Geographic Support For Vega-Lite

Youying Lin, Vivek Paramasivam

University of Washington Dept. of Computer Science & Engineering

Problem and Motivation

Vega is a declarative format for creating, saving, and sharing interactive visualization designs. Vega-Lite is a more concise, higher-level syntax designed to enable rapid creation of visualizations. A Vega-lite specification compiles into a Vega specification.

Currently Vega-Lite does not support cartographic projections on geographic coordinates. Meanwhile, Vega has geo and geopath transforms which can perform these projections but without a tile layer. To make Vega-Lite support projections with a tile layer, we want to add the tile layer at the Vega and Vega-Embed level. Then, we adapt the Vega-Lite to allow users to visualize geographic coordinates.

Vega, Vega-Lite, and Vega-Embed are developed by the UW Interactive Data Lab.

Approach

In this section, we will discuss the choices we made in implementing our changes, and the reasoning behind those choices.

Leaflet Plugin for Vega and Vega-Embed

Instead of implementing our own tile layers at the Vega level, we opted to use Leaflet, an "open-source javascript library for mobile-friendly interactive maps." Leaflet's rich variety of interaction features made it an optimal choice for this task. We opted to perform this implementation as a plugin because we wanted to make use of the Vega runtime API instead of modifying the actual Vega codebase. Synchronization between the Vega layer and the Leaflet layer was the biggest hurdle we needed to overcome. We solved this by outputting the center, translate, and scale from the geo and and geo-path transforms as signals which the Leaflet plugin can read and and adapt to accordingly. An example of this visualization can be seen in *Fiigure 3*.

Adding Geo Projections to Vega-Lite

The high-level goal for adding geo projections was simple: we modify Vega-Lite to allow users to note that a data element should be projected in a certain way. The actual implementation of this is actually fairly complex. In order to support this feature, the first step was to determine the Vega-Lite syntax which would produce a Vega specification that projects latitude/longitude data onto a euclidean plane. We explored a number of approaches to this, and decided on the syntax described in *Figure 1.* The resulting visualization from that spec is in *Figure 2*.

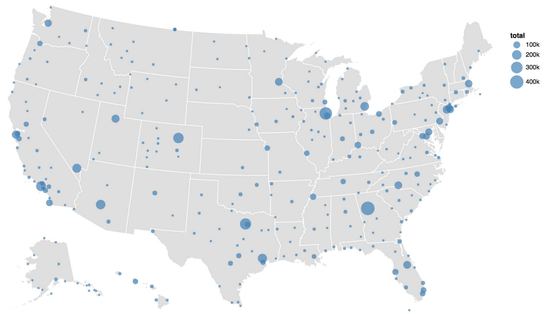
Approach (cont.)

The next step was to identify and modify the parts of the Vega-Lite codebase which needed to be modified in order to implement the changes we needed. A hurdle we ran into while performing these changes was that we had to perform some refactoring work on the current Vega-lite codebase to adapt the changes for the map projections part. Specifically, in order to use currently marks, such as points and lines, for geographic projections, we had to modify the code for each of those types of marks.

Figure 1

```
{
  "layers": [{
    "data":{
      "url":"data/us-10m.json",
      "format": {"type": "topojson", "feature": "states"}
    },
    "mark":"path",
    "projection":{
      "type":"albersUsa",
      "zoom":200,
      "translate":[120, 200],
      "center":[0, 0]
    },
    "encoding":{
      "geopath":{
        "field":"geodata",
        "type":"geojson"
      },
      "color": { "value": "#dedede"}
    },
    "config": { "mark": {"stroke": "white"} }
  }, {
    "data": {"url": "data/airports.csv"},
    "mark": "point",
    "projection": {
      "type": "albersUsa",
      "zoom": 200,
      "translate": [120, 200],
      "center": [0, 0]
    },
    "encoding": {
      "x": {"field": "longitude","type": "longitude"},
      "y": {"field": "latitude","type": "latitude"}
    },
    "config": { "mark": {"filled": true} }
  }]
}
```

Figure 2



The figure above was created by the Vega-Lite specification on the left, in Figure 1.

Future Work

There are a number of directions in which this work could move forward from here. One extension would be to add on-hover handlers for marks in Vega-Lite, which don't exist yet. Another natural extension is to implement auto-zoom, which would automatically adjust the translation, center, and zoom to fit given data. In addition, our work with Leaflet is not yet integrated with our work from Vega-Lite. Performing this merge is a necessary extension.

References

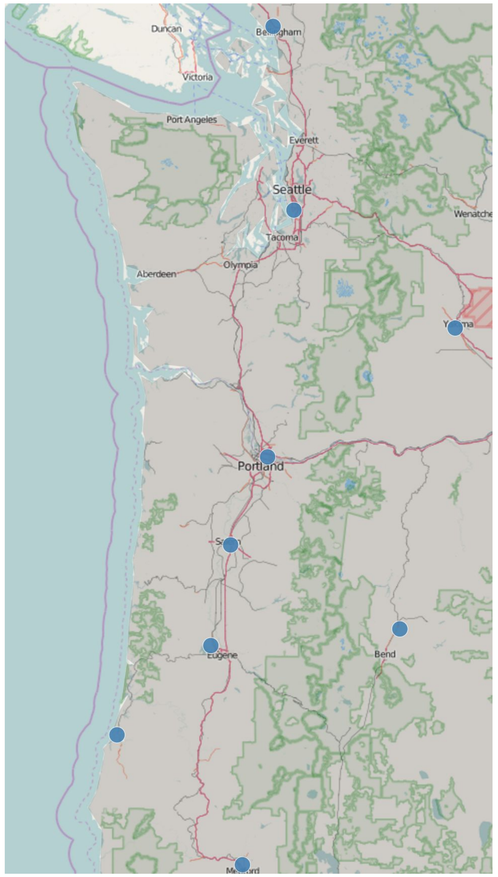
D3: Data-Driven Documents. Michael Bostock, Vadim Ogievetsky, Jeffrey Heer

Declarative Interaction Design for Data Visualization. Arvind Satyanarayan, Kanit Wongsuphasawat, Jeffrey Heer

Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization. Arvind Satyanarayan, Ryan Russell, Jane Hoffswell, Jeffrey Heer

Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, Jeffrey Heer

Figure 3



The figure above is an example of the kind of data augmentation provided by integrating a leaflet plugin with Vega.