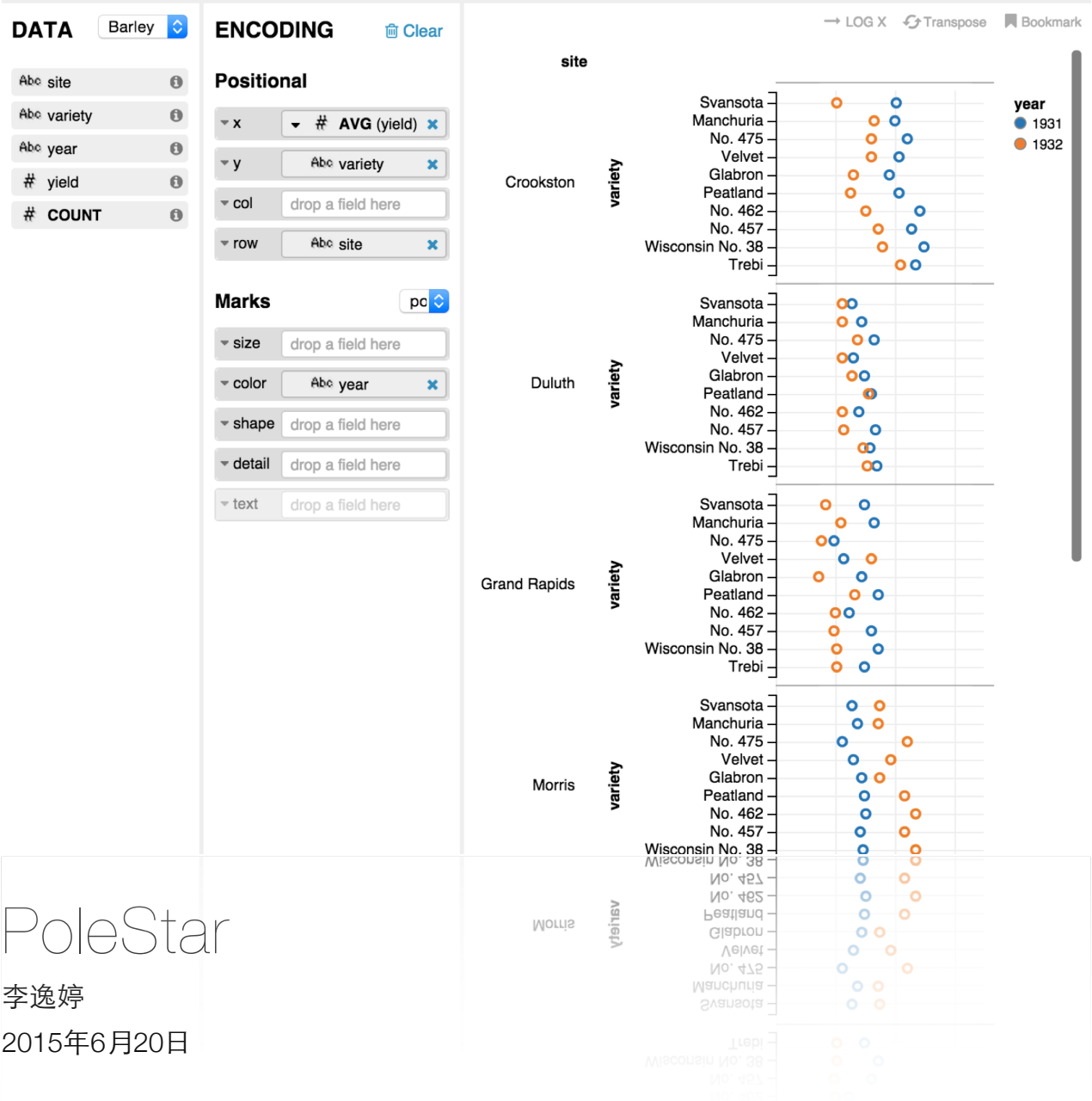


PoleStar

李逸婷

2015年6月20日

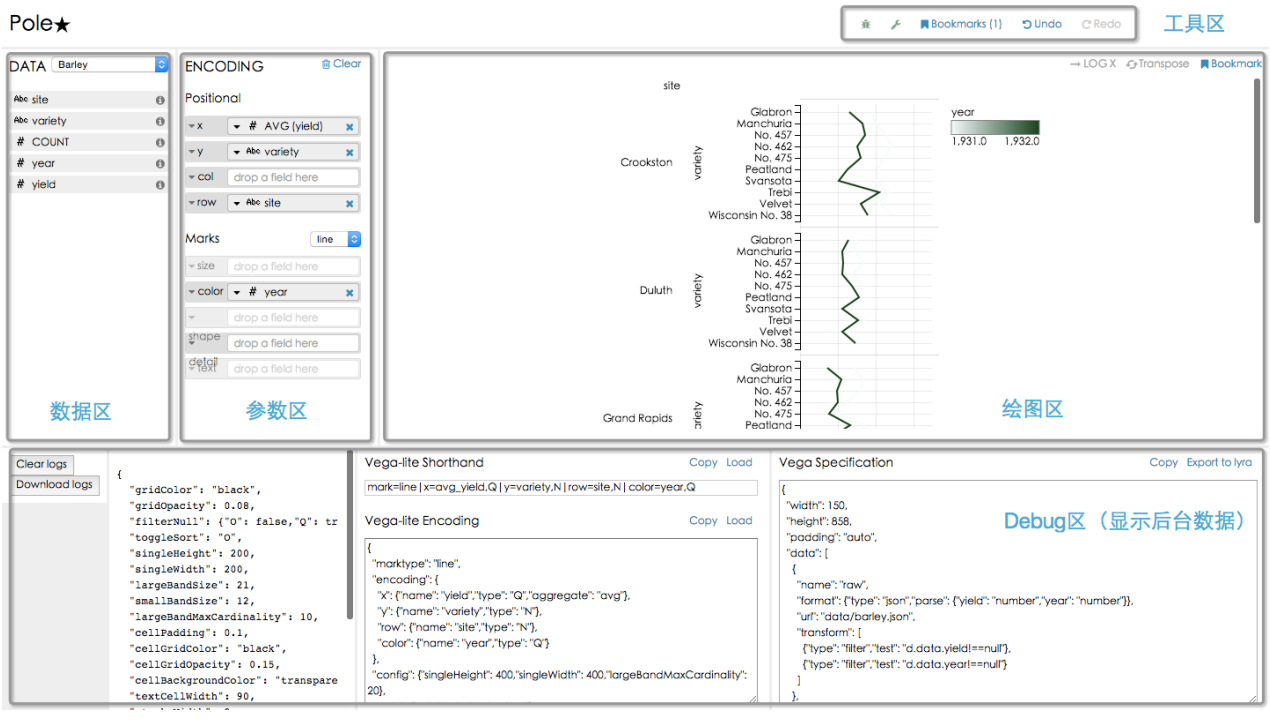
POLESTAR简介

功能定位

POLESTAR是基于web的可视化接口，受Tableau的启发。数据分析者可以快速地通过POLESTAR来建立视图，作为数据挖掘处理过程中的一部分。

在我的理解中，Polestar的主要定位就是在快速上面，所以界面功能非常简洁浅显上手很快。在数据可视化上的功能可能没有那么强劲，但是通过快速地对于数据建立不同地视图，可以让启发分析者对于数据之间关系的有很快的把握和理解。

