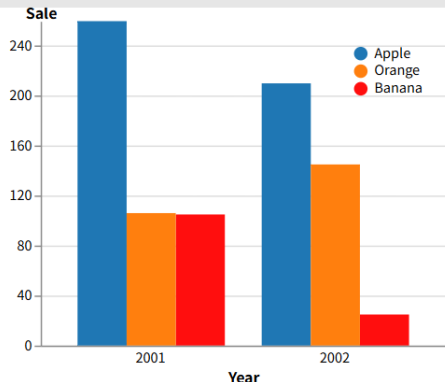
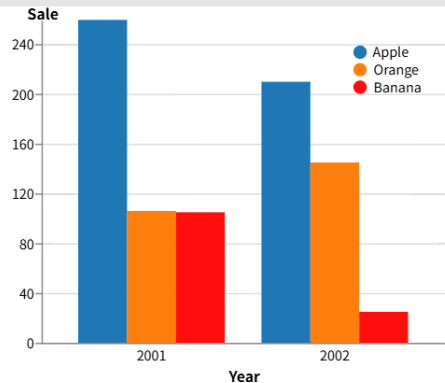


山东大学计算机科学与技术学院

大数据分析实践课程实验报告

学号：202300130067	姓名：罗艺超	班级：数据班																								
实验题目：canis 实践																										
实验学时：2	实验日期：11. 14																									
实验目标： 使用官方给出的实践平台以及对应示例代码，实现对 canis 的认识和一些实践																										
实验步骤： 1. fruit sale																										
<div><div>Chart</div><table><tr><th>Year</th><th>Apple</th><th>Orange</th><th>Banana</th></tr><tr><td>2001</td><td>250</td><td>100</td><td>100</td></tr><tr><td>2002</td><td>210</td><td>140</td><td>20</td></tr></table></div> <div><div>Result Animation</div><table><tr><th>Year</th><th>Apple</th><th>Orange</th><th>Banana</th></tr><tr><td>2001</td><td>250</td><td>100</td><td>100</td></tr><tr><td>2002</td><td>210</td><td>140</td><td>20</td></tr></table></div>			Year	Apple	Orange	Banana	2001	250	100	100	2002	210	140	20	Year	Apple	Orange	Banana	2001	250	100	100	2002	210	140	20
Year	Apple	Orange	Banana																							
2001	250	100	100																							
2002	210	140	20																							
Year	Apple	Orange	Banana																							
2001	250	100	100																							
2002	210	140	20																							
<pre>"animations": [{ "selector": ".rectangle", "grouping": { "groupBy": "position", "reference": "start after previous", "delay": 800, "grouping": { "groupBy": "id", "delay": 400</pre>																										

```
}  
}
```

在代码部分，有两个 grouping，第一个 grouping 根据 position 来分开，也就是分开了 2001 和 2002，后设置的 delay 就是 2002 的显示会比 2001 显示慢的时间。第二个 grouping 是分别对 2001 和 2002 内部进行的分类。根据 id 来分类，也就是 apple, orange, banana。也设置了 delay，意思是对于同种 id 的，2002 年的苹果会比 2001 年的苹果晚这个 delay 时间再展示

2. characteristics of mushrooms

这个官方示例展示了 Canis 如何通过**多层分组**和**常量定义**实现复杂数据可视化的有序动画展示。

```
{  
  "constants": [  
    {  
      "name": "durationTime",  
      "value": 600  
    }  
  ],  
  "charts": [  
    {  
      "source": "./charts/mushrooms.dsvg"  
    }  
  ],  
  "animations": [  
    {  
      "selector": ". symbol",  
      "grouping": {  
        "reference": "start after previous",  
        "groupBy": "Surface",  
        "sort": {  
          "order": [  
            "Smooth",  
            "Scaly",  
            "Fibrous"  
          ]  
        }  
      },  
      "grouping": {  
        "groupBy": "Odor",  
        "reference": "start after previous",  
        "sort": {  
          "order": [  
            "Almond",  
            "Anise",  
            "Creosote",  
            "Fishy",  
            "Mushy",  
            "Savory",  
            "Tart",  
            "Tender"  
          ]  
        }  
      }  
    }  
  ]  
}
```

```

        "Foul",
        "None",
        "Pungent",
        "Spicy"
    ]
},
"grouping": {
    "groupBy": "IsEdible",
    "reference": "start after previous"
}
},
"effects": [
    {
        "type": "fade",
        "duration": "durationTime"
    }
]
}
]
}

