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：202300130098	姓名：马浩鑫	班级：23 级数据
实验题目：表格实践		
实验学时：2 课时	实验日期：	

实验目标：

在 x-spreadsheet 系统中添加可视化数据的功能

实验描述：

利用 x-spreadsheet 进行表格操作，利用 d3 进行可视化

以下是具体实现方法：

1. 在 src 文件夹中添加了 visualization 文件夹来实现可视化功能
2. 其中添加了 get_data 函数，用于将获取选中区域的数据，并返回给可视化函数。

```
import Sheet from "../component/sheet";

// src/visualization/get_data.js

export function get_data(sheet) {
  try {
    const { data, selector } = sheet;

    if (!data || !selector) {
      console.error('getSelectedData: sheet 未定义或缺少必要属性');
      return [];
    }

    // 获取选区范围
    const range = selector.range;
    if (!range) {
      console.error('getSelectedData: 未找到选区');
      return [];
    }

    const { sri, sci, eri, eci } = range; // 起始行/列，结束行/列

    const result = [];
    for (let r = sri; r <= eri; r++) {
      const row = [];
      for (let c = sci; c <= eci; c++) {
        const cell = data.getCell(r, c);
        if (cell) {
          row.push(cell.text || cell.value || '');
        } else {
          row.push('');
        }
      }
      result.push(row);
    }

    console.log('选中区域数据：', result);
    return result;
  } catch (err) {
    console.error('get_data.js 出错：', err);
    return [];
  }
}
```

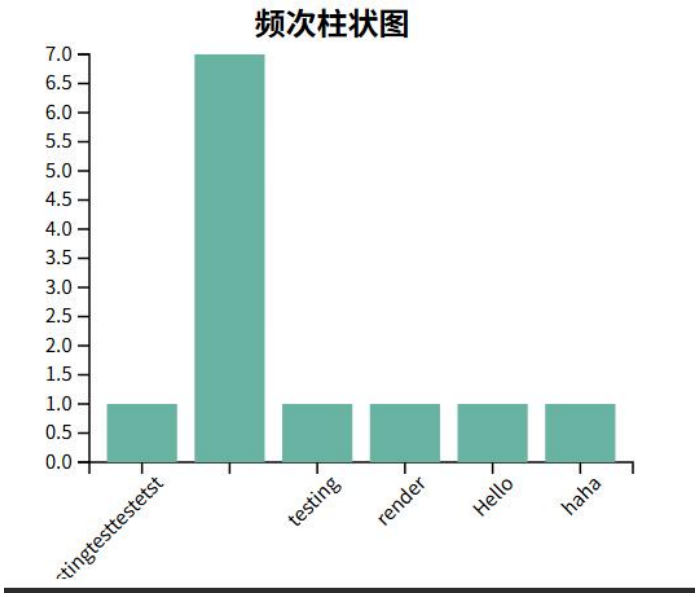
3. 同时添加了 vis 函数，用于将得到的数据进行可视化。同时支持生成频率柱状图、扇形图和折线图

```
/** 柱状图逻辑 **/
function drawBar(selection, data, width, height, margin) {
  const svg = selection.append('svg').attr('width', width).attr('height', height);

  const x = d3.scaleBand()
    .domain(data.map(d => d.label))
    .range([margin.left, width - margin.right])
    .padding(0.2);

  const y = d3.scaleLinear()
    .domain([0, d3.max(data, d => d.value)]).nice()
    .range([height - margin.bottom, margin.top]);

  svg.append('g')
    .attr('transform', `translate(0,${height - margin.bottom})`)
    .call(d3.axisBottom(x))
    .selectAll("text").attr("transform", "rotate(-45)").style("text-anchor", "end");

  svg.append('g')
    .attr('transform', `translate(${margin.left},0)`)
    .call(d3.axisLeft(y));

  svg.selectAll('rect')
    .data(data)
    .enter().append('rect')
    .attr('x', d => x(d.label))
    .attr('y', d => y(d.value))
    .attr('width', x.bandwidth())
    .attr('height', d => height - margin.bottom - y(d.value))
    .attr('fill', '#69b3a2');

  svg.append('text').attr('x', width/2).attr('y', 20).attr('text-anchor', 'middle').text('频次柱状图').style('font-weight', 'bold');
}
```

