# ECharts: 一个用于快速构建基于网络的可视化的声明性框架✩

## 李德清a, 梅红辉b, 沈毅a, 苏爽a, 张文利a, 王俊廷a, 祖明a, 陈伟b,\*。

a 百度公司，中国

b 浙江大学CAD&CG国家重点实验室，中国

概要：

虽然已经有十几个用于可视化设计和开发的创作系统和编程工具包，但不具备编程能力的用户，如数据分析师或界面设计师，仍然会觉得有效地实现基于网络的可视化很麻烦。

在本文中，我们提出了ECharts，一个开源的、基于网络的、跨平台的框架，支持快速构建交互式可视化。其动机是由三个目标驱动的：易于使用，丰富的内置交互，以及高性能。ECharts的内核是一套声明性的视觉设计语言，可以定制内置的图表类型。底层的流媒体架构，加上基于HTML5画布的高性能图形渲染器，使得ECharts具有很高的扩展性和性能。我们报告了ECharts的设计、实现和应用，并列举了各种各样的例子。我们将ECharts与C3.js、HighCharts和Chart.js的效用和性能进行了比较。实验的结果证明了我们框架的效率和可扩展性。自2013年6月首次发布以来，ECharts已经迭代了63个版本，在GitHub上吸引了超过22,000个星星计数和1700多个相关项目。ECharts被认为是世界上领先的可视化开发工具，在GitHub的可视化标签中排名第三。

© 2018 由Elsevier B.V.代表浙江大学和浙江大学出版社出版。这是一篇在CC BY-NC-ND许可下的开放存取文章

## 1. 绪论

随着数据的增长，人们对数据的呈现和分析有了迫切的需求（Wang等人，2016），引发了数据可视化的快速构建工具。虽然已经有十几个用于可视化设计和开发的创作系统和编程工具包（Mei等人，2018），但对于用户来说，如数据分析师或界面设计师，要快速实现基于网络的交互式可视化仍然很麻烦（Grammel等人，2010）。

最近的一个趋势是在图形用户界面（GUI）中实现可视化构建，而不需要文本编程（Satya- narayan和Heer，2014）。通常情况下，这些工具缺乏表达能力，特别是在指定交互方面。同时，图形语法（Ichikawa等人，2013）的设计对于浏览设计空间至关重要（例如，Lyra（Satyanarayan和Heer，2014）是建立在可视化语法Vega（Satyanarayan等人，2016）之上的）。

D3.js（Bostock等人，2011）和Vega（Satyanarayan等人，2016）等声明性语言是构建可视化的流行工具。随着底层数据传输的封装和控制流暴露在用户面前，这些声明性语言允许用户专注于可视化设计。然而，用户必须对网络开发非常熟练。例如，D3.js要求用户熟悉HTML、CSS、SVG和DOM。同样地，Vega要求用户掌握一套新的图形语法。这些要求使开发工作变得非同小可。

我们认为，视觉标识的灵活性和复杂性不应受到编程技能要求的限制（Heer等人，2008）。这项工作的基本动机是通过声明性对象选项和可组合的可视化组件来填补这一空白，这些组件是用用户可配置的声明性对象选项建模的。当用户创建可视化的时候，他们不需要精通网络相关的编程。相反，他们只需要花几分钟时间熟悉所提供的可视化组件，并通过指定数据、可视化编码、注释和可视化风格来配置这些组件。

在这里，我们贡献了ECharts，一个易于使用的框架来构建交互式可视化。主要的贡献是确认了三个目标。

易于使用。如果采用陈述性语言，用户在学习视觉表达方面会有一些困难。我们希望能让用户专注于可视化的设计，而不是使用一些工具。

丰富的内置交互。高效的数据探索和分析需要大量的可配置的交互。ECharts设计并实现了丰富的内置交互，这些交互附属于每个图表类型，最大限度地减少了用户定制的要求。

