



下载APP



33 | 数据处理（一）：可视化数据处理的一般方法是什么？

2020-09-14 月影

跟月影学可视化

[进入课程 >](#)**讲述：月影**

时长 13:33 大小 12.42M



你好，我是月影。

在数据处理的过程中，我们经常遇到两种情况：一种是数据太少，我们没法找到有用的信息，也就无法进行可视化呈现。另一种是数据太多，信息纷繁复杂，我们经常会迷失在信息海洋中，无法选择合适的可视化呈现方式，最终也表达不了多少有意义的内容。

那你可能想问了，想要解决这两种情况，我们能用上节课讲过的三种数据处理方法吗？事实上，上节课的方法是数据可视化的基本方法论，你可以在可视化过程中借鉴它们的思路，但是它们并不系统。



因此，我们在探索数据可视化的时候，还需要一个合理的数据可视化分析过程作为参照。从这一节课开始，我们就来系统地讨论数据处理的一般方法。

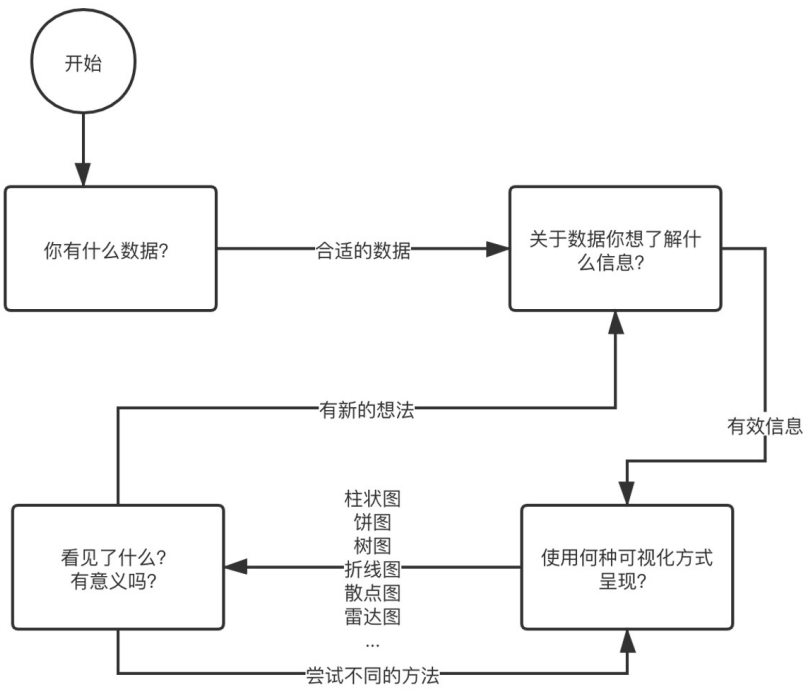
数据可视化的一般过程

针对课程一开始这两种情况，就算是不学数据处理的一般思路，我们也知道，如果你的数据太少，你要想办法获取更多的数据，而如果你的数据太多，那你就需要学会运用正确的方法不断迭代、筛选。而且，数据过多的情况我们遇到得更多。

当你学会在众多复杂的数据中准确地抽取信息，把这些数据的某一面可视化出来的时候，你就已经能够轻松地从中得到你想要的内容。通过这个过程，有可能让你从数据的一面获得启发，从而发现数据其他方面的有趣内容，进而产生出更多不同的图表，让数据呈现出更多的意义。

所以在数据可视化中，你有什么样的数据其实是最重要的，而我们的可视化手段会随着具体数据集的不同而不同。因此，我们一般会围绕 4 个问题对可视化过程进行迭代，它们分别是你有什么数据？你想从数据中了解什么信息？你想用什么样的可视化方式呈现？以及你看到了什么，它有意义吗？并且，前一个问题的答案都会影响后一个问题的答案。

结合这几个问题，我把数据可视化的一般过程用一个流程图画了出来。接下来，我们就分别说说每一步具体是怎么操作的。



可视化迭代过程

你有什么数据？

我们在实际处理可视化数据的时候，经常会遇到可视化项目最终产出的结果与设计预期和客户的期望相差甚远。

比如说，产品经理经常会提出一些需求，可能是要我们实现某种复杂的可视化大屏，或者指定一些竞品中的图表形式让我们模仿，有时候甚至会在数据给到我们之前先完成 UI 设计。可是，等到我们最终拿到数据之后，却发现数据的信息不足以支撑这些图表的展现形式，或者数据的内容在这些图表上表现得不好。

其实这不是产品经理能力不足、也不是可视化工程师能力不足，而是我们一开始就搞错了步骤，我们应该先分析真实数据，找到数据的特点，比如我们可以按照时间、地点、性别对数据进行分类。然后，我们再研究具体的数据呈现形式。因此，做可视化项目的第一步，就是要先了解自己掌握的数据，而不同的数据要了解的内容不同，我们要根据实际情况来具体分析。