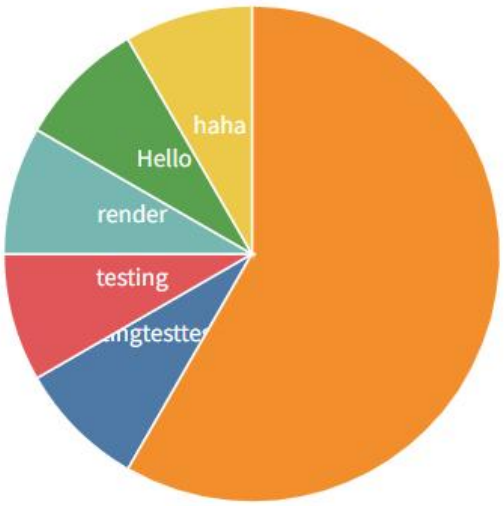
```
/** 扇形图逻辑 **/
function drawPie(selection, data, width, height) {
  const svg = selection.append('svg').attr('width', width).attr('height', height);
  const radius = Math.min(width, height) / 2 - 40;
  const color = d3.scaleOrdinal(d3.schemeTableau10);

  const g = svg.append('g').attr('transform', `translate(${width / 2},${height / 2 + 10})`);

  const pie = d3.pie().value(d => d.value);
  const arc = d3.arc().innerRadius(0).outerRadius(radius);

  const arcs = g.selectAll('.arc')
    .data(pie(data))
    .enter().append('g');

  arcs.append('path')
    .attr('d', arc)
    .attr('fill', (d, i) => color(i))
    .attr('stroke', '#fff');

  // 简单的标签显示（仅显示值较大的）
  arcs.filter(d => d.endAngle - d.startAngle > 0.2)
    .append('text')
    .attr('transform', d => `translate(${arc.centroid(d)})`)
    .attr('text-anchor', 'middle')
    .text(d => d.data.label)
    .style('font-size', '10px').style('fill', '#fff');

  svg.append('text').attr('x', width/2).attr('y', 20).attr('text-anchor', 'middle').text('占比扇形图').style('font-weight', 'bold');
}
```

```
/** 折线图逻辑 **/
function drawLine(selection, data, width, height, margin) {
  const svg = selection.append('svg').attr('width', width).attr('height', height);

  const x = d3.scalePoint()
    .domain(data.map(d => d.label))
    .range([margin.left, width - margin.right]);

  const y = d3.scaleLinear()
    .domain([0, d3.max(data, d => d.value)]).nice()
    .range([height - margin.bottom, margin.top]);

  svg.append('g').attr('transform', `translate(0,${height - margin.bottom})`).call(d3.axisBottom(x));
  svg.append('g').attr('transform', `translate(${margin.left},0)`).call(d3.axisLeft(y));

  const line = d3.line()
    .x(d => x(d.label))
    .y(d => y(d.value))
    .curve(d3.curveMonotoneX); // 使线条圆滑

  svg.append('path')
    .datum(data)
    .attr('fill', 'none')
    .attr('stroke', '#4e79a7')
    .attr('stroke-width', 2)
    .attr('d', line);

  svg.selectAll('circle')
    .data(data)
    .enter().append('circle')
    .attr('cx', d => x(d.label))
    .attr('cy', d => y(d.value))
    .attr('r', 4)
    .attr('fill', '#4e79a7');

  svg.append('text').attr('x', width/2).attr('y', 20).attr('text-anchor', 'middle').text('趋势折线图').style('font-weight', 'bold');
}
```

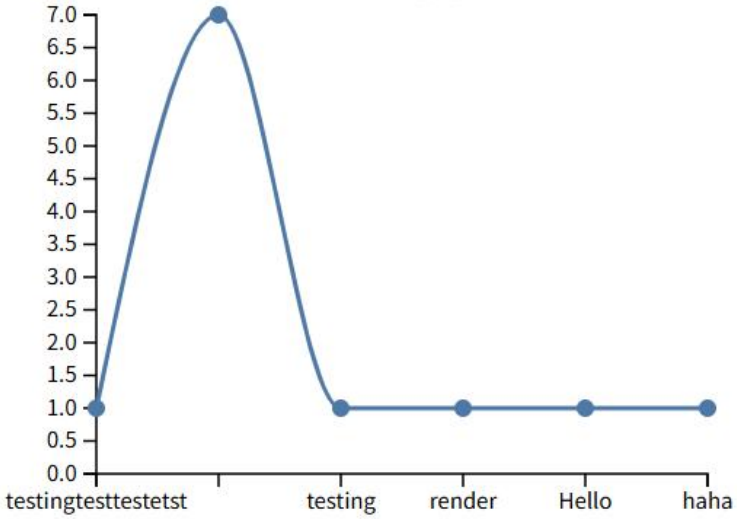
4. 在 src/component/contextmenu.js 中找到 menuItems，在其中添加可视化的功能键，点击后可以在下方得到可视化结果



占比扇形图


haha
Hello
render
testing

趋势折线图


7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
testingtesttestetst
testing
render
Hello
haha

结论分析：
技术实现分析
数据预处理逻辑：
通过 data.flat() 将二维表格数组降维，利用 JavaScript 对象（Hash Map）实现了 O(n) 时间复杂度的频率统计。这一步是可视化的核心，确保了无论是离散的字符串还是连续的数字都能被正确归类。

多图表并行渲染：
在单次函数调用中，通过动态操作 DOM（flex 布局）创建了三个独立的渲染空间。这种模块化绘图逻辑（drawBar, drawPie, drawLine）不仅提高了代码的可维护性，也保证了不同图表类型之间的数据一致性。
. 可视化维度分析
通过生成的三种图表，我们可以得出以下数据洞察结论：

图表类型	表达重点	结论分析价值
柱状图	频数分布	能够直观对比不同数值/类别的出现频率，快速识别出数据集中的“众数”或最频繁出现的项。
扇形图	构成比例	自动计算各部分占总选区的百分比。适合分析数据的成分构成，判断某一类数据是否占据主导地位（如：A 类数据占比 > 50%）。
折线图	趋势/波动	虽然数据是频率统计，但折线图展示了不同分类间权重的“起伏”。若 X 轴具有序性（如日期或等级），则能揭示数据分布的演变规律。