Eurographics Conference on Visualization (EuroVis) 2017
J. Heer, T. Ropinski and J. van Wijk
(Guest Editors)

Volume 36 (2017), Number 3

Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images

Jorge Poco[1] and Jeffrey Heer[1]

[1]University of Washington

Appendix A: Vega Chart Generation

Generating Vega specifications

We first generate a large set of visual specifications using the Vega language. For that, we use the Compass library (part of Voyager [WMA*16]). Compass is a recommendation engine that receives a data file as input and it enumerates, ranks and prunes visualizations. The output of Compass is a collection of visual specifications in the Vega language. Figure 2(a) shows the Vega specification of the scatter plot in Figure 1. We use 11 real-world datasets from different domains – *e.g.*, *iris*, a common data in machine learning community; *movies*, a dataset of motion picture. The average number of records is 2,002 (min=16, max=10,000), and the average number of variables is 8.73 (min=4, max=25). In total we have 96 variables (46 nominal, 46 quantitative, and 3 temporal). These data files are freely available at `https://github.com/vega/vega-datasets`.

dataset	# variables			# records
	quantitative	nominal	temporal	
barley	1	2	–	120
crimea	3	1	–	24
iris	4	1	–	150
population	3	1	–	570
birdstrikes	4	9	1	10,000
campaings	9	15	1	58
cars	5	3	1	406
driving	3	1	–	55
jobs	3	2	–	7,650
movies	8	8	–	3,201
burtin	3	3	–	16

Table 1: ***Data Files use for Vega Char Generation***.

In Voyager, users specify which variables they are interested in, then Compass uses the selected variables to generate recommendations. Initially (with no selected variables), Voyager displays univariate summaries. Then, when variables are selected, Compass includes those variables in the generated visualizations. Our tool simulates users' selection of 1-3 data variables. In total, we generated 5,542 Vega specifications.

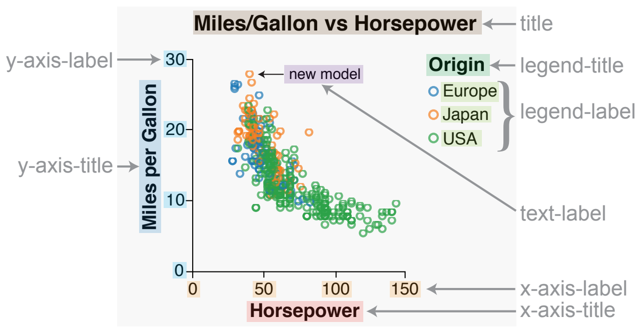


Figure 1: *Text role labels, shown for a Vega-generated scatter plot.*

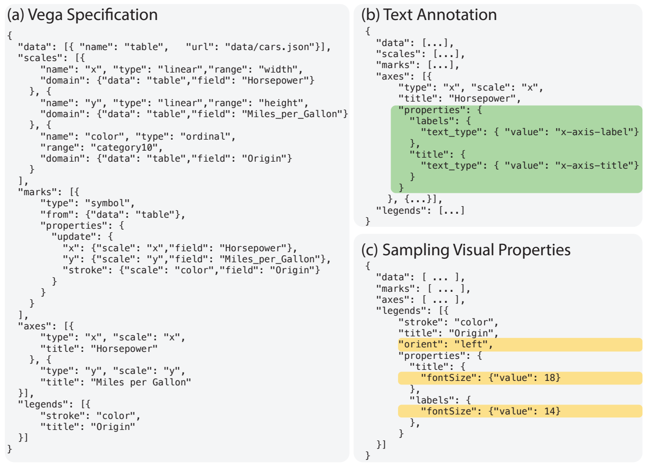
Annotating text on Vega specifications

In Vega, we describe the visual appearance of a visualization but we do not explicitly describe the text elements (*i.e.*, neither text content nor text location is provided). This information is generated later when Vega's compiler processes the specification and generates a *scene graph*. Traversing this *scene graph*, we detect text elements (text content and localization), however, we still do not have enough information to infer text roles (in Vega version 2.6). In order to assign roles to text elements, we annotate Vega specifications with properties that are preserved by the compiler. For example, in Figure 2(b) we annotate title and labels for the x-axis using `x-axis-title` and `x-axis-label` respectively.

