标图，钻取顺序 `species` → `sepal_length_bins` → `Category`；用颜色区分另外一个特征

```

# 将species分类转为数值
a df["species_numerical"] = df["species"].map(
b   {"setosa": 0, "versicolor": 1, "virginica": 2})
c cmap = px.colors.sequential.Plotly3

# 可视化
fig = px.parallel_categories(df, dimensions = dims_2,
                             color = "species_numerical",
                             color_continuous_scale = cmap,
                             width = 800, height = 500)

fig.show()

```


代码 17. 分类数据平行坐标图，增加颜色特征；使用时配合前文代码 |  Bk1_Ch23_01.ipynb

图 21 所示为**冰柱图** (icicle plot)，这种可视化方案常用来展示分层数据的结构和关系。

冰柱图的外观类似于树状图；但与树状图不同，冰柱图的主要目的是在有限的空间内用矩形有效地表示分类层次化信息。

图 21 左侧最大矩形代表样本数据整体，从左往右代表钻取顺序为 `species` → `Category` → `sepal_length_bins`。

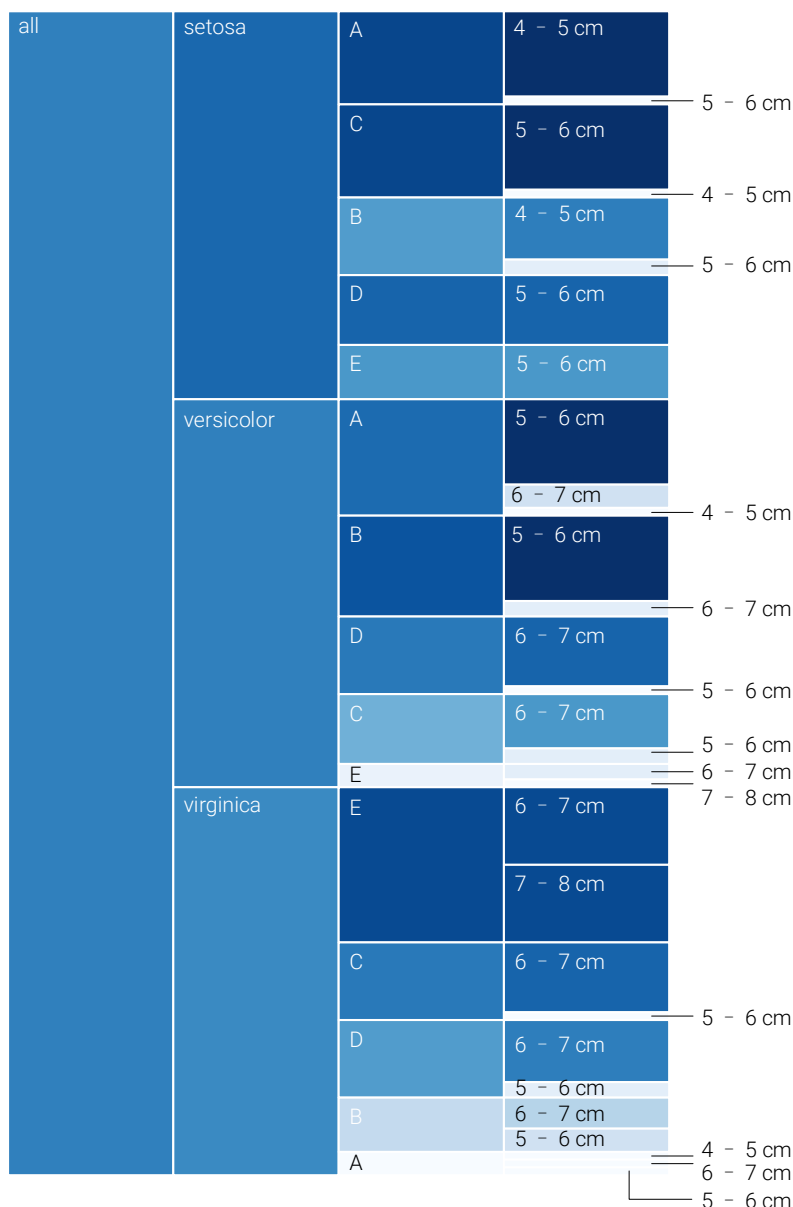


图 21. 冰柱图可视化计数；钻取顺序 species → Category → sepal_length_bins

代码 18 中 ^a 还是使用大家应该很熟悉的 `pandas.crosstab()` 创建交叉制表。

参数 `index = [df.species, df.Category]` 指定交叉制表的行索引。它包括两个变量，这意味着将数据集按照 `df.species` 和 `df.Category` 这两个变量的组合进行分组。

