实验报告

课　　程　可视语言与信息可视化

题　　目　城市天气数据的柱形图可视化

分　　数

学院名称　软件工程学院

专　　业　软件工程

学生姓名　杨宇鑫

学　　号　**3022207128**

年　　级　大学三年级

班　　级　软件工程 **4** 班

目录

一、 实验目的

对某城市 365 天的天气数据。包括每天的温度，湿度，云量等等。
统计
"windSpeed","moonPhase","dewPoint","humidity","uvIndex","windBearing","temperatureMin","temperatureMax"几个数据项的数据分布，生成对应的柱状图，并且完成之间的切换动画以及交互功能。

二、 实验内容

对某城市 365 天的天气数据。包括每天的温度，湿度，云量等等。
统计
"windSpeed","moonPhase","dewPoint","humidity","uvIndex","windBearing","temperatureMin","temperatureMax"几个数据项的数据分布，生成对应的柱状图，并且完成之间的切换动画以及交互功能。

三、 实验步骤

1. 使用 D3.js 创建 8 个直方图,展示不同天气数据的分布情况

每个直方图包含:

- 标题
- X 轴(数值范围)
- Y 轴(频数)
- 可交互的柱状图(鼠标悬停时会改变透明度)

图表布局采用:

- 固定宽度(480px)和高度(300px)
- 合适的边距以容纳轴标签和标题
- 自动计算的比例尺

数据处理:

- 自动计算数据范围
- 将数据分成 10 个区间
- 计算每个区间的频数

样式设计:

- 使用 steelblue 颜色的柱状图
- 添加鼠标悬停效果
- 清晰的字体大小和位置

要使用这个页面:

- 将 HTML 文件保存到本地
- 确保 weather_data.json 文件在同一目录下
- 通过 web 服务器访问该页面

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>天气数据分析</title>
    <script src="https://d3js.org/d3.v7.min.js"></script>
    <style>
        .chart-container {
            width: 960px;
            margin: 0 auto;
        }
        .histogram {
            margin: 20px 0;
        }
        .bar {
            fill: steelblue;
            opacity: 0.8;
        }
        .bar:hover {
            opacity: 1;
        }
        .axis-label {
            font-size: 12px;
        }
        .title {
            font-size: 16px;
            text-anchor: middle;
        }
    </style>
</head>
<body>
    <div class="chart-container" id="charts"></div>

    <script>
        // 定义要分析的字段
        const fields = [
            {name: 'windSpeed', title: '风速分布'},
            {name: 'moonPhase', title: '月相分布'},
            {name: 'dewPoint', title: '露点分布'},
            {name: 'humidity', title: '湿度分布'},
            {name: 'uvIndex', title: '紫外线指数分布'},
            {name: 'windBearing', title: '风向分布'},
            {name: 'temperatureMin', title: '最低温度分布'},
```

```
            {name: 'temperatureMax', title: '最高温度分布'}
        ];
        // 设置图表尺寸
        const margin = {top: 40, right: 30, bottom: 40, left: 50};
        const width = 480 - margin.left - margin.right;
        const height = 300 - margin.top - margin.bottom;
        // 加载数据
        d3.json("./weather_data.json").then(data => {
            // 为每个字段创建直方图
            fields.forEach((field, i) => {
                createHistogram(data, field.name, field.title, i);
            });
        });
        function createHistogram(data, fieldName, title, index) {
            // 提取数据
            const values = data.map(d => d[fieldName]);
            // 创建容器
            const svg = d3.select("#charts")
                .append("svg")
                .attr("width", width + margin.left + margin.right)
                .attr("height", height + margin.top + margin.bottom)
                .append("g")
                .attr("transform",
`translate(${margin.left},${margin.top})`);
            // 创建直方图数据
            const histogram = d3.histogram()
                .domain(d3.extent(values))
                .thresholds(10);
            const bins = histogram(values);
            // 创建比例尺
            const x = d3.scaleLinear()
                .domain([bins[0].x0, bins[bins.length - 1].x1])
                .range([0, width]);
            const y = d3.scaleLinear()
                .domain([0, d3.max(bins, d => d.length)])
                .range([height, 0]);
            // 绘制柱状图
            svg.selectAll("rect")
                .data(bins)
                .enter()
                .append("rect")
                .attr("class", "bar")
                .attr("x", d => x(d.x0))
                .attr("width", d => Math.max(0, x(d.x1) - x(d.x0) - 1))
```