Sampling visual properties

In order to increase the visual variability in our chart images, our tool randomly pick values for some visual properties (*e.g.*, font size, legend position). This component reads a configuration file (in a JSON format) where the supported properties are listed, including possible values and their probabilities. We then use a Roulette Wheel algorithm [Bak87] to pick a value given the probabilities. For instance, in the following JSON text, we provide values 16 and 18 for the legend fontSize and their probabilities are 0.5 and 0.5 respectively.

```
"legends" : {
  "title-fontSize": [
    {"value": 16,  "p": 0.5},
    {"value": 18,  "p": 0.5}
  ]
}
```



```
{
  "data": [{ "name": "table",   "url": "data/cars.json"}],
  "scales": [{
      "name": "x", "type": "linear","range": "width",
      "domain": {"data": "table","field": "Horsepower"}
    }, {
      "name": "y", "type": "linear","range": "height",
      "domain": {"data": "table","field": "Miles_per_Gallon"}
    }, {
      "name": "color", "type": "ordinal",
      "range": "category10",
      "domain": {"data": "table","field": "Origin"}
    }
  ],
  "marks": [{
      "type": "symbol",
      "from": {"data": "table"},
      "properties": {
        "update": {
          "x": {"scale": "x","field": "Horsepower"},
          "y": {"scale": "y","field": "Miles_per_Gallon"},
          "stroke": {"scale": "color","field": "Origin"}
        }
      }
    }
  ],
  "axes": [{
      "type": "x", "scale": "x",
      "title": "Horsepower"
    }, {
      "type": "y", "scale": "y",
      "title": "Miles per Gallon"
  }],
  "legends": [{
      "stroke": "color",
      "title": "Origin"
  }]
}
```

```
{
  "data": [...],
  "scales": [...],
  "marks": [...],
  "axes": [{
      "type": "x", "scale": "x",
      "title": "Horsepower",
      "properties": {
        "labels": {
          "text_type": { "value": "x-axis-label"}
        },
        "title": {
          "text_type": { "value": "x-axis-title"}
        }
      }
    }, {...}],
  "legends": [...]
}
```

```
{
  "data": [ ... ],
  "marks": [ ... ],
  "axes": [ ... ],
  "legends": [{
      "stroke": "color",
      "title": "Origin",
      "orient": "left",
      "properties": {
        "title": {
          "fontSize": {"value": 18}
        },
        "labels": {
          "fontSize": {"value": 14}
        },
      }
  }]
}
```

Figure 2: ***Vega Charts Generation:*** *(a) Vega specification for scatter plot in Figure 1. (b) Annotating text elements in x-axis. (c) Setting legend properties* `orient`*, title-*`fontSize`*, and labels-*`fontSize` *with values* `left`*,* `18` *and* `14` *respectively.*

Table 2 describes the supported visual properties and the possibles values. In Figure 2(c) we set the legend properties: `orient`, title-`fontSize`, and label-`fontSize` with the values `left`, `18` and `14` respectively.

Extracting bounding boxes and text

Finally, using the Vega specifications with the text annotations and the visual properties, we call the Vega compiler to render the visualization. Before final image is generated, we traverse the scene graph to detect the text elements (*i.e.*, content, bounding box, and role.). We then export this information into a CSV file.

Pruning wrong charts

Examining the generated visualizations, we note some incorrect charts. Some charts have legends covering mark elements. Some scatter plots have one or more points over the x-axis line. This happens when the y-axis encodes an aggregated variable (*e.g.*, count) and the x-axis is placed on the top. Similarly, there are line charts with a single line over the x-axis. Another common problem is legends without labels (only legend title). These charts are generated when a color channel encodes the autogenerated variable *count* (number of records).