`columns = df.sepal_length_bins` 是交叉制表的列索引。

`values = df.petal_length` 是要进行统计的数据帧的列。

`aggfunc = 'count'` 指定交叉制表进行聚合操作的函数。'count' 表示要计算每个分类组合样本的频数。

^b 对生成的多级列索引数据帧先后进行堆叠 (`stack`)、重置索引 (`reset_index`) 操作。请大家思考每步操作后数据帧会发生怎样变化。

^c 对特定列进行重命名。

d 删除 'count' 为 0 行，这是因为当前版本 `plotly.express.icicle()` 遇到子集样本数为 0 时，会报错。

e 利用 `plotly.express.icicle()` 绘制冰柱图。

参数 `path` 用于指定冰柱图中的路径或层次结构，也就是钻取顺序。注意，`px.Constant('all')` 代表在冰柱图中加入顶级（样本总体）节点，也就是图 21 左侧矩形。

参数 `values='count'` 用于指定在每个冰柱图节点上显示的数值。

参数 `color_continuous_scale='Blues'`，指定颜色映射。

参数 `color='count'` 指定了矩形着色的参考依据。

```
# 交叉计数，计数
a count_matrix = pd.crosstab(index = [df.species, df.Category],
                             columns = df.sepal_length_bins,
                             values=df.petal_length, aggfunc='count')

b count_matrix = count_matrix.stack().reset_index()
c count_matrix.rename(columns = {0:'count'}, inplace = True)
d count_matrix = count_matrix[count_matrix['count'] != 0]
# 删除 'count' 列值为0的行

# 冰柱图
e fig = px.icicle(count_matrix,
                 path=[px.Constant("all"),
                     'species', 'Category', 'sepal_length_bins'],
                 values = 'count',
                 color_continuous_scale='Blues',
                 color = 'count',
                 width = 600, height = 800)

fig.show()
```

