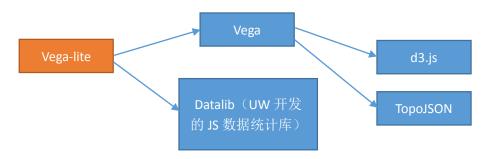
# Vega-lite 调研

### 简介

Vega-lite(https://github.com/uwdata/vega-lite)是一款基于 Vega 的高层抽象标记式语言,用于简化 Vega 的繁琐语法。官网提到当前 Vega-lite 尚处于活跃开发阶段,5 日前尚有 Github 代码更新。相比于 Vega 的大段 JSON 描述,Vega-lite 简化了可视设计部分,使用类似模板的方式定义可视化形式,用户仅需设定数据输入、维度映射等接口,即可输出可视化结果。

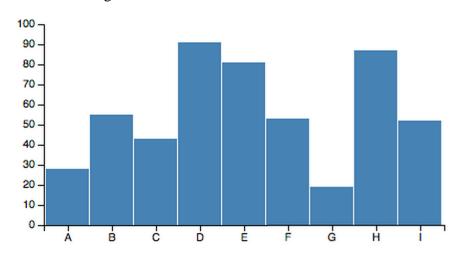
Vega-lite 依赖 UW 开发的一系列可视化库,其依赖关系如下:



## 语法

Vega 的描述是以基本几何形状为最小单位,通过绑定数据生成对应图元。 Vega-lite 预制了一系列可视化模板,仅需输入数据和指定模板所需接口即可实现 同样效果,相当于 Vega 的封装。

下面展示了使用 Vega-lite 生成简单 bar chart 的过程 (摘自官方示例):



```
{
 "data": {
   "values": [
                                                                 数据定义,其中属性
     {"x":"A", "y":28}, {"x":"B", "y":55}, {"x":"C", "y":43},
                                                                x 为类别型, y 为数
     {"x":"D", "y":91}, {"x":"E", "y":81}, {"x":"F", "y":53},
                                                                值型。
     {"x":"G", "y":19}, {"x":"H", "y":87}, {"x":"I", "y":52}
                                                          指定可视化形式为
 "marktype": "bar",
                                                          bar chart。这是与
 "encoding": {
                                                          Vega 最大的不同,直
   "y": {"type": "Q", "name": "y"},
                                    指定 bar chart 的横
                                                          接封装了 bar chart
   "x": {"type": "0", "name": "x"}
                                    纵坐标绑定关系
                                                          模板。
 }
}
```

其对应的原始 Vega 代码为:

```
"width": 400,
                                                                        画布设置
 "height": 200,
 "padding": {"top": 10, "left": 30, "bottom": 20, "right": 10},
 "data": [
   {
     "name": "table",
     "values": [
       {"x":"A", "y":28}, {"x":"B", "y":55}, {"x":"C", "y":43},
                                                                        输入数据
       {"x":"D", "y":91}, {"x":"E", "y":81}, {"x":"F", "y":53},
       {"x":"G", "y":19}, {"x":"H", "y":87}, {"x":"I", "y":52}
     ]
   }
 ],
 "scales": [
   {"name":"x", "type":"ordinal", "range":"width",
"domain":{"data":"table", "field":"data.x"}},
                                                                             Scaling 与
   {"name":"y", "range":"height", "nice":true, "domain":{"data":"table",
                                                                             轴设置。
"field":"data.y"}}
                                                                             这里轴是
 ],
                                                                             visible的。
 "axes": [
   {"type":"x", "scale":"x"},
   {"type":"y", "scale":"y"}
 ],
 "marks": [
     "type": "rect",
```

```
"from": {"data":"table"},
   "properties": {
     "enter": {
       "x": {"scale":"x", "field":"data.x"},
                                                                指定矩形图元, 生成
       "width": {"scale":"x", "band":true, "offset":-1},
                                                               Bar。Vega-lite 将这一
       "y": {"scale":"y", "field":"data.y"},
                                                                大段封装为 bar chart
       "y2": {"scale":"y", "value":0}
                                                                类型模板。
     },
     "update": { "fill": {"value":"steelblue"} },
     "hover": { "fill": {"value":"red"} }
   }
 }
]
```

#### 编辑器

UW 还开发了一套在线编辑器来即时展示 Vega-lite 代码的结果。



直接在 Vega-lite 代码区输入代码,即可在右侧显示出可视化结果。

## 总结

- 1. 从设计哲学角度考虑: UW 工具系偏向于 UNIX 哲学,每个小工具都只完成一个简单的任务并完成得十分扎实;小工具之间使用一定方式通信。工具前后继承关系明显,这样有利于尽快出阶段性成果,对科研来说是一种比较好的模式。VisComposer 更偏向 Windows 哲学,所有功能都包含于一个强大的GUI下,虽然与UW工具系殊途同归,都能够为用户提供强大的可视化工具,但前期成本略高。
- 2. 从可视化贡献角度考虑: UW 工具系较朴素,仅实现相应基础功能。事后看 Lyra 发表 EuroVis,先下手为强的因素占主要地位。
- 3. 从功能角度考虑:

- a) 可能可以借鉴 Vega-lite 对 Vega 的封装策略,对我们的可编程模块部分进行一定的设计和包装。这部分或许能提炼、设计出一个形式化的表达,而不是当前直接写 JS 代码的形式,加强论文中这部分的贡献。不过工作量方面值得斟酌。
- b) VisComposer 的学习成本比 UW 工具系中的可视化工具高,这点可能需要想办法改进,比如减少交互动作等。
- 4. 从工程角度考虑: VisComposer 暂时比不过。