Plotly Data Visualization and Storytelling

23 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.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

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。本章最后用鸢尾花数据再举一个例子，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 (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。

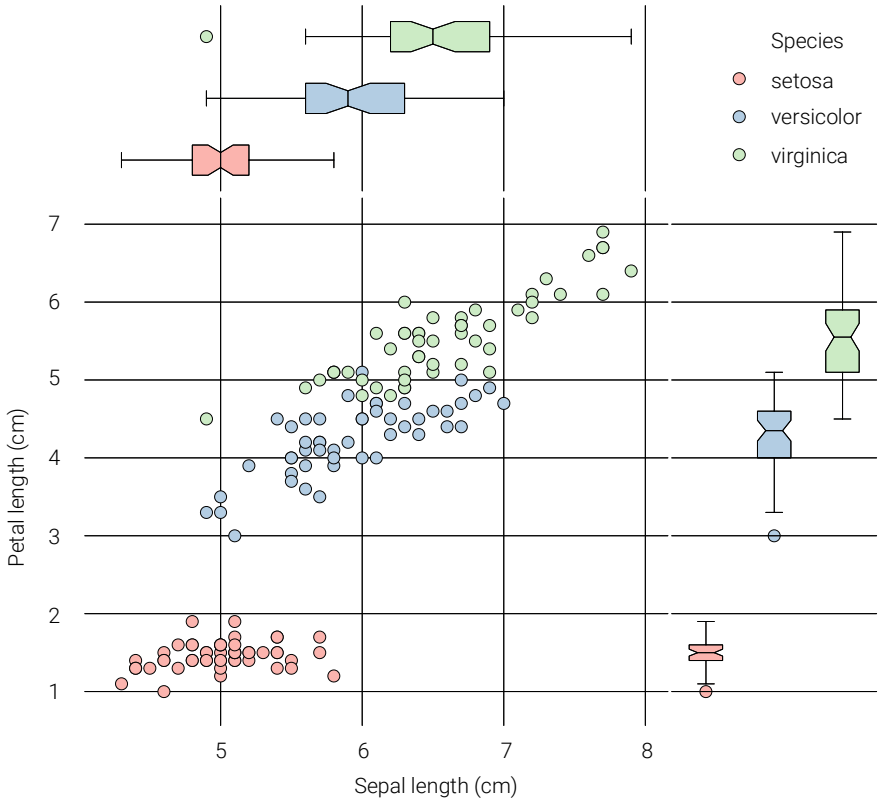

图 1. 用 Plotly 绘制散点图 (横轴为花萼长度，纵轴为花瓣长度)，边缘分布为箱型图，考虑鸢尾花分类

图 2 代码绘制图 1，下面聊聊其中核心语句。

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
a 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',
b                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)"})
c fig.show()
```

图 2. 用 Plotly 散点图可视化鸢尾花数据

```
# 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()
```

图 3．用 Plotly 绘制成对散点图

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。“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

	Area (cm^2)						
Category	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

图 4 代码完成上述运算分析，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 JupyerLab 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')
```

图 4. 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；

图 5 所示为不同'Category'条件下的“花萼面积”散点图，对应图 7 中 a。图 6 不同'Category'条件下的“花萼面积”的“箱型图 + 散点图”，对应图 7 中 b。

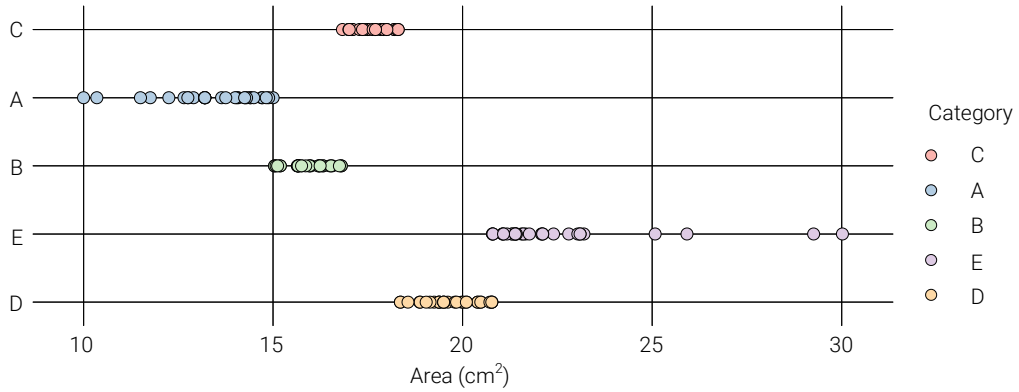

图 5. 用“花萼面积”对鸢尾花数据集再分割

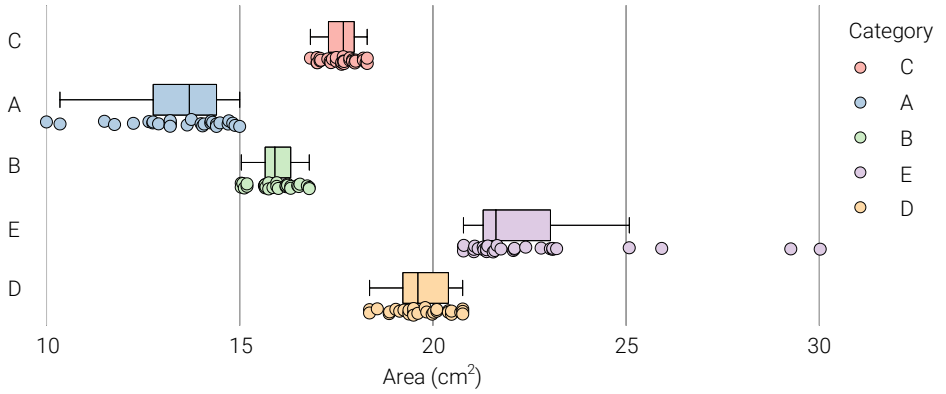

图 6. “花萼面积”的箱型图，考虑'Category'分类

```
# 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()
```

图 7. 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

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 8 所示为考虑'Category'分类情况下，鸢尾花长度的箱型图。图 9 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。

