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：202300130041	姓名： 徐守政	班级： 数据
实验题目：电子表格实践 I		
实验学时：2	实验日期： 2025/10/10	
实验目的：为 spreadsheet 添加一份可视化方案		
硬件环境： 计算机一台		
软件环境： Linux 或 Windows		

实验步骤与内容：

首先通过 CDN 引入实验所需的外部资源，包括 x-spreadsheet 表格库用于数据录入与编辑，以及 D3.js 用于图表绘制

在 HTML 文件中构建两大核心功能区域，一是表格与复选框区域，通过 容器承载表格，并在其中添加 “barchart”（柱状图）和 “linechart”（折线图）两个复选框，用于控制图表的显示与隐藏；二是独立图表容器区域，通过 包裹两个子容器 和 ，分别作为柱状图和折线图的专属渲染区域，避免图表重叠。

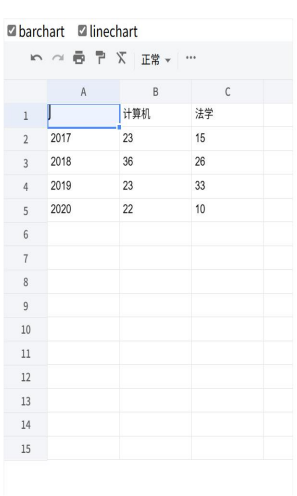
表格初始化与数据配置：先调用 x_spreadsheet.locale("zh-cn")将表格语言设置为中文，再通过 x_spreadsheet("#xspreadsheet", { ... })初始化表格，配置表格模式为可编辑（edit），显示工具栏、网格线与右键菜单，设置表格行列参数（15 行 8 列，列宽 100px、行高 25px）及默认单元格样式（字体、对齐方式、颜色等）。接着为表格设置初始数据，通过 xs.cellText(row, col, text).reRender()方法录入初始数据，同时绑定 cell-edited 事件，确保表格修改会触发图标更新

定义 getTableData()函数实现表格数据的统一读取与处理，该函数先遍历表格第 0 列获取年份列表（ytitle），遍历第 0 行获取专业列表（xtitle），再遍历表格数据区域读取数值，过滤非数字无效数据，最后计算数值最大值，并将处理后的原始数据、年份列表、专业列表、最大值封装为对象返回，为柱状图和折线图提供统一的数据来源。

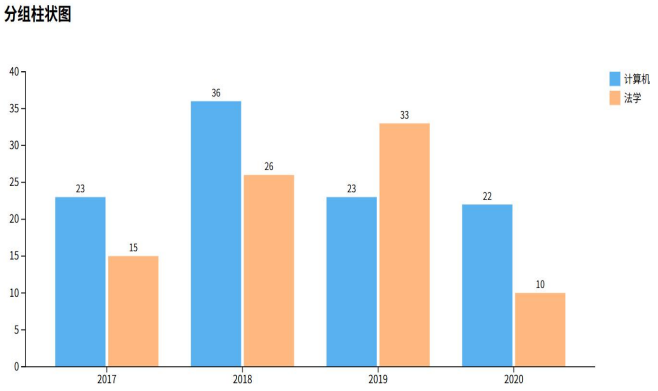
分别编写 drawBarChart(dataObj)和 drawLineChart(dataObj)函数实现柱状图与折线图的独立绘制。对于柱状图，先获取专属容器尺寸，计算图表绘图区域，清除容器内旧图表后创建 SVG；通过 D3.js 的 scaleBand 设置 X 轴（年份分组）与子分组（专业），scaleLinear 设置 Y 轴（数值），再通过 selectAll("g").data().join("g")绑定数据并绘制分组柱子，添加柱子数值标签，最后绘制坐标轴与图例。对于折线图，流程与柱状图类似，但使用 d3.line()创建平滑折线生成器（curveMonotoneX），绘制折线后额外添加数据点及数值标签，并为数据点添加 line-point 类实现 hover 交互效果，同样绘制坐标轴与图例。