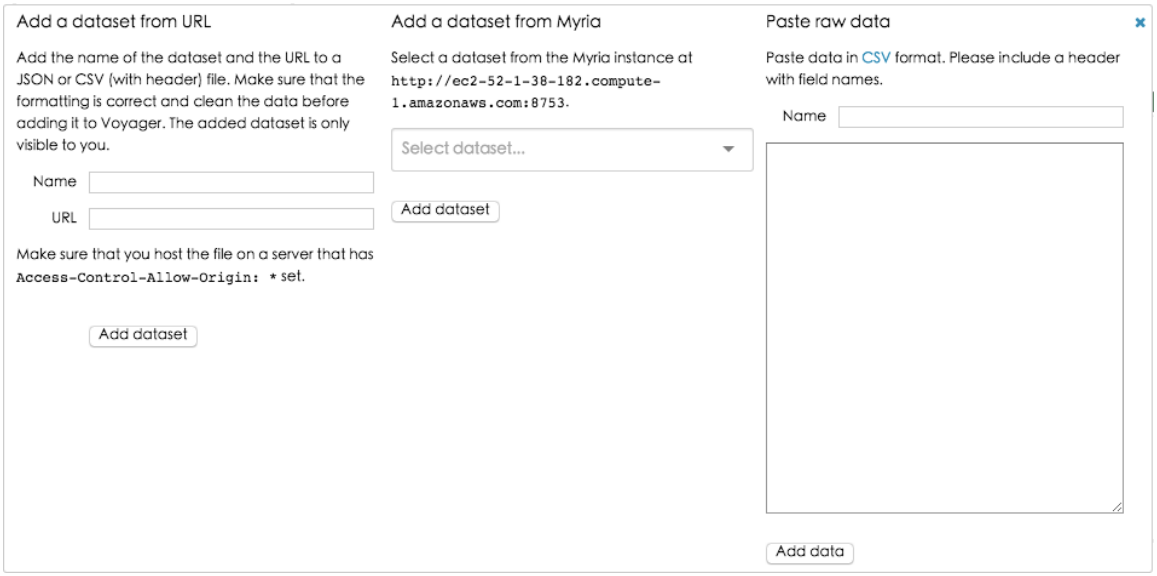
界面



使用方法

1. Data Section

数据部分Online Web上给出了一些示例的数据，格式都是JSON，当然也支持自己导入Data Set。



导入方式有三种：

- URL导入（支持JSON或CSV）
- Myria导入

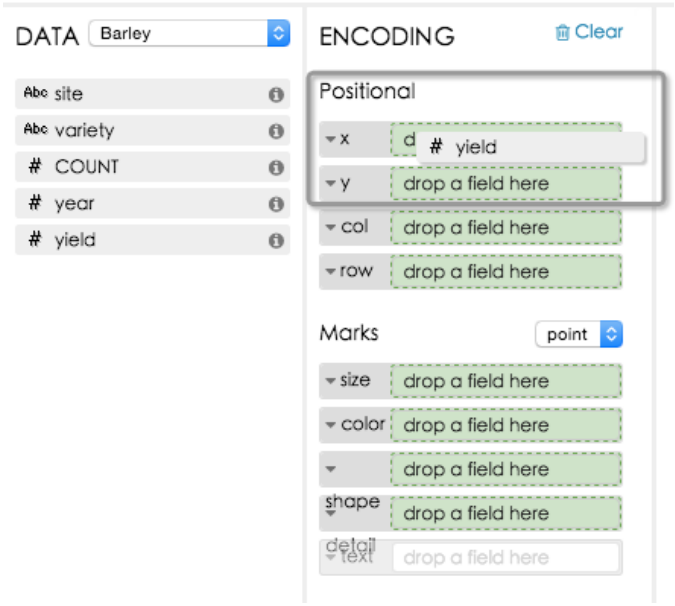
调查了一下Myria，也是华盛顿大学的一个项目，Myria是一个大数据的管理系统，可以做一些data的query之类的操作。系统本身就集成了一些数据集，如下如所示。这些数据集都可以导入到Polestar中，我觉得就是为和大数据接轨做的准备，通过Myria中处理过的一些大数据集可以导出到Polestar中可视化。

Datasets in Myria

Relation name	Creating query	Create time	Download
JustX	159	3 minutes ago	JSON CSV TSV
NumProfilings	86	13 days ago	JSON CSV TSV
TwitterK	4	17 days ago	JSON CSV TSV
TwoHopsInTwitter	132	4 days ago	JSON CSV TSV
powersOfTwo	85	13 days ago	JSON CSV TSV
Profiling	1	17 days ago	not available
Resource	3	17 days ago	not available
Sending	2	17 days ago	not available

