实验 3：电子表格实践 I

大数据分析实践实验报告

姓　　名: 郑坤武
学　　号: 202200150184
班　　级: 22 级公信班
学　　院: 政治学与公共管理学院

2025 年 10 月 15 日

1 实验目的

本实验基于开源电子表格库 **x-data-spreadsheet**（简称 x-spreadsheet）与 **D3.js**，实现“表格—可视化”的联动，构建一个在网页端可编辑、可视化同步更新的教学演示系统。实验重点包括：

- 掌握 **x-spreadsheet** 的基本用法（初始化、中文本地化、读写单元格、事件绑定）。
- 基于 **D3 v6** 实现三种可视化：分组柱状图、堆叠柱状图、折线图，并与表格数据双向联动（编辑即更新）。
- 通过 **复选框开关**控制多图共存渲染，探索 **响应式全屏布局**与图例/轴标签/数值标签等观感优化策略。
- 建立 **数据读取与校验**流程（空值与非数值诊断），保证图形渲染稳定性与教学可用性。

2 实验环境

- **操作系统：** macOS 12.7
- **开发工具：** VS Code
- **前端库：**
  - x-data-spreadsheet `@1.1.5`（核心）与 `@1.1.9` 中文本地化包
  - D3.js `v6`

3 具体实验步骤与结果分析

3.1 项目初始化与库引入

使用 CDN 方式快速集成，HTML `<head>` 中引入样式与脚本，确保顺序为：x-spreadsheet 样式 → x-spreadsheet 脚本 → 中文本地化 → D3。

3.2 页面布局与主题美化

采用“**顶部导航 + 左表右图**”的两列栅格，左侧为电子表格与开关控件，右侧为可视化容器，支持小屏断点为上下布局。通过 `view.width/height` 与容器 `clientWidth/Height` 动态绑定，实现 **自适应**与 **无滚动干扰**的观感。页面风格卡片化，增强教学演示可读性（图 1）。

3.3 电子表格初始化与示例数据

启用中文本地化，设置表格行列、字体与交互模式，并预置一组示例数据（第一行 = 系列，第一列 = 分组）。这便于课堂上“即开即演示”。

```
// 中文本地化 & 初始化
x_spreadsheet.locale('zh-cn');
const xs = x_spreadsheet('#xspreadsheet', {
mode:'edit', showToolbar:true, showGrid:true, showContextmenu:true,
view: {
```

