山东大学计算机科学与技术学院

大数据分析实践课程实验报告

学号：202300300198	姓名：朱宣玮	班级：数据 23
实验题目：Canis/Cast/Libra 实践		
实验学时：2	实验日期：2025.11.14	

实验目标：

从 Canis/Cast/Libra 这三个工具中，任选其一完成实践。

本实验选择 Libra 可视化库进行实践。

实验环境：

Windows 11

Edge 浏览器

VS Code + Live Server 插件

实验步骤与结果：

1. 导入需要的库

导入 D3.js 可视化库和 Libra 交互式可视化库，分别对应基础可视化和交互模块。

Libra 库使用 ES 模块格式，需要通过动态 import()加载，并导出到全局对象以便后续使用。

```
<script src="https://d3js.org/d3.v7.min.js"></script>
<!-- Libra 库使用 ES 模块格式，需要使用 type="module" -->
<script type="module">
    // 优先加载 libra.min.js（压缩版本），如果失败则加载 libra.js
    const libraPaths = [
        'https://cdn.jsdelivr.net/gh/Ideas-Laboratory/Libra@main/dist/libra.min.js',
        'https://cdn.jsdelivr.net/gh/Ideas-Laboratory/Libra@main/dist/libra.js'
    ];

    let currentIndex = 0;
    async function loadLibra() {
        if (currentIndex >= libraPaths.length) {
            console.error('Libra 库加载失败！');
            window.libraLoaded = false;
            window.dispatchEvent(new Event('libraLoadComplete'));
            return;
        }
```

2. 创建页面结构和样式

创建简单的页面布局，包含标题、可视化区域和交互说明面板。

```
<div class="container">
    <h1>散点图可视化</h1>
    <p class="subtitle">Libra 交互式可视化</p>

    <div id="visualization"></div>

    <div class="info-panel">
        <h3>交互说明</h3>
        <ul>
            <li><strong>悬停</strong>：显示数据点详细信息</li>
            <li><strong>点击</strong>：高亮同类别所有数据点</li>
        </ul>
    </div>
</div>
```

3. 生成示例数据

创建数据生成函数，生成 4 个类别，每个类别 30 个数据点的散点图数据。每个点包含坐标值、类别、标签和数值等信息。

```
// 生成示例数据
function generateSampleData() {
    const categories = ['A', 'B', 'C', 'D'];
    const data = [];

    categories.forEach((category, catIdx) => {
        for (let i = 0; i < 30; i++) {
            // 为每个类别生成不同分布的数据点
            const baseX = catIdx * 200 + 100;
            const baseY = 100 + catIdx * 100;

            data.push({
                x: baseX + (Math.random() - 0.5) * 150,
                y: baseY + (Math.random() - 0.5) * 150,
                category: category,
                label: `${category}-${i + 1}`,
                value: Math.floor(Math.random() * 100),
                description: `这是类别 ${category} 的第 ${i + 1} 个数据点，值为 ${Math.floor(Math.random() * 100)}`
            });
        }
    });

    return data;
}
```

4. 实现静态可视化

使用 D3.js 绘制散点图的基础元素，包括坐标轴、数据点和图例。

```
// 渲染静态可视化（坐标轴、标题等）
globalThis.renderStaticVisualization = function() {
    const svg = d3.select("#visualization")
        .append("svg")
        .attr("width", globalThis.WIDTH)
        .attr("height", globalThis.HEIGHT)
        .style("display", "block")
        .style("margin", "0 auto");

    // 创建主图形容器
    const g = svg.append("g")
        .attr("transform", `translate(${globalThis.MARGIN.left}, ${globalThis.MARGIN.top})`);

    // 添加 X 轴
    g.append("g")
        .attr("transform", `translate(0, ${globalThis.HEIGHT - globalThis.MARGIN.top - globalThis.MARGIN.bottom})`)
        .call(d3.axisBottom(globalThis.x))
        .append("text")
        .attr("x", (globalThis.WIDTH - globalThis.MARGIN.left - globalThis.MARGIN.right) / 2)
        .attr("y", 40)
        .attr("fill", "black")
        .style("text-anchor", "middle")
        .text("X 坐标");

    // 添加 Y 轴
    g.append("g")
        .call(d3.axisLeft(globalThis.y))
        .append("text")
```