图 10 中 a 和 b 分别绘制图 8 和图 9，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。

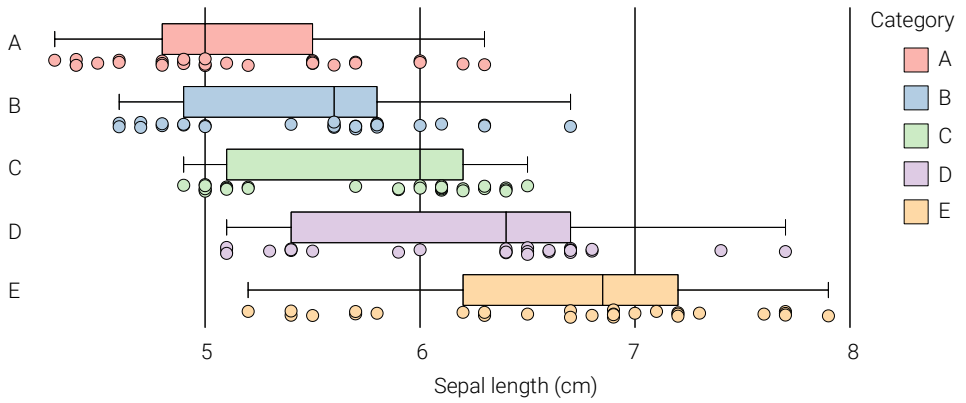

图 8. 花萼长度的箱型图，考虑'Category'分类

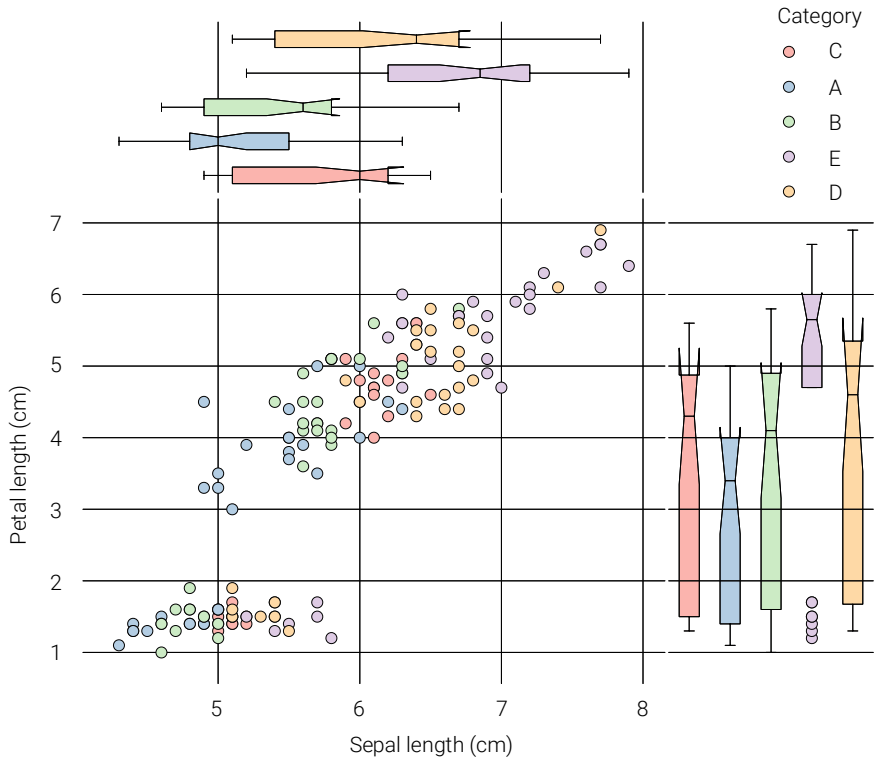

图 9. 用 Plotly 绘制散点图 (横轴为花萼长度，纵轴为花瓣长度)，边缘分布为箱型图，考虑“花萼面积”分类

```
# 花萼长度的箱型图，考虑'Category'分类
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
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()
```

图 10. 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

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 11 (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 11 (a) 的数值为频数 (frequency)，即计数。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 11 (a) 所有值之和是否为 150。然后再分别计算图 11 (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 11 (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 11 (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。

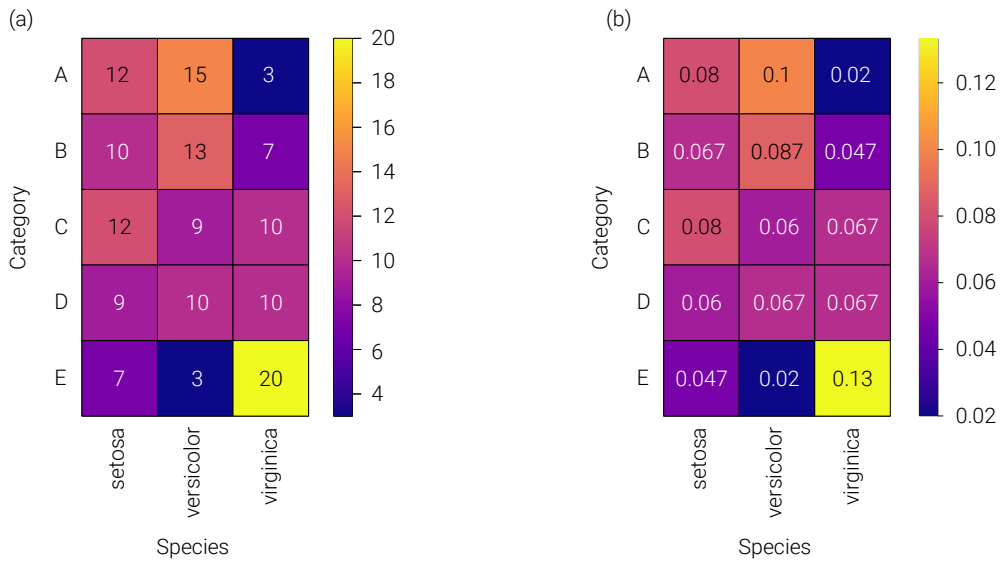

图 11. 用 plotly.express.imshow() 绘制频率和概率热图，两个分割维度

```
a  # 从Category和species两个维度切割鸢尾花数据，结果为二维频率
   freq_matrix = pd.crosstab(index = df['Category'],
                             columns = df['species'])

b  # 可视化二元频率数组
   fig = px.imshow(freq_matrix, text_auto=True)
   fig.show()

   # 从Category和species两个维度切割鸢尾花数据，结果为二维概率
c  prob_matrix = pd.crosstab(index = df['Category'],
                             columns = df['species'],
                             normalize = 'all')

d  # 可视化二元概率数组
   fig = px.imshow(prob_matrix, text_auto='.3f')
   fig.show()
```