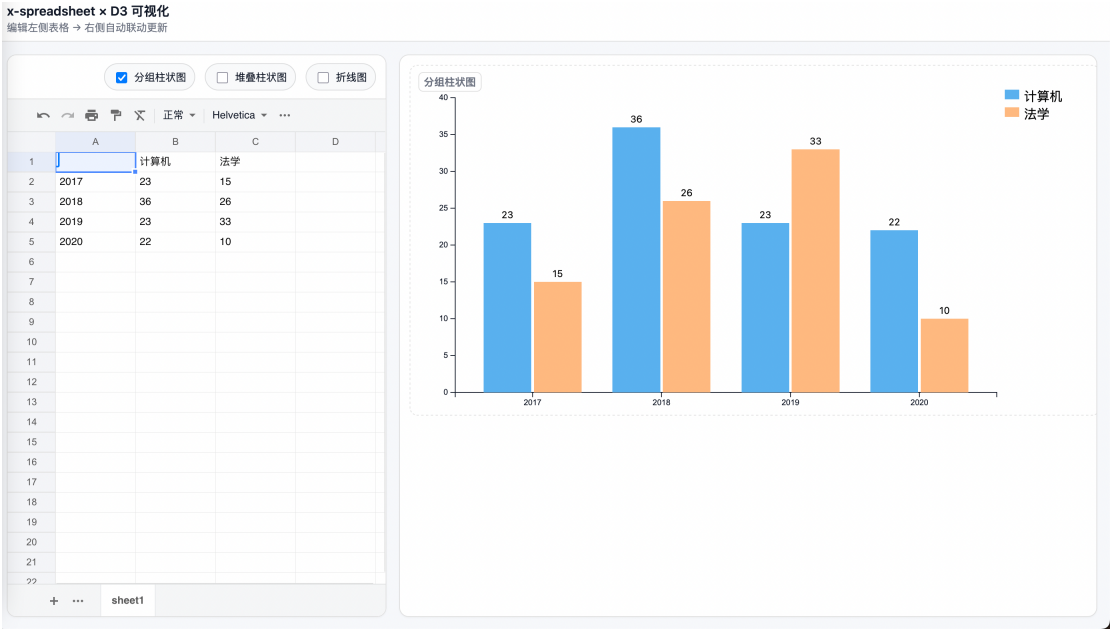

图 1: 页面整体布局与样式

```
height: () => document.getElementById('xspreadsheet').clientHeight,
width: () => document.getElementById('xspreadsheet').clientWidth,
},
row:{ len:40, height:25 }, col:{ len:16, width:100, indexWidth:60, minWidth:60 }
});

// 示例数据（系列：计算机/法学；分组：2017–2020）
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();
```

3.4 数据读取与校验（核心）

统一通过 `readTable()` 从表格读出 `xTitles`(系列)、`yTitles`(分组)与二维数值 `values[y][x]`。若出现空值或非数值，**立即抛错**并提示，保证渲染稳定性。

```
function readTable(){
const xTitles = [], yTitles = [], values = [];
// 第一列：分组
let r=1; for(; r<500; r++){
const c = xs.cell(r,0); const t = c && c.text!=null ? String(c.text).trim() : '';
if(!t) break; yTitles.push(t);
}
// 第一行：系列
```

```
let cidx=1; for(; cidx<500; cidx++){
const c0 = xs.cell(0,cidx); const t0 = c0 && c0.text!=null ? String(c0.text).trim() : '';
if(!t0) break; xTitles.push(t0);
}
// 数值
for(let i=1;i<r;i++){
const row=[]; for(let j=1;j<cidx;j++){
const cc = xs.cell(i,j); const txt = cc && cc.text!=null ? String(cc.text).trim() : '';
if(txt===''||isNaN(+txt)) throw new Error(第 ${i} 行 第 ${j} 列 不是数值);
row.push(+txt);
} values.push(row);
}
return { xTitles, yTitles, values };
}
```

3.5 三种可视化渲染与联动

复选框控制三种图形可同时渲染：**分组柱状图**、**堆叠柱状图**、**折线图**。每种图形封装独立渲染函数，接受统一数据结构与挂载容器，实现“多卡片”纵向排布（图 2）。

联动机制： 绑定 `xs.on('cell-edited', update)`、`checkbox.change` 与 `window.resize`。一旦数据或开关改变，`update()` 读取表格并按勾选顺序渲染对应卡片。

```
// 事件绑定
xs.on('cell-edited', update);
d3.selectAll('.checkbox').on('change', update);
window.addEventListener('resize', update);

// 联动更新
function update(){
const showGrouped = document.getElementById('chk-grouped').checked;
const showStacked = document.getElementById('chk-stacked').checked;
const showLine = document.getElementById('chk-line').checked;

const host = d3.select('#my_dataviz'); host.selectAll('*').remove();
if(!showGrouped && !showStacked && !showLine) return;

let table; try{ table = readTable(); } catch(e){ alert('数据格式错误: '+e.message); return; }

if(showGrouped){ const block = host.append('div').attr('class','viz-block').node();
d3.select(block).append('h4').attr('class','viz-title').text('分组柱状图');
renderGroupedBar(table, block);
}
if(showStacked){ const block = host.append('div').attr('class','viz-block').node();
```

```
22  d3.select(block).append('h4').attr('class','viz-title').text('堆叠柱状图');
23  renderStackedBar(table, block);
24  }
25  if(showLine){ const block = host.append('div').attr('class','viz-block').node();
26  d3.select(block).append('h4').attr('class','viz-title').text('折线图');
27  renderLine(table, block);
28  }
29  }
```

