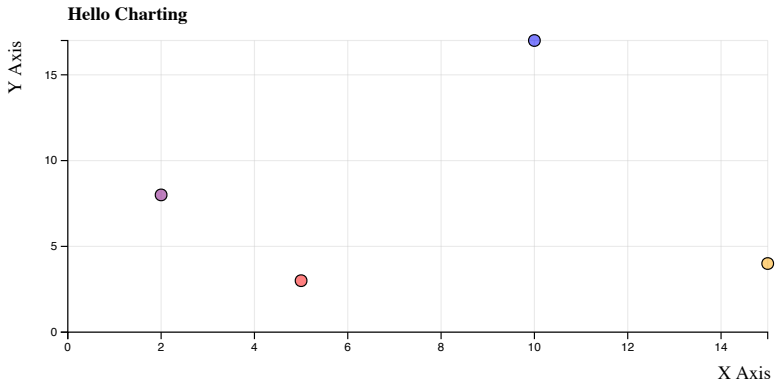
Getting started with *ivml.js*

J. Gentile and A. Meyers

August 25, 2014

1 Getting Started

IVML, the Interactive Visualization Markup Language, is a JavaScript library that leverages popular JavaScript technologies Angular, D3 and JQuery to enable developers to quickly implement interactive data visualizations in a browser. It has predefined a collection of embeddable Angular directives that bind to underlying JavaScript objects. Let's demonstrate its functionality by quickly drawing this graph:



The code for generating this plot is listed below:

```
1  <!DOCTYPE html>
2  <html>
3  <meta charset="utf-8">
4  <script type="text/javascript" src="./vendor/angular/angular.js"></
       script>
5  <script type="text/javascript" src="./vendor/d3/d3.js"></script>
6  <script type="text/javascript" src="./vendor/ivml/ivml.0.5.0.js"></
       script>
7
8  <body ng-app="app">
9      <div ng-controller="controller">
10         <div>
11             <plot height="250" width="600" plot-label-text="'Hello
                   Charting'" yaxis-label-text="'Y Axis'" xaxis-label-
                   text="'X Axis'" xmin="0" xmax="xmax" yticks="5"
                   ymin="0" ymax="ymax">
12                 <points data="data" yfunction="y" xfunction="x"
                       fill='cfunc' radius="5" fill-opacity="0.5"
```

```
13                          stroke="'black'">
14                      </points>
15                  </plot>
16              </div>
17          </div>
18  </body>
19  <script>
20  angular.module('app',['ivml'])
21          .controller('controller',function($scope){
22              $scope.data = [[5,3], [10,17], [15,4], [2,8]];
23              $scope.xmax = d3.max($scope.data, function(d) { return
                    d[0]; });
24              console.log($scope.data)
25              console.log($scope.xmax)
26              $scope.ymax = d3.max($scope.data, function(d) { return
                    d[1]; });
27              $scope.x = function(d){
28                  return d[0];
29              }
30              $scope.y = function(d){
31                  return d[1];
32              }
33              $scope.cfunc = function(d,i){
34                  var c = ['red', 'blue', 'orange', 'purple' ]
35                  return c[i]
36              }
37          })
38  </script>
39  </html>
```

There are at least three things you will probably notice. First, a couple of HTML tags look unfamiliar on lines 11 and 12 (`<plot>`, `<points>`), these directives are provided by IVML. Additionally, there is some *angular.js* boilerplate code (lines 20, 21, and the `$scope` variable). Finally, various properties are defined on the `$scope` object. The next sections provide detail on graphing by directives and data management in *angular.js*.

1.1 Graphing by IVML Directive

IVML provides a set of Angular directives to make graphing very easy. There are three types of these, charts, visual elements, and groups. Charts are high-level directives describing the canvas that visual elements are plotted on. Visual elements are graphics that represent data. In the provided example, the `plot` directive describes the axes while `points` represent the data for plotting. Groups are collections of visual elements that are necessary for generating some plots.

This approach allows us to embed an arbitrary number of visual elements inside a chart so it's easy to have points, line segments, error bars, and rectangles on the same plot. Each visual element can bind to data so expressive and custom graphs can be quickly generated with sets of simple visual elements.