```
                .attr("y", d => y(d.length))
                .attr("height", d => height - y(d.length));
            // 添加x轴
            svg.append("g")
                .attr("transform", `translate(0,${height})`)
                .call(d3.axisBottom(x));
            // 添加y轴
            svg.append("g")
                .call(d3.axisLeft(y));
            // 添加标题
            svg.append("text")
                .attr("class", "title")
                .attr("x", width / 2)
                .attr("y", -margin.top / 2)
                .text(title);
            // 添加x轴标签
            svg.append("text")
                .attr("class", "axis-label")
                .attr("text-anchor", "middle")
                .attr("x", width / 2)
                .attr("y", height + margin.bottom)
                .text("数值范围");
            // 添加y轴标签
            svg.append("text")
                .attr("class", "axis-label")
                .attr("text-anchor", "middle")
                .attr("transform", "rotate(-90)")
                .attr("x", -height / 2)
                .attr("y", -margin.left + 15)
                .text("频数");
        }
    </script>
</body>
</html>
```

2. 交互

- 添加了提示框的 CSS 样式
- 为柱状图添加了鼠标悬停（mouseover）和移出（mouseout）事件

鼠标悬停时：

- 改变当前柱形颜色为紫色
- 显示包含数据量的提示框

鼠标移出时：

- 恢复柱形原始颜色
- 隐藏提示框

```
<style>
    // 添加提示框样式
    .tooltip {
        position: absolute;
        padding: 8px;
        background: white;
        border: 1px solid #ddd;
        border-radius: 4px;
        pointer-events: none;
        font-size: 12px;
    }
</style>

<script>
        // 创建提示框
        const tooltip = d3.select("body")
            .append("div")
            .attr("class", "tooltip")
            .style("opacity", 0);
        // 修改柱状图的绘制部分
        svg.selectAll("rect")
            .data(bins)
            .enter()
            .append("rect")
            .attr("class", "bar")
            .attr("x", d => x(d.x0))
            .attr("width", d => Math.max(0, x(d.x1) - x(d.x0) - 1))
            .attr("y", d => y(d.length))
            .attr("height", d => height - y(d.length))
            // 添加鼠标事件
            .on("mouseover", function(event, d) {
                // 改变当前柱形颜色
                d3.select(this)
                    .style("fill", "purple");

                // 显示提示框
                tooltip.style("opacity", 1)
                    .html(`${d.length} 天`)
                    .style("left", (event.pageX + 10) + "px")
                    .style("top", (event.pageY - 10) + "px");
            })
            .on("mouseout", function() {
```

```
                // 恢复原始颜色
                d3.select(this)
                    .style("fill", "steelblue");

                // 隐藏提示框
                tooltip.style("opacity", 0);
            });
    }
</script>
```

3.动画

使用 enter() 处理新增数据：

- 新柱形从底部开始，高度为 0
- 通过 transition() 动画过渡到实际高度

使用 transition() 处理数据更新：

- 平滑过渡到新的位置和高度
- 使用 exit() 处理移除的数据：
- 通过动画将高度降为 0
- 然后移除元素
- 动画持续时间设置为 1 秒（1000 毫秒）

实现柱状图的动画效果：

- 新增数据时，柱形会从底部向上增长
- 更新数据时，柱形会平滑过渡到新的位置和高度
- 移除数据时，柱形会逐渐消失
- 添加了控制按钮的样式
- 添加了右侧控制按钮面板
- 实现了切换功能：
- 记录当前显示的直方图
- 点击按钮时，先淡出当前直方图
- 创建新直方图时添加淡入动画
- 柱形从底部向上增长的动画效果
- 切换动画效果：
- 当前图表淡出（500ms）
- 新图表淡入（500ms）
- 新柱形从底部向上增长（1000ms）

实现了：

- 右侧显示所有可切换的图表按钮
- 每次只显示一个柱形图
- 切换时有平滑的过渡动画
- 新数据带有动画效果

```
<style>
    .controls {
```

```
        position: fixed;
        right: 20px;
        top: 50%;
        transform: translateY(-50%);
        display: flex;
        flex-direction: column;
        gap: 10px;
    }

    .control-button {
        padding: 8px 16px;
        background-color: #4CAF50;
        color: white;
        border: none;
        border-radius: 4px;
        cursor: pointer;
    }

    .control-button:hover {
        background-color: #45a049;
    }
</style>

<body>
    <div class="chart-container" id="charts"></div>
    <div class="controls" id="controls"></div>
    <script>
        let currentIndex = 0;
        let currentHistogram = null;
        // 修改主要逻辑
        d3.json("./weather_data.json").then(data => {
            // 创建控制按钮
            fields.forEach((field, i) => {
                d3.select("#controls")
                    .append("button")
                    .attr("class", "control-button")
                    .text(field.title)
                    .on("click", () => switchHistogram(data, i));
            });
            // 显示第一个直方图
            switchHistogram(data, 0);
        });
        function switchHistogram(data, index) {
            // 如果当前有直方图，先移除它
```