5. 创建 Libra 图层并绘制数据点

使用 Libra 创建 D3 图层，在图层中绘制数据点。

数据点以灰色填充显示，带有类别颜色的边框，便于区分不同类别的数据。

```
function renderMainVisualization() {
    // 查找 SVG 元素
    const svg = d3.select("#visualization svg");

    // 创建主图层
    const mainLayer = Libra.Layer.initialize("D3Layer", {
        name: "mainLayer",
        width: globalThis.WIDTH - globalThis.MARGIN.left - globalThis.MARGIN.right,
        height: globalThis.HEIGHT - globalThis.MARGIN.top - globalThis.MARGIN.bottom,
        offset: { x: globalThis.MARGIN.left, y: globalThis.MARGIN.top },
        container: svg.node(),
    });

    const g = d3.select(mainLayer.getGraphic());

    // 绘制数据点
    g.selectAll("circle")
        .data(globalThis.data)
        .join("circle")
        .attr("class", "mark")
        .attr("cx", (d) => globalThis.x(d[globalThis.FIELD_X]))
        .attr("cy", (d) => globalThis.y(d[globalThis.FIELD_Y]))
        .attr("fill", "lightgray")
        .attr("fill-opacity", 0.7)
        .attr("stroke", (d) => globalThis.color(d[globalThis.FIELD_COLOR]))
        .attr("stroke-width", 1.5)
        .attr("r", 5);

    return mainLayer;
}
```

6. 实现点击高亮交互

使用 Libra 的 ClickInstrument、FilterService 和 SelectionTransformer 实现点击高亮功能。

点击某个数据点后，通过 FilterService 过滤出同类别数据，使用 SelectionTransformer 在选择层中高亮显示。

```
async function mountInteraction(layer) {
    // 点击高亮功能
    Libra.Interaction.build({
        inherit: "ClickInstrument",
        layers: [layer],
        remove: [
            {
                find: "SelectionTransformer",
            },
        ],
        insert: [
            {
                find: "SelectionService",
                flow: [
                    {
                        comp: "FilterService",
                        sharedVar: {
                            fields: [globalThis.FIELD_COLOR],
                        },
                    },
                    {
                        comp: "SelectionTransformer",
                        layer: layer.getLayerFromQueue("selectionLayer"),
                    },
                ],
            },
        ],
        sharedVar: {
            highlightColor: (d) => globalThis.color(d[globalThis.FIELD_COLOR]),
        },
    });
```