你想从数据中了解到什么信息？

在得到并且了解原始数据之后，接下来我们该怎样着手获取想要的信息呢？我们以第 27 节课用到的 GitHub 贡献数据为例。当时我们想实现 GitHub 用户代码贡献图表，GitHub 贡献数据已经被平台加工过，平台已经把每个用户的贡献信息整理出来了，所以这类数据非常简单，我们直接用就可以了。

不过，我们可以想象一下，GitHub 贡献数据的原始数据长什么样。我认为，它应该包含了每个 GitHub 代码仓库提交的日志，并且每一条日志对应一个提交人和一个提交时间。要把这些数据整理成 GitHub 用户代码贡献数据，GitHub 平台需要对这些原始数据按日期和提交人进行归总，统计出每个用户在具体某个日期的代码提交次数。

假设，现在我们不想要统计每个用户每天的代码贡献数据，而是要统计每个代码仓库每天的代码贡献数据。那么，我们只需要对原始数据换一种方式处理，将日志不按照提交人进行归总，而是按照提交的仓库进行归总。这样我们就可以得到数据的另一面，从而得到一个完全不同的可视化内容，也就是代码仓库每日的代码贡献次数。

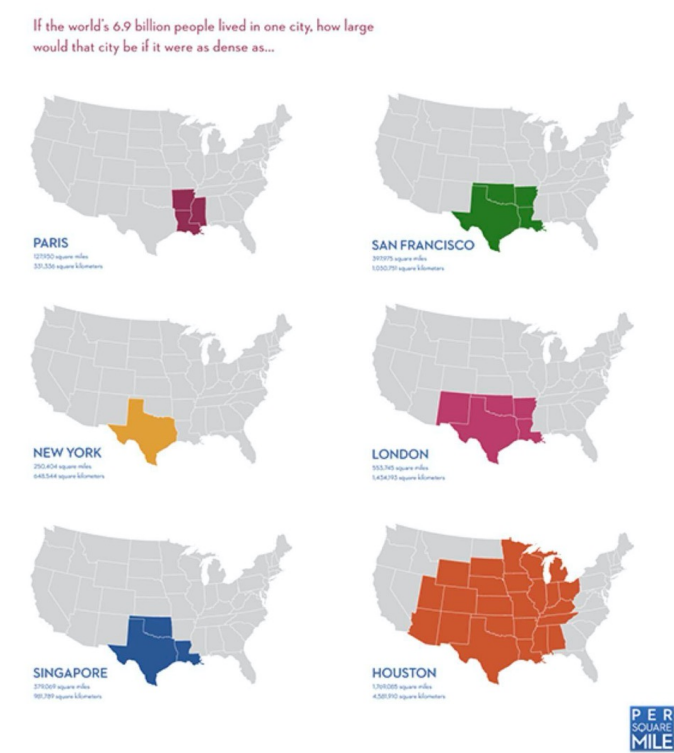
所以，当我们确定了想要表达的信息之后，如果数据中有，我们就直接拿来用，如果没有，我是需要自己来处理的。

使用何种可视化方式呈现？

当确定了想要了解的信息之后，会有很多视觉呈现方式供我们选择。那我们该如何选择呢？核心原则就是一条：当为数据选择正确的呈现形式时，我们要学会从不同的角度思考数据，把握住最希望被用户关注到的那些内容，然后用直观的、有趣的以及更加印象深刻的方式来进行表达。

你可能会说，道理我都懂，可具体该怎么做呀？

其实具体怎么做，并没有定论，我们只能通过例子慢慢积累经验，随着经验的丰富，你就能慢慢找到设计合适的可视化方式的感觉。比如说，现在你就可以想想，我在下面给出的这个例子。例子中给出的数据图表是 Tim De Chant 的世界人口地图。其中一共有 6 个城市，每个城市的人口密度不同，如果我们要把全球 70 亿人都放在任意密度的城市中，这个城市会有多大呢？



我的建议你可以好好想到底怎么做，有机会的话一定要去实践。在实践的过程中，当你尝试不同的标尺、颜色、形状、大小和几何图形的时候，就会看到可以值得探索的视觉呈现形式了。另外，给数据增加直观性和趣味性，也能够让朴实的数据立刻生动起来。这也是实践你学过的视觉呈现技术的最好时机。

你看见了什么？它有意义吗？


可视化数据之后，我们需要从中挖掘出一些有价值的信息，包括但不限于数据信息的规律（增加、减少、离群值等），以及数据信息的异常（明显差异、规律改变）等等，然后将这些规律和异常归因，帮助我们真正了解数据背后有价值的内容。这些正是可视化的意义所在。