代码 18. 冰柱图可视化计数；使用时配合前文代码 |  Bk1_Ch23_01.ipynb

图 22 给出的可视化方案叫做矩形树形图，它的功能和冰柱图类似。

代码 19 采用 `plotly.express.treemap()` 绘制矩形树形图，请大家逐行注释。此外请大家尝试其他钻取顺序绘制本节的冰柱图和矩形树形图。

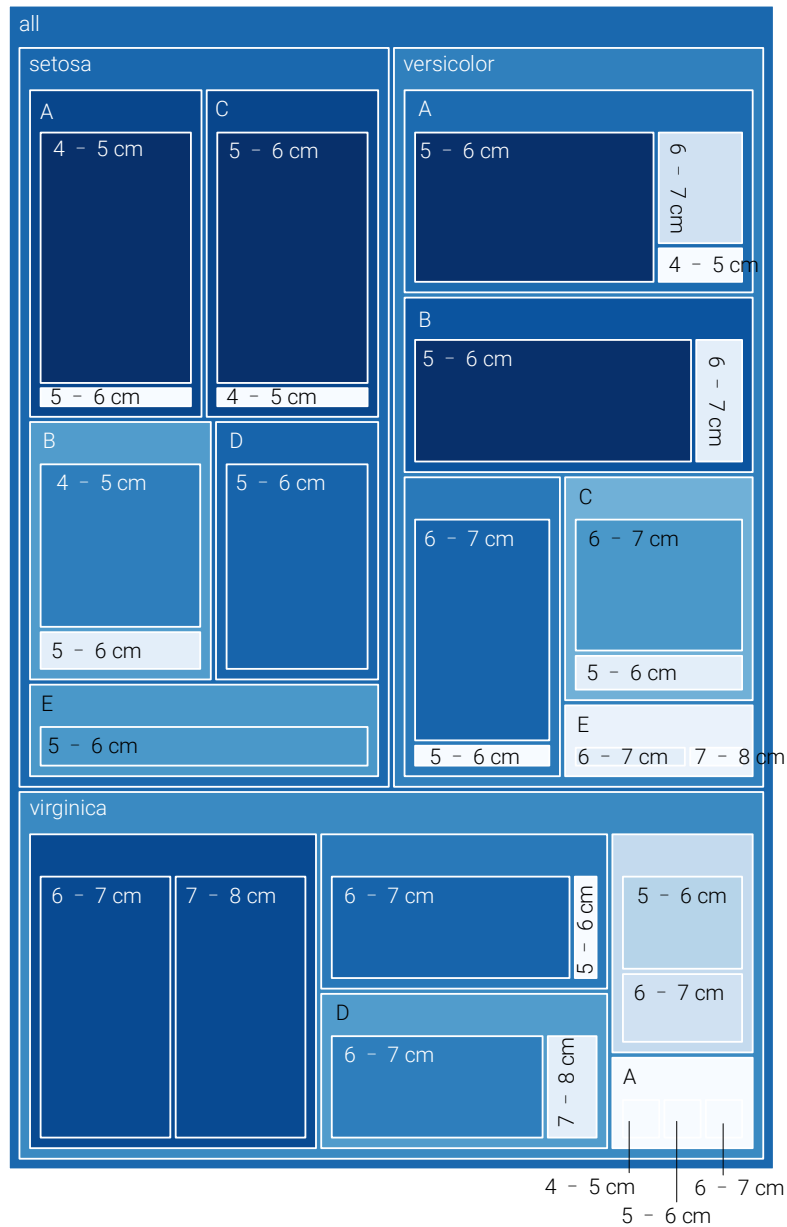


图 22. 矩阵树形图可视化计数；钻取顺序 species → Category → sepal_length_bins

```

# 矩阵树形图
fig = px.treemap(count_matrix,
                 path=[px.Constant("all"),
                       'species', 'Category', 'sepal_length_bins'],
                 values = 'count',
                 color_continuous_scale='Blues',
                 color = 'count',
                 width = 600, height = 800)
fig.show()

```

代码 19. 矩阵树形图可视化计数；使用时配合前文代码 | Bk1_Ch23_01.ipynb

23.8 平均值的钻取：全集 vs 子集

对于鸢尾花数据集，我们可以计算 150 个样本点在花瓣长度特征上的平均值（约为 3.578 cm）。但是，如果要问鸢尾花类别为 *setosa* 的样本花瓣长度平均值，我们就需要选取满足条件的子集，然后再计算平均值。

而图 23 便能回答刚才提出的这个问题。

图 23 中给出的这种均值又叫**条件均值** (conditional average)，或**条件期望** (conditional expectation)。

大家如果好奇图 23 这三个条件均值和样本整体均值（约为 3.578 cm）的关系，可以自己算一下。我们会发现，图 23 中三个值求和再求平均结果约为 3.578（注意，图中数值仅仅保留两位小数）。

但是，请大家注意这是一个明显错误的运算。虽然最终结果是正确的，但是计算时采用的数学工具完全错误。

用条件均值计算样本整体均值时要考虑每个条件均值的“贡献度”，也就是权重。而如图 10 所示，鸢尾花数据中，*setosa*、*versicolor*、*virginica* 各占 1/3，因此正确的计算过程应该为加权平均，即 $1.46 \times (1/3) + 4.26 \times (1/3) + 5.55 \times (1/3)$ 。



鸢尾花书《统计至简》会专门介绍条件概率、条件期望、条件方差等概念。

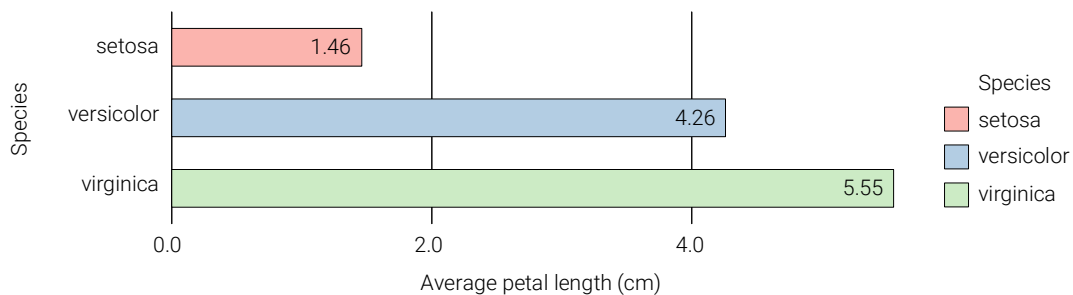


图 23. 可视化花瓣长度条件均值，species 维度；仅保留两位小数

代码 20 计算条件均值并绘制图 23，代码相对简单，下面简单介绍。

- a 利用 `groupby()` 方法以 `species` 分组计算 `petal_length` 均值，然后重置索引。
- b 利用 `plotly.express.bar()` 绘制水平柱状图。请大家逐行注释。


```

# 分别计算每个子类 (species) petal_length均值
petal_length_mean_by_species = df.groupby([
    'species'])['petal_length'].mean().reset_index()

fig = px.bar(petal_length_mean_by_species,
             x='petal_length', y='species',
             color='species',
             color_discrete_sequence=px.colors.qualitative.Pastel1,
             width=600, height=300,
             text_auto='.2f', orientation='h',
             template="plotly_white")

fig.show()