```
            if (currentHistogram) {
                d3.select("#charts").selectAll("svg")
                    .transition()
                    .duration(500)
                    .style("opacity", 0)
                    .remove();
            }
            // 创建新的直方图
            currentIndex = index;
            createHistogram(data, fields[index].name, fields[index].title, index);
        }
        function createHistogram(data, fieldName, title, index) {
            const svg = d3.select("#charts")
                .append("svg")
                .style("opacity", 0)  // 初始透明度为0
                .attr("width", width + margin.left + margin.right)
                .attr("height", height + margin.top + margin.bottom)
                .append("g")
                .attr("transform", `translate(${margin.left},${margin.top})`);
            // 绘制柱状图
            const bars = svg.selectAll("rect")
                .data(bins);
            // 添加新数据的动画
            bars.enter()
                .append("rect")
                .attr("class", "bar")
                .attr("x", d => x(d.x0))
                .attr("width", d => Math.max(0, x(d.x1) - x(d.x0) - 1))
                .attr("y", height)
                .attr("height", 0)
                .style("fill", "steelblue")
                .transition()
                .duration(1000)
                .attr("y", d => y(d.length))
                .attr("height", d => height - y(d.length));
            // 整体淡入动画
            d3.select("#charts").select("svg")
                .transition()
                .duration(500)
                .style("opacity", 1);
            currentHistogram = svg;
        }
    </script>
</body>
```

4. 布局

主要布局优化：

- 创建了一个主容器，使用 flex 布局
- 图表区域和控制按钮并排放置
- 添加了阴影和圆角效果
- 优化了按钮样式和间距
- 改善了整体视觉层次

布局效果：

- 左侧是图表区域，占据主要空间
- 右侧是控制按钮，固定宽度
- 整体居中对齐
- 添加了适当的留白和间距
- 增加了阴影效果提升层次感

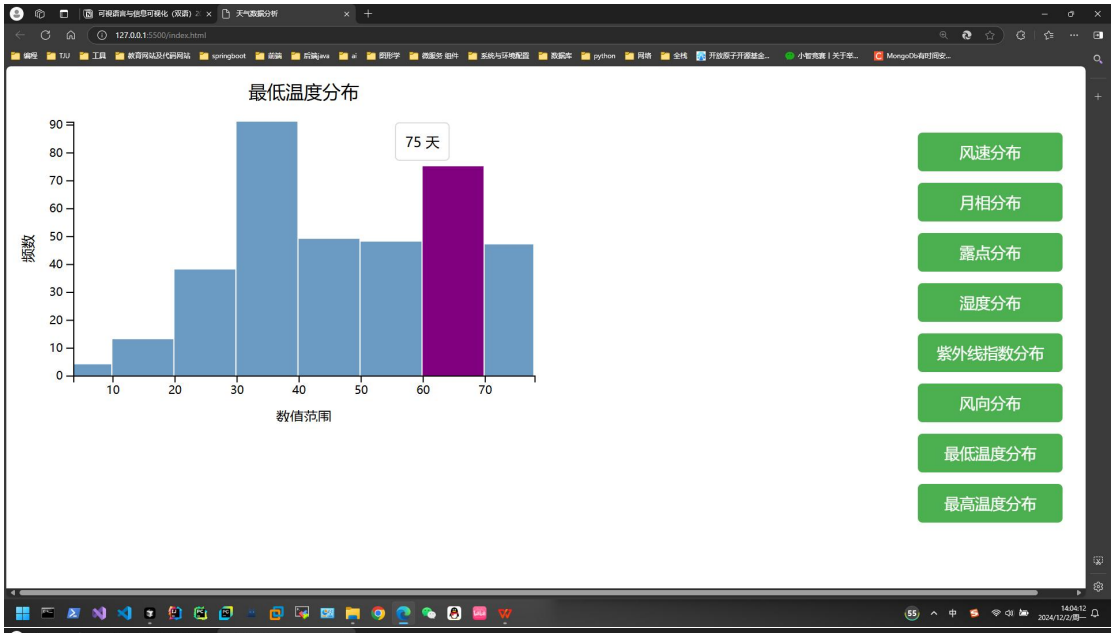
这样的布局更加紧凑和专业，视觉效果也更好。图表和按钮的位置固定，不会随着页面滚动而散乱。

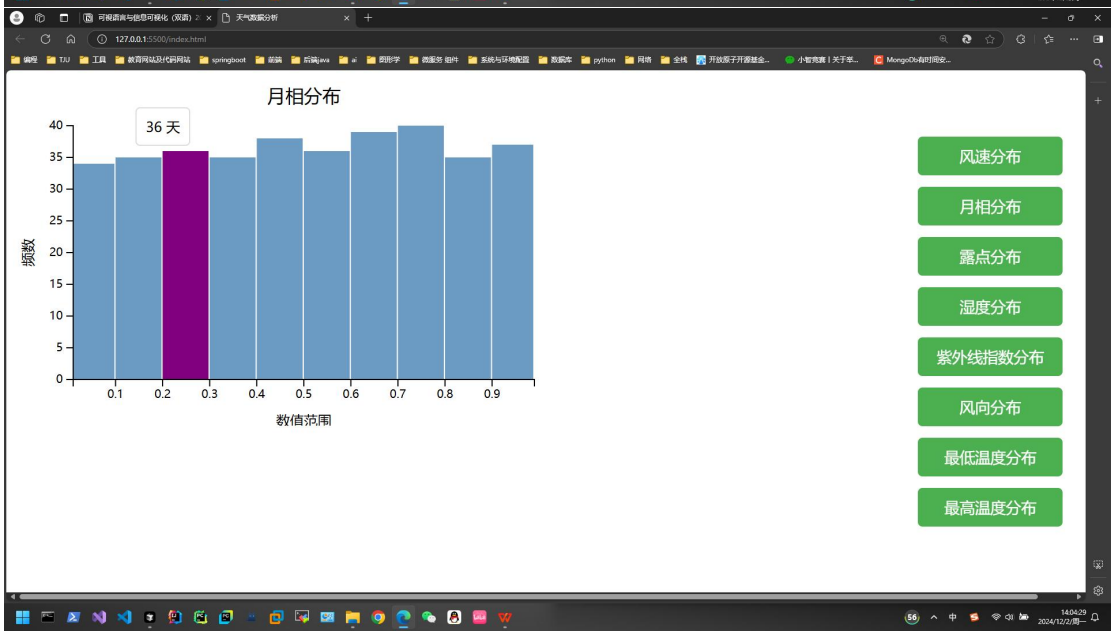
```
<style>
    body {
        margin: 0;
        padding: 20px;
        font-family: Arial, sans-serif;
    }