图 12. 用 plotly.express.imshow() 绘制频率和概率热图；使用时配合前文代码

图 11 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 13 所示。

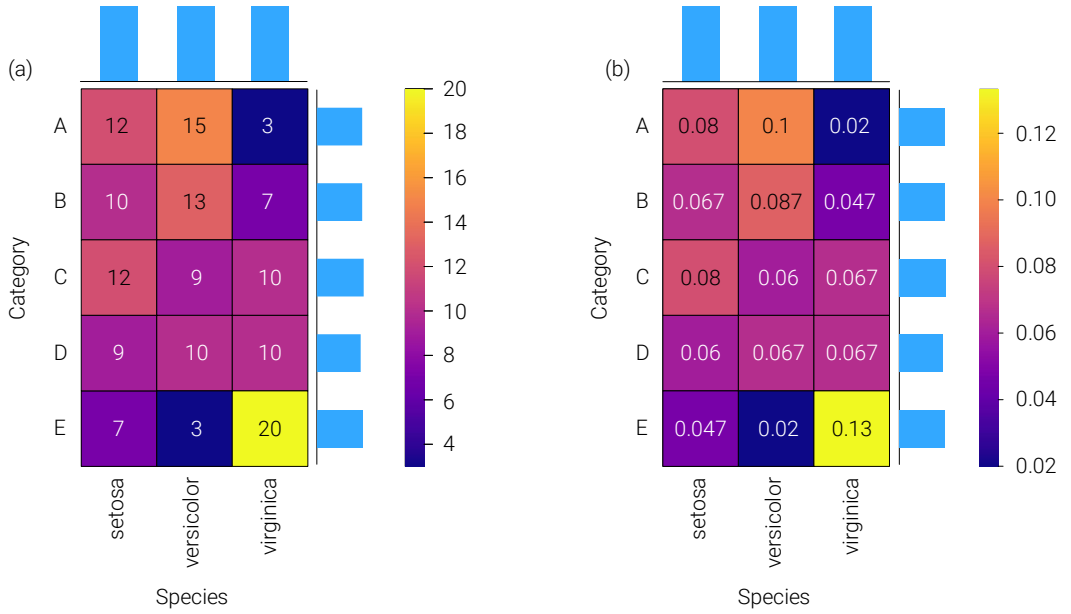

图 13. 用 plotly.express.density_heatmap() 绘制频率和概率热图，边缘分布为直方图，两个分割维度

```python
# 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()
```

图 14. 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

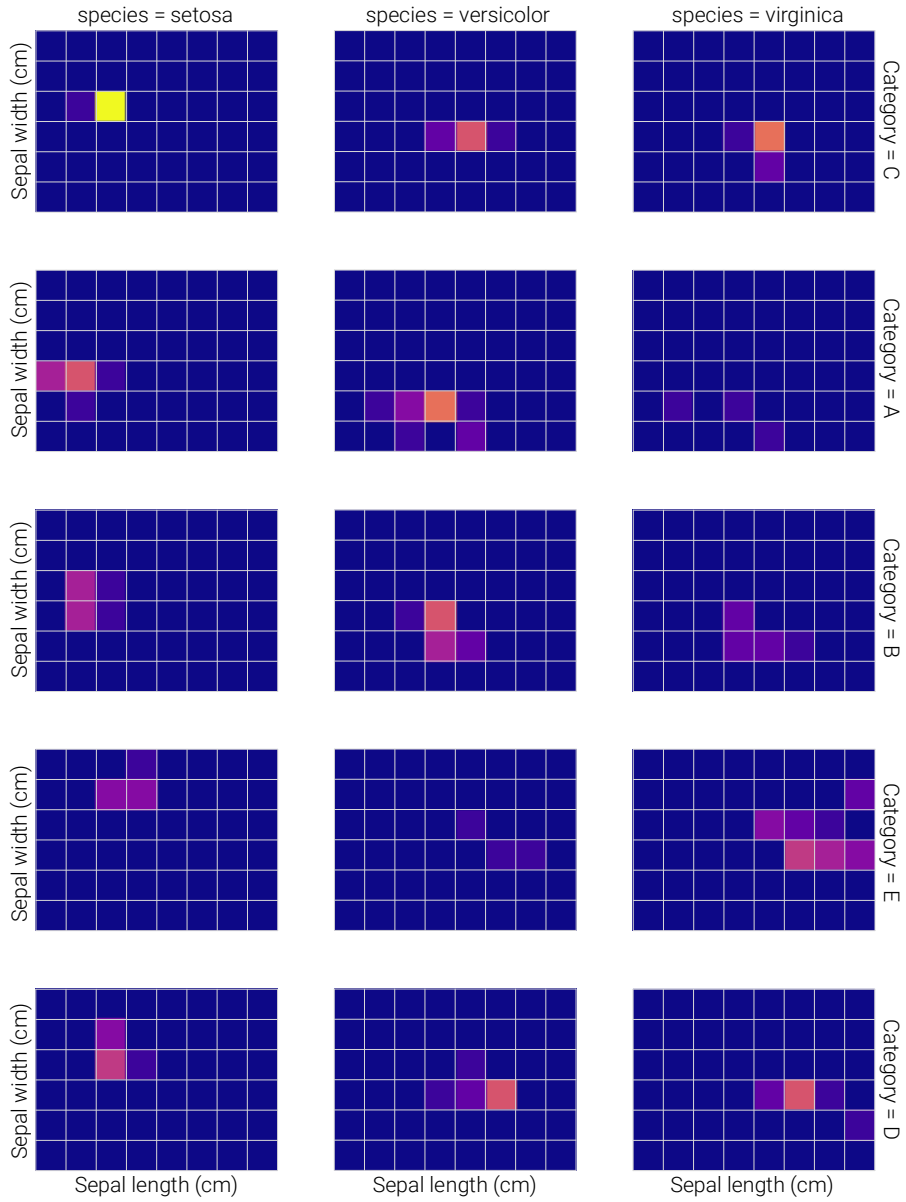

图 15. 频率热图，子图布局

```
# 绘制频率热图，子图布置
fig = px.density_heatmap(df,
                         x="sepal_length", y="sepal_width",
                         facet_row="Category", facet_col="species",
                         width = 500, height = 800, text_auto=True)
fig.show()
```

图 16. 用 plotly.express.density_heatmap() 绘制频率热图，子图布置

23.4 可视化比例：柱状图、饼图

如果我们好奇鸢尾花数据集中不同标签数据对应的数据比例，则可以用 `pandas.Series.value_counts(normalize=True)` 完成计算。