- 黏贴导入（只支持CSV）

2. 参数区



（1）Position参数

Position下的每个参数可以从Data Section拖拽一个属性来绑定，如上图所示。Position可选的参数有以下几个：

- x x轴
- y y轴
- col 列属性
- row 行属性

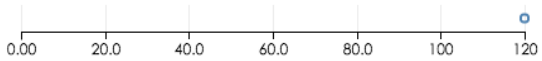
如果点Position参数前的下拉三角按钮，可以进一步地做一些设置：

A. scale(只对数值型的数据有用)

* reverse:该轴或行列上的数据标示顺序反一下，e.g.原来x轴0~120，现在则是120~0。

* type:数据标示类型。可选：linear/log/pow/sqrt。

e.g.**linear**：线性增长（均匀地）



* zero:数据标示是否从零开始标记，false时就适应最小值的数据，修改数据标示的起点。

B. axis

* format:规定数据格式

* grid:是否显示背景的栅格

* maxLabelL:最大标签数

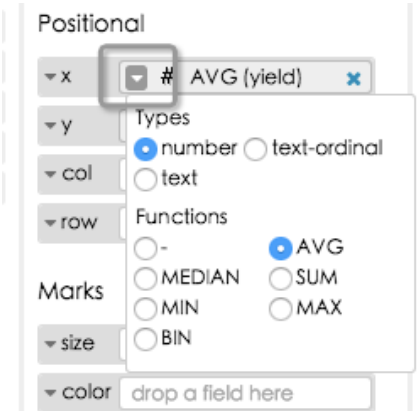
* title:是否显示坐标轴绑定的属性名称

* titleOffset:属性名称显示位置的偏移量

C. bin

* maxbins:最大聚类数

如果点击拖拽进来的的数据属性之前的下拉三角，也可以对数据的格式以及func做一些设定



A. Types

* number：数值型

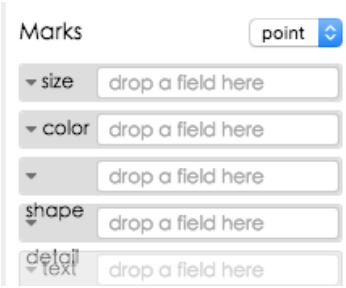
* text-ordinal：

* text：字符串型

B. Functions

对数值做一些基本的转换，比如AVG（均值），MEDIAN（中位数），SUM（求和），MIN（最小值），MAX（最大值），BIN（聚类）

（2）标记参数



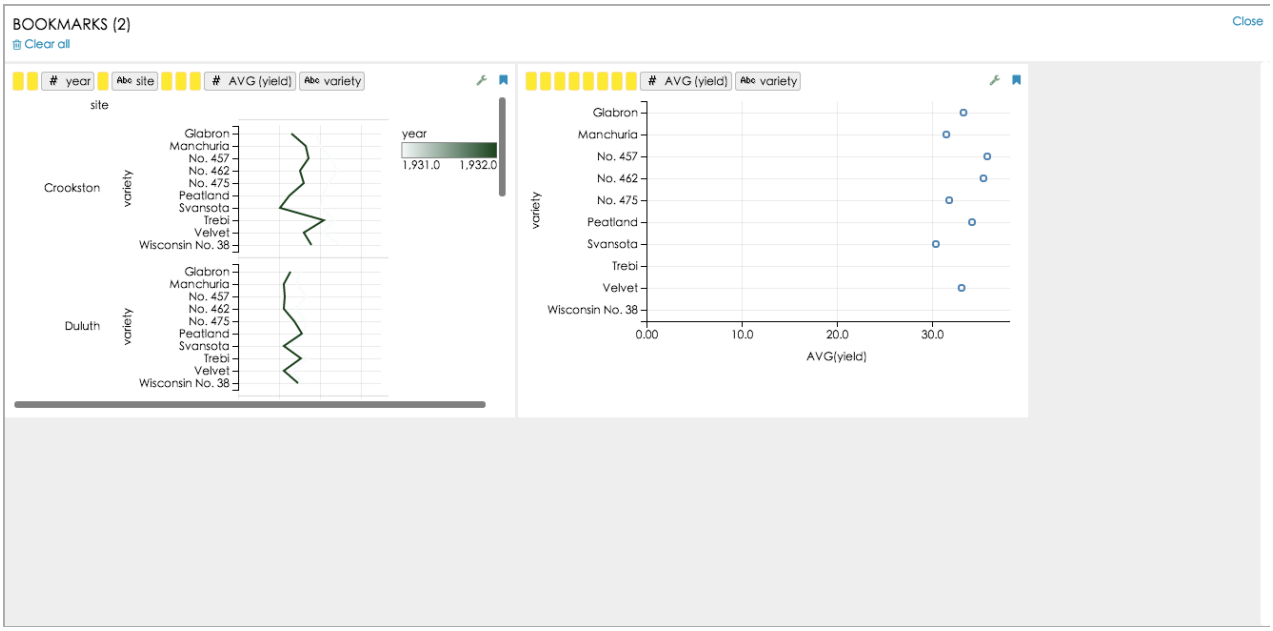
一共分为五个参数，

- size 改变标记的尺寸，数值参数
- color 改变标记颜色，可填写英文颜色名或者十六进制RGB