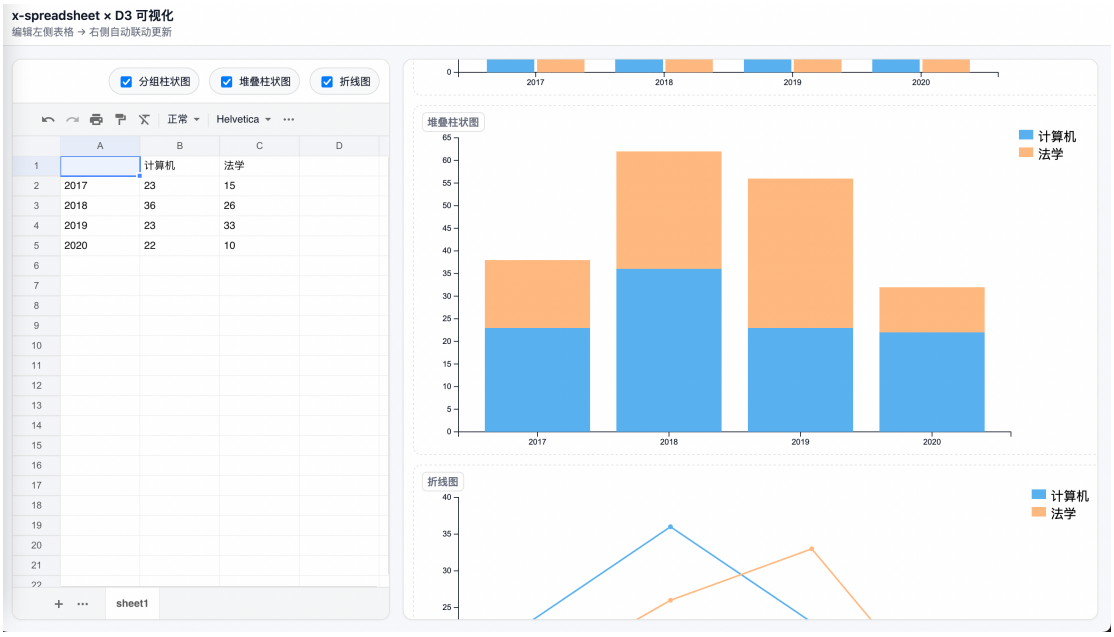

图 2: 三图联动示例：分组柱、堆叠柱、折线 (可滑动)

3.6 响应式与观感优化

为确保“一页展示”与小屏适配，采用以下策略：

- **容器驱动尺寸**：图表宽高由挂载容器 `clientWidth/Height` 计算，SVG 加 `viewBox` 提升缩放清晰度。
- **轴/标签自适应**：根据分组/系列数量调整字号与是否显示数值标签；当分组较多时自动旋转 X 轴标签。
- **图例换行**：图例在右侧竖向分布，避免压缩绘图区；小屏断点将整体布局切换为“上下结构”。

3.7 实验结果与分析

- **正确性：**编辑表格中任意数值，三种图形均可即时更新；当输入非数值时，系统能给出明确错误提示，避免渲染异常。
- **可用性：**复选框支持多图同时显示，适合教学对比；布局简洁，信息元素（轴、图例、数值标签）层级清晰。
- **扩展性：**渲染函数采用同一数据接口，后续可平滑增加如面积图、堆叠面积、雷达图等。

4 实验总结与收获

本实验实现了基于 x-spreadsheet 的网页端“可编辑表格—可视化联动”范例，涵盖了数据采集/清洗校验/可视化呈现的端到端流程。通过通用的 `readTable()` 数据出口与独立渲染函数，我们实现了良好的工程分层与复用性。响应式布局和观感优化使得该系统既适合课堂演示、也能用于课程作业验收与答辩展示。后续可在此基础上加入数据导入导出（CSV/JSON）、图表主题切换、交互提示（tooltip）与选择刷选（brush）等高级功能，形成完整的“前端数据可视化实验平台”。