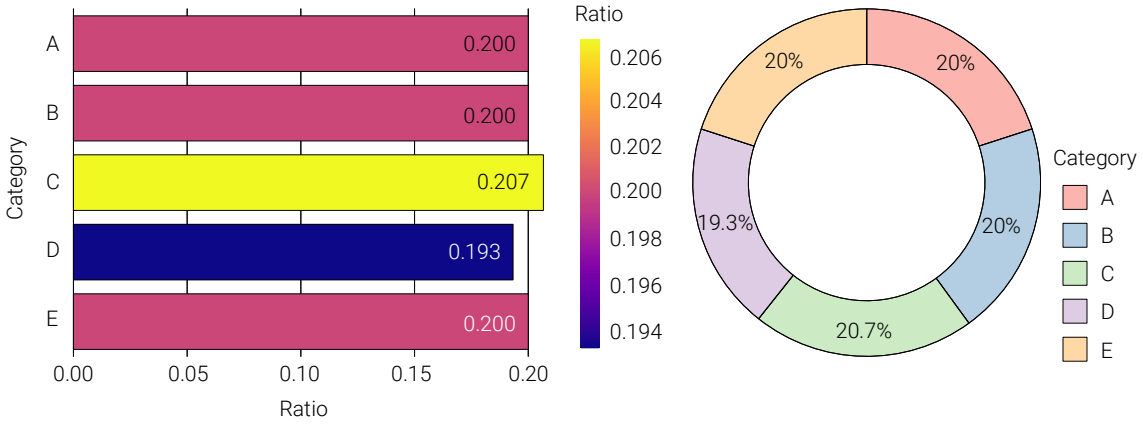

图 17. 用 `plotly.express.bar()`和 `plotly.express.pie()`可视化`'Category'`分类比例

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

# 用直方图展示 Category 分类比例
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
fig = px.pie(ctg_percent,
             category_orders={"Category": ["A", "B", "C", "D", "E"]},
             color_discrete_sequence=px.colors.qualitative.Pastel1,
             values='Ratio', names='Category')
fig.update_traces(hole=.68)
fig.show()
```

图 18. 可视化`'Category'`分类比例，柱状图和饼图；使用时配合前文代码

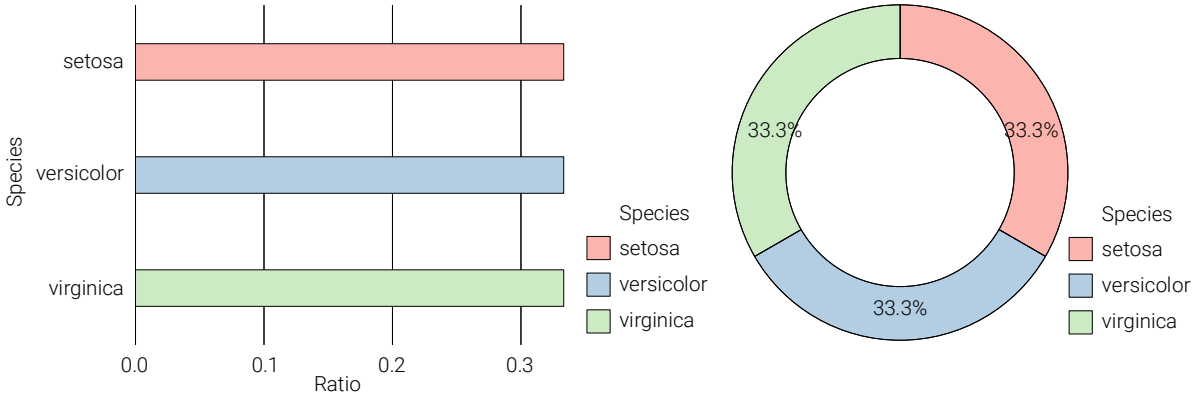

图 19. 用 plotly.express.bar()和 plotly.express.pie()可视化'species'分类比例

```
a  # 计算 species 分类比例
   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()
```

图 20. 可视化'species'分类比例，柱状图和饼图；使用时配合前文代码

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。下面，我们就利用 Pandas 和 Plotly 试着完成钻取运算和可视化。

注意，图 21 和图 22 中所有分段柱子之和均为 1。

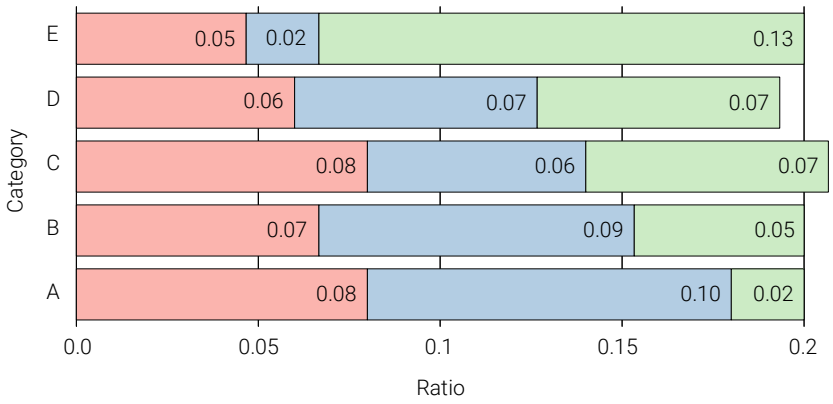

图 21. 用堆叠柱状图可视化 Category 中 species 的绝对比例，所有分段柱子之和为 1

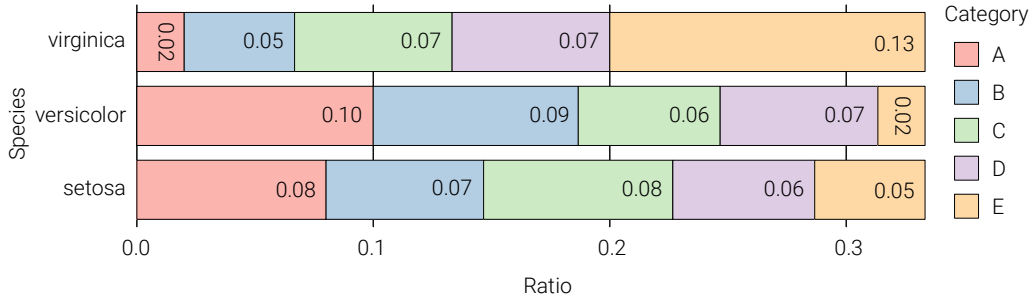

图 22. 用堆叠柱状图可视化 species 中 Category 的绝对比例，所有分段柱子之和为 1

