山东大学计算机科学与技术学院

可视化技术课程实验报告

学号：202300130220	姓名：刘傲宇	班级：数据科学与大数据技术班
实验题目：电子表格实践 I		
实验学时：2	实验日期：2025/10/24	

实验目标：

Add a new vis function based on the open source spreadsheet codes:https://github.com/myliang/x-spreadsheet

实验步骤与内容：

1、基础 HTML 框架

引入 x-spreadsheet 的 CSS 样式文件

引入 x-spreadsheet 核心 JavaScript 库

引入中文语言包

引入 D3.js v6 版本用于数据可视化

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>电子表格实践 I - x-spreadsheet 与 D3 可视化</title>
    <!-- 导入需要的官方库 -->
    <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>
</head>
```

2、用户界面布局

创建复选框控制面板，支持多种图表类型选择

创建可视化容器#my_dataviz 用于显示图表

```
    <!-- 加入多个 check box 可以选择不同的可视化类型 -->
    <div id="xspreadsheet">
        <div class="chart-controls">
            <h3>可视化选项：</h3>
            <label><input type="checkbox" class="checkbox" value="barchart" />
柱状图</label>
            <label><input type="checkbox" class="checkbox" value="linechart" />
```

```
折线图</label>
            <label><input type="checkbox" class="checkbox" value="piechart" />
饼图</label>
            <label><input type="checkbox" class="checkbox" value="scatterchart"
/>散点图</label>
        </div>
    </div>
    <div id="my_dataviz"></div>
```

3、样式配置
定义表格和图表容器的尺寸
美化复选框控制面板的样式
设置坐标轴文本样式

```
    <style>
        #xspreadsheet {
            width: 400px;
            height: 500px;
            padding: 0px;
            /* border: 1px solid rgba(0, 0, 0, 0.815); */
            margin: 0px;
        }
        #my_dataviz {
            width: 1000px;
            height: 900px;
            padding: 0px;
            /* border: 1px solid rgba(0, 0, 0, 0.815); */
            margin: 0px;
        }
        .ticktext {
            font-size: 20;
            stroke: black;
            stroke-width: 0.05em;
        }
        .chart-controls {
            margin-bottom: 20px;
            padding: 10px;
            border: 1px solid #ddd;
            border-radius: 5px;
            background-color: #f9f9f9;
        }
        .chart-controls h3 {
            margin: 0 0 10px 0;
            font-size: 16px;
            color: #333;
        }
        .chart-controls label {
            display: block;
            margin: 5px 0;
            font-size: 14px;
            cursor: pointer;
        }
        .chart-controls input[type="checkbox"] {
            margin-right: 8px;
        }
    </style>
```

4、表格初始化与配置
设置表格为中文界面
配置表格为编辑模式，显示工具栏和网格
设置行列数量和尺寸
配置表格样式（背景色、字体、对齐方式等）

```
        // 设置中文语言环境
        x_spreadsheet.locale("zh-cn");

        // 初始化表格
        var xs = x_spreadsheet("#xspreadsheet", {
            mode: 'edit', // edit | read
            showToolbar: true,
            showGrid: true,
            showContextmenu: true,
            view: {
                height: () => document.documentElement.clientHeight,
                width: () => document.documentElement.clientWidth,
            },
            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,
                },
            },
        });
```

5、初始数据设置与事件绑定
绑定单元格编辑事件到 update 函数
设置预设数据：2017-2020 年计算机和法学专业的人数
使用链式调用设置多个单元格内容

```
        // 设置初值
        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();
```

6、颜色调色板函数
定义 20 种颜色的调色板
根据索引返回对应颜色，支持循环使用

```
        // 颜色调色板函数
        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];
        }
```

7、主更新函数 - 数据收集与处理
检查复选框状态，判断是否需要绘制图表
从表格中读取行标题（年份）和列标题（专业）
读取数值数据并进行格式验证
将数据存储到 localStorage 中

```
        function update() {
            // 获取所有复选框状态
            const barCheckbox = d3.select('input[value="barchart"]');
            const lineCheckbox = d3.select('input[value="linechart"]');
            const pieCheckbox = d3.select('input[value="piechart"]');
            const scatterCheckbox = d3.select('input[value="scatterchart"]');

            const hasAnyChart = barCheckbox.property("checked.equals") ||
                                lineCheckbox.property("checked") ||
                                pieCheckbox.property("checked") ||
                                scatterCheckbox.property("checked");

            if (hasAnyChart) {
                var data = [];
                var ytitle = [];
                var xtitle = [];
                var col = 0;

