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：202300130067	姓名：罗艺超	班级：数据班
实验题目：实验 3		
实验学时：2	实验日期：10.17	

实验目标：

1. 学习使用开源电子表格库 X-Spreadsheet 进行表格操作
2. 掌握 D3.js 数据可视化库的基本使用方法
3. 实现表格数据与可视化图表的实时联动功能
4. 开发基于表格数据变化自动更新图表的交互系统

实验步骤:

1. 环境搭建与库引入

首先创建 HTML 文件，引入必要的 CSS 和 JavaScript 库：

```html
<link rel="stylesheet" href="https://unpkg.com/x-data-spreadsheet@1.1.9/dist/xspreadsheet.css">
<script src="https://unpkg.com/x-data-spreadsheet@1.1.9/dist/xspreadsheet.js"></script>
<script src="https://d3js.org/d3.v7.min.js"></script>
```

2. 页面结构设计

设计包含表格容器、图表容器和提示框的页面布局：

```
html
<div id="xspreadsheet" style="width: 800px; height: 600px;"></div>
<div id="chart-container" style="width: 800px; height: 400px; margin-top: 20px;"></div>
<div class="tooltip"></div>
```

3. X-Spreadsheet 表格初始化

配置表格参数并初始化：

```
javascript
const spreadsheet = x_spreadsheet('#xspreadsheet', {
    mode: 'edit',
    showToolbar: true,
    showGrid: true,
    showContextmenu: true,
    row: { len: 100, height: 25 },
    col: { len: 26, width: 100, indexWidth: 60, minWidth: 60 }
});
```

4. 事件监听机制实现

设置单元格编辑事件监听器，实时捕获数据变化：

```
javascript
spreadsheet.on('cell-edited', (cell, ri, ci) => {
```

```
    console.log("单元格更新：", "行", ri+1, "列", String.fromCharCode(65+ci));
    if(ci === 0 || ci === 1) {
        setTimeout(updateChart, 10);
    }
});
```

5. 数据提取功能开发

实现从表格中提取 A 列和 B 列数据的函数：

```
javascript
function gettableData() {
    const newData = [];
    for(let ri = 0; ri < 20; ri++) {
        try {
            const nameCell = spreadsheet.cell(ri, 0);
            const valueCell = spreadsheet.cell(ri, 1);
            const name = nameCell?.text || '';
            const value = valueCell ? Number(valueCell.text) : null;

            if(name && value !== null && !isNaN(value)) {
                newData.push({name, value});
            }
        } catch(e) {
            console.log("处理行", ri, "时出错:", e);
        }
    }
    return newData;
}
```

6. D3.js 图表初始化

创建 SVG 画布并设置比例尺和坐标轴：

```
javascript
function initChart() {
    const width = 600 - margin.left - margin.right;
    chartHeight = 400 - margin.top - margin.bottom;
    const svg = d3.select("#chart-container")
        .append("svg")
        .attr("width", width + margin.left + margin.right)
        .attr("height", chartHeight + margin.top + margin.bottom)
        .append("g")
        .attr("transform", `translate(${margin.left}, ${margin.top})`);

    xScale = d3.scaleBand().range([0, width]).padding(0.2);
    yScale = d3.scaleLinear().range([chartHeight, 0]);

    xAxis = svg.append("g").attr("transform", `translate(0,${chartHeight})`);
    yAxis = svg.append("g");

    window.chartSvg = svg;
```

```
}
```

7. 图表更新功能实现

开发根据表格数据动态更新柱状图的函数：

```
javascript
function updateChart() {
    const newData = gettableData();
    if(newData.length === 0) return;

    const minValue = d3.min(newData, d => d.value);
    const maxValue = d3.max(newData, d => d.value);

    xScale.domain(newData.map(d => d.name));
    yScale.domain([Math.min(0, minValue), maxValue]);

    // 数据绑定与柱状图绘制
    const bars = window.chartSvg.selectAll("rect")
        .data(newData, d => d.name);

    // 进入、更新、退出模式实现
    bars.enter()
        .append("rect")
        .attr("fill", "steelblue")
        .merge(bars)
        .attr("x", d => xScale(d.name))
        .attr("width", xScale.bandwidth())
        .attr("y", d => yScale(d.value))
        .attr("height", d => Math.max(0, chartHeight - yScale(d.value)));

    bars.exit().remove();

    // 坐标轴更新
    xAxis.call(d3.axisBottom(xScale));
    yAxis.call(d3.axisLeft(yScale));
}
```