Directives are configured by their attributes. In `plot`, the attributes configure the plotting canvas and axes. Four of the attributes are constant values (attributes surrounded by `' '`), while the value of `xmaximum` and `ymaximum` corresponds to variables in the JavaScript.

Some plots require visual elements to be part of a group. For example, collections of data in a bar graph should be centered around values on a nominal

axis. IVML provides group directives for this purpose. Consider this code snippet from an HTML body:

```
1 <plot xaxis-label-text="'Rects_x'" yaxis-label-text="' Rects y'"
      xordinal-domain="odomain" yminimum="'-2'" ymaximum="2" height=
      "'200'" width="'200'">
2       <bar-group padding="'2'" type="type">
3           <bar data="data1" value-function="'/m'" position-function
                ="'/o'" fill="'blue'" width="'4'"></bar>
4           <bar data="data2" value-function="'/m'" position-function
                ="'/o'" fill="'blue'" width="'4'"></bar>
5           <bar data="data3" value-function="'/m'" position-function
                ="'/o'" fill="'blue'" width="'4'"></bar>
6       </bar-group>
7 </plot>
```

Notice there are multiple `<bar>` tags in a `<bar-group>`. Each `<bar>` is associated with a unique data set but the position of each visual element will depend on other element positions so they must be grouped.

1.2 Designing with *angular.js*

IVML leverages *angular.js* to define its controllers and directives because interactive data visualization are web applications. *angular.js* is a 'model-view-whatever' framework that allows you to develop very powerful web applications. We will cover just enough *angular.js* to give us an understanding of what is going on in our example code but we recommend looking into *angular.js* to learn all it can do.

Notice on lines `8` and `9` there are `ng-*` attributes in the `div`s. These are defining the context of the embedded code for *angular.js*. Our application is called `app`; our controller is `controller`; and we define those in lines `20` and `21`. Note that the controller is defined with a function that takes a `$scope` variable.

Variables referenced in the `ng-controller` HTML context are in the "scope" so `xmaximum="xmax"` means the `xmaximum` attribute for the chart will be equal to `$scope.xmax`. The controller script defines the values referenced by IVML directives and when the referenced values change, the charts will automatically update. We define `$scope.data` and set it as the data attribute in `points` so every time `$scope.data` is changed, the `points` will update to reflect the new data values.

1.3 Role of Callback Functions

Notice that there are three functions described in the controller as members of the `$scope` object, `x(d)`, `y(d)`, and `cfunc(d,i)`. These functions are given as the `xfunction`, `yfunction`, and `fill` of `points`. Points are drawn by adding `circle` elements to the DOM and each element is given a set of attributes defined in the directive. When attribute values are given as callback functions, IVML passes the data object along with its index or key to the function and the returned value should be appropriate for the attribute (i.e. colors or numbers when necessary).

The data being plotted is an array of arrays (`$scope.data`) so each element in the higher-level array represents each point. The accessor for the x position returns the first element of a data object, and the y position returns the second

element. The fill of a point is dependent on its data's position in the list. Any visual element attribute can be passed in as a callback function except for `data`. This should be familiar to users who know *d3.js* (in fact, IVML uses *d3.js* to bind data to DOM elements).

IVML has a construct to simplify callback functions that just return the value at an index or key. In the first example, `$scope.x(d)` and `$scope.y(d)` just returned the value at index `0` and `1`. This can be simplified an attribute a string with '/' as the leading character followed by the index or key. Therefore, on line 12, `yfunction="y"` can be replaced with `yfunction="'/1'"` and the declaration of `$scope.y` can be removed.

1.3.1 Event Callbacks

IVML supports several events for when the user may mouse-over, mouse-out or click on a visual object. Callback functions for these events be set as an attribute. IVML will call event functions with three parameters: the data, index and DOM element (abbreviated as `d,i,e`). Element properties can be changed by JavaScript methods (inkling d3 and jQuery functions). The examples directory provides samples of these events in action.

2 Documentation

This section will describe the attributes and directives provided by IVML in detail. Recall that there are three types of directives, charts, visual elements, and groups. Visual objects need to be in a chart and groups are collections of visual objects.

