可视化实验二报告

201900161140 张文浩

实验完成时间：9.20

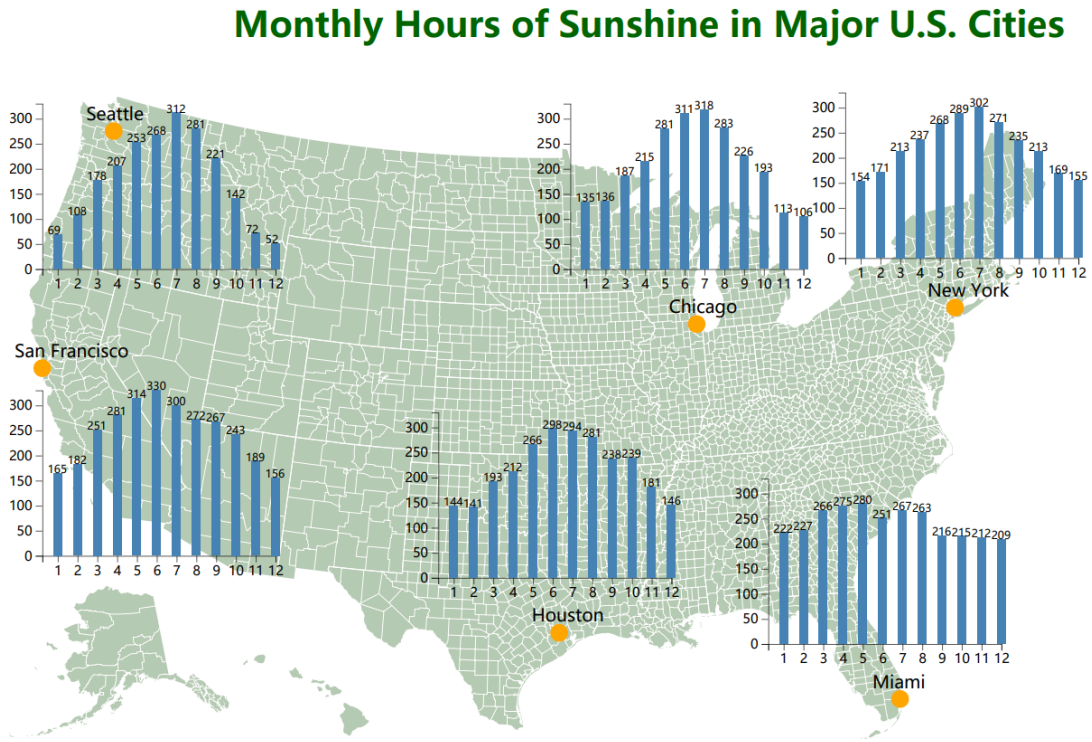
软件环境：vscode

一、实验任务:

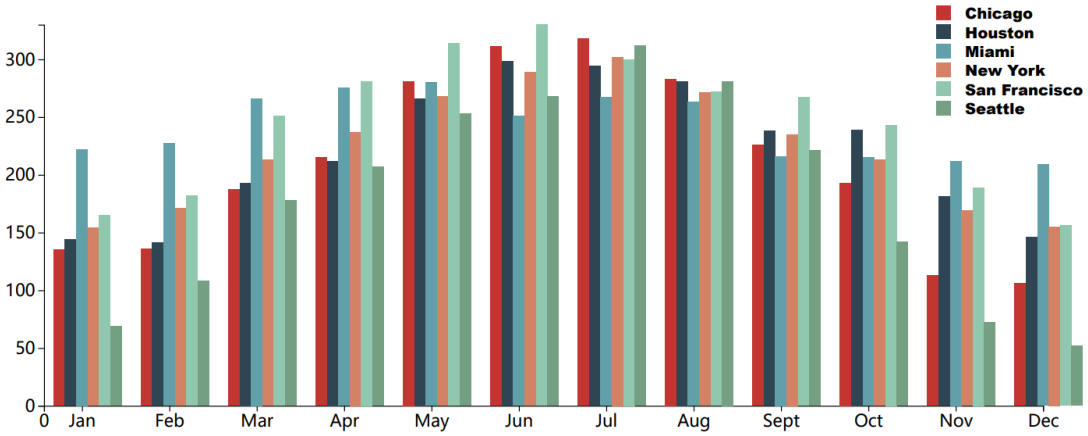
可视化数据集美国主要城市每月光照时长（Monthly Hours of Sunshine in Major U.S. Cities）