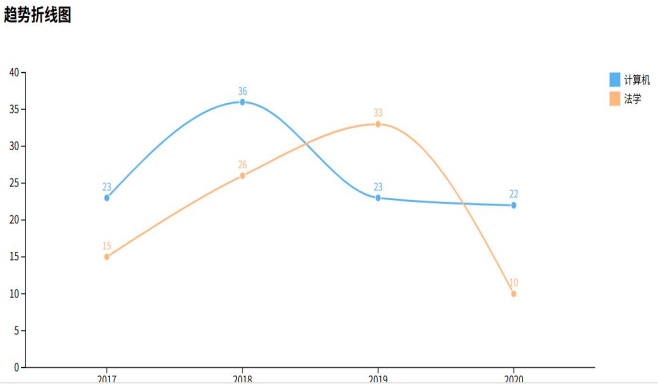
实验效果图展示：

原始图像

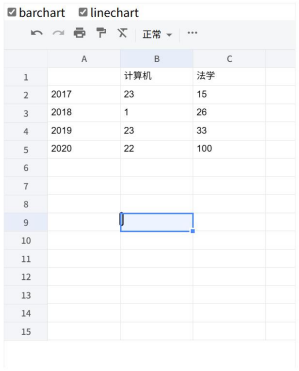


	A	B	C
1		计算机	法学
2	2017	23	15
3	2018	36	26
4	2019	23	33
5	2020	22	10

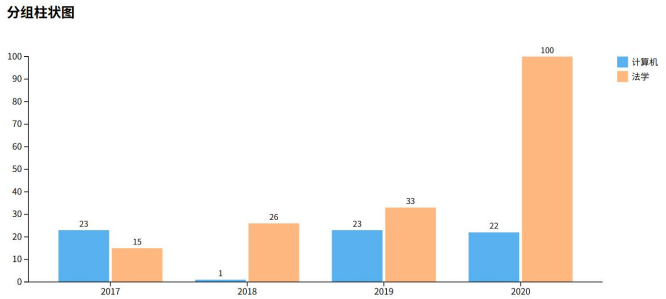


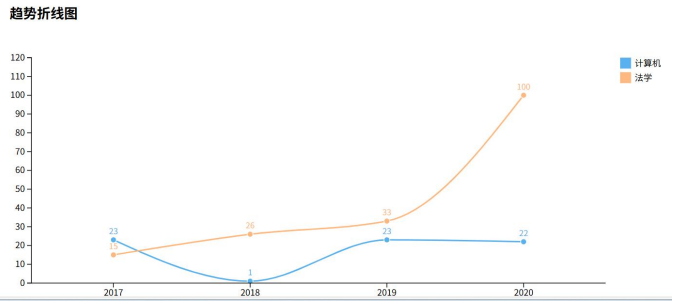


双击表格单元格改变数据



	A	B	C
1		计算机	法学
2	2017	23	15
3	2018	1	26
4	2019	23	33
5	2020	22	100





代码展示：

```
<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>

<!-- 表格与复选框区域 -->
<div id="xspreadsheet">
    <input type="checkbox" class="checkbox" value="barchart" /><label>barchart</label>
    <input type="checkbox" class="checkbox" value="linechart" style="margin-left:15px;" /><label>linechart</label>
</div>

<!-- 独立图表容器：柱状图和折线图各占一个容器，避免重叠 -->
<div id="chart-container">
    <div id="bar-chart-container" class="chart-box"> <!-- 柱状图专属容器 -->
        <h3 style="margin-left:50px;">分组柱状图</h3>
    </div>
    <div id="line-chart-container" class="chart-box"> <!-- 折线图专属容器 -->
        <h3 style="margin-left:50px;">趋势折线图</h3>
    </div>
</div>

<style>
/* 表格样式（保留原配置） */
#xspreadsheet {
    width: 400px;
    height: 500px;
    padding: 0px;
    margin: 20px;
    border: 1px solid #eee;
    float: left; /* 表格靠左，图表靠右 */
}

/* 图表容器整体样式 */
#chart-container {
    width: 1000px;
    height: 1000px;
    padding: 0px;
    margin: 20px 20px 20px 450px; /* 避开左侧表格 */
}

/* 单个图表容器：固定高度，上下排列不重叠 */
.chart-box {
    width: 100%;
    height: 480px; /* 每个图表占480px，预留间距 */
    margin-bottom: 40px; /* 图表之间间距40px */
```

```
}

/* 折线图数据点交互样式 */
.line-point {
    cursor: pointer;
}
.line-point:hover {
    stroke: #000;
    stroke-width: 2;
}

/* 原 ticktext 样式保留 */
.ticktext {
    font-size: 20;
    stroke: black;
    stroke-width: 0.05em;
}
</style>

<script>
// 表格初始化（完全保留原逻辑）
x_spreadsheet.locale("zh-cn");
var xs = x_spreadsheet("#xspreadsheet", {
    mode: 'edit', // edit | read
    showToolbar: true,
    showGrid: true,
    showContextmenu: true,
    view: {
        height: () => document.documentElement.clientHeight - 50,
        width: () => 400,
    },
    row: {
        len: 15,
        height: 25,
    },
    col: {
        len: 8,
        width: 100,
        indexWidth: 60,
        minWidth: 60,
    },
    style: {
        bgcolor: '#ffffff',
        align: 'left',
        valign: 'middle',
        textwrap: false,
```

```
        strike: false,
        underline: false,
        color: '#0a0a0a',
        font: {
            name: 'Helvetica',
            size: 10,
            bold: false,
            italic: false,
        },
    },
})

// 表格初始数据（保留原配置）
xs.on('cell-edited', update)
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();

// 颜色函数（保留原配置）
function getColor(idx) {
    var 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];
}

// 通用数据读取函数（两种图表共用，避免重复）
function getTableData() {
    var data = [];