高性能。通过引入流式系统结构和增量渲染模式，ECharts实现了高性能，即使在处理数百万个数据点时也是如此。

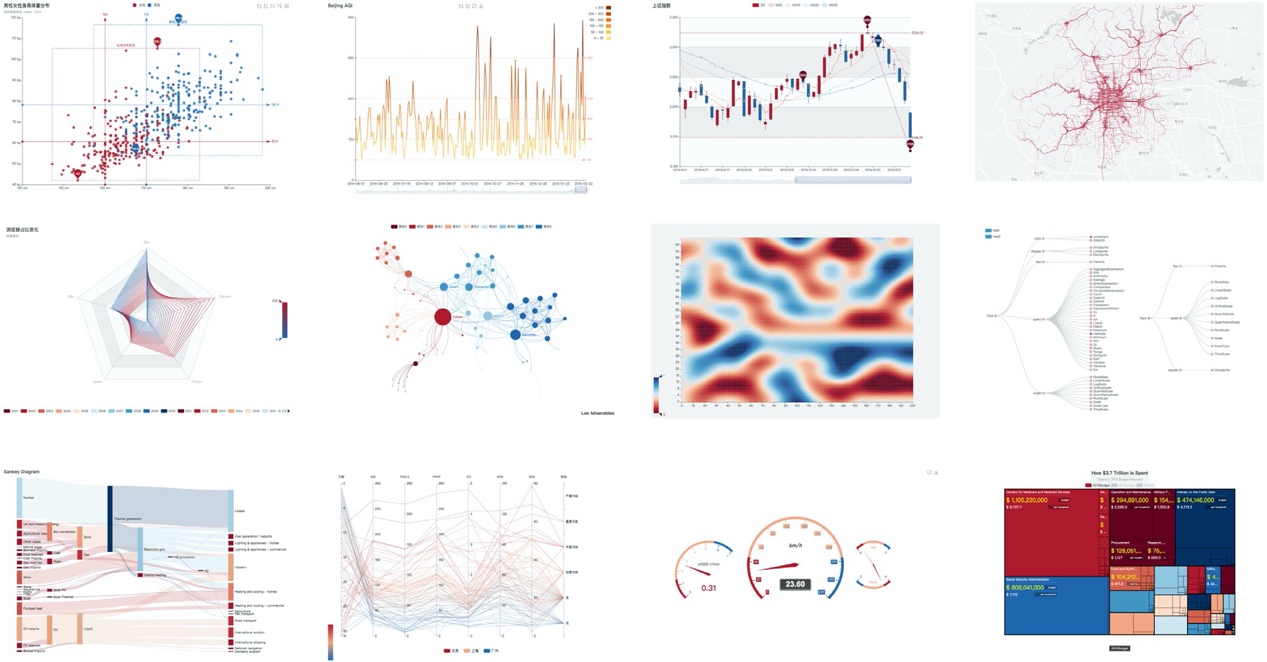


图1. ECharts图表类型的例子。从上到下，从左到右：散点图、折线图、烛台图、地理图、雷达图、节点链接图、热图、树状图、桑基图、平行坐标、仪表图、树状图。

## 2. 相关工作

2.1. 视觉化的语法

图表工具，如Excel（Microsoft Excel, 2017）和ManyEyes（Viegas等人，2007）支持通过从预定义的可视化模板中选择适当的形式来快速生成图表。这种方案的一个主要缺点是，可视化的表现力完全被提供的模板所约束。

Wilkinson (Wilkinson, 2005)介绍了The Grammar of Graph- ics，用于更广泛的图形规范和 "顺应图表类型学"。它具有深远的影响，形式语言被设计用来描述生成图形的规则。遵循这一思想，许多软件和框架的实现为用户带来了更多的定制，如Tableau（Tableau soft- ware，2017）（前身是Polaris（Stolte等人，2002）），ggplot2（Wick- ham，2009，2010），和ggivs（ggvis 0.4 overview，2017）。一些可视化框架对底层图形绘制进行抽象，以实现更简洁的规范，如InfoVis Toolkit（Fekete，2004）、Improvise（Weaver，2004）、prefuse（Heer等人，2005）和Flare（Flare，2017）。在这些框架中，引入了可继承的可视化部件或可组合的操作符来减轻用户的负担。另一方面，那些用于信息可视化的声明性、特定领域的语言（DSL），包括ProtoVis（Bo-stock和Heer，2009；Heer和Bostock，2010）、D3.js（Bostock等人，2011）和Vega（Satyanarayan等人，2016），允许用户通过直接将数据映射到视觉元素来指定可视化，而无需计算细节。在此基础上，Vega-lite（Satya- narayan等人，2017）对数据模型、图形标记、视觉编码和其他详细的规范进行了更多的抽象。这种高层次的抽象通过利用省略低层次细节的部分规范和用默认值解决含糊不清的问题，能够快速构建视觉形式。