二、可视化效果

图一:



图二：



三、实现方法

思路：通过观察 sunshine.csv 数据集，我了解到我想要可视化的数据集包括城市、经纬度坐标、月份和光照时长这些属性标签。相比于上一个实验来说，这个数据集包含的信息量比较大，维度比较高，想要明确清晰的表达比较困难。

在尝试找到合适的图像类型的时候，我发现城市的月份和光照时长这三个关系比较好表达，但经纬度的信息很难清晰的表示，及时通过折线图或者散点图表示出来，对于看图的人来说也不够直观。

最终我想到的办法是先绘制一张美国地图，再将月份和光照时长的信息通过绘制柱状图 bar chart 的方式直接表示在地图上，这样做的好处就是可以非常直观的表示出六个城市之间地理位置的关系和差异。

做完这张图后我发现这张图不能很好地对比城市之间光照时长的差异，于是我又绘制了第二张 bar chart 图，来更好的对比每个月每个城市之间光照时长的差异。

图一:

绘制美国地图:

```
                req=new XMLHttpRequest();
                req.open("GET",'https://raw.githubusercontent.com/no-stack-dub-sack/testable-projects-fcc/master/src/data/choropleth_map/for_user_education.json',true);
                req.send();
                req.onload=function(){
                json=JSON.parse(req.responseText);

                var svg = d3.select("svg");
                var path = d3.geoPath();

            d3.json("https://raw.githubusercontent.com/no-stack-dub-sack/testable-projects-fcc/master/src/data/choropleth_map/counties.json", function(error, us) {
                if (error) throw error;

            //绘制美国地图
            svg.append("g")
                .attr("class", "county")
                .selectAll("path")
                .data(topojson.feature(us, us.objects.counties).features)
                .enter().append("path")
                .attr("d", path)
                .style('fill', (d) => {
                            return ("rgba(12, 81, 0, .3)")
                        }//end for loop that checks for matching county IDs
                    //end of loop through ALL json data
                )//end style - fill

            //绘制美国边界
            svg.append("path")
                .attr("class", "county-borders")
                .attr("d", path(topojson.mesh(us, us.objects.counties, function(a, b) { return a !== b; })));
            //绘制各州边界
            svg.append("path")
                .attr("class", "state-borders")
                .attr("d", path(topojson.mesh(us, us.objects.states, function(a, b) { return a !== b; })));
```

通过画小圆和添加文字的方式标识出六个城市的位置：

（以 Chicago 为例）

```
//Chicago
svg.append('circle')
    .attr('cx',625)
    .attr('cy',220)
    .attr('r',8)
    .attr('fill',"orange")
svg.append('text')
    .attr('x',600)
    .attr('y',210)
    .text("Chicago")
```

输入数据集：

```
var dataset1 = [135,136,187,215,281,311,318,283,226,193,113,106]
var dataset2 = [144,141,193,212,266,298,294,281,238,239,181,146]
var dataset3 = [222,227,266,275,280,251,267,263,216,215,212,209]
var dataset4 = [154,171,213,237,268,289,302,271,235,213,169,155]
var dataset5 = [165,182,251,281,314,330,300,272,267,243,189,156]
var dataset6 = [69,108,178,207,253,268,312,281,221,142,72,52]
var maxx = 330
```