                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);
                }

                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);
                }

                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(i, j, xs.cell(i, j));
                            // alert("error data format");
                            return;
                        } else {
                            data[i - 1][j - 1] = +xs.cell(i, j).text;
                        }
                    }
                }

                window.localStorage.data = data;
                window.localStorage.xTitle = xtitle;
                window.localStorage.yTitle = ytitle;
                console.log(window.localStorage.data);
```

8、数据格式转换与图表绘制调度
从 localStorage 中恢复数据并重新格式化
计算数据最大值用于 Y 轴范围设定
清除现有图表
根据复选框状态调用相应的图表绘制函数

```
                var xTitle = Array.from(window.localStorage.xTitle.split(","));
                var yTitle = Array.from(window.localStorage.yTitle.split(","));
                var list_data = window.localStorage.data.split(",");
                var pos = 0;

                var data1 = [];
                for (var i = 0; i < yTitle.length; i++) {
                    let tmp = [];
                    for (var j = 0; j < xTitle.length; ++j) {
                        tmp.push(+list_data[pos++]);
                    }
                    data1.push(tmp);
                }

                var max = 0;
                var data = [];
                for (var i = 0; i < yTitle.length; i++) {
                    var jsd = {};
                    jsd["group"] = yTitle[i];
                    data.push(jsd);
                }

                for (var i = 0; i < yTitle.length; i++) {
                    for (var j = 0; j < xTitle.length; j++) {
                        if (data1[i][j] > max)
                            max = data1[i][j];
                        data[i][xTitle[j]] = data1[i][j];
                    }
                }

                console.log(data);
                console.log(max);

                // 清除现有图表
                d3.selectAll('svg').remove();

                // 创建主容器
                const container = d3.select("#my_dataviz");

                // 绘制柱状图
                if (barCheckbox.property("checked")) {
                    drawBarChart(container, data, xTitle, yTitle, max);
                }

                // 绘制折线图
                if (lineCheckbox.property("checked")) {
                    drawLineChart(container, data, xTitle, yTitle, max);
                }

                // 绘制饼图
                if (pieCheckbox.property("checked")) {
                    drawPieChart(container, data1, xTitle);
                }

                // 绘制散点图
                if (scatterCheckbox.property("checked")) {
                    drawScatterChart(container, data1, xTitle, yTitle, max);
                }
            } else {
                d3.selectAll('svg').remove();
            }
        }
```

9、柱状图绘制函数
创建 SVG 画布和坐标系
设置 X 轴（年份）和 Y 轴（数值）比例尺
绘制分组柱状图，每个年份下有多个专业柱子
添加数值标签和图例

```
        function drawBarChart(container, data, xTitle, yTitle, max) {
            const margin = { top: 40, right: 30, bottom: 40, left: 50 };
            const width = 600 - margin.left - margin.right;
            const height = 500 - margin.top - margin.bottom;
            const svg = container
                .append("svg")
                .attr("width", width + margin.left + margin.right + 100)
                .attr("height", height + margin.top + margin.bottom)
                .append("g")
                .attr("transform",                  `translate(${margin.left},
${margin.top})`);

            const subgroups = xTitle;
            const groups = yTitle;

            const      x      =      d3.scaleBand().domain(groups).range([0,
width]).padding([0.2]);
            svg
                .append("g")
                .attr("transform", `translate(0, ${height})`)
                .call(d3.axisBottom(x).tickSizeOuter(0));

            const   y   =   d3.scaleLinear().domain([0,   max]).range([height,
0]).nice();
            svg.append("g").call(d3.axisLeft(y));

            const xSubgroup = d3
                .scaleBand()
                .domain(subgroups)
                .range([0, x.bandwidth()])
                .padding([0.05]);

            svg
                .append("g")
                .selectAll("g")
                .data(data)
                .join("g")
                .attr("class", "bar")
                .attr("transform", (d) => `translate(${x(d.group)}, 0)`)
                .selectAll("rect")
                .data(function (d) {
                    return subgroups.map(function (key) {
                        return { key: key, value: d[key] };
                    });
                })
                .join("rect")
                .attr("x", (d) => xSubgroup(d.key))
                .attr("y", (d) => y(d.value))
                .attr("width", xSubgroup.bandwidth())
                .attr("height", (d) => height - y(d.value))
                .attr("fill", function(d, i) { return getColor(i); });