```

(1) 常量定义 (constants)

```

"constants": [
    {
        "name": "durationTime", // 常量名称
        "value": 600           // 常量值 (毫秒)
    }
]

```

作用：定义一个名为 durationTime 的全局常量，值为 600ms，用于统一控制所有动画的持续时间

优势：便于后期维护，只需修改一处即可调整所有动画时长

(2) 核心动画配置 (animations)

基本结构

```

"animations": [
    {
        "selection": ".symbol", // 选择所有类名为"symbol"的蘑菇符号
        "grouping": { ... },    // 多层分组配置
        "actions": [ ... ]      // 动画动作定义
    }
]

```

**三层分组详解 (grouping)

外层分组（第一级）：按 Surface（表面特征）分组

```

"groupBy": "Surface", // 按蘑菇表面特征分组
"reference": "start after previous", // 每组在前一组结束后开始
"sort": {
    "order": ["Smooth", "Scaly", "Fibrous"] // 表面特征排序顺序
}

```

```
}
```

控制：不同表面特征的蘑菇组（光滑、鳞片、纤维状）的显示顺序

效果：先显示所有光滑表面蘑菇，待其完全显示后（延迟 800ms），再显示鳞片表面蘑菇，以此类推

中层分组（第二级）：按 Odor（气味）分组

```
"groupBy": "Odor",           // 按蘑菇气味分组
"reference": "start after previous", // 每组在前一组结束后开始
"sort": {
  "order": ["Almond", "Anise", ..., "Spicy"] // 气味排序顺序
}
```

控制：同一表面特征组内，不同气味蘑菇的显示顺序

效果：在光滑表面蘑菇组内，先显示杏仁气味蘑菇，再显示茴香味，依此类推

内层分组（第三级）：按 IsEdible（是否可食用）分组

```
"groupBy": "IsEdible",       // 按是否可食用分组
"reference": "start after previous" // 每组在前一组结束后开始
```