定义比例尺：

```
var rectWidth = 8
//柱状图比例尺
var linear = d3.scaleLinear().domain([0, maxx]).range([0,150])
//x 轴的比例尺
var xScale = d3.scaleOrdinal()
.domain(["",1,2,3,4,5,6,7,8,9,10,11,12])
.range([60,74,92,110,128,146,164,182,200,218,236,254,272])
//y 轴的比例尺
var yScale = d3.scaleLinear()
.domain([0,maxx])
.range([150, 0]);
//定义 x 轴
var xAxis = d3.axisBottom()
.scale(xScale)
//定义 y 轴
var yAxis = d3.axisLeft()
.scale(yScale)
```

绘制六个城市的 bar chart 柱状图：

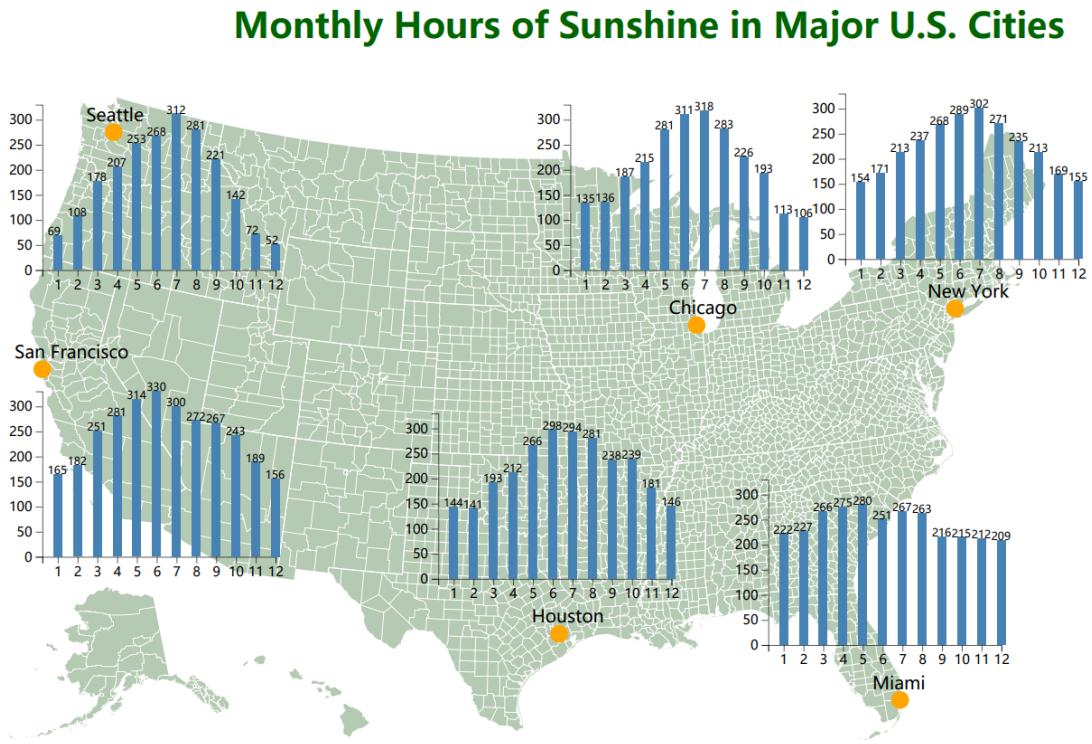
（以 Chicago 为例）：

```
//Chicago
            //添加 x 轴
            svg.append("g")
            .attr("class","axis")
            .attr("transform","translate(" + dx1 + "," + (420+dy1) + ")")
            .call(xAxis);
            //添加 y 轴
            svg.append("g")
            .attr("class","axis")
            .attr("transform","translate(" + (60+dx1) + "," + (270+dy1) + ")")
            .call(yAxis);

            //设置柱形图元素

            var rec1 = svg.selectAll("rect1")
                .data(dataset1)
                .enter()//选中只有数据没有元素的部分
                .append('rect')//由于没有元素所以要添加元素
                .attr('fill','steelblue')//设置颜色
                .attr('x',(d,i)=>i*(rectWidth+10)+70+dx1)//设置横坐标,+3 是设置柱
形图之间的间隙
                .attr('y',(d)=>420-linear(d)+dy1)//反转
                .attr('width',rectWidth)
                .attr('height',(d)=>linear(d))
            //添加文字元素
            var text1 = svg.selectAll("t1")
                        .data(dataset1)
                        .enter()
                        .append("text")
                        .attr("font-size",10)
                        .attr("transform","translate(" + dx1 + "," + (dy1-
20) + ")")
                        .attr("x", (d,i)=>i*(rectWidth+10)+65)
                        .attr("y",function(d){
                        return 420-linear(d);
                        })
                        .attr("dx",function(){
                        return 0;
                        })
                        .attr("dy",function(d){
                        return 20;
                        })
```

```
                    .text(function(d){
                    return d;
                    })
```