            // 数值标签
            svg
                .append("g")
                .selectAll("g")
                .data(data)
                .join("g")
                .attr("class", "bar")
                .attr("transform", (d) => `translate(${x(d.group)}, 0)`)
                .selectAll("text")
                .data(function (d) {
                    return subgroups.map(function (key) {
                        return { key: key, value: d[key] };
                    });
                })
                .join("text")
                .attr("x", (d) => xSubgroup(d.key) + xSubgroup.bandwidth() * 0.5)
                .attr("y", (d) => y(d.value) - 10)
                .text(d => d.value)
                .attr('text-anchor', 'middle');

            // 图例
            drawLegend(svg, xTitle, width);
        }
```

10、折线图绘制函数
创建折线图坐标系
将数据转换为适合折线图的格式
使用 D3 的 line 生成器绘制连接线
在每个数据点添加圆圈标记

```
        function drawLineChart(container, data, xTitle, yTitle, max) {
            const margin = { top: 40, right: 30, bottom: 40, left: 50 };
            const width = 600 - margin.left - margin.right;
            const height = 500 - margin.top - margin.bottom;

            const svg = container
                .append("svg")
                .attr("width", width + margin.left + margin.right + 100)
                .attr("height", height + margin.top + margin.bottom)
                .append("g")
                .attr("transform",                  `translate(${margin.left},
${margin.top})`);

            const x = d3.scaleBand().domain(yTitle).range([0, width]);
            const    y    =    d3.scaleLinear().domain([0,    max]).range([height,
0]).nice();

            svg
                .append("g")
                .attr("transform", `translate(0, ${height})`)
                .call(d3.axisBottom(x));

            svg.append("g").call(d3.axisLeft(y));

            // 生成折线数据
            const lineData = xTitle.map((category, idx) => ({
                name: category,
                values: data.map(d => ({ year: d.group, value: d[category] }))
            }));

            const line = d3.line()
                .x(d => x(d.year) + x.bandwidth() / 2)
                .y(d => y(d.value));

            // 绘制折线
            svg.selectAll(".line")
                .data(lineData)
                .enter()
                .append("path")
                .attr("class", "line")
                .attr("d", d => line(d.values))
                .attr("fill", "none")
                .attr("stroke", (d, i) => getColor(i))
                .attr("stroke-width", 3);

            // 绘制数据点
            svg.selectAll(".dot")
                .data(lineData)
                .enter()
                .selectAll("circle")
                .data(d => d.values)
                .enter()
                .append("circle")
                .attr("class", "dot")
                .attr("cx", d => x(d.year) + x.bandwidth() / 2)
                .attr("cy", d => y(d.value))
                .attr("r", 5)
                .attr("fill",          (d,          i,          nodes)          =>
getColor(nodes[i].parentNode.__data__.name === xTitle[0] ? 0 : 1));
```

```
            drawLegend(svg, xTitle, width);
        }
```

11、饼图绘制函数
计算各专业的总体数据用于饼图
使用 D3 的 pie 和 arc 生成器创建饼图
在每个扇形中添加百分比标签

```
        function drawPieChart(container, data1, xTitle) {
            const width = 400;
            const height = 400;
            const radius = Math.min(width, height) / 2 - 40;

            const svg = container
                .append("svg")
                .attr("width", width)
                .attr("height", height)
                .append("g")
                .attr("transform", `translate(${width/2}, ${height/2})`);

            // 计算总和用于百分比
            let totalSum = 0;
            data1.forEach(row => {
                row.forEach(value => totalSum += value);
            });

            const pieData = [];
            xTitle.forEach((category, categoryIdx) => {
                let categorySum = 0;
                data1.forEach(row => {
                    categorySum += row[categoryIdx];
                });
                pieData.push({
                    name: category,
                    value: categorySum,
                    percentage: ((categorySum / totalSum) * 100).toFixed(1)
                });
            });

            const pie = d3.pie()
                .value(d => d.value)
                .sort(null);

            const arc = d3.arc()
                .innerRadius(0)
                .outerRadius(radius);
            const arcs = svg.selectAll(".arc")
                .data(pie(pieData))
                .enter()
                .append("g")
                .attr("class", "arc");

            arcs.append("path")
                .attr("d", arc)
                .attr("fill", (d, i) => getColor(i))
                .attr("stroke", "#fff")
                .attr("stroke-width", 2);

            arcs.append("text")
                .attr("transform", d => `translate(${arc.centroid(d)})`)
                .attr("text-anchor", "middle")
                .style("font-size", "12px")
                .style("font-weight", "bold")
                .text(d => `${d.data.name}: ${d.data.percentage}%`);
        }
```

12、散点图绘制函数
创建散点图坐标系
将数据转换为散点格式，每个点代表一个具体数值
绘制圆圈散点并添加数值标签