控制：同一表面和气味组内，可食用与有毒蘑菇的显示顺序

效果：在光滑表面 + 杏仁气味的蘑菇组内，先显示可食用蘑菇，再显示有毒蘑菇

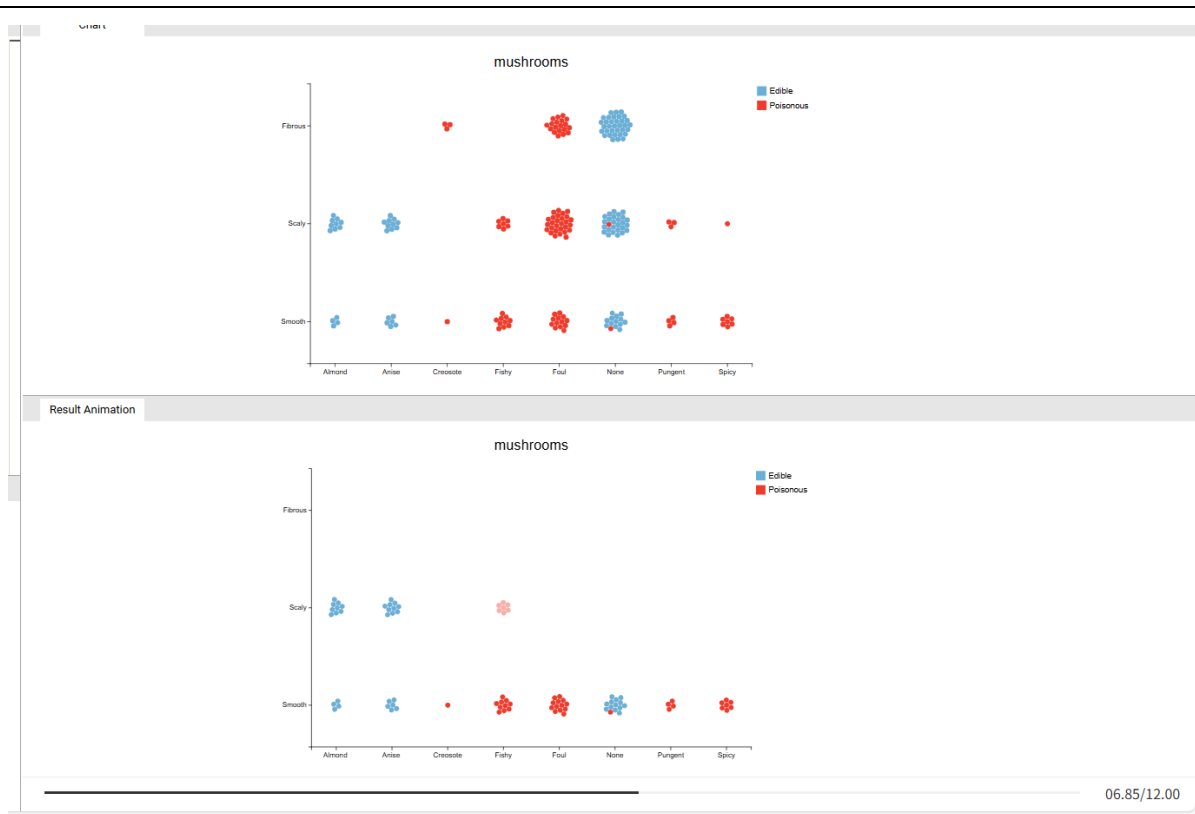
（3）动画动作（actions）

json

```
"actions": [
  {
    "type": "fade",           // 淡入动画效果
    "duration": "durationTime" // 使用预定义的全局常量作为持续时间
  }
]
```