    .container {
        display: flex;
        gap: 20px;
        max-width: 1200px;
        margin: 0 auto;
        height: 100vh;
    }
    .chart-container {
        flex: 1;
        background: #fff;
        border-radius: 8px;
        box-shadow: 0 2px 8px rgba(0,0,0,0.1);
        padding: 20px;
        display: flex;
        justify-content: center;
        align-items: center;
    }
    .controls {
        width: 200px;
        padding: 20px;
```

```
        background: #f5f5f5;
        border-radius: 8px;
        box-shadow: 0 2px 8px rgba(0,0,0,0.1);
    }
    .control-button {
        width: 100%;
        padding: 12px;
        margin-bottom: 10px;
        background-color: #4CAF50;
        color: white;
        border: none;
        border-radius: 4px;
        cursor: pointer;
        transition: background-color 0.3s;
    }
    .control-button:hover {
        background-color: #45a049;
    }
    .tooltip {
        position: absolute;
        padding: 8px 12px;
        background: white;
        border: 1px solid #ddd;
        border-radius: 4px;
        pointer-events: none;
        font-size: 12px;
        box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    }
    .title {
        font-size: 18px;
        font-weight: bold;
        text-anchor: middle;
    }
    .axis-label {
        font-size: 12px;
    }
</style>
<body>
    <div class="container">
        <div class="chart-container" id="charts"></div>
        <div class="controls" id="controls"></div>
    </div>
    <script>
```