```
        function drawScatterChart(container, data1, xTitle, yTitle, max) {
            const margin = { top: 40, right: 30, bottom: 40, left: 50 };
            const width = 600 - margin.left - margin.right;
            const height = 500 - margin.top - margin.bottom;

            const svg = container
                .append("svg")
                .attr("width", width + margin.left + margin.right + 100)
                .attr("height", height + margin.top + margin.bottom)
                .append("g")
                .attr("transform",                  `translate(${margin.left},
${margin.top})`);

            const x = d3.scaleBand().domain(yTitle).range([0, width]);
            const   y   =   d3.scaleLinear().domain([0,   max]).range([height,
0]).nice();

            svg
                .append("g")
                .attr("transform", `translate(0, ${height})`)
                .call(d3.axisBottom(x));
            svg.append("g").call(d3.axisLeft(y));

            // 生成散点数据
            const scatterData = [];
            data1.forEach((row, rowIdx) => {
                row.forEach((value, colIdx) => {
                    scatterData.push({
                        year: yTitle[rowIdx],
                        category: xTitle[colIdx],
                        value: value,
                        x: x(yTitle[rowIdx]) + x.bandwidth() / 2 + (colIdx -
xTitle.length/2 + 0.5) * 10,
                        y: y(value)
                    });
                });
            });

            svg.selectAll(".scatter")
                .data(scatterData)
                .enter()
                .append("circle")
                .attr("class", "scatter")
                .attr("cx", d => d.x)
                .attr("cy", d => d.y)
                .attr("r", 8)
                .attr("fill", (d, i) => getColor(xTitle.indexOf(d.category)))
                .attr("stroke", "#fff")
                .attr("stroke-width", 2);

            // 添加数值标签
            svg.selectAll(".scatter-label")
                .data(scatterData)
                .enter()
                .append("text")
                .attr("class", "scatter-label")
                .attr("x", d => d.x)
                .attr("y", d => d.y - 15)
                .attr("text-anchor", "middle")
                .style("font-size", "10px")
                .text(d => d.value);

            drawLegend(svg, xTitle, width);
        }
```

13、图例绘制函数
创建统一的图例绘制函数

为每个数据系列生成颜色方块和标签

```
        function drawLegend(svg, xTitle, width) {
            const legendData = xTitle.map((name, idx) => ({
                name: name,
                color: getColor(idx)
            }));

            const legend = svg.selectAll(".legend")
                .data(legendData)
                .enter()
                .append("g")
                .attr("class", "legend")
                .attr("transform", (d, i) => `translate(30, ${i * 20 + 120})`);

            legend.append("rect")
                .attr("x", width - 25)
                .attr("y", 8)
                .attr("width", 40)
                .attr("height", 5)
                .style("fill", d => d.color);