2.1 *ivml*, *svg*, and *event* Attributes

Visual elements have three types of attributes: *ivml*, *svg*, and *event*. In general, the data attribute in *ivml* will point to a JavaScript object and the accessors will be javascript functions. *svg* attributes can be constant values (surrounded by `''` or callback functions. *event* attributes must be callback functions and the parameters are the data, index and DOM element.

ivml attributes are required by the toolkit and generally relate to the data and accessor routines required for rendering data in the correct position on a chart.

***svg* attributes** directly map to the svg attributes of visual elements so any formats or units recognized by the browser can be utilized.

***event* attributes** are functions that are called when a specified event occurs on a data object. The callback function's parameters are the data, key and DOM element .

2.2 How to Read the Directive Pages

Each IVML directive is documented on the following pages. We provide a brief description, define the directive's type and list its attributes. An underlined

attribute means it's required by the directive. Examples for each directive's usage can be found in the accompanying files.

`<plot>`

Cartesian chart with x and y axes.

ivml attributes

yodomain: array of nominal values for discrete y axes (overrides ymin, ymax, yticks)

xaxis-tick-size: tick size for x axis

xgridlines-visibility: visibility for x axis gridlines

ygridlines-stroke: stroke color for y axis gridlines

yaxis-truncate-ending: string to append to end of y axis text truncated due to exceeding yaxis-text-max-width

ygridlines-fill: fill color for y axis gridlines

margin-right: size in pixels of the right margin

height: height in pixels of the plot area

brush-stroke: stroke color for brush

xgridlines-stroke: stroke color for x axis gridlines

margin-left: size in pixels of the left margin

xmin: minimum value of the x axis

xaxis-label-text: x axis label

ymin: minimum value of the y axis

yaxis-tick-size: tick size for x axis

xgridlines-shape-rendering: shape rendering for x axis gridlines

xgridlines-fill: fill color for x axis gridlines

brush-fill-opacity: fill opacity for brush

ymax: maximum value of the y axis

ytick-format-function: formatter for y axis tick labels

yaxis-shape-rendering: shape rendering for y axis

margin-bottom: size in pixels of the bottom margin

xticks: number of tick marks to be shown on continuous x axis

width: width in pixels of the plot area

brush-fill: fill color for brush

ygridlines-opacity: opacity for y axis gridlines

brush-shape-rendering: shape rendering brush

plot-label-font-color: font color of the main label of the plot

yaxis-stroke: stroke color for y axis

xaxis-font-color: font color for x axis

yaxis-visibility: visibility value for y axis

plot-background: background color of the plot area

yaxis-font-size: font size for y axis

xaxis-text-max-width: maximum width of x axis text in pixels

xaxis-fill: fill color for x axis

xaxis-visibility: visibility value for x axis

ygridlines-visibility: visibility for y axis gridlines

xaxis-truncate-ending: string to append to end of x axis text truncated due to exceeding xaxis-text-max-width

background: background color of the entire element

xtick-format-function: formatter for x axis tick labels

ygridlines-shape-rendering: shape rendering for y axis gridlines

xaxis-shape-rendering: shape rendering for x axis

yaxis-font-color: font color for y axis

brush-clear-on-redraw: set to true if brush should clear when plot is redrawn

plot-label-font-size: font size of the main label of the plot

yaxis-font-family: font family for y axis

xodomain: array of nominal values for discrete x axes (overrides xmin, xmax, xticks)

xaxis-font-size: font size for x axis

yaxis-label-text: y axis label

xaxis-font-family: font family for x axis

yaxis-fill: fill color for y axis

margin-top: size in pixels of the top margin

plot-label-text: text for the main label of the plot

yaxis-text-max-width: maximum width of y axis text in pixels

xmax: maximum value of the x axis

yticks: number of tick marks to be shown on continuous y axis

xgridlines-opacity: opacity for x axis gridlines

xaxis-stroke: stroke color for x axis

event attributes

ybrushstart: function that will be called when the vertical brush starts. Will pass the d3.svg.brush element of the plot as the first parameter.