```

代码 20. 计算并可视化花瓣长度条件均值, species 维度; 使用时配合前文代码 |  Bk1_Ch23_01.ipynb

如果我们想知道 *setosa* 标签鸢尾花中, Category 为 A 的子类样本花瓣均值 (条件均值), 我们就可以使用图 24 来回答。

? 再请大家算一遍, 图 24 中 *setosa* 对应的五个数值的均值是否为图 23 中的 1.46。然后, 再用图 14 给出的权重, 再算一遍“加权平均数”, 以便加强印象。

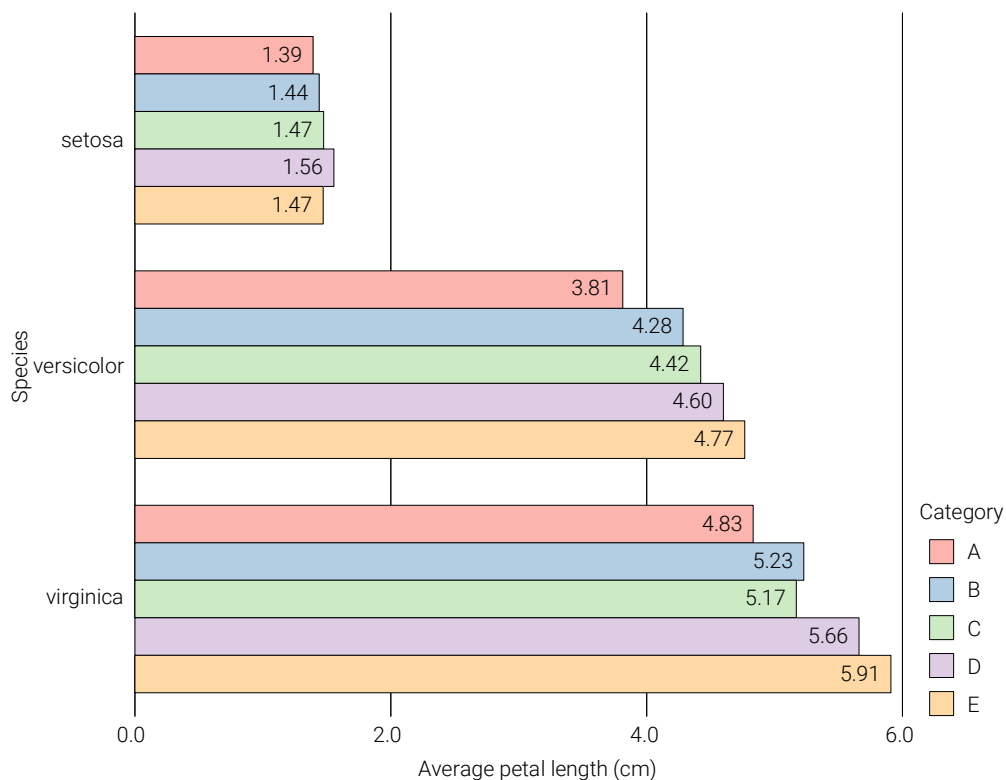



图 24. 可视化花瓣长度均值, 钻取顺序 species → Category

代码 21 中  采用 `groupby()` 方法计算条件均值, 分组时采用两个维度。

- b 给出第二种方法，采用 `pandas.crosstab()` 函数，请大家考虑下一步如何把宽格式改为长格式。
- c 绘制水平柱状图。请大家自行注释。

```

# 分别计算每个子类 (x Category y species) petal_length均值
petal_length_mean_by_species_ctgr = df.groupby([
    'Category', 'species'])['petal_length'].mean().reset_index()

# 另外一种计算方法
# 创建交叉指标，计算petal_length平均值
# 行: species; 列: Category
pd.crosstab(index = df.species, columns = df.Category,
            values=df.petal_length, aggfunc='mean')

# 可视化petal_length均值，先species分类再Category分类
fig = px.bar(petal_length_mean_by_species_ctgr,
             x = 'petal_length', y = 'species',
             color = 'Category', barmode = 'group',
             text_auto = '.2f', orientation = 'h',
             width=600, height=600,
             color_discrete_sequence=px.colors.qualitative.Pastel1,
             template = "plotly_white")

fig.show()