客观的数据是有说服力的，因为客观数据一旦产生就不会改变，所以，我们一般认为这样的数据就是事实。但有用的信息往往隐藏得比较深，因此，数据可视化最终目的就是将这信息挖掘出来，呈现在人们眼前。

到这里，数据可视化的一般过程我们就讲完了。总之，在对数据进行四个步骤的迭代过程中，不同的数据以及不同的数据处理方式，在迭代中会产生不同的呈现形式。为了让你更深刻地理解这一点，我就以上节课游乐场的的数据为例，带你体验一次数据处理的全过程。

实战演练：对公园中的游客进行数据可视化

首先我们来看我们有什么数据。我们的原始数据的格式就像下面记录的一样，有时间、地点和性别。

 复制代码

```
1 [{
2   "x": 456,
3   "y": 581,
4   "time": 12,
5   "gender": "f"
6 }, {
7   "x": 293,
8   "y": 545,
9   "time": 12,
10  "gender": "m"
11 }, {
12  ...
13 }]
```

接下来我们看一下我们想了解什么，假设我们想了解公园一天中的**游客变化规律**，那么我们可以用分类的思路处理数据。在上节课，我们就对数据进行了简单的**时间分类**、**地点分类**和**性别分类**。这里，我们用 d3 的数据变换（Transformations）模块将原始数据处理成我们想要的模式。

这里，我详细说说我们用到的 `d3.rollups`，它可以对数据进行分组，然后汇总。这个接口设计得比较函数式 (functional)，它接受 3 个参数，第一个参数是要处理的数据，也就是上面的原始数据，后面两个参数是两个函数算子，第一个算子表示对数据分组进行汇总的方式，这里是使用 `length` 来汇总，也就是统计数据的条目数。第二个算子则表示对数据进行分组的属性，这里是用时间属性进行分组。最后，我们在分组之后，再对数据进行一次排序，因为我们要按照时间从小到大进行排序。

[复制代码](#)

```
1 (async function() {  
2   const data = await (await fetch('data.json')).json();  
3   const dataset = d3.rollups(data, v => v.length, d => d.time).sort(([a],[b])  
4     ...  
5 }());
```

经过分组和排序之后，我们从原始数据得到了如下的新数据：

[复制代码](#)

```
1 [[8, 145], [12, 141], [18, 191], [20, 23]]
```


现在，我们只有 8 点、12 点、18 点、20 点这 4 个时间段的数据，我们还需要把游客为 0 的时间信息补全。假设公园是早晨 6 点开门，晚上 22 点关门，那么 6 点、22 点的游客数应该是 0，补充的数据如下：

[复制代码](#)

```
1 dataset.unshift([6, 0]);  
2 dataset.push([22, 0]);
```


我们补全的数据是一个二维数组，其中每个元素的第一个值是时间，第二个值是当前时间的公园内人数。

接着，我们就要进行第三步了：确定用哪种可视化方式呈现数据。因为要呈现游客的变化规律，所以我们最终决定使用折线图来呈现，那我们就要把数据转换成要显示的折线上的点坐标。

 复制代码

```
1  const points = [];  
2  dataset.forEach((d, i) => {  
3    const x = 20 + 20 * d[0];  
4    const y = 300 - d[1];  
5    points.push(x, y);  
6  });
```

然后，我们用 SpriteJS 创建 Polyline 元素，把这个折线点坐标传给它。最后，我们把这个元素给添加到 layer 上，就可以将它显示出来了。

 复制代码

```
1  const p = new Polyline();  
2  p.attr({  
3    points,  
4    lineWidth: 4,  
5    strokeColor: 'green',  
6    smooth: true,  
7  });  
8  
9  fglayer.append(p);
```

因为还要考虑到游客是随时间变化的，所以我们要给它增加一个坐标轴。这里，我们是用 SpriteSvg 来绘制坐标轴的，SpriteSvg 是一个特殊元素，它能够创建一个 SVG 对象，然后把它以图像方式绘制到 Canvas 上。你发现了没有，这就是我们前面说过的 SVG 和 Canvas 的混合使用方式，它可以把 SVG 作为图像绘制，这样既能使用 SVG 来灵活地改变图形，又可以用 Canvas 来高性能地渲染。