            legend.append("text")
                .attr("x", width + 20)
                .attr("y", 15)
                .style("text-anchor", "end")
                .text(d => d.name);
        }
```

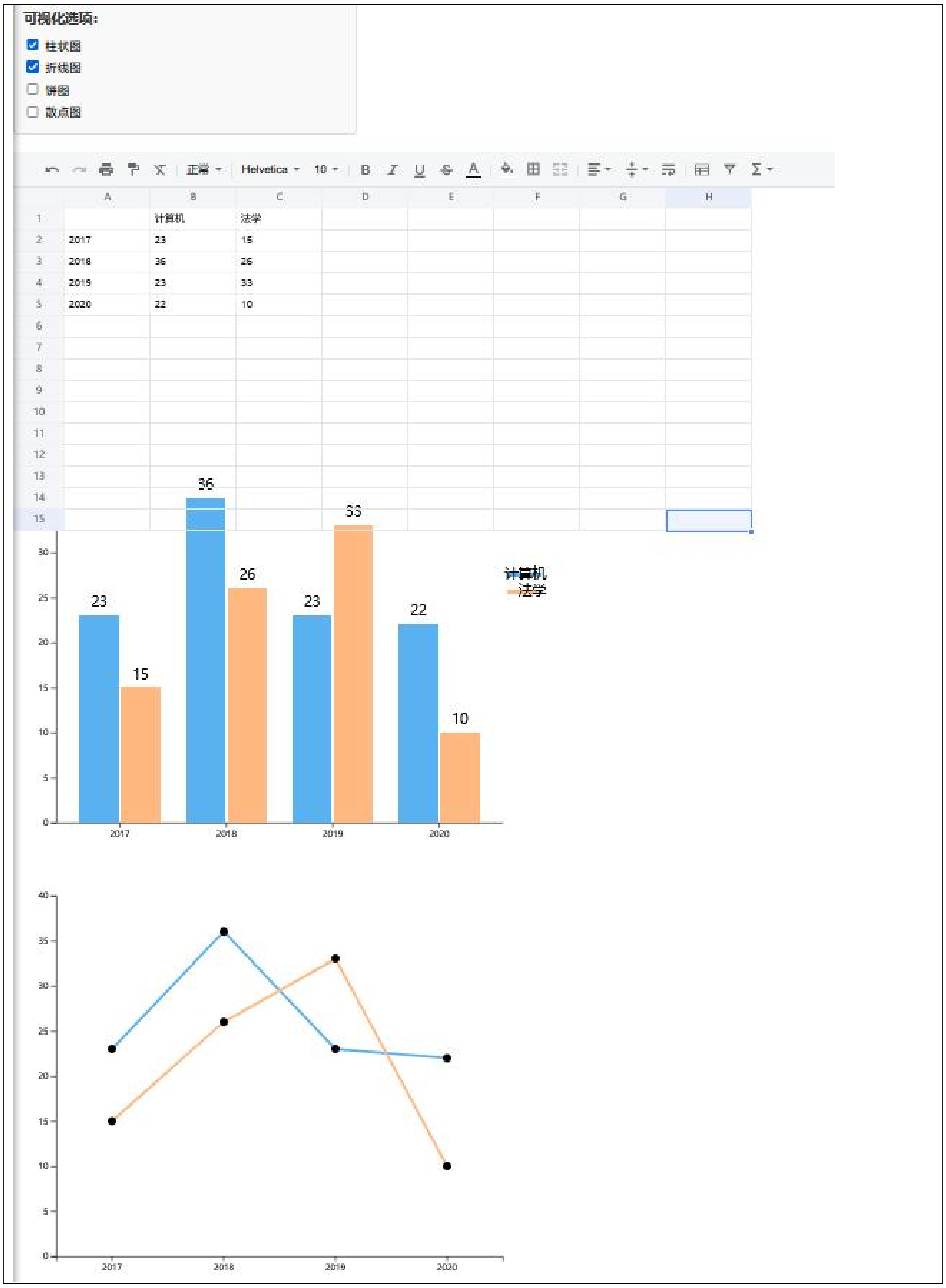
14、事件绑定

将所有复选框的 change 事件绑定到 update 函数

实现用户交互时的图表实时更新