```

```
var ytitle = []; // 行标题：年份（X 轴共用）
var xtitle = []; // 列标题：专业（系列名共用）
var col = 0;
var rows = 0;

// 读取年份（行标题）
for (var i = 1; i < 20; i++) {
    if (xs.cell(i, 0) === null || xs.cell(i, 0).text === undefined || xs.cell(i, 0).text === "") {
        rows = i;
        break;
    }
    data.push([]);
    ytitle.push(xs.cell(i, 0).text);
}

// 读取专业（列标题）
for (var i = 1; i < 20; i++) {
    if (xs.cell(0, i) === null || xs.cell(0, i).text === undefined || xs.cell(0, i).text === "") {
        col = i;
        break;
    }
    xtitle.push(xs.cell(0, i).text);
}

// 读取数值（过滤非数字）
for (var i = 1; i < rows; i++) {
    for (var j = 1; j < col; j++) {
        if (xs.cell(i, j) === null || xs.cell(i, j).text === undefined || isNaN(+xs.cell(i, j).text)) {
            console.log("无效数据：单元格(" + i + "," + j + ")");
            return null;
        }
        data[i - 1][j - 1] = +xs.cell(i, j).text;
    }
}

// 计算最大值（两种图表 Y 轴共用）
var max = 0;
data.forEach(row => {
    var rowMax = Math.max(...row);
    if (rowMax > max) max = rowMax;
});

return {
    rawData: data,    // 原始二维数据
    yearList: ytitle, // 年份列表（X 轴）
    majorList: xtitle,// 专业列表（系列名）
```

```
        maxValue: max     // 数值最大值（Y 轴定义域）
    };
}

// 1. 独立柱状图绘制函数（绑定到专属容器）
function drawBarChart(dataObj) {
    const { rawData, yearList, majorList, maxValue } = dataObj;
    const margin = { top: 30, right: 30, bottom: 60, left: 80 };
    const containerWidth = document.getElementById("bar-chart-container").offsetWidth;
    const width = containerWidth - margin.left - margin.right - 100; // 适配容器宽度
    const height = 400 - margin.top - margin.bottom; // 适配容器高度

    // 清除柱状图容器内旧图表
    d3.select("#bar-chart-container svg").remove();

    // 创建柱状图专属 SVG
    const svg = d3.select("#bar-chart-container")
        .append("svg")
        .attr("width", containerWidth)
        .attr("height", 400)
        .append("g")
        .attr("transform", `translate(${margin.left},${margin.top})`);

    // 柱状图 X 轴（年份分组）
    const x = d3.scaleBand()
        .domain(yearList)
        .range([0, width])
        .padding(0.2);

    // 柱状图 Y 轴（人数）
    const y = d3.scaleLinear()
        .domain([0, maxValue])
        .range([height, 0])
        .nice();

    // 子分组（专业）
    const xSubgroup = d3.scaleBand()
        .domain(majorList)
        .range([0, x.bandwidth()])
        .padding(0.05);

    // 绘制柱子
    svg.append("g")
        .selectAll("g")
        .data(yearList.map((year, i) => ({ year, values: rawData[i] })))
        .join("g")
```

```
    .attr("transform", d => `translate(${x(d.year)}, 0)`)
    .selectAll("rect")
    .data((d, i) => majorList.map((major, j) => ({ major, value: d.values[j], idx: j })))
    .join("rect")
    .attr("x", d => xSubgroup(d.major))
    .attr("y", d => y(d.value))