7. 实现悬停提示交互

使用 D3.js 直接实现 tooltip，显示数据点的详细信息。

Tooltip 跟随鼠标位置动态显示，包含标签、类别、坐标值和数值等信息。

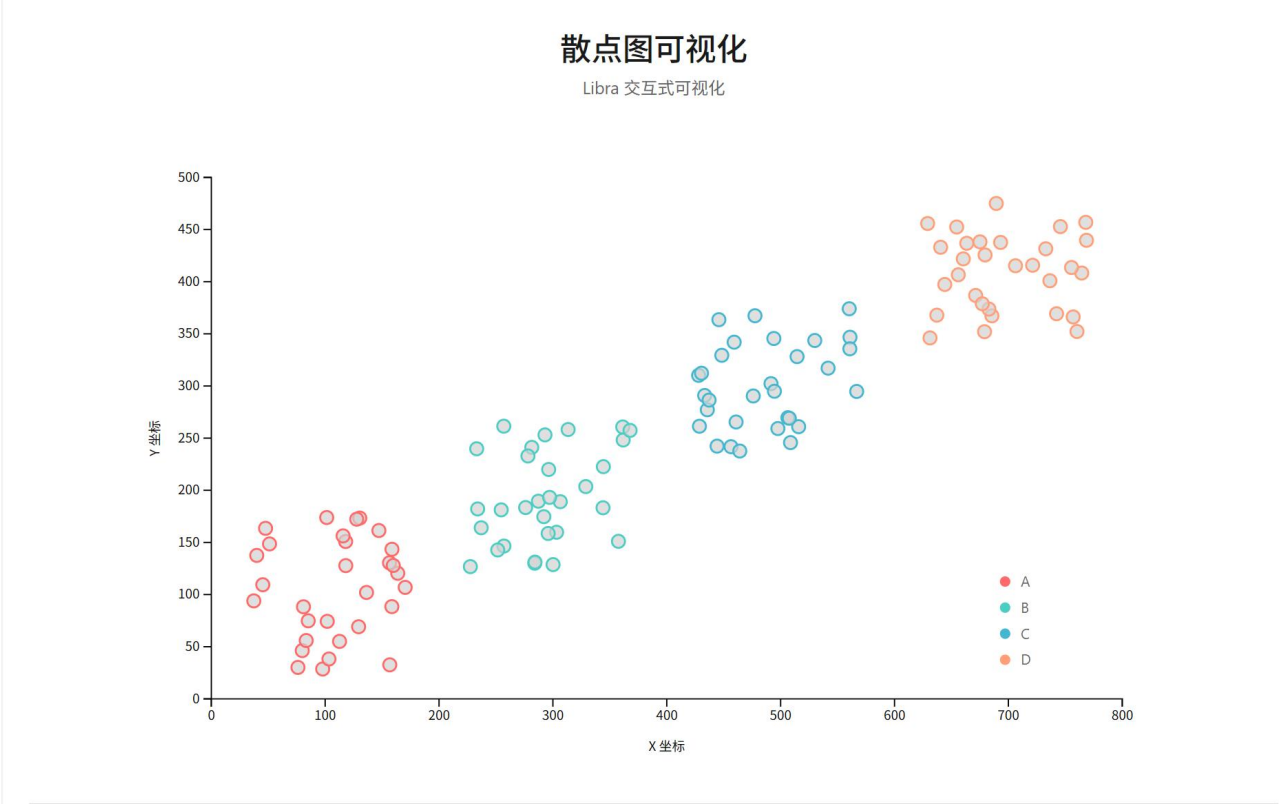
```
// 悬停展示信息功能 - 使用 D3 实现 tooltip
const tooltip = d3.select("body")
    .append("div")
    .attr("class", "tooltip")
    .style("opacity", 0)
    .style("position", "absolute")
    .style("background", "rgba(0, 0, 0, 0.9)")
    .style("color", "white")
    .style("padding", "10px")
    .style("border-radius", "5px")
    .style("pointer-events", "none")
    .style("font-size", "12px")
    .style("z-index", "1000")
    .style("box-shadow", "0 2px 10px rgba(0,0,0,0.3)");

// 使用 Libra 的 HoverInstrument，但自定义 tooltip 显示
const g = d3.select(layer.getGraphic());
g.selectAll("circle.mark")
    .on("mouseover", function(event, d) {
        tooltip.transition()
            .duration(200)
            .style("opacity", 0.9);
        tooltip.html(`
            <div style="line-height: 1.6;">
                <strong>标签:</strong> ${d[globalThis.FIELD_LABEL]}<br/>
                <strong>类别:</strong> ${d[globalThis.FIELD_COLOR]}<br/>
                <strong>X值:</strong> ${d[globalThis.FIELD_X].toFixed(2)}<br/>
                <strong>Y值:</strong> ${d[globalThis.FIELD_Y].toFixed(2)}<br/>
                <strong>数值:</strong> ${d.value}
            </div>
        `)
            .style("left", (event.pageX + 10) + "px")
            .style("top", (event.pageY - 10) + "px");
    })
```