We remove these charts because they violate assumptions in our pipeline. For instance, the post-processing in the text role classification assumes that legends may or may not have a title but they must have at least one label. In the case of charts with lines or points over the x-axis line, these are visually blank plots and our mark type classifier does not have any visual clue to classify them. After manually deleting such files, we have 4,318 chart images.

Visual Property	X-Axis	Y-Axis	Legend
orient	top/bottom	left/right	top/bottom right/left
title font	font name	font name	font name
title font size	number	number	number
label font	font name	font name	font name
label font size	number	number	number
background	–	–	color
tick size	number	number	–
grid line	boolean	boolean	–

Table 2: ***Visual Properties****: List of supported visual properties and possible values.*

Implementation

The data generator component is implemented as a client-side JavaScript application using Node.js. This version has been tested using the libraries Vega 2.6, Vega-Lite 1.2 and Compass 0.6. The tool used for the manual text annotation was implemented as a web application.

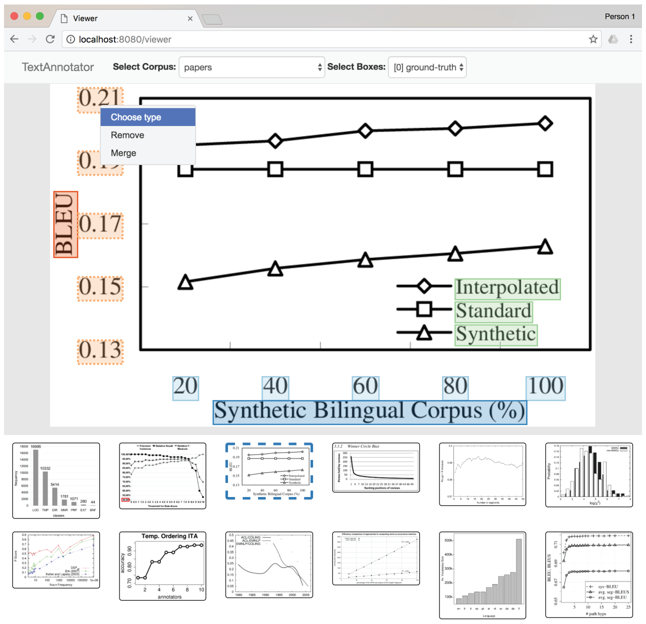
Configuration File

Below is the configuration file we used to generate the Vega corpus.

```
{
  "legends": {
    "orient": [
      {"value": "left",         "p": 0.3},
      {"value": "right",        "p": 0.3},
      {"value": "top-left",     "p": 0.1},
      {"value": "top-right",    "p": 0.1},
      {"value": "bottom-left",  "p": 0.1},
      {"value": "bottom-right", "p": 0.1}
    ],
    "title-fontSize": [
      {"value": 16,   "p": 0.1},
      {"value": 18,   "p": 0.1},
      {"value": 20,   "p": 0.2},
      {"value": 22,   "p": 0.2},
      {"value": 24,   "p": 0.4}
    ],
    "title-font": [
      {"value": "Gill Sans",       "p": 0.25},
      {"value": "Helvetica",       "p": 0.25},
      {"value": "Helvetica Neue",  "p": 0.25},
      {"value": "sans-serif",      "p": 0.25}
    ],
    "labels-fontSize": [
      {"value": 14,   "p": 0.1},
      {"value": 15,   "p": 0.2},
      {"value": 16,   "p": 0.4},
      {"value": 17,   "p": 0.2},
      {"value": 18,   "p": 0.1}
    ],
    "labels-font": [
      {"value": "Gill Sans",       "p": 0.25},
      {"value": "Helvetica",       "p": 0.25},
      {"value": "Helvetica Neue",  "p": 0.25},
      {"value": "sans-serif",      "p": 0.25}
    ],
    "legend-stroke": [
      {"value": "#fff",            "p": 0.5},
      {"value": "#ccc",            "p": 0.25},
      {"value": "#000",            "p": 0.25}
    ]
  },
  "x-axes": {
    "orient": [
      {"value": "bottom",   "p": 0.7},
      {"value": "top",      "p": 0.3}
    ],
    "grid": [
      {"value": true,   "p": 0.7},
      {"value": false,  "p": 0.3}
    ],
    "tickSize": [
      {"value": 0, "p": 0.3},