然而，复杂的视觉设计总是与简单性和效率相对立。ECharts的设计试图在快速构建和富有表现力的视觉设计之间取得一个良好的平衡。

2.2. 图形库

随着网络技术的飞速发展，如Cas-cading Style Sheets（CSS）、Java applets、JavaScripts和AJAX，越来越多的应用程序通过网站部署。对于可视化工具来说，基于网络的方法提供了一个简单的解决方案，支持跨平台的部署，并使合作者和受众之间能够轻松分享和交流。这些网络工具使用户专注于核心问题（Mwalongo等人，2016），无需维护。

早期基于网络可视化的构建工具利用Java和Flash。Processing（Processing，2017；Reas和Fry，2003，2005）在其第一版中提供了对Java小程序的支持。Flare（Flare, 2017）是一个用于创建可视化的ActionScript库，在Adobe Flash Player中运行。后来，可视化构建工具转向使用JavaScript和可扩展矢量图形（SVG），包括Raphaël（Raphaël-JavaScript Library，2017）、JavaScript InfoVis Toolkit（JIT）（JavaScript InfoVis Toolkit，2017）、ProtoVis和D3.js。从本质上讲，SVG可以产生质量最好的二维图纸，而JavaScript在指定用户交互时，对DOM元素接收到的用户输入事件提供了灵活而易用的回调。

尽管它很方便，但使用SVG的性能很低，因为所有SVG元素的DOM结构以及相应的样式和事件都需要在网络浏览器中维护。另一种方案是利用HTML5画布进行2D绘图。例如，在Vega（Satyanarayan等人，2016）中使用画布，比SVG实现了2倍至4倍的性能提升。其他工具，如iVisDesigner（Ren等人，2014年）、iVolVER（Méndez等人，2016年）和Data-Driven Guides（Kim等人，2017年）采用画布来实现高性能。然而，由于 HTML5 画布不支持单个图形对象的交互规范，使用画布的工具必须设计自己的事件处理机制。

ECharts 实现了一个名为 ZRen- der 的二维矢量绘图库，它支持在 HTML5 画布中显示视觉形式，并管理图形元素、渲染以及事件。ZRender的细节可以在第5.2节找到。

2.3. 视觉设计的定制

最流行的基于网络的可视化设计引擎，D3.js，通过将链接数据引导到原生的网络表现形式，即SVG，充分挖掘了网络表现的能力。这导致了高度的表现力。然而，当视觉设计包含复杂的视觉映射到SVG路径元素的''d''处时，用户仍然会感到很麻烦。为此，D3.js提供了模块来封装常见的视觉任务，包括形状、比例、布局和交互。这为用户提供了一个灵活的选择，但当高层和低层的声明混合在一起时，仍然会造成混乱。Vega克服了这个问题，它将复杂的形式建模为基本形状的组合，而这些形状是由预定义的布局安排的。

最近提出的数据驱动指南（DDG）（Kim等人，2017）通过数据指南机制支持方便地创建图表和灵活的视觉设计。这些指南是简单的形状，数据被映射到其属性之一，例如，一系列平行线，其长度被绑定到数据属性之一。基于这样的指南，设计者可以在数据指南的基础上绘制SVG图形。

