山东大学 计算机科学与技术 学院

大数据分析与实践 课程实验报告

学号：202300130005	姓名：于佳杭	班级：23 数据
实验题目：电子表格实践 I		
实验学时：2	实验日期： 2025.10.15	

实验步骤与内容：

1、HTML 结构与资源引入

首先进行 HTML 结构布局与资源引入。页面主体采用 Flex 布局，划分为左右两大区域。左侧区域由#spreadsheet-container div 构成，内部包含一个用于控制图表显示的复选框（checkbox）以及 x-spreadsheet 电子表格的实例。右侧区域则是#my_dataviz div，它作为 SVG 容器，专门用于承载 D3.js 生成的动态图表。所需的核心库文件，即 x-spreadsheet 和 d3.js，均通过<link>和<script>标签从公共 CDN 地址引入。

```html
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>X-Spreadsheet with D3.js Visualization</title>

    <link rel="stylesheet" href="https://unpkg.com/x-data-spreadsheet@1.1.5/dist/xspreadsheet.css" />
    <script src="https://unpkg.com/x-data-spreadsheet@1.1.5/dist/xspreadsheet.js"></script>
    <script src="https://unpkg.com/x-data-spreadsheet@1.1.9/dist/locale/zh-cn.js"></script>
    <script src="https://d3js.org/d3.v6.js"></script>

    <style>
        body {
            display: flex;
            padding: 20px;
        }
        #spreadsheet-container {
            padding: 0;
            margin: 0;
        }
        #xspreadsheet {
            width: 500px;
            height: 400px;
            border: 1px solid rgba(0, 0, 0, 0.1);
        }
        #my_dataviz {
            width: 700px;
            height: 500px;
            padding: 0;
            margin-left: 20px;
        }
    </style>
</head>
```

2、表格初始化

页面加载后，首先对 x-spreadsheet 电子表格进行初始化。通过 x_spreadsheet.locale('zh-cn')将界面语言设置为中文。接着，使用 x_spreadsheet()函数在指定的 div 中创建表格实例，并配置其视图、行列数及默认样式。为了演示，使用 xs.cellText().reRender()方法向表格中填充初始数据，并遵循一个约定好的结构：A 列（从 A2 开始）作为分组名（如年份），第一行（从 B1 开始）作为系列名（如学科），而中间的交叉区域则为对应的数值。为了实现动态更新，通过 xs.on('cell-edited', update)为表格实例绑定了单元格编辑事件监听，确保用户的任何修改都能立刻触发 update()函数以重绘图表。

```
// 设置表格初始数据
xs.cellText(0, 1, "计算机").cellText(0, 2, "法学").reRender();
xs.cellText(1, 0, "2017").cellText(1, 1, "23").cellText(1, 2, "15").reRender();
xs.cellText(2, 0, "2018").cellText(2, 1, "36").cellText(2, 2, "26").reRender();
xs.cellText(3, 0, "2019").cellText(3, 1, "23").cellText(3, 2, "33").reRender();
xs.cellText(4, 0, "2020").cellText(4, 1, "22").cellText(4, 2, "10").reRender();

// 绑定事件：当单元格被编辑时，调用 update 函数
xs.on('cell-edited', update);

// 绑定事件：当复选框状态改变时，调用 update 函数
d3.selectAll(".checkbox").on("change", update);
```

3、颜色函数与读取表格数据

为了保证图表系列颜色的统一性和稳定性，定义了一个辅助颜色函数 getColor(idx)，它会根据传入的索引从一个预定义的调色板数组中返回一个固定的颜色值，从而确保每个数据系列在图表和图例中拥有一致的颜色。数据的读取与解析在 update 函数内部直接进行。当函数被触发时，它会遍历电子表格，通过循环读取 A 列（A2 及以下）来获取分组标题 yTitle，读取第一行（B1 及以后）来获取系列标题 xTitle。随后，通过双重循环读取交叉区域的数值，进行数据校验以跳过非数字内容，并最终将这些数值组织成一个对象数组 processedData，其中每个对象代表一个完整的分组数据，以便后续的 D3 绘图使用。

```
// 颜色生成函数
function getColor(idx) {
    const palette = [
        '#5ab1ef', '#ffb980', '#d87a80', '#2ec7c9', '#b6a2de',
        '#8d98b3', '#e5cf0d', '#97b552', '#95706d', '#dc69aa',
        '#07a2a4', '#9a7fd1', '#588dd5', '#f5994e', '#c05050',
        '#59678c', '#c9ab00', '#7eb00a', '#6f5553', '#c14089'
    ];
    return palette[idx % palette.length];
}
```

```
function update() {
    const checkbox = d3.select('.checkbox');

    // 检查复选框是否被选中
    if (checkbox.property("checked")) {
        let data = [];
        let yTitle = [];
        let xTitle = [];
        let rows = 0;
        let cols = 0;

        // 读取行标题 (Y轴分组)
        for (let i = 1; i < 20; i++) {
            const cell = xs.cell(i, 0);
            if (cell === null || cell.text === undefined || cell.text === "") {
                rows = i;
                break;
            }
            yTitle.push(cell.text);
        }

        // 读取列标题 (X轴子分组)
        for (let i = 1; i < 20; i++) {
            const cell = xs.cell(0, i);
            if (cell === null || cell.text === undefined || cell.text === "") {
                cols = i;
                break;
            }
            xTitle.push(cell.text);
        }
```