8. 交互功能增强

实现鼠标悬停提示效果：

```
javascript
bars.enter()
    .append("rect")
    .on("mouseover", function(event, d) {
        tooltip.html(`数值:${d.value}`)
            .style("opacity", 1)
            .style("left", (event.pageX + 10) + "px")
            .style("top", (event.pageY - 20) + "px");
        d3.select(this).attr("fill", "orange");
    })
```

```
    .on("mouseout", function() {
        tooltip.style("opacity", 0);
        d3.select(this).attr("fill", "steelblue");
    });
```

9. 初始数据设置

预设示例数据并初始化图表：

```
javascript
const data = [
    { name: "A", value: 40 }, { name: "B", value: 25 },
    { name: "C", value: 55 }, { name: "D", value: 30 },
    { name: "E", value: 60 }, { name: "F", value: 70 }
];

data.forEach((item, index) => {
    spreadsheet.cellText(index, 0, item.name);
    spreadsheet.cellText(index, 1, item.value.toString());
});

spreadsheet.reRender();
initChart();
updateChart();
```

结果分析:

1. 功能实现效果

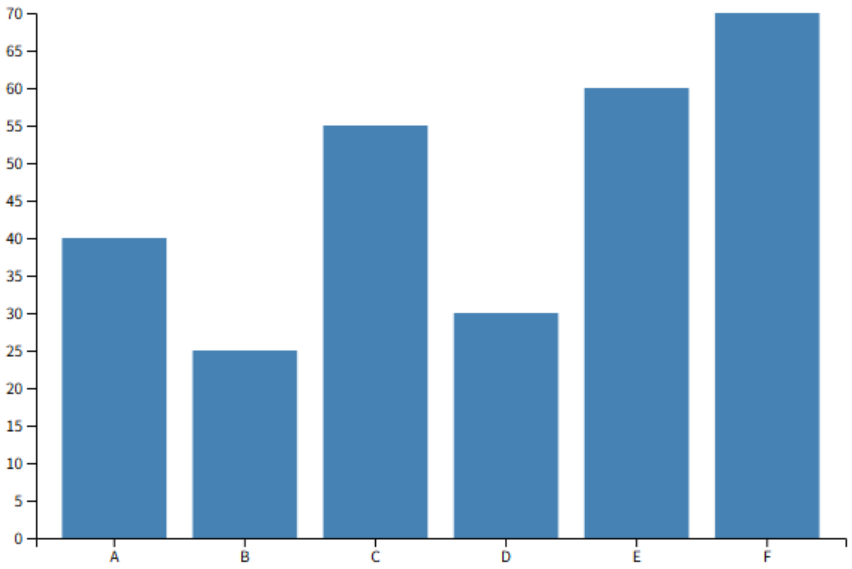
- 成功实现了基于 X-Spreadsheet 的在线电子表格
- 完成了 D3.js 柱状图可视化组件的开发
- 实现了表格数据与图表的实时联动
- 添加了鼠标悬停交互提示功能
- 支持动态数据更新和图表重绘

2. 技术难点与解决方案

- **数据同步问题**：通过 cell-edited 事件监听和 setTimeout 延迟确保数据完全更新后再刷新图表
- **比例尺动态调整**：使用 d3.min()和 d3.max()自动计算数据范围，确保图表自适应
- **数据绑定优化**：采用 D3.js 的 enter-update-exit 模式高效管理 DOM 元素