图一绘制结束。

效果：



图二：

图二绘制就比较简单了，绘制的思路与实验一的第一个 bar chart 一样，画出六组 rect，再通过调整“x”属性让它们正好错开。

添加比例尺，定义 x，y 轴：

```
    //柱状图比例尺
    var linear = d3.scaleLinear().domain([0, maxx]).range([0,400])
    //x 轴的比例尺
    var xScale = d3.scaleOrdinal()
        .domain(['0','Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sept','Oc
t','Nov','Dec'])
        .range([40,80,172,264,356,448,540,632,724,816,908,1000,1092])
    //y 轴的比例尺
    var yScale = d3.scaleLinear()
        .domain([0,maxx])
```

```
        .range([400, 0]);
    //定义 x 轴
    var xAxis = d3.axisBottom()
        .scale(xScale)
    //定义 y 轴
    var yAxis = d3.axisLeft()
        .scale(yScale)
    //添加 x 轴
    svg1.append("g")
      .attr("class","axis1")
      .attr("transform","translate(" + 0 + "," + (height-30) + ")")
      .call(xAxis);
    //添加 y 轴
    svg1.append("g")
      .attr("class","axis1")
      .attr("transform","translate(" + 40 + "," + 20 + ")")
      .call(yAxis);
```

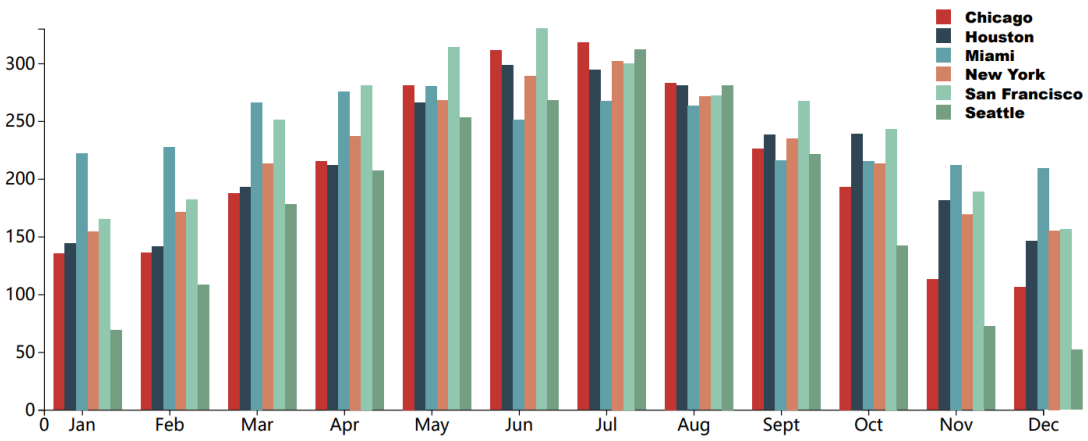
绘制六组 rect：

```
    //设置柱形图元素
    var rectWidth = 12
    var rec1 = svg1.selectAll("rect1")
                   .data(dataset1)
                   .enter()//选中只有数据没有元素的部分
                   .append('rect')//由于没有元素所以要添加元素
                   .attr('fill','#c23531')//设置颜色
                   .attr('x',(d,i)=>i*(rectWidth + 80)+50)//设置横坐标,+3 是设置柱形图之间的间隙
                   .attr('y',(d)=>height-linear(d)-30)//反转
                   .attr('width',rectWidth)
                   .attr('height',(d)=>linear(d))
    var rec2 = svg1.selectAll("rect2")
                   .data(dataset2)
                   .enter()//选中只有数据没有元素的部分
                   .append('rect')//由于没有元素所以要添加元素
                   .attr('fill','#2f4554')//设置颜色
                   .attr('x',(d,i)=>i*(rectWidth + 80)+62)//设置横坐标,+3 是设置柱形图之间的间隙
                   .attr('y',(d)=>height-linear(d)-30)//反转
                   .attr('width',rectWidth)
                   .attr('height',(d)=>linear(d))
    var rec3 = svg1.selectAll("rect3")
                   .data(dataset3)
                   .enter()//选中只有数据没有元素的部分