```

代码 21. 计算并可视化花瓣长度均值，钻取顺序 species → Category; 使用时配合前文代码 | Bk1_Ch23_01.ipynb

图 25 所示为在不同 Category 分类条件下，花瓣长度条件均值。比如，给定 `Category == 'A'` 的条件下，花瓣长度条件均值为 2.95。Category == 'A' 就是所谓“条件”。

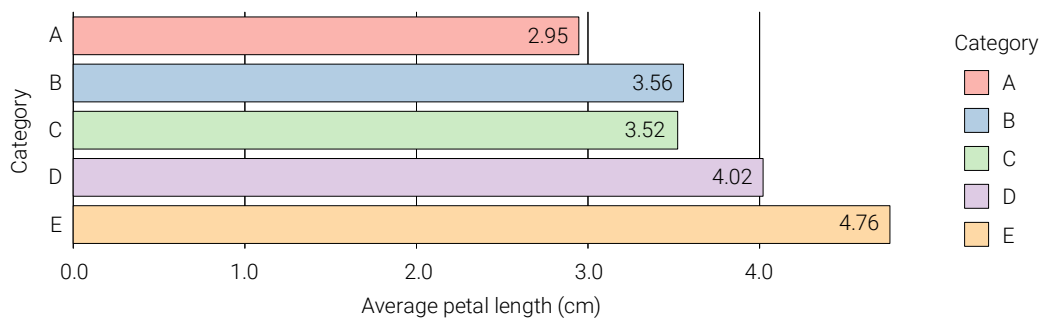


图 25. 可视化花瓣长度条件均值，Category 分类

代码 22 绘制图 25，请大家自行注释。

```

# 分别计算每个子类 (Category) petal_length均值
petal_length_mean_by_ctgr = df.groupby(['Category'])['petal_length'].mean().reset_index()
fig = px.bar(petal_length_mean_by_ctgr,
             x='petal_length', y='Category',
             color='Category',
             color_discrete_sequence=px.colors.qualitative.Pastel1,
             width=600, height=300,
             text_auto='.2f', orientation='h',
             template="plotly_white")
fig.show()