    .attr("width", xSubgroup.bandwidth())
    .attr("height", d => height - y(d.value))
    .attr("fill", d => getColor(d.idx));

// 柱子数值标签
svg.append("g")
    .selectAll("g")
    .data(yearList.map((year, i) => ({ year, values: rawData[i] })))
    .join("g")
    .attr("transform", d => `translate(${x(d.year)}, 0)`)
    .selectAll("text")
    .data((d, i) => majorList.map((major, j) => ({ major, value: d.values[j] })))
    .join("text")
    .attr("x", d => xSubgroup(d.major) + xSubgroup.bandwidth() / 2)
    .attr("y", d => y(d.value) - 5)
    .text(d => d.value)
    .attr("text-anchor", "middle")
    .attr("font-size", "11px");

// 柱状图X轴刻度
svg.append("g")
    .attr("transform", `translate(0, ${height})`)
    .call(d3.axisBottom(x).tickSizeOuter(0))
    .selectAll("text")
    .attr("font-size", "12px");

// 柱状图Y轴刻度
svg.append("g")
    .call(d3.axisLeft(y))
    .selectAll("text")
    .attr("font-size", "12px");

// 柱状图图例
const barLegend = svg.selectAll(".bar-legend")
    .data(majorList)
    .enter()
    .append("g")
    .attr("class", "bar-legend")
    .attr("transform", (d, i) => `translate(${width + 20}, ${i * 20})`);
```

```
    barLegend.append("rect")
        .attr("width", 15)
        .attr("height", 15)
        .attr("fill", (d, i) => getColor(i));

    barLegend.append("text")
        .attr("x", 20)
        .attr("y", 12)
        .text(d => d)
        .attr("font-size", "12px");
}

// 2. 独立折线图绘制函数（绑定到专属容器）
function drawLineChart(dataObj) {
    const { rawData, yearList, majorList, maxValue } = dataObj;
    const margin = { top: 30, right: 30, bottom: 60, left: 80 };
    const containerWidth = document.getElementById("line-chart-container").offsetWidth;
    const width = containerWidth - margin.left - margin.right - 100; // 适配容器宽度
    const height = 400 - margin.top - margin.bottom; // 适配容器高度

    // 清除折线图容器内旧图表
    d3.select("#line-chart-container svg").remove();

    // 创建折线图专属 SVG
    const svg = d3.select("#line-chart-container")
        .append("svg")
        .attr("width", containerWidth)
        .attr("height", 400)
        .append("g")
        .attr("transform", `translate(${margin.left},${margin.top})`);

    // 折线图 X 轴（年份）
    const x = d3.scaleBand()
        .domain(yearList)
        .range([0, width])
        .padding(0.2);

    // 折线图 Y 轴（人数，留 10% 余量避免顶边）
    const y = d3.scaleLinear()
        .domain([0, maxValue * 1.1])
        .range([height, 0])
        .nice();

    // 折线生成器（平滑曲线）
    const lineGenerator = d3.line()
        .x((d, i) => x(yearList[i]) + x.bandwidth() / 2) // 年份居中
```

```
        .y(d => y(d))
        .curve(d3.curveMonotoneX); // 平滑折线

    // 为每个专业绘制折线
    majorList.forEach((major, idx) => {
        // 提取当前专业的所有年份数据
        const seriesData = rawData.map(row => row[idx]);

        // 绘制折线
        svg.append("path")