```python
# 对 Category 比例值在 species 维度上钻取
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
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()
```

图 23. 用 plotly.express.bar()绘制堆叠柱状图；使用时配合前文代码

相对比例钻取

图 21 有个缺陷，它不能直接回答“Category 为 A 时，setosa 的占比为多少？”

也就是说，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)。

注意，图 24 和图 25 中所有分段柱子之和均为 1。

鸢尾花书《统计至简》将专门讲解条件概率。

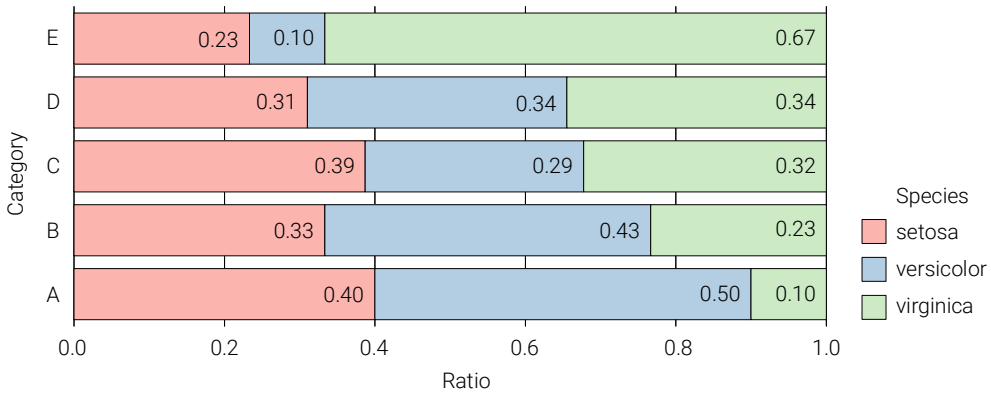

图 24. 用堆叠柱状图可视化 Category 中 species 的相对比例，每个柱子分段长度之和为 1

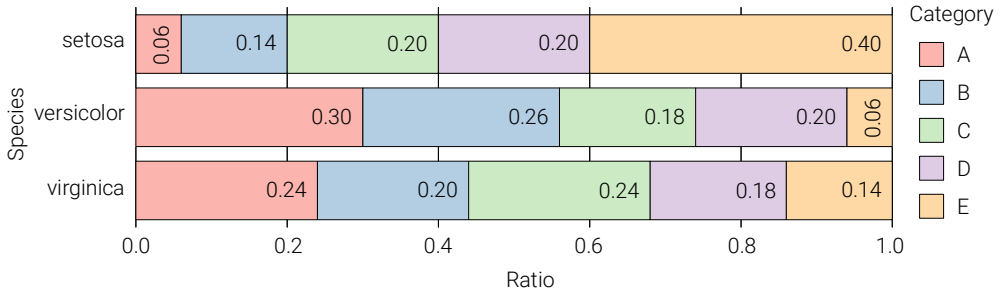

图 25. 用堆叠柱状图可视化 species 中 Category 的相对比例，每个柱子分段长度之和为 1

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

# 可视化
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的比例
ratio_category_in_species = pd.crosstab(index = df['Category'],
                                        columns = df['species'],
                                        normalize = 'columns').T

# 可视化
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()
```

图 26. 计算并可视化相对比例；使用时配合前文代码

23.6 太阳爆炸图：展示层次结构

plotly.express.sunburst() 是 Plotly Express 库中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。

下面，我们用太阳爆炸图可视化上述比例钻取。

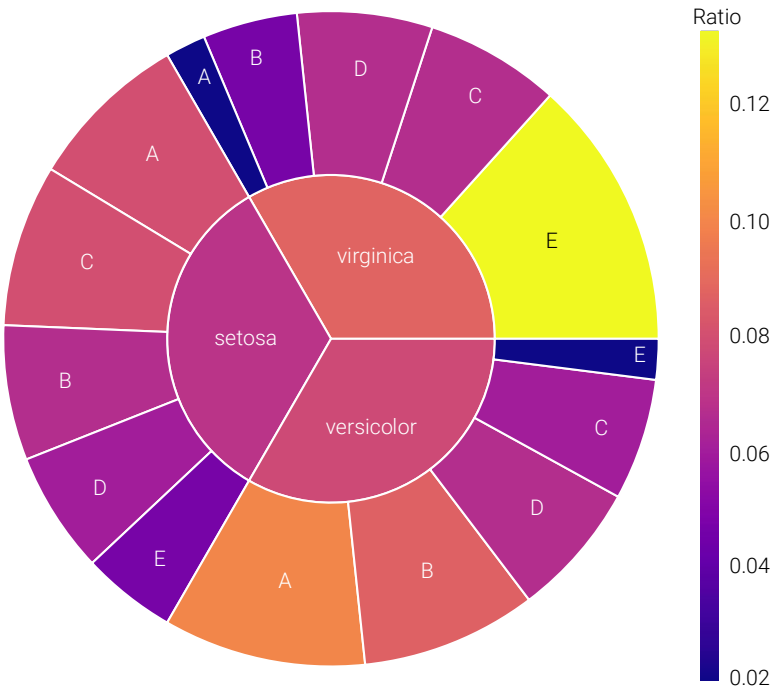

图 27. 太阳爆炸图可视化绝对比例钻取，先 species 后 Category

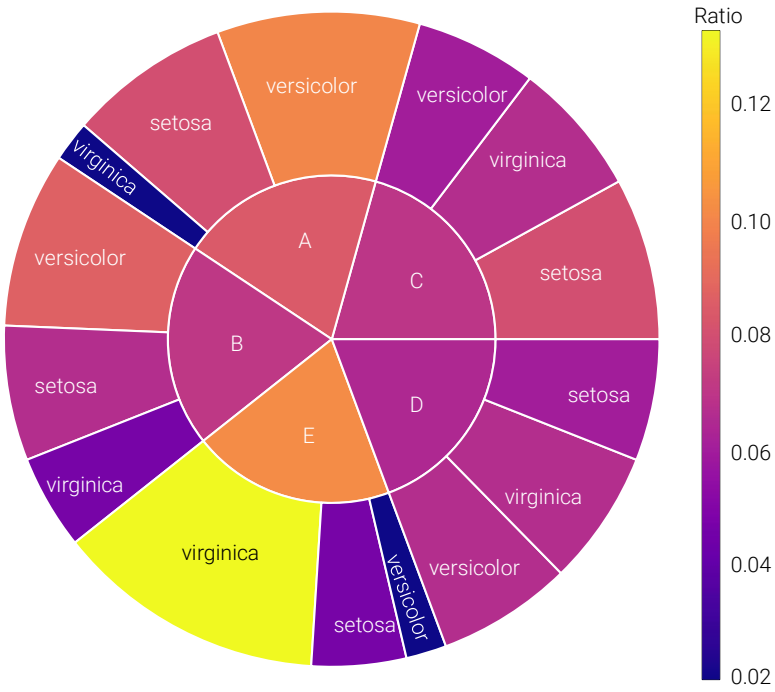

图 28. 太阳爆炸图可视化绝对比例钻取，先 Category 后 species

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。

```
# 将概率值（比例值）stack 起来
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
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
fig = px.sunburst(prob_matrix_stacked,
                  path=['Category', 'species'],
                  values='Ratio', color='Ratio',
                  width = 600, height = 600)
fig.show()
```