- shape 改变标记图形，可选circle/square/cross/diamond/triangle-up/down
- detail
- text 改变字体参数

3. 工具区

从左到右功能分别是：报告app错误，打开debug面板，书签，撤销，重做

所谓书签（Bookmarks）就是，可以保存一个制作好的视图放在localstorage里面，刷新页面或者重启浏览器，也会从缓存中取出。存储在Bookmarks中的数据，其实就是一些用来描绘渲染所必须的后台数据。随时可以根据这些Json数据重新绘制这些图。



4. 视图区

视图区除了显示绘制的图表以外，还有一些额外的功能：

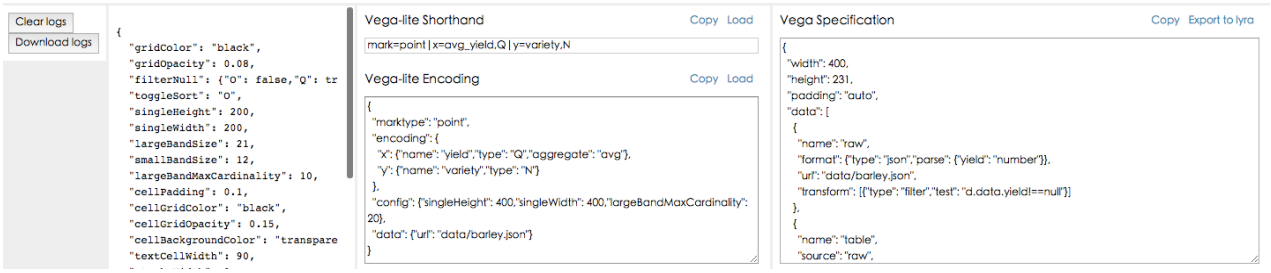
* LOGX：对于x坐标轴的进行log映射

* Sort：排序

* Transpose：x，y轴对换

* Bookmark：存为书签

5. debug后台数据区



这里其实显示的都是一些后台的数据，

其中包括

Vega-lite Shorthand	速记信息，记录关键信息
Vega-lite Encoding	Vega-lite的后台数据json编码
Vega Specification	Vega的图标描述json编码

由于PoleStar其实是使用的vega-lite的图形绘制的库，最后每一张图表都可以用json的方式来记录下来，并且通过这些数据可以再次绘制出来。所以通过修改这些数据，也可以直接修改图表。

并且通过这些数据，可以兼容vega，vega-lite，polestar，他们都是可以互相转换的，通过这些后台数据。其实也就是是一套东西。

技术手段

前端框架：Angular.js

Angular.js在写PoleStar的好处我觉得是：

Angular.js非常适合这个应用。它本身最大的特点就是数据双向绑定，我们在controller里面处理完了数据之后，前端也会同时自动更新数据。在这个应用中，我们经常要在前端处理很多信息，然后又要反馈显示出来，Angular.js能够很好地完成一个双向绑定的作用。不用去我们自己手动地更新很多东西。