实验结果

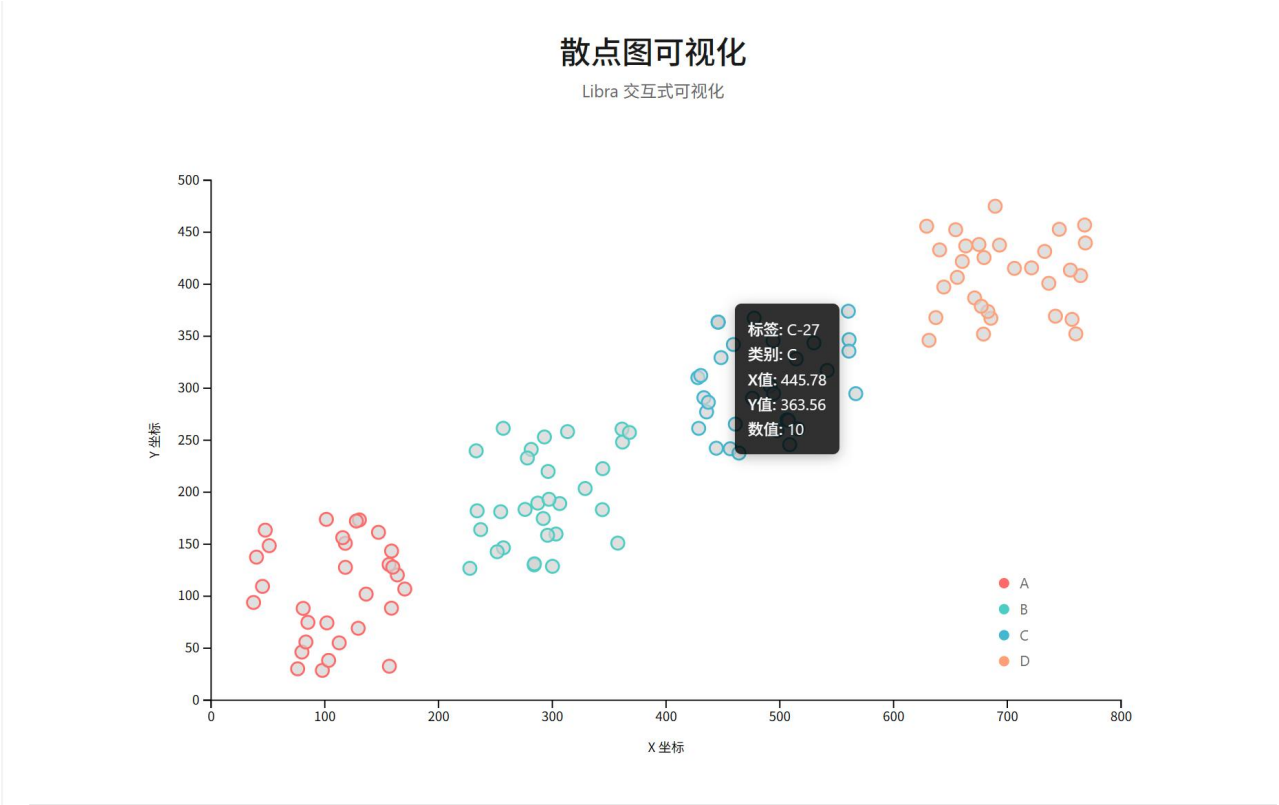
实验成功实现了基于 Libra 的交互式散点图可视化系统。

初始状态：所有数据点以灰色填充显示，带有类别颜色的边框。



悬停交互：鼠标悬停在数据点上时，显示详细信息提示框，包含标签、类别、X/Y 坐标值、数

值等信息。



点击交互：点击某个数据点后，该类别所有数据点被高亮显示。
由于截图功能限制，无法截图点击交互状态下的网页。

结论分析与体会：
结论分析
本次实验成功基于 Libra 和 D3.js 实现了交互式散点图可视化系统。系统支持鼠标悬停显示详细信息、点击高亮同类别数据点等交互功能，能够实时响应用户操作，页面设计简约美观，交互体验良好。
体会
通过本次实验，掌握了 Libra 库的基本使用方法，了解了它的设计理念和核心优势。
Libra 与 D3.js 无缝集成，便于在现有 D3 代码基础上添加交互功能。
Libra 通过图层、工具和交互效果的分离，使得交互逻辑可以独立于可视化代码进行开发和复用，这种模块化设计使得代码结构清晰，便于维护。
通过 inherit 和 insert 机制，可以轻松扩展现有交互功能，实现按类别过滤和高亮等功能。

参考资料：
Libra GitHub 仓库：https://github.com/Ideas-Laboratory/Libra
Libra 官方文档：https://ideas-laboratory.github.io/libra-js/
D3.js 官方文档：https://d3js.org/
jsDelivr CDN：https://www.jsdelivr.com/