初始时的表格和可视化：

	A	B	C
1	A	40	
2	B	25	
3	C	55	
4	D	30	
5	E	60	
6	F	70	

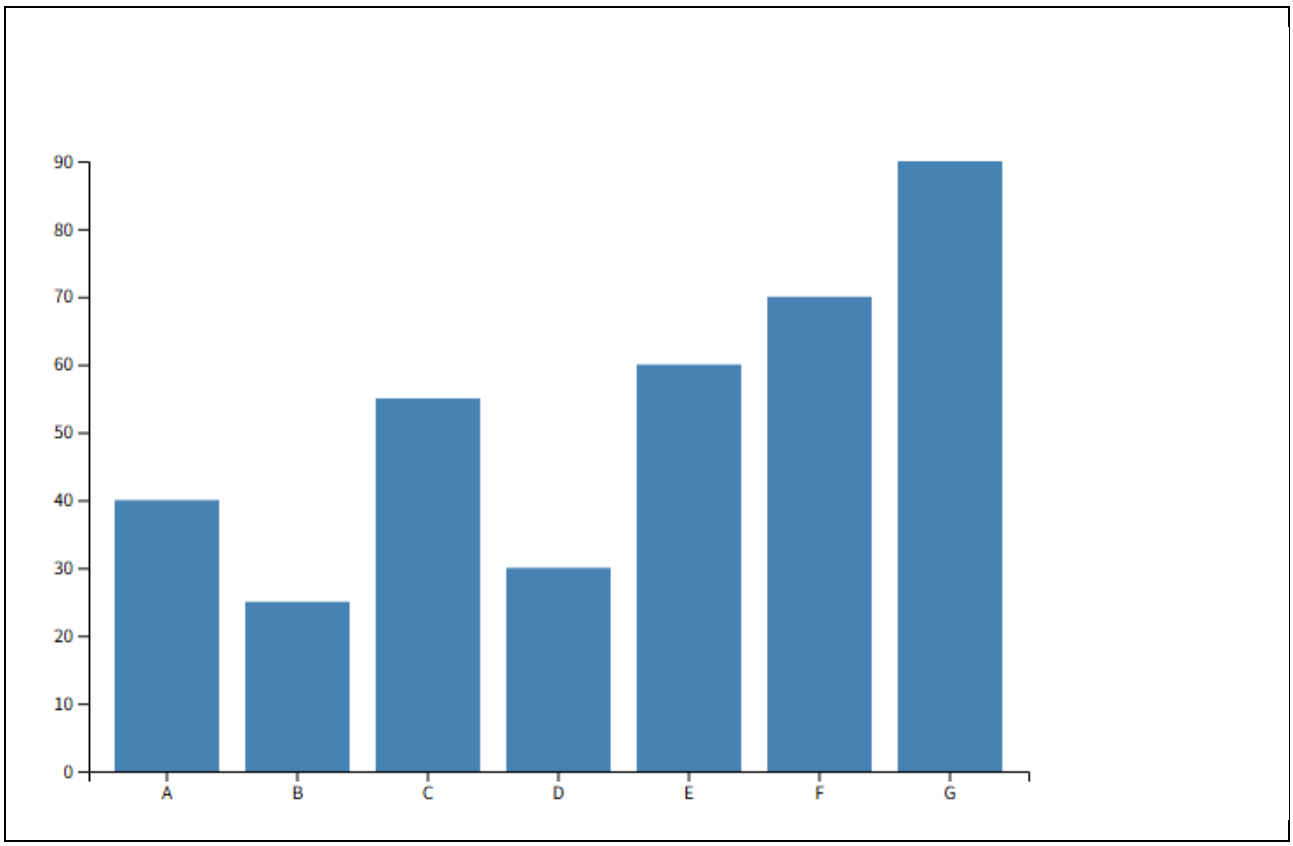


手动添加数据后，监听到数据变化

	A	B	C	D
1	A	40		
2	B	25		
3	C	55		
4	D	30		
5	E	60		
6	F	70		
7	G	90		
8				
9				
10				

单元格更新触发： 行 7 列 B 内容： undefined practice.html:81
已调用 updateChart 更新图表 practice.html:85
提取到的数据： ▸ Array(7) practice.html:114
单元格更新触发： 行 7 列 B 内容： undefined practice.html:81
已调用 updateChart 更新图表 practice.html:85
提取到的数据： ▸ Array(7) practice.html:114
选中单个单元格: practice.html:119
内容: undefined practice.html:120
位置:行 8 列 B practice.html:121

可视化结果同步更新


90
80
70
60
50
40
30
20
10
0
A
B
C
D
E
F
G