ybrush: function that will be called when the vertical brush is brushed. Will pass the d3.svg.brush element of the plot as the first parameter.

xbrushend: function that will be called when the horizontal brush ends. Will pass the d3.svg.brush element of the plot as the first parameter. Setting the function disables ybrushstart, ybrush, ybrushend.

xbrushstart: function that will be called when the horizontal brush starts. Will pass the d3.svg.brush element of the plot as the first parameter. Setting the function disables ybrushstart, ybrush, ybrushend.

xbrush: function that will be called when the horizontal brush is brushed. Will pass the d3.svg.brush element of the plot as the first parameter. Setting the function disables ybrushstart, ybrush, ybrushend.

ybrushend: function that will be called when the vertical brush ends. Will pass the d3.svg.brush element of the plot as the first parameter.

brush: function that will be called when the two dimensional brush is brushed. Will pass the d3.svg.brush element of the plot as the first parameter. Setting the function disables xbrushstart, xbrush, xbrushend, ybrushstart, ybrush, ybrushend.

brushend: function that will be called when the two dimensional brush ends. Will pass the d3.svg.brush element of the plot as the first parameter. Setting the function disables xbrushstart, xbrush, xbrushend, ybrushstart, ybrush, ybrushend.

brushstart: function that will be called when the two dimensional brush starts. Will pass the d3.svg.brush element of the plot as the first parameter. Setting the function disables xbrushstart, xbrush, xbrushend, ybrushstart, ybrush, ybrushend.

`<paths>`

Paths are visual elements that are defined by a series of points with x and y values

ivml attributes

yfunction: accessor for the y value of an element of the points array

points-function: returns an array of JavaScript objects that represent the points of the path.

data: the javascript data object to plot

xfunction: accessor for the x value of an element of the points array

svg attributes

stroke-opacity: opacity of object's outline

fill-opacity: fill opacity of object

interpolate: interpolation mode of the object (https://github.com/mbostock/d3/wiki/SVG-Shapes#line_interpolate)

stroke: color of object's outline

stroke-dasharray: dashing of object's outline

stroke-width: width of object's outline

fill: color opacity of object

`<bars>`

Vertical or horizontal bar that is part of a group. The bar's magnitude is it's length along the independent dimension (vertical for horizontal bar charts).

ivml attributes

value-function: accessor for the the bar's value (size and direction)

position-function: accessor for the bar's position on the nominal axis

data: javascript object to plot

stroke: color of the bar's outline

fill-opacity: fill opacity of bar

thickness: the bar's thickness (size parallel to the dependent dimension)

stroke-opacity: opacity of the bar's outline

fill: fill color of the bar

event attributes

click-e: mouse click event

mouse-over-e: mouse over event

mouse-out-e: mouse out event

`<cylinders>`

Disks defined by a radius and height.

ivml attributes

adjustxfunction: TODO

data: javascript data object

adjustyfunction: TODO

centerxfunction: center function for x position

centeryfunction: center function for y position

width: width of the object

height: height of the object

svg attributes

stroke-opacity: stroke opacity

fill-opacity: fill opacity

stroke: stroke color

stroke-dasharray: stroke dashing

radius: radius of the cirle

fill: fill color

event attributes

click-e: mouse click event

mouse-over-e: mouse over event

mouse-out-e: mouse out event

`<donut-charts>`

Donut charts display data as slices of a circle or arch

ivml attributes

yfunction: y position function of the object

xfunction: x position function of the object

data: javascript data object

slice-function: function that determines the size of a slice

fill-function: function determining the fill of a slice

svg attributes

stroke: stroke color of slices

inner-radius: inner radius of slices

outer-radius: outer radius of slices

stroke-opacity: stroke opacity of slices

fill-opacity: fill opacity of slices

event attributes

click-e: mouse click event

mouse-over-e: mouse over event

mouse-out-e: mouse out event

`<points>`

---

ivml attributes

yfunction: accessor for data's y value

xfunction: accessor for data's x value

data: the javascript data object to be plotted

svg attributes

stroke-opacity: opacity of the point's outline

fill-opacity: opacity of the points fill

cursor: hover cursor style

stroke: color of the point's