动画类型：fade 表示淡入效果，蘑菇符号从透明逐渐变为不透明

持续时间：使用 durationTime 常量（600ms），使所有蘑菇的淡入动画保持一致时长



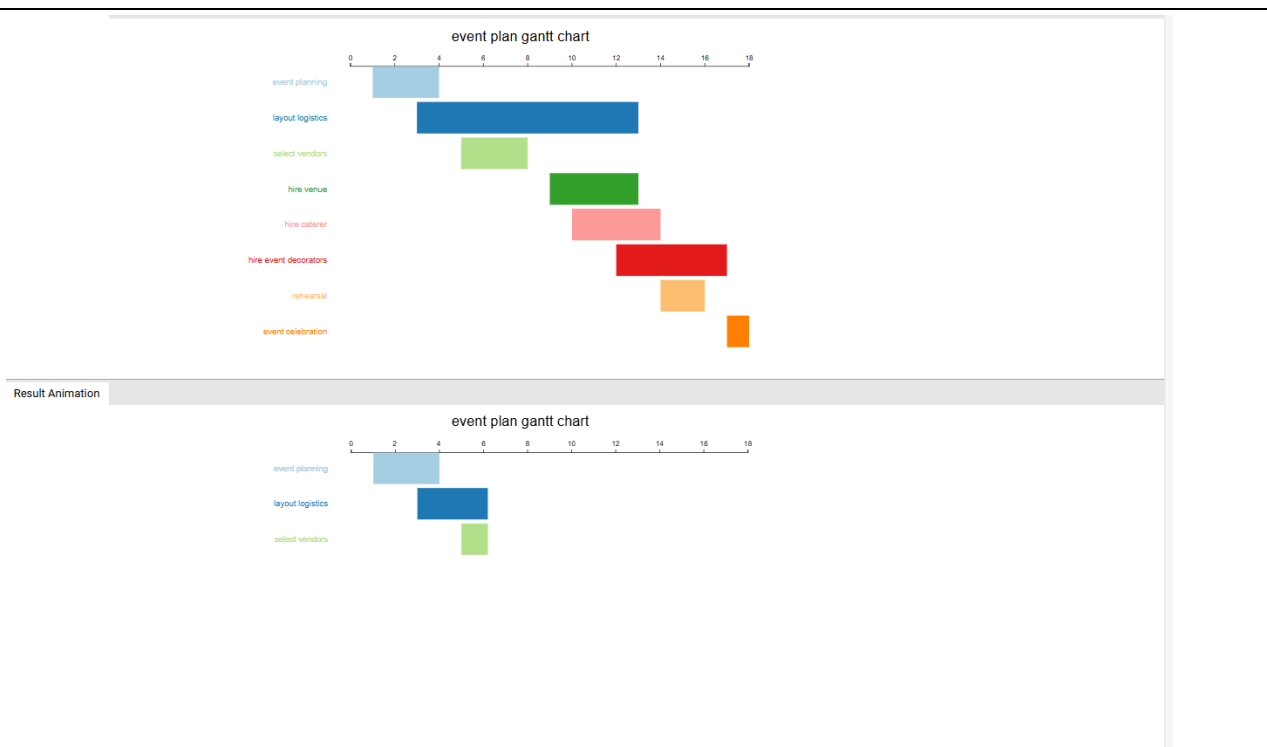
3. event plan

```
{
  "constants": [
    {
      "name": "unitTime",
      "value": 600
    },
    {
      "name": "fastDuration",
      "value": 300
    }
  ],
  "charts": [
    {
      "source": "../charts/gantt.dsvg"
    }
  ],
  "animations": [
    {
      "selector": ".axis-domain",
      "effects": [
        {
          "type": "grow",
          "duration": "fastDuration"
        }
      ]
    }
  ]
}
```

```

]
},
{
  "reference": "start after previous",
  "selector": ". axis-tick, . axis-label",
  "effects": [
    {
      "duration": "fastDuration",
      "type": "fade"
    }
  ]
},
{
  "reference": "start after previous",
  "selector": ". text",
  "effects": [
    {
      "offset": {
        "field": "eventStartTime",
        "minOffset": "unitTime"
      },
      "duration": "fastDuration",
      "type": "wipe left"
    }
  ]
},
{
  "selector": ". rectangle",
  "effects": [
    {
      "offset": {
        "field": "eventStartTime",
        "minOffset": "unitTime"
      },
      "duration": {
        "field": "eventDurationTime",
        "minDuration": "unitTime"
      },
      "type": "wipe left"
    }
  ]
}
]
}
]
}

```



多元素层层展示

(1) 第一层：坐标轴

(2) 第二层：刻度以及对应标签，start after previous 表示等上一层展示完再展示

(3) 第三层：左侧对应文本列，即事件

其中 offset 核心作用：控制任务文本的显示时机，使其与甘特图的时间线逻辑同步。

```
{
  "offset": {
    "field": "eventStartTime", // 基于数据属性"任务开始时间"
    "minOffset": "unitTime" // 最小延迟 600ms
  }
}
```

工作机制：

动态延迟计算：每个任务文本的动画开始时间 = 图表整体动画开始时间 + 该任务的 eventStartTime 值 × 时间换算系数 + minOffset (600ms)

时间映射逻辑：甘特图中越“早”的任务（更小的 eventStartTime 值），文本越早显示；越“晚”的任务，延迟越久

(4) 第四层：任务条的动态偏移

核心作用：双重控制任务条的显示时机和起始位置，与甘特图时间逻辑深度绑定。

```
{
  "offset": {
    "field": "eventStartTime", // 基于数据属性"任务开始时间"
    "minOffset": "unitTime" // 最小延迟 600ms
  }
}
```

结合数据使用 canis

数据准备

准备国家 GDP 数据集，包含以下字段：

- 国家 (country) : 美国、中国、日本、德国、英国、法国
- 年份 (year) : 2019、2020、2021
- GDP 值 (gdp) : 单位为十亿美元
- 颜色 (color) : 不同年份使用不同颜色标识