```
    </script>
</body>
```

四、 实验结果





最终成果视频打包在了文件中

五、 实验结论

六、 源代码

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>天气数据分析</title>
    <script src="https://d3js.org/d3.v7.min.js"></script>
    <style>
        .chart-container {
            width: 960px;
            margin: 0 auto;
        }
        .histogram {
            margin: 20px 0;
        }
        .bar {
            fill: steelblue;
            opacity: 0.8;
        }
        .bar:hover {
            opacity: 1;
        }
        .axis-label {
            font-size: 12px;
        }
        .title {
            font-size: 16px;
            text-anchor: middle;
        }
        .tooltip {
            position: absolute;
            padding: 8px;
            background: white;
            border: 1px solid #ddd;
            border-radius: 4px;
            pointer-events: none;
            font-size: 12px;
        }
        .controls {
            position: fixed;
```

```
            right: 20px;
            top: 50%;
            transform: translateY(-50%);
            display: flex;
            flex-direction: column;
            gap: 10px;
        }

        .control-button {
            padding: 8px 16px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }

        .control-button:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <div class="chart-container" id="charts"></div>
    <div class="controls" id="controls"></div>

    <script>
        // 定义要分析的字段
        const fields = [
            {name: 'windSpeed', title: '风速分布'},
            {name: 'moonPhase', title: '月相分布'},
            {name: 'dewPoint', title: '露点分布'},
            {name: 'humidity', title: '湿度分布'},
            {name: 'uvIndex', title: '紫外线指数分布'},
            {name: 'windBearing', title: '风向分布'},
            {name: 'temperatureMin', title: '最低温度分布'},
            {name: 'temperatureMax', title: '最高温度分布'}
        ];
        let currentHistogram = null;
        let globalData = null;  // 存储全局数据
        // 设置图表尺寸
        const margin = {top: 40, right: 30, bottom: 40, left: 50};
        const width = 480 - margin.left - margin.right;
        const height = 300 - margin.top - margin.bottom;
```

```
        // 加载数据
        d3.json("./weather_data.json").then(data => {
            globalData = data;  // 保存数据到全局变量

            // 创建控制按钮
            fields.forEach((field, i) => {
                d3.select("#controls")
                    .append("button")
                    .attr("class", "control-button")
                    .text(field.title)
                    .on("click", () => switchHistogram(i));  // 简化调用
            });
            // 显示第一个直方图
            switchHistogram(0);
        });
        function switchHistogram(index) {
            const field = fields[index];

            // 清除现有图表
            d3.select("#charts").selectAll("svg").remove();

            // 获取当前字段的数据
            const values = globalData.map(d => d[field.name]);

            // 创建 SVG
            const svg = d3.select("#charts")
                .append("svg")
                .attr("width", width + margin.left + margin.right)
                .attr("height", height + margin.top + margin.bottom)
                .append("g")
                .attr("transform",
`translate(${margin.left},${margin.top})`);
            // 创建直方图数据
            const histogram = d3.histogram()
                .domain(d3.extent(values))
                .thresholds(10);
            const bins = histogram(values);
            // 创建比例尺
            const x = d3.scaleLinear()
                .domain([bins[0].x0, bins[bins.length - 1].x1])
                .range([0, width]);
            const y = d3.scaleLinear()
                .domain([0, d3.max(bins, d => d.length)])
                .range([height, 0]);
```

```
            // 创建提示框
            const tooltip = d3.select("body")
                .append("div")
                .attr("class", "tooltip")
                .style("opacity", 0);
            // 绘制柱状图（带动画）
            svg.selectAll("rect")
                .data(bins)
                .enter()
                .append("rect")
                .attr("class", "bar")
                .attr("x", d => x(d.x0))
                .attr("width", d => Math.max(0, x(d.x1) - x(d.x0) - 1))
                .attr("y", height)  // 开始时的位置
                .attr("height", 0)  // 开始时的高度
                .transition()  // 添加过渡动画
                .duration(1000)
                .attr("y", d => y(d.length))
                .attr("height", d => height - y(d.length));
            // 添加交互效果
            svg.selectAll("rect")
                .on("mouseover", function(event, d) {
                    d3.select(this)
                        .style("fill", "purple");

                    tooltip.style("opacity", 1)
                        .html(`${d.length} 天`)
                        .style("left", (event.pageX - 25) + "px")
                        .style("top", (y(d.length) + margin.top - 30) +
"px");
                })
                .on("mouseout", function() {
                    d3.select(this)
                        .style("fill", "steelblue");

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

```
            // 添加标题
            svg.append("text")
                .attr("class", "title")
                .attr("x", width / 2)
                .attr("y", -margin.top / 2)
                .text(field.title);
            // 添加x轴标签
            svg.append("text")
                .attr("class", "axis-label")
                .attr("text-anchor", "middle")
                .attr("x", width / 2)
                .attr("y", height + margin.bottom)
                .text("数值范围");
            // 添加y轴标签
            svg.append("text")
                .attr("class", "axis-label")
                .attr("text-anchor", "middle")
                .attr("transform", "rotate(-90)")
                .attr("x", -height / 2)
                .attr("y", -margin.left + 15)
                .text("频数");
            // 整体淡入动画
            d3.select("#charts").select("svg")
                .transition()
                .duration(500)
                .style("opacity", 1);
            currentHistogram = svg;
        }
    </script>
</body>
</html>
```