图 29. 用 plotly.express.sunburst()绘制太阳爆炸图；使用时配合前文代码

23.7 增加第三分类维度

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 4 到 5，左闭右开，之间的花萼长度。

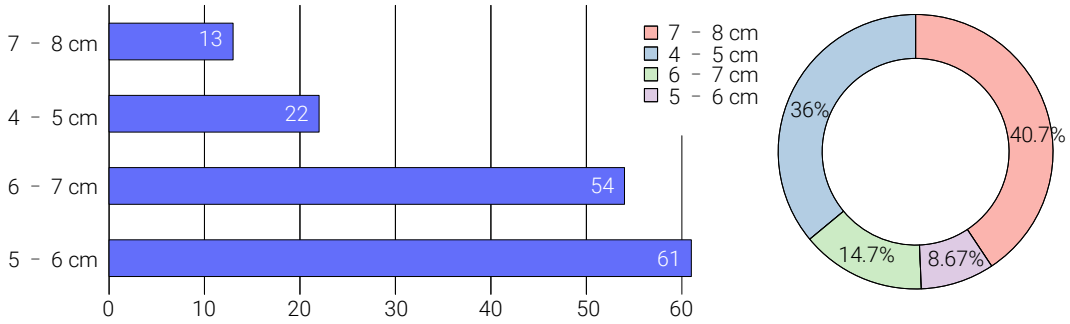

图 30. 可视化第三维度的样本计数

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

# 计算频数
sepal_length_bins_counts = df["sepal_length_bins"].value_counts()
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
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
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()
```

图 31. 根据花萼长度再增加一个分类维度；使用时配合前文代码

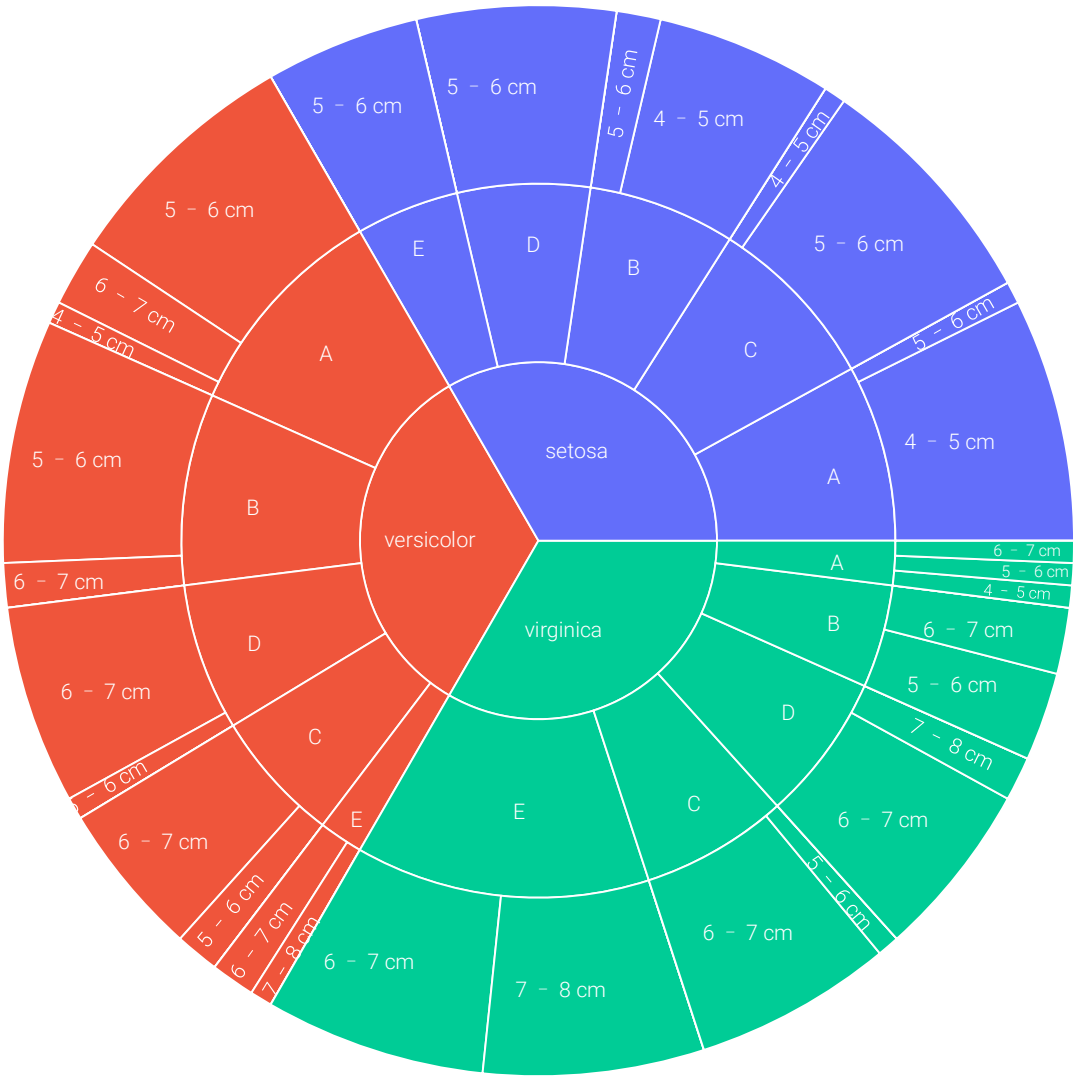

图 32. 太阳爆炸图可视化绝对比例钻取，先 species 再 Category 最后 sepal_length_bins

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

prob_matrix_by_3 = df.groupby(dims)['sepal_length'].apply(
    lambda x: x.count()/len(df))
prob_matrix_by_3 = prob_matrix_by_3.reset_index()
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
fig = px.sunburst(prob_matrix_by_3,
                  path=dims,
                  values='Ratio',
                  width = 600, height = 600)
fig.show()
```