            .datum(seriesData)
            .attr("fill", "none")
            .attr("stroke", getColor(idx))
            .attr("stroke-width", 2)
            .attr("d", lineGenerator);

        // 绘制数据点
        svg.append("g")
            .selectAll("circle")
            .data(seriesData)
            .join("circle")
            .attr("class", "line-point")
            .attr("cx", (d, i) => x(yearList[i]) + x.bandwidth() / 2)
            .attr("cy", d => y(d))
            .attr("r", 4)
            .attr("fill", getColor(idx))
            .attr("stroke", "#fff")
            .attr("stroke-width", 1);

        // 数据点数值标签
        svg.append("g")
            .selectAll("text")
            .data(seriesData)
            .join("text")
            .attr("x", (d, i) => x(yearList[i]) + x.bandwidth() / 2)
            .attr("y", d => y(d) - 8)
            .text(d => d)
            .attr("text-anchor", "middle")
            .attr("font-size", "11px")
            .attr("fill", getColor(idx));
    });

    // 折线图X轴刻度
    svg.append("g")
        .attr("transform", `translate(0, ${height})`)
        .call(d3.axisBottom(x).tickSizeOuter(0))
```

```
        .selectAll("text")
        .attr("font-size", "12px");

    // 折线图Y轴刻度
    svg.append("g")
        .call(d3.axisLeft(y))
        .selectAll("text")
        .attr("font-size", "12px");

    // 折线图图例
    const lineLegend = svg.selectAll(".line-legend")
        .data(majorList)
        .enter()
        .append("g")
        .attr("class", "line-legend")
        .attr("transform", (d, i) => `translate(${width + 20}, ${i * 20})`);

    lineLegend.append("rect")
        .attr("width", 15)
        .attr("height", 15)
        .attr("fill", (d, i) => getColor(i));

    lineLegend.append("text")
        .attr("x", 20)
        .attr("y", 12)
        .text(d => d)
        .attr("font-size", "12px");
}

// 核心更新函数：控制两个图表独立显示/隐藏
function update() {
    // 1. 获取两个复选框状态
    const showBar = d3.select('.checkbox[value="barchart"]').property("checked");
    const showLine = d3.select('.checkbox[value="linechart"]').property("checked");

    // 2. 读取表格数据（共用逻辑）
    const dataObj = getTableData();
    if (!dataObj) return; // 数据无效时终止

    // 3. 分别控制两个图表的显示/隐藏
    if (showBar) {
        drawBarChart(dataObj); // 绘制柱状图（专属容器）
    } else {
        d3.select("#bar-chart-container svg").remove(); // 隐藏柱状图
    }
```

```
    if (showLine) {
        drawLineChart(dataObj); // 绘制折线图（专属容器）
    } else {
        d3.select("#line-chart-container svg").remove(); // 隐藏折线图
    }

    // 4. 保留原本地存储功能
    window.localStorage.data = dataObj.rawData;
    window.localStorage.xTitle = dataObj.majorList;
    window.localStorage.yTitle = dataObj.yearList;
}

// 绑定事件：两个复选框都触发更新
d3.selectAll(".checkbox").on("change", update);
</script>
```