                   .append('rect')//由于没有元素所以要添加元素
                   .attr('fill','#61a0a8')//设置颜色
```

```
                .attr('x',(d,i)=>i*(rectWidth + 80)+74)//设置横坐标,+3 是设置柱形图之间的间隙
                .attr('y',(d)=>height-linear(d)-30)//反转
                .attr('width',rectWidth)
                .attr('height',(d)=>linear(d))
var rec4 = svg1.selectAll("rect4")
                .data(dataset4)
                .enter()//选中只有数据没有元素的部分
                .append('rect')//由于没有元素所以要添加元素
                .attr('fill','#d48265')//设置颜色
                .attr('x',(d,i)=>i*(rectWidth + 80)+86)//设置横坐标,+3 是设置柱形图之间的间隙
                .attr('y',(d)=>height-linear(d)-30)//反转
                .attr('width',rectWidth)
                .attr('height',(d)=>linear(d))
var rec5 = svg1.selectAll("rect5")
                .data(dataset5)
                .enter()//选中只有数据没有元素的部分
                .append('rect')//由于没有元素所以要添加元素
                .attr('fill','#91c7ae')//设置颜色
                .attr('x',(d,i)=>i*(rectWidth + 80)+98)//设置横坐标,+3 是设置柱形图之间的间隙
                .attr('y',(d)=>height-linear(d)-30)//反转
                .attr('width',rectWidth)
                .attr('height',(d)=>linear(d))
var rec6 = svg1.selectAll("rect6")
                .data(dataset6)
                .enter()//选中只有数据没有元素的部分
                .append('rect')//由于没有元素所以要添加元素
                .attr('fill','#749f83')//设置颜色
                .attr('x',(d,i)=>i*(rectWidth + 80)+110)//设置横坐标,+3 是设置柱形图之间的间隙
                .attr('y',(d)=>height-linear(d)-30)//反转
                .attr('width',rectWidth)
                .attr('height',(d)=>linear(d))
```

添加注释和图例……

效果:



四、实验总结

本次实验前期准备工作做的还是挺多的，主要是在思考如何直观的表达经纬度坐标信息的一步，我也去 D3 的官网上看了很多 example，但没有找到类似的数据集，但看到了有关地图绘制的 example 给了我灵感，于是最终想到了在地图上绘制 bar chart 的方案，虽然这样的方式也有很多不足，比如每个 bar chart 相对较小、不能很清晰地比对城市之间数据的差异等问题，但至少这是我能想到的最好的表达方式了。为了弥补第二个问题，我又绘制了第二张图来表达。最终的效果我自己还是比较满意的。

通过本次实验我对 D3 语言更加熟悉，通过学习官网上的 example 探索了很多表达方法。在实验过程中也学到了很多知识，比如为了绘制图一，我去网上学习了绘制地图的方法。同时也在实践中发现了自己在编程方面的问题，比如代码比较冗余，不如六个 bar chart 的绘制完全可以用一个 for 来缩减代码长度，提高程序可读性，但需要更多的参数设定，这一点我也会在不断地实践中不断改进。