图 33. 太阳爆炸图，三个维度；使用时配合前文代码

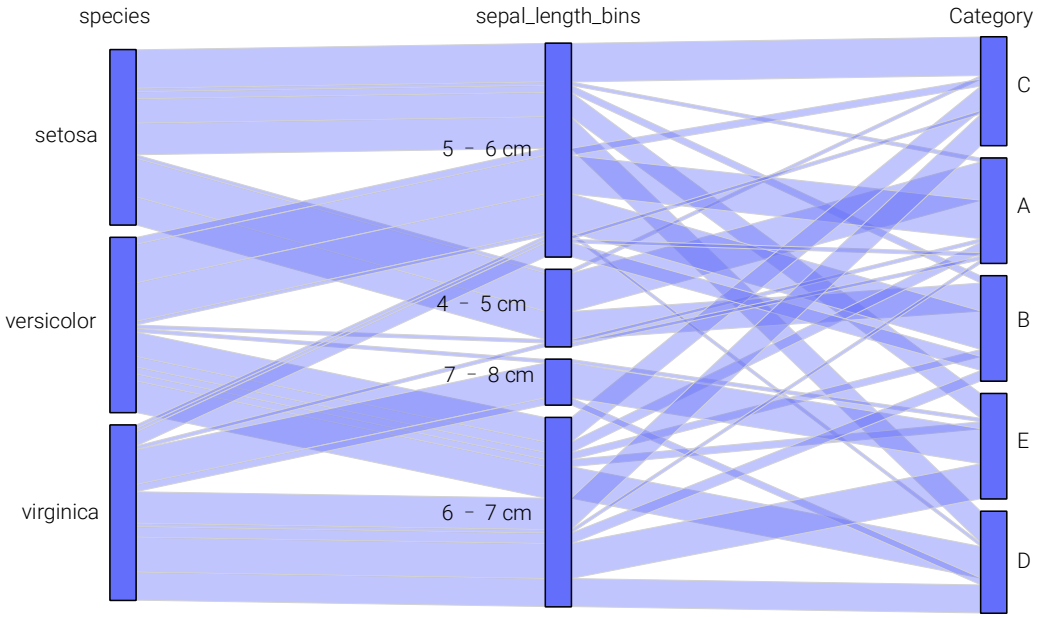

图 34. 分类数据平行坐标图，单一颜色

```python
# 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()
```

图 35. 分类数据平行坐标图；使用时配合前文代码

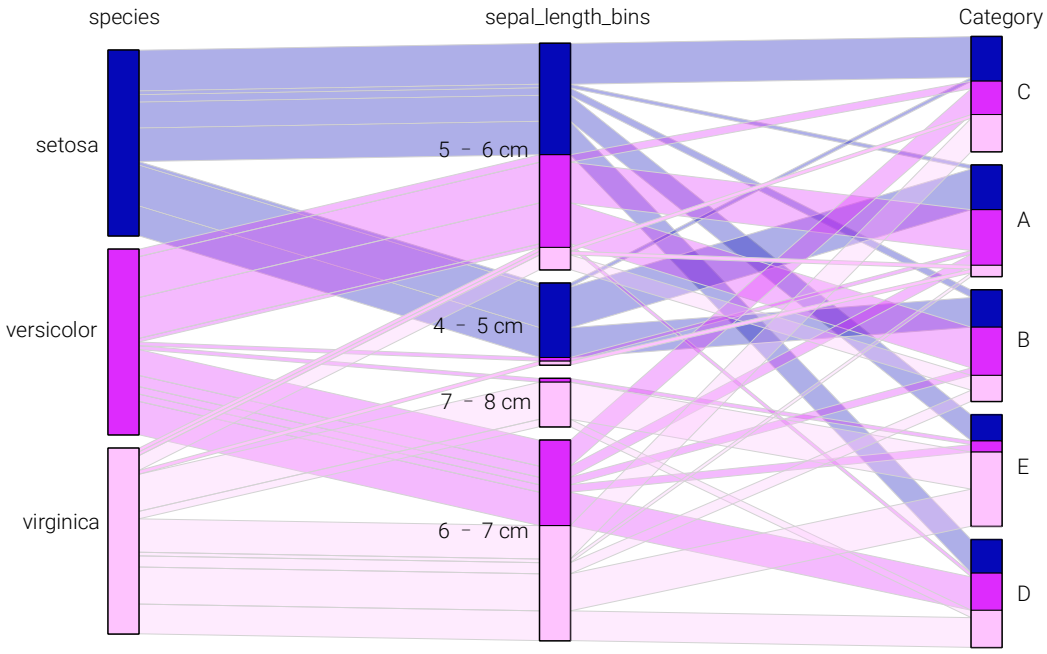

图 36. 分类数据平行坐标图，用颜色区分另外一个特征

```python
# 将species分类转为数值
df["species_numerical"] = df["species"].map(
    {"setosa": 0, "versicolor": 1, "virginica": 2})
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()
```

图 37. 分类数据平行坐标图，增加颜色特征；使用时配合前文代码

23.8 平均值的钻取

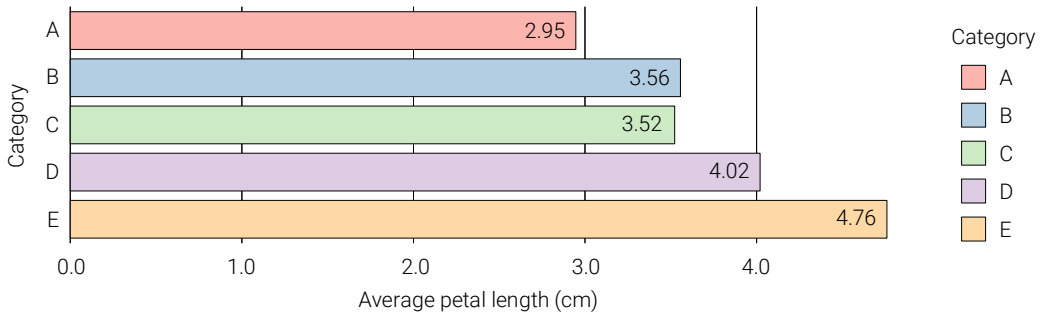

图 38. 可视化花萼长度均值，Category 分类

```python
# 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()
```

图 39. 计算并可视化花萼长度均值，Category 维度；使用时配合前文代码

大家可以计算一下，图 40 中 A 中三个数值的平均值不等于图 38 中 A 的平均值。大家可以自己算算其他 Category 分类，都支持这个结论。这是为什么？