      {"value": 3, "p": 0.3},
      {"value": 5, "p": 0.4}
    ],
    "title-fontSize": [
      {"value": 16,  "p": 0.1},
      {"value": 18,  "p": 0.1},
      {"value": 20,  "p": 0.2},
      {"value": 22,  "p": 0.2},
      {"value": 24,  "p": 0.4}
    ],
    "title-font": [
      {"value": "Gill Sans",       "p": 0.25},
      {"value": "Helvetica",       "p": 0.25},
      {"value": "Helvetica Neue",  "p": 0.25},
      {"value": "sans-serif",      "p": 0.25}
    ],
    "labels-fontSize": [
      {"value": 14,  "p": 0.1},
      {"value": 15,  "p": 0.2},
      {"value": 16,  "p": 0.4},
      {"value": 17,  "p": 0.2},
      {"value": 18,  "p": 0.1}
    ],
    "labels-font": [
      {"value": "Gill Sans",       "p": 0.25},
      {"value": "Helvetica",       "p": 0.25},
      {"value": "Helvetica Neue",  "p": 0.25},
      {"value": "sans-serif",      "p": 0.25}
    ]
  },
  "y-axes": {
    "orient": [
      {"value": "left",   "p": 0.7},
      {"value": "right",  "p": 0.3}
    ],
    "grid": [
      {"value": true,  "p": 0.7},
      {"value": false, "p": 0.3}
    ],
    "tickSize": [
      {"value": 0, "p": 0.3},
      {"value": 3, "p": 0.3},
      {"value": 5, "p": 0.4}
    ],
    "title-fontSize": [
      {"value": 16,  "p": 0.1},
      {"value": 18,  "p": 0.1},
      {"value": 20,  "p": 0.2},
      {"value": 22,  "p": 0.2},
      {"value": 24,  "p": 0.4}
    ],
    "labels-fontSize": [
      {"value": 14,  "p": 0.1},
      {"value": 15,  "p": 0.2},
      {"value": 16,  "p": 0.4},
      {"value": 17,  "p": 0.2},
      {"value": 18,  "p": 0.1}
    ],
    "title-font": [
      {"value": "Gill Sans",       "p": 0.25},
      {"value": "Helvetica",       "p": 0.25},
      {"value": "Helvetica Neue",  "p": 0.25},
      {"value": "sans-serif",      "p": 0.25}
    ],
    "labels-font": [
      {"value": "Gill Sans",       "p": 0.25},
      {"value": "Helvetica",       "p": 0.25},
      {"value": "Helvetica Neue",  "p": 0.25},
      {"value": "sans-serif",      "p": 0.25}
    ]
  }
}
```

Appendix B: Graphical Interface for Labelling

Figure 3 shows the graphical interface we implemented to facilitate manual labeling. It displays the charts and overlays their text bounding boxes. The interface supports operations such as adding, deleting, merging and resizing boxes, as well as assigning role labels to each.



TextAnnotator Select Corpus: papers Select Boxes: [0] ground-truth

Figure 3: *Graphical interface for manually refining bounding boxes and assigning role labels to text elements.*

References

[Bak87] Baker J. E.: Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application* (Hillsdale, NJ, USA, 1987), L. Erlbaum Associates Inc., pp. 14–21. 1

[WMA*16] Wongsuphasawat K., Moritz D., Anand A., Mackinlay J., Howe B., Heer J.: Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 649–658. 1