```
        // 将 update 函数作为事件添加到 check-box 与表格修改时
        d3.selectAll(".checkbox").on("change", update);
```

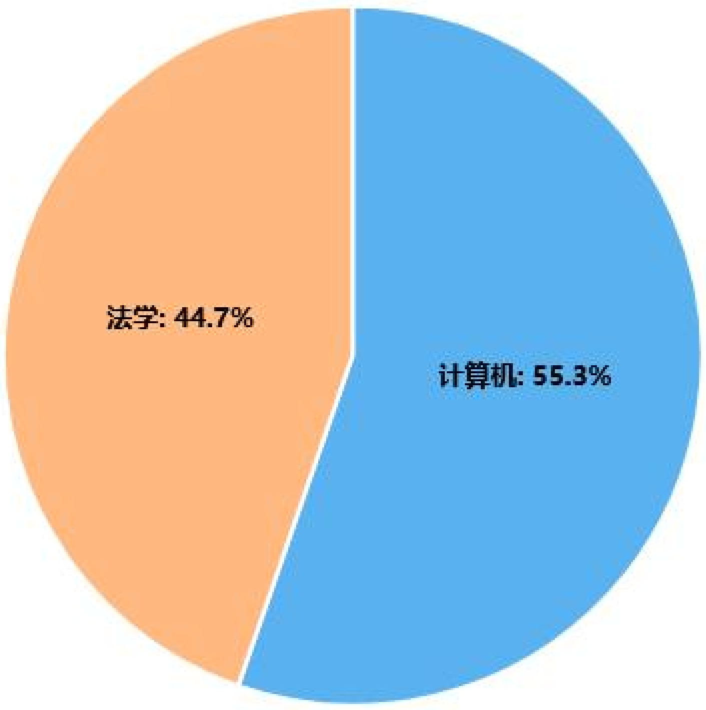
结果:

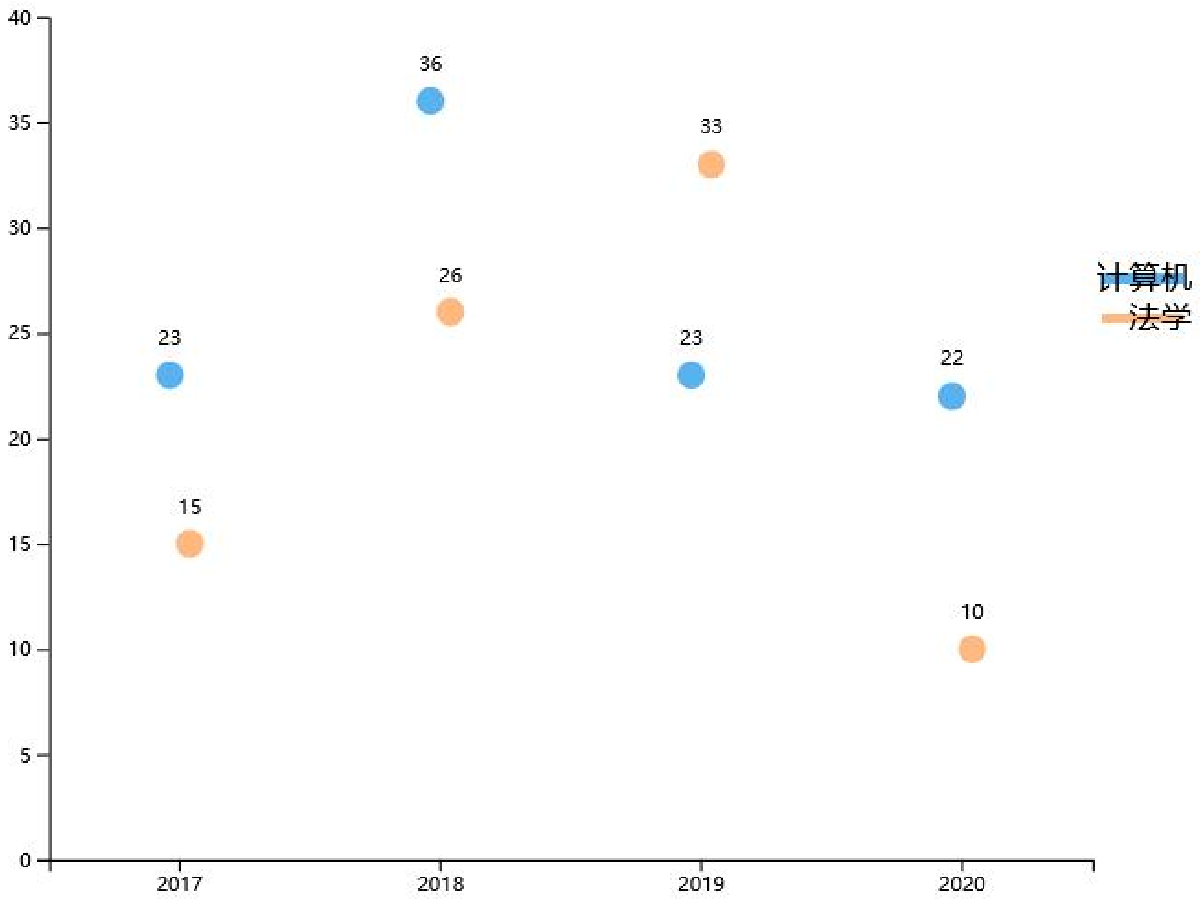


可视化选项:

- [✓] 柱状图
- [✓] 折线图
- [ ] 饼图
- [ ] 散点图

	A	B	C	D	E	F	G	H
1		计算机	法学					
2	2017	23	15					
3	2018	36	26					
4	2019	23	33					
5	2020	22	10					
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								





结论分析与体会：
掌握了 x-spreadsheet 进行表格操作，并利用 d3 进行可视化的方法。