为了在保持简单性的同时提供灵活性，ECharts提供了一个特殊的组件系列，允许用户通过改变其渲染过程来修改预定义的图表。

在创建可视化的过程中，最复杂的任务之一是指定用户的交互方式。一个常见的解决方案是附加事件监听器。事件监听器是回调函数，当指定的事件发生时被调用，并对用户输入做出反应。在此基础上，D3.js引入了称为行为的事件处理程序，以实现交互技术的重用。在Vega中，事件被作为信号从输入流中提取，并触发谓词，可以影响视觉映射。另一方面，反应式编程方法（Satyanarayan等人，2014）避免了可怕的 "回调地狱"，但对新手来说很难设计。同样地，Vega-Lite允许用户通过转换预先设定的交互来简单地定义一个新的交互。然而，Vega-Lite只支持选择操作。

在ECharts中，丰富的交互被自动附加到生成的图表上，而且新的交互仍然可以通过一个易于使用的事件系统来指定。

## 3. 声明性的可视化设计

ECharts采用了一体化的JSON格式选项来声明组件、样式、数据和交互，从而形成了一个无逻辑和无状态的模式。JSON格式的主要优势在于，它可以安全地存储、传输和执行，并且容易做进一步的验证。

遵循微软Excel等知名工具的惯例，ECharts使用系列来抽象出一组由数据映射和编码的图形元素。例如，在一个直角坐标系中的所有线条或条形图构成一个系列。同样，饼状图、树状图、图表或其他图表类型也可以被抽象为一个系列。换句话说，一个系列是一种类型的图表的实例。此外，ECharts使用组件来命名功能单元，如数据缩放、视觉编码和工具箱。

ECharts提供了一个方法setOption来创建或更新组件和系列。如果setOption是为了更新数据，ECharts采用了一种基于键的差异算法来寻找数据的差异，并使用适当的过渡来显示它。

3.1. 声明性选项

声明性选项是一个分层的JSON对象。在顶层，所有组件、系列和全局设置都被声明。特别是，全局设置，如调色板、字体和动画，是通用的设置，在不同的系列和组件上都有。ECharts检查顶层的键是否被注册为一个组件，然后创建所有的组件。否则，这些值将被设置为全局设置。

像图例、工具提示、刷子、visualMap等组件，都是可选的。不同的组件可以为不同的可视化目的而组成。

在组件层面上，属性的配置取决于相关的组件，包括布局、样式和状态。ECharts组件中的大多数状态都存储在全局选项中，并在用户交互发生时同步改变。在获得了包括组件状态在内的当前全局选项后，该视图可以很容易地在另一个环境中重复。这对于调试、重放和自动测试非常有用。

还有一个系列字段，包含一个或多个系列的数据集和它们的图表类型。大多数的图表类型都有一个坐标系统，比如笛卡尔、极地和地理系统。一个坐标系统也被定义为顶层的一个组件。对于一个系列，有三种方法来索引相应的坐标系：索引、ID和名称，这对于索引任何其他组件都是常见的。