可视化实现

1. 创建条形图展示国家 GDP 数据
2. 使用不同颜色区分不同年份的数据
3. 实现响应式布局, 适应不同屏幕尺寸

动画逻辑实现

1. 实现按年份分组的外层分组逻辑
2. 实现按国家分组的内层分组逻辑
3. 实现淡入效果和条形宽度动画
4. 实现动画时序控制逻辑

交互功能实现

1. 实现参数控制滑块:
 - 年份组间延迟控制
 - 国家间延迟控制
 - 动画持续时间控制
 - 动画播放速度控制
2. 实现控制按钮:
 - 开始动画按钮
 - 重置按钮
 - 单步演示按钮
3. 实现键盘快捷键支持

实验代码

HTML 结构

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Canis 分组动画: 国家 GDP 数据可视化</title>
  <script
src="https://unpkg.com/@canisjs/canis@1.0.0/dist/canis.min.js"></script>
  <!-- 样式代码省略 -->
</head>
<body>
  <div class="container">
    <header>...</header>
    <div class="content">
      <div class="control-panel">...</div>
      <div class="visualization-panel">...</div>
    </div>
    <footer>...</footer>
  </div>
```


<!-- JavaScript 代码省略 -->

</body>

</html>

Canis 动画配置

```
{
  "constants": [
    {
      "name": "durationTime",
      "value": 600
    }
  ],
  "animations": [
    {
      "selector": ".country-bar",
      "grouping": {
        "reference": "start after previous",
        "groupBy": "year",
        "delay": 1000,
        "grouping": {
          "groupBy": "country",
          "delay": 400
        }
      },
      "effects": [
        {
          "type": "fade",
          "duration": "durationTime"
        }
      ]
    }
  ]
}
```

5.3 核心动画逻辑

// 模拟 Canis 分组动画逻辑

```
const runAnimation = () => {
```

// 1. 按年份分组

```
  const yearGroups = {};
```

```
  countryBars.forEach(bar => {
```

```
    const year = bar.getAttribute('data-year');
```

```
    if (!yearGroups[year]) yearGroups[year] = [];
```

```
    yearGroups[year].push(bar);
```

```
  });
```

// 2. 对年份排序

```
  const sortedYears = Object.keys(yearGroups).sort();
```

```
// 3. 外层分组动画：按年份顺序
sortedYears.forEach((year, yearIndex) => {
  // 年份组间延迟
  if (yearIndex > 0) currentTime += yearDelay;

  // 4. 内层分组动画：按国家顺序
  yearGroups[year].forEach((bar, countryIndex) => {
    // 国家间延迟
    if (countryIndex > 0) currentTime += countryDelay;