其次，Angular.js的模块化编程也非常适合。因为在项目中所需要构建的模块非常多，Angular.js的MVC模式在PoleStar中中心app controller来调度所有模块，再分发给下级的功能模块controller去完成任务，结构清晰，逻辑简单。

依赖：

Datalib 数据处理

Vega-lite 绘制功能

Vega-lite-ui PoleStar和Voyager（另一款产品）的ui库

与VISCOMPOSER的分析比较

差异与优劣

我对VisComposer的接触时间不是很长，还不是特别了解。但是在我使用下来的感觉上来说。

打个比方，VisComposer的定位可能偏向于功能强大的IDE的感觉，而PoleStar只是一个有基础功能的编辑器。

可以说VisComposer的预期功能要比PoleStar强大的多。

VisComposer有自己模板，更多的图元，更多的表现形式。而PoleStar则没有，PoleStar只有一个栅格系统和至多二维的坐标系，也无法完成饼图这样的任务。VisoComposer有自己的复杂多变的工作管道，而PoleStar只是在给坐标和行列绑定属性，设置参数而已，基本的数据过滤功能也没有。

而且VisComposer不是纯交互的，可以半编程，这是最大的不同于PoleStar的地方。

PoleStar的定位在于“快速”，整个操作面板非常地简洁很容易上手，想要达到的目的就是，数据分析者可以通过PoleStar对于数据进行快速地探索性分析，能够通过简单的步骤去发掘处理数据之间的关系。如果需要更多的功能，完全可以使用debug面板中的vega的json数据，导出到vega或者lyra去做更强大的分析和可视化。

熟悉编辑器很简单，只要会打字就可以轻松地运用。但是熟悉IDE就不那么容易了，如何去将代码跑起来，如何debug都需要慢慢去学习IDE的那套体系。

作为一个刚刚接触VisComposer和PoleStar的人来说，无疑PoleStar上手难度低，逻辑简单，不看任何说明，没有提示的情况下我都可以很快速地去绘制我想要的图表。

然而VisComposer却需要去熟悉他的一套体系，比如scenegraph，workflow等等。很多功能在没有说明的情况下，很难知道怎么去达到我想要的目的。

所以从功能上来说VisComposer完胜，但是从易用性和上手难度上PoleStar完胜。

当然这也是由于两个项目的不同定位所造成的原因。

借鉴

研究完PoleStar我觉得在Viscomposer教程和提示上可以有很多改进的地方：

1. 最大的问题还是在应用使用的提示性和引导性上有很多可以改进的地方。

（1） 第一次打开网页的时候，可以做一个类似于新手引导一样的简单引导性过程，至少应该让使用者知道如何操作去创建一个新的视图，添加坐标轴等等。

（2） 有很多拖拽的过程，可以在拖拽的同时，给予一定的引导和提示信息。

比如我在拖拽Blank View的时候这时候Visualization的模板上可以弹出一个tooltip来提示我，Blank View应该是拖拽到这里。因为拖拽的操作很多，有时候很迷茫，不知道应该拖去哪里。

还有在workflow的时候我的接线的拖拽也最好能够将可连接的结点高亮出来，或者给予一些提示。

2. 如果要修改Scenegraph中结点的属性，必须先点进workflow，然后再去找到相应的panel去修改，这个最好可以能够在Scenegraph中就能够统一实现，比如右键结点可以弹出选择修改结点属性的弹框。

3. 利用好localstorage来做一些存储，不然如果浏览器错误退出的话，数据就直接没了。可以学习一下PoleStar的Bookmark的功能。

4. 最好还能够加上redo和undo的功能，这样从体验上来说会好很多。

5. 数据导入方式可以借鉴一下PoleStar的三种导入方式，可以接受URL或者黏贴的数据。最关键就是还可以接入大数据，这样可视化的价值就会更大。