原因其实藏在图 24 中！

也就是说，A 中 setosa、versicolor、virginica 相对占比不同，计算平均值时要考虑图 24 权重。

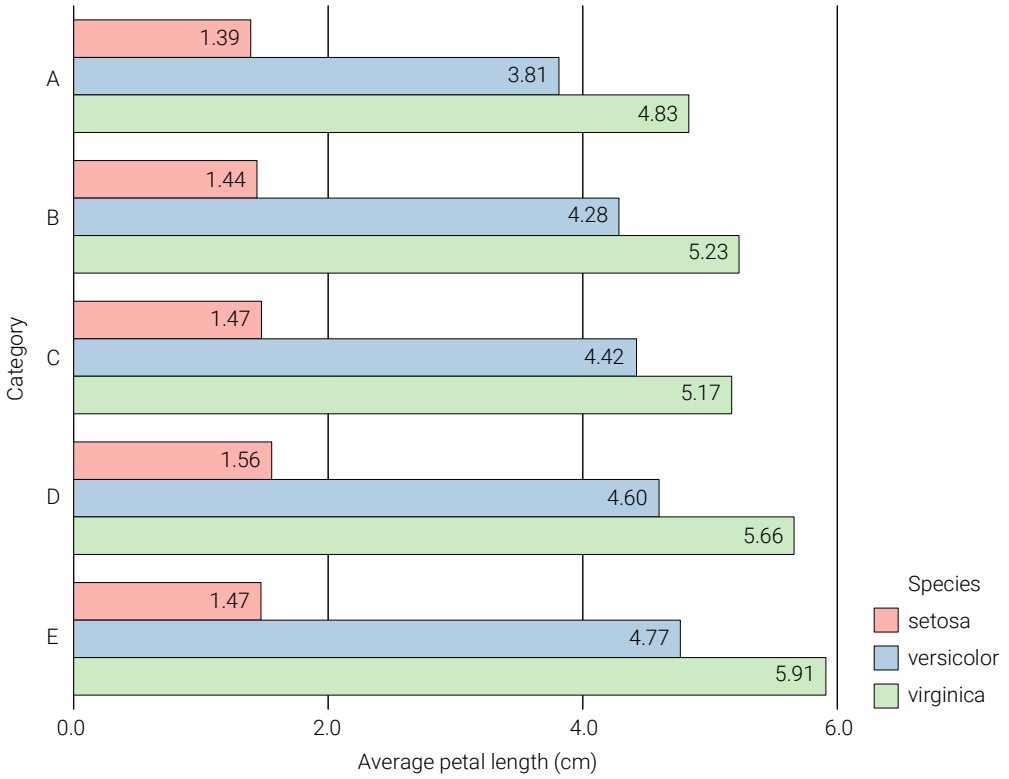

图 40. 可视化花萼长度均值，先 Category 分类再 species 分类

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

图 41. 计算花萼长度均值，先 Category 分类再 species 分类；使用时配合前文代码

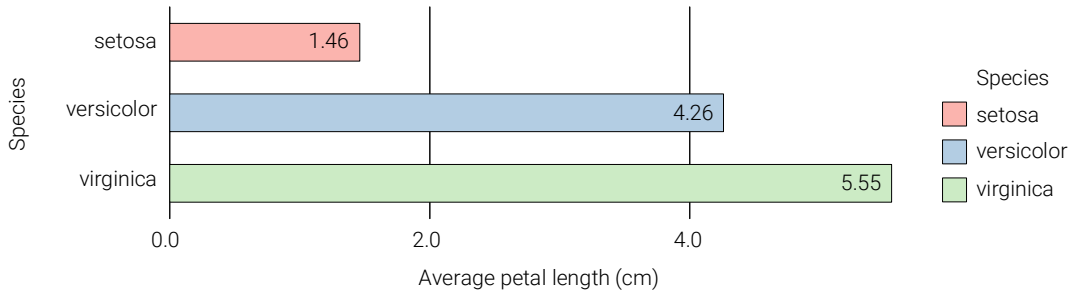

图 42. 可视化花萼长度均值，species 分类

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

# 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()
```

图 43. 计算并可视化花萼长度均值，specie 维度；使用时配合前文代码

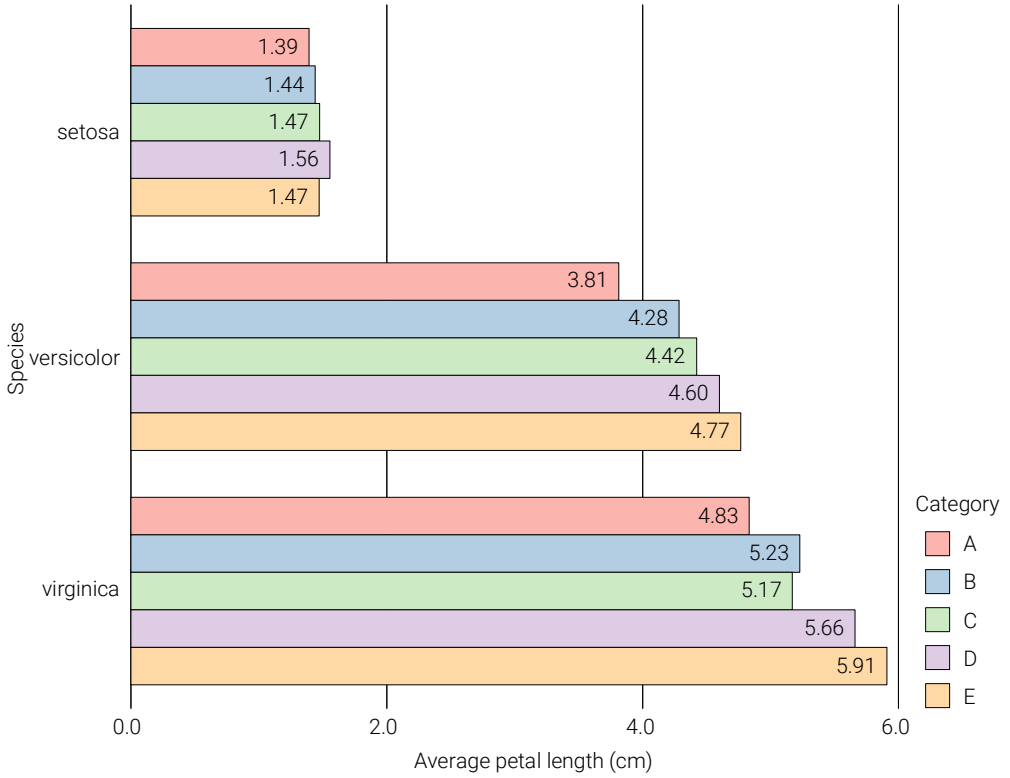

图 44. 可视化花萼长度均值，先 species 分类再 Category 分类

```python
# 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()
```

图 45. 计算并可视化花萼长度均值，specie 维度；使用时配合前文代码

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。