```

代码 22. 计算并可视化花瓣长度条件均值, Category 维度; 使用时配合前文代码 | Bk1_Ch23_01.ipynb

相比图 24, 图 26 在可视化花瓣长度 (条件) 均值时采用的钻取顺序为 Category → species。

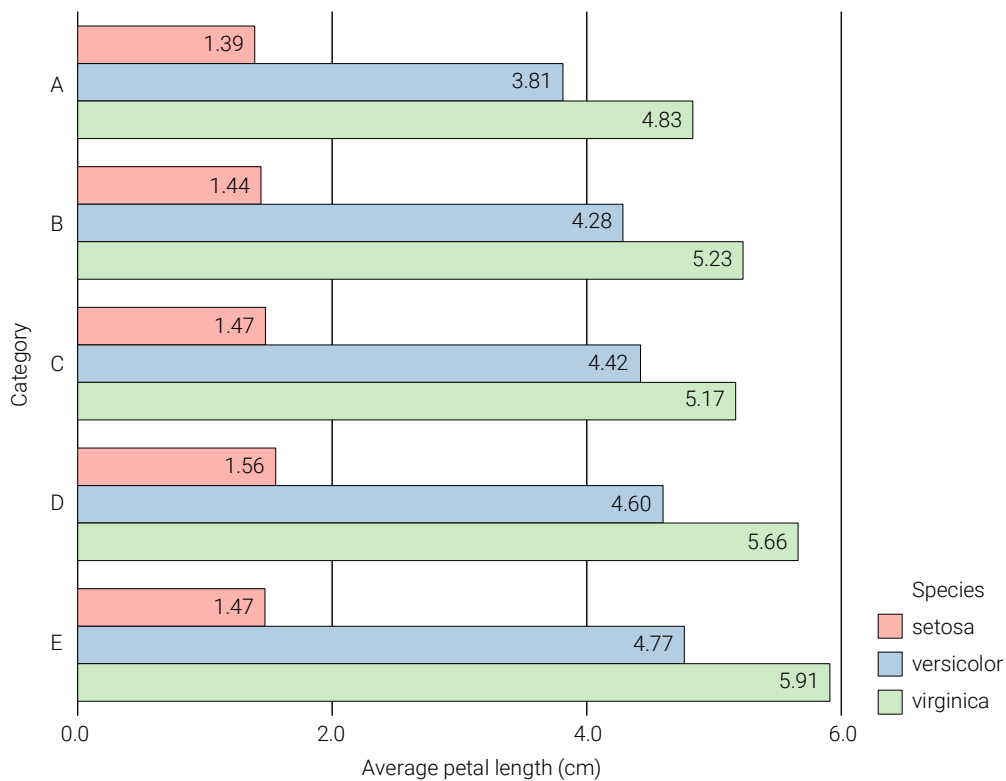



图 26. 可视化花瓣长度均值, 钻取顺序 Category → species

请大家自行分析代码 23。

```

# 绘制水平分组柱状图
fig = px.bar(petal_length_mean_by_species_ctgr,
             x='petal_length', y='Category',
             color='species', barmode='group',
             color_discrete_sequence=px.colors.qualitative.Pastel1,
             width=600, height=600,
             text_auto='.2f', orientation='h',
             template="plotly_white")
fig.show()

```

代码 23. 计算花瓣长度均值, 钻取顺序 Category → species; 使用时配合前文代码 |  Bk1_Ch23_01.ipynb

函数 `pandas.crosstab()` 的使用方法可以很灵活。表 7 和表 8 条件均值都是用这个函数计算得到的, 对应的代码为代码 24, 请大家自行学习并注释。

表 7. 花瓣长度均值; 行: sepal_length_bins; 列: species → Category

species	setosa					versicolor					virginica				
Category	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
sepal_length_bins															
4 - 5 cm	1.37	1.48	1.40			3.30					4.50				
5 - 6 cm	1.60	1.30	1.48	1.56	1.47	3.73	4.15	4.20	4.80		5.00	5.05	5.10		
6 - 7 cm						4.30	5.00	4.49	4.58	4.80	5.00	5.47	5.18	5.45	5.56
7 - 8 cm										4.70				6.50	6.26

表 8. 花瓣长度均值; 行: species → Category; 列: sepal_length_bins


species	Category	4 - 5 cm	5 - 6 cm	6 - 7 cm	7 - 8 cm
setosa	A	1.37	1.60		
	B	1.48	1.30		
	C	1.40	1.48		
	D		1.56		
	E		1.47		
versicolor	A	3.30	3.73	4.30	
	B		4.15	5.00	
	C		4.20	4.49	
	D		4.80	4.58	
	E			4.80	4.70
virginica	A	4.50	5.00	5.00	
	B		5.05	5.47	
	C		5.10	5.18	
	D			5.45	6.50
	E			5.56	6.26


```

# 计算花萼长度条件均值;
# 行: sepal_length_bins
# 列: species > Category
a pd.crosstab(index = df.sepal_length_bins,
              columns = [df.species, df.Category],
              values=df.petal_length, aggfunc='mean')

# 计算花萼长度条件均值;
# 行: species > Category
# 列: sepal_length_bins
b pd.crosstab(index = [df.species, df.Category],
              columns = df.sepal_length_bins,
              values=df.petal_length, aggfunc='mean')

```

代码 24. 计算并可视化花瓣长度条件均值，更复杂的条件组合；使用时配合前文代码 |  Bk1_Ch23_01.ipynb

 大家会发现本节在设置不同 Plotly 可视化方案时，有几个共享设置，比如风格、类别顺序等。请大家思考我们如何仅仅创建一个变量 kwarg，然后在代码不同位置拆包调用 kwarg。



请大家完成如下题目。

Q1. 在本章例子基础之上，计算鸢尾花花瓣长宽比值，以此作为划分样本数据的第四维度。用 `pandas.cut()` 将样本数据分为 4 类，然后再用太阳爆炸图、冰柱图、矩形树形图可视化四个维度的钻取。

* 这道题目很基础，本书不给答案。



这一章用鸢尾花数据为例给大家展示了“Pandas + Plotly”讲故事的力量！Pandas 有处理数据的强大能力，而 Plotly 的交互式可视化方案让数据跃然纸上。

下一章将继续用“Pandas + Plotly”分析时间序列数据。此外，鸢尾花书《可视之美》将继续介绍更多 Plotly 的可视化方案。