4、绘图逻辑（update 函数）

update()函数是整个应用的核心，它根据复选框的选中状态来决定是否绘制柱状图。函数首先通过 d3.select('.checkbox').property("checked")检查复选框的当前状态。如果复选框被选中，则执行完整的数据读取和图表绘制流程：它会根据读取的数据计算出数值的最大值，然后创建 X 轴（基于分组的离散比例尺 scaleBand）、Y 轴（基于数值的线性比例尺 scaleLinear）以及用于分组内部条柱定位的子比例尺 xSubgroup。接下来，它会为每个分组绘制一组矩形（rect），并在每个矩形上方添加数值标签，最后根据系列名和颜色在图表右侧绘制出清晰的图例。反之，如果复选框被取消选中，则函数会调用 d3.selectAll('svg').remove()来清空绘图区域，实现图表的移除。

```
function update() {
    const checkbox = d3.select('.checkbox');

    // 检查复选框是否被选中
    if (checkbox.property("checked")) {
        let data = [];
        let yTitle = [];
        let xTitle = [];
        let rows = 0;
        let cols = 0;

        // 读取行标题 (Y轴分组)
        for (let i = 1; i < 20; i++) {
            const cell = xs.cell(i, 0);
            if (cell === null || cell.text === undefined || cell.text === "") {
                rows = i;
                break;
            }
            yTitle.push(cell.text);
        }

        // 读取列标题 (X轴子分组)
        for (let i = 1; i < 20; i++) {
            const cell = xs.cell(0, i);
            if (cell === null || cell.text === undefined || cell.text === "") {
                cols = i;
                break;
            }
            xTitle.push(cell.text);
        }

        // 读取数据
        let max_val = 0;
        let processedData = [];
        for (let i = 1; i < rows; i++) {
            let rowData = { group: yTitle[i - 1] };
            for (let j = 1; j < cols; j++) {
                const cell = xs.cell(i, j);
                if (cell === null || cell.text === undefined || isNaN(+cell.text)) {
                    console.error(`Error data format at cell (${i}, ${j})`);
                    // d3.selectAll('svg').remove(); // 清空图表避免显示错误
                    // return; // 发现错误数据时提前退出
                    continue; // 或者跳过这个单元格
                }
                const value = +cell.text;
                rowData[xTitle[j - 1]] = value;
                if (value > max_val) {
                    max_val = value;
                }
            }
            processedData.push(rowData);
        }
```

5、事件绑定

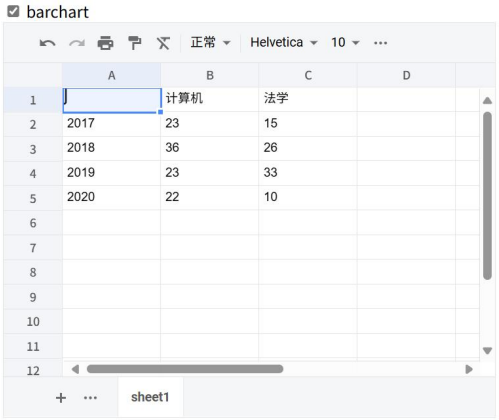
为了让交互生效，需要将 update 函数绑定到相应的用户操作事件上。通过 d3.selectAll(".checkbox").on("change", update)为控制图表显示的复选框绑定了 change 事件监听器，这样当用户勾选或取消勾选时，都会立即调用 update 函数。同时，在表格初始化阶段

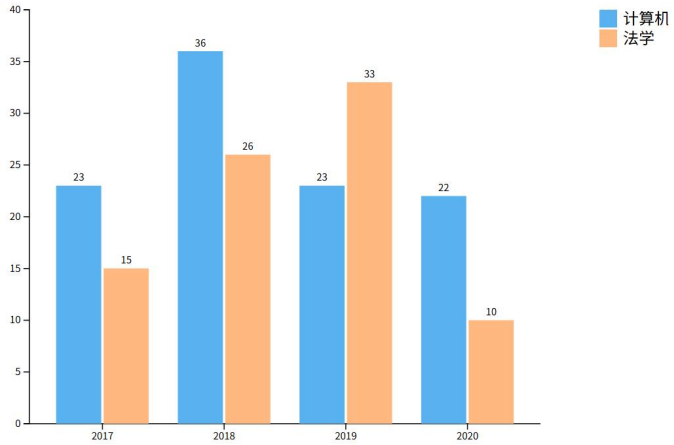
已经通过 xs.on('cell-edited', update)绑定了单元格编辑事件，这确保了在复选框为勾选状态的前提下，任何对表格数据的修改都能触发图表的实时更新。

```
// 绑定事件：当单元格被编辑时，调用 update 函数
xs.on('cell-edited', update);

// 绑定事件：当复选框状态改变时，调用 update 函数
d3.selectAll(".checkbox").on("change", update);
```

效果展示：





结论分析与体会：

数据可视化的本质是一个将原始数据转化为图形语言的过程，其核心在于建立清晰的数据到视觉的映射规则，而非绘图技术本身。这次实验也很好地演示了如何通过库的组合来各司其职：x-spreadsheet 充当数据源和编辑器，而 D3.js 则扮演了强大的渲染引擎。这种策略展现了前端生态的灵活性与强大功能。在开发过程中我也发现，应用的最终体验往往取决于细节，例如对非法数据输入的容错处理、确保交互事件的实时响应等，这些环节直接影响了应用的可用性和专业度，值得不断打磨。