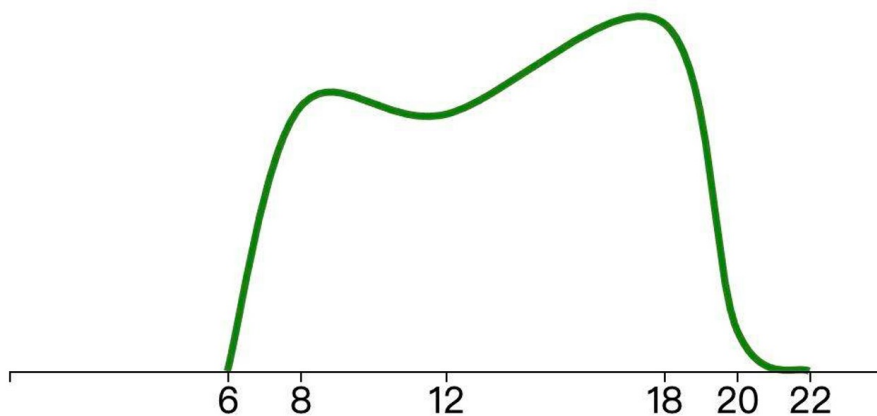
 复制代码

```
1  const scale = d3.scaleLinear()  
2    .domain([0, 24])  
3    .range([0, 480]);  
4  
5  const axis = d3.axisBottom(scale)  
6    .tickValues(dataset.map(d => d[0]));  
7  
8  const axisNode = new SpriteSvg({  
9    x: 20,  
10   y: 300,  
11   flexible: true,  
12 });  
13  
14 d3.select(axisNode.svg)
```

```
15 .attr('width', 600)
16 .attr('height', 60)
17 .append('g')
18 .call(axis);
19
20 axisNode.svg.children[0].setAttribute('font-size', 20);
21 fglayer.append(axisNode);
```

在创建坐标轴的时候，我们需要给坐标轴设置一个 scale，d3 中间应用了很多函数式编程思想，这里的 scale 也是一个函数算子，它是由高阶函数 `d3.scaleLinear` 创建的，我们给它设置 domain 从 0 到 24，表示一天的 24 个小时，range 从 0 到 480，表示占据 480 像素宽度。

然后，我们再通过 `d3.axisBottom` 高阶函数，用创建的 scale 来生成一个具体的坐标轴算子 `axis`，然后对 SVG 对象应用这个算子，就可以绘制出坐标轴的图像了。最终的效果如下图所示：



公园24小时游客人数变化图

最后，我们能从这张图上看出来什么信息呢？在这张图上，我们看到公园的游客数有两个高峰期，分别是早 8 点和傍晚 6 点，而 12 点的游客是比较少，晚上 8 点之后到 10 点闭园之前游客的数量是最少的。

这样，我们就得到了一天中游园人数的变化趋势，这对公园的管理策略是有一些参考价值的。这个数据还很粗糙，因为我们的原始数据的信息量并不大。不过如果我们继续收集不同天的数据，并进一步分析游客在不同日期、不同地点的人数情况，或者对游客的性别进行分析，我们就可能得到更多有趣的信息，但究竟能获得什么样的有用信息，还要看原始数据情况和我们实际着手进行迭代的方式。

总之，当我们把更多的信息集中在一起的时候，我们就能做更多的事情了，比如，我们可以分析游客数和当月平均气温的关系，或者分析交通趋势和公园游客趋势的关系，以及分析天气与游园游客性别的相关性等等，并且我们还能利用这些数据来帮助公园后续的建设和管理决策。

要点总结

这节课，我们讲了数据可视化的一般过程，它是一个需要反复迭代的过程，而且在每一轮迭代中，我们都可以以这 4 个问题作为参照，分别是你有什么数据？你想从数据中了解什么信息？你想用哪种可视化方式呈现？你看见了什么，它有意义吗？

按照这四个问题，我们在迭代过程中可以用分类的方式来处理数据，对数据分类，是最常用的处理原始数据的方式，也符合我们的思维习惯。数据的分类可以带来信息结构化，从而帮助我们提取想要的内容。也就能一步步达成我们想要的结果，然后我们再逐步细化迭代，就能做出更好的可视化效果来。

最后，我还想再强调一下，从本质上来说，可视化过程是对数据进行分析、提取有效信息、设计展现形式的不断迭代过程，今天我们讲的例子虽然简单，但更加复杂的例子，也无外乎是同样的处理过程，只不过那个时候，我们可能需要处理更多的数据，经过更多轮迭代。

小试牛刀

我们的原始数据里有性别数据，你可以根据性别数据进行过滤，修改上面的例子，看看按照性别分类的游客趋势图与不按性别分类的游客趋势图有什么不同吗？或者，除了折线图，你能把每个时间段的游客性别比例用饼状图显示出来吗？以及，你还可以试着按照公园地点进行分组，再显示更细分的饼图效果。

欢迎在留言区和我讨论，分享你的答案和思考，也欢迎你把这节课分享给你的朋友，我们下节课见！

源码

🔗 课程中完整示例代码 [GitHub 仓库](#)

提建议

更多课程推荐

数据结构与算法之美

为工程师量身打造的数据结构与算法私教课

王争

前 Google 工程师



立省 ¥40 🖱️

破 90000 订阅特惠，到手价 ¥89

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 32 | 数据之美：如何选择合适的方法对数据进行可视化处理？

下一篇 34 | 数据处理（二）：如何处理多元变量？

精选留言

写留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。