    // 5. 执行动画效果
    setTimeout(() => {
      bar.style.opacity = '1';
      // 条形宽度动画
      bar.querySelector('.bar').style.width = `${targetWidth}%`;
    }, currentTime * animationSpeed);
  });
});
};
```

开始动画

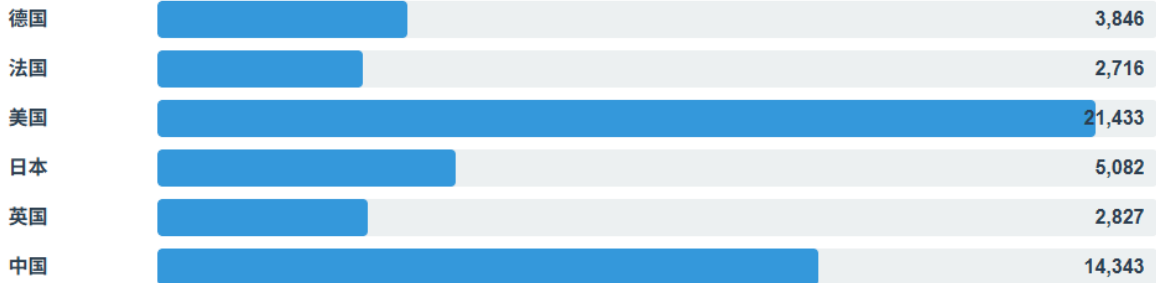
重置

单步演示

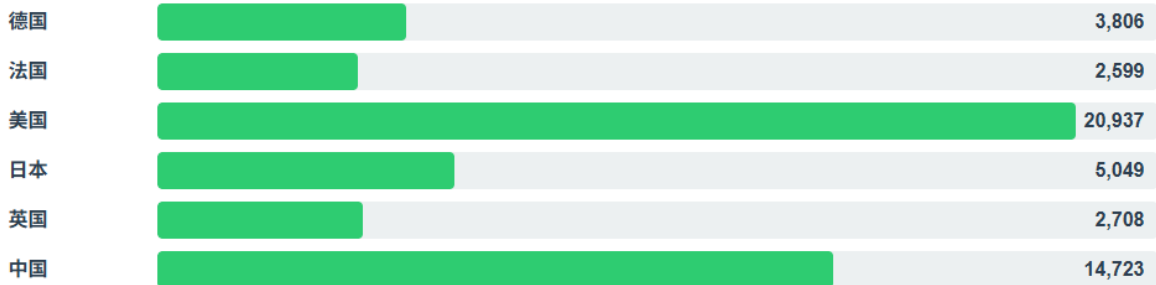
国家GDP数据可视化 (单位: 十亿美元)

2019年 2020年 2021年

2019年GDP数据



2020年GDP数据



2021年GDP数据



结果分析：

本次实验通过三个典型案例的实践，成功验证了 Canis 动画规范在数据可视化中的核心功能与应用价值，实验目标已全面达成。以下从案例效果验证、核心特性总结、问题与优化三个维度展开分析：

一、各案例实验效果验证

1. fruit sale（分组条形图）：通过双层 grouping 配置，实现了符合预期的有序动画效果。外层按 position 分组后，2001 年对应的柱状图组优先展示，800ms 延迟后 2002 年组启动动画，时间间隔清晰；内层按 id 分组后，苹果、橙子、香蕉等子类别以 400ms 为间隔依次显现，既区分了年份维度，又明确了类别层级，动画逻辑与数据结构完全匹配。
2. characteristics of mushrooms（蘑菇分类散点图）：三层嵌套分组与常量复用机制发挥了关键作用。动画严格按照“表面特征→气味→是否可食用”的层级顺序推进，sort 配置确保了各维度按预设顺序（如表面特征的“Smooth→Scaly→Fibrous”）展示，600ms 统一动画时长使整体视觉流畅，成功将多维分类数据通过动画时序转化为可直观感知的信息层级。
3. event plan（甘特图）：分阶段动画与动态参数绑定效果符合预期。坐标轴→刻度标签→任务文本→任务条的时序推进，符合视觉认知逻辑；offset 绑定 eventStartTime 实现了任务文本与任务条的时间同步触发，duration 绑定 eventDurationTime 使任务条动画时长与实际任务周期成正比，完美还原了甘特图的时间线关系，未出现位置错位或时序混乱。

二、Canis 核心功能实践总结

1. 多层分组机制：支持 1-3 层嵌套分组，可精准匹配多维数据的层级结构（如年份 - 类别、表面 - 气味 - 可食用性），通过 groupBy 指定分组维度、reference 控制时序关系、delay 设置间隔，实现了复杂数据的有序动画叙事。
2. 参数复用与动态绑定：constants 配置实现了动画时长的统一管理，修改一处即可批量调整，大幅提升了配置维护效率；offset 与 duration 支持绑定数据属性（如 eventStartTime、eventDurationTime），使动画从“固定效果”升级为“数据驱动”，实现了动画与数据的深度耦合。
3. 分阶段时序控制：通过 reference: "start after previous" 实现了多动画对象的顺序执行（如甘特图的四阶段渲染），避免了多元素动画的重叠混乱，同时 minOffset、minDuration 等参数为动画提供了安全边界，确保了视觉连贯性。
4. 动画与 SVG 的协同：实验验证了“SVG 预设位置 + Canis 控制显示”的核心逻辑，只要原始 SVG 文件中元素的 x、y 等位置属性准确，Canis 动画仅控制显示时机、效果与时长，不会出现位置偏差，二者协同保障了可视化的准确性与美观性。