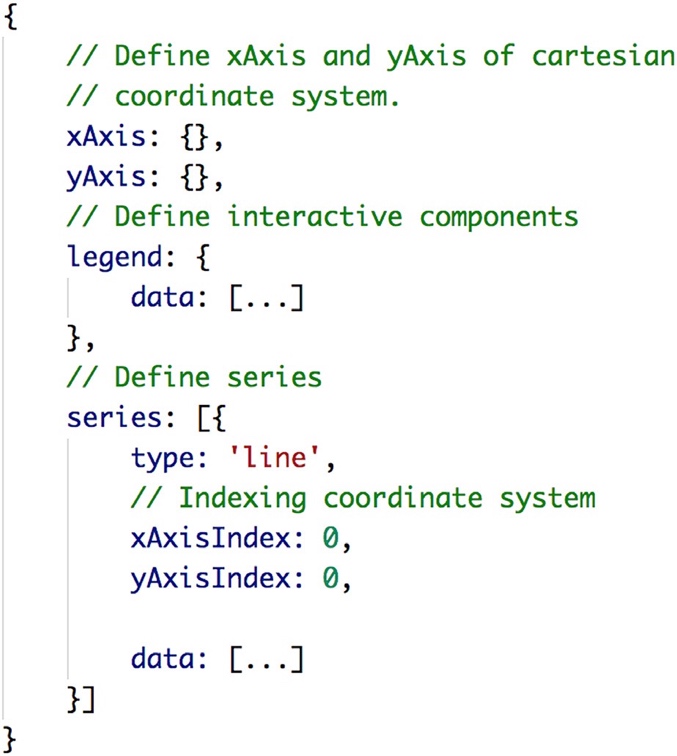


图 2. 基本option的结构。

3.2. 图表类型

为了适应各种情况，ECharts提供了多种图表类型，可以分为三类，包括内置、定制和扩展。

具体来说，内置图表类型为创建最常用的图表提供了方便；如果需要特定的布局或图表类型，而内置图表类型没有提供，用户可以使用自定义系列来指定布局；扩展模式作为制作自定义图表类型的替代品，对渲染过程和交互有更多的控制。

一般来说，内置模式是最简单的一种。另外，如果一个图表类型可以在其他情况下重复使用，我们建议将其制定为扩展模式。尽管如此，用户化模式具有最大的灵活性。

3.2.1. 内置图表类型

内置的图表类型包括一般用途的图表类型，如散点图、线形图、条形图、饼图、地理图和烛台图。

ECharts支持19种内置图表类型。其中一些显示在图1中。

3.2.2. 定制的图表类型

有了定制的图表类型，用户只需要专注于渲染逻辑，而不需要实现诸如创建或释放图形元素、过渡动画或视觉地图等细节。第4.2节介绍了关于定制过程的更多细节。

图3显示了自定义图表类型的例子。

3.2.3. 扩展图表类型

ECharts提供了一个扩展机制来支持添加新的功能（第4.3节）。图3(e)显示了一个标签云扩展的例子。  
3.3. 坐标系统

在ECharts中，坐标系可以在声明性对象模型中进行配置。基本的坐标系是笛卡尔坐标系和极坐标系。前者可以通过配置xAxis和yAxis来使用；而后者则用：polar、radiusAxis和angleAxis来指定；这两种坐标系都可以接受两个以上的轴，它们由索引来识别。ECharts笛卡尔坐标系统支持三种轴：值、类别和时间。

ECharts提供了两种特殊的坐标系统：日历和地理地图。此外，ECharts自动将数据中的时间或空间信息映射到相应的坐标系中。这些系统可以接收针对特定位置的任意图形元素。这意味着，其他图表，如饼状图，可以在坐标系统内绘制。

3.4. 组件

ECharts中的每个组件都包含三种类型的功能：视觉编码、引导和互动。相应的模块被集成到流水线的不同阶段。一些组件，如tooltip和markPoint，侧重于引导。有些组件，如dataZoom，则侧重于交互。大多数组件负责两个以上的功能。

3.4.1. 视觉编码

每个图表都是数据驱动的视觉编码的组合，比如条形图中的高度、散点图中的位置、饼图中的面积等等。视觉编码组件被用于此目的。像颜色、颜色亮度、大小、字形这样的视觉通道是可组合的，可以从不同层次的数据中对这些组件进行编码。例如，一个图例组件对不同系列的颜色进行编码。在此基础上，visualMap组件可以修改颜色亮度以表示每个系列的数值比较。特别是，如果包括刷子组件，颜色的alpha通道被用来编码数据选择状态。

大多数可视化编码组件是交互式的，比如用图例组件选择系列，用visualMap组件过滤范围内的数据。

3.4.2. 指南

指南功能为可视化提供描述性的信息。组件可以使用标签、引导线、字形来标出特定的项目或显示附加信息。

ECharts中的大多数组件都扮演着引导组件的角色。坐标系统使用标签和刻度线来显示数据的范围并帮助阅读数值。图例使用带有编码颜色的标签来显示视图中每个元素的系列是什么。VisualMap可以作为一个色条使用。

3.4.3. 交互

ECharts提供了一组交互式组件来支持交互式操作，如：平移、缩放、选择。特别是，图例组件可以取消选择不必要的系列。数据缩放组件可以沿着指定的维度过滤数据，刷新组件可以在指定的区域内突出选择的数据。每个交互组件都会触发一个事件；这些事件在连接多个图表实例中是必要的。当需要对数据进行统计或显示详细信息时，事件中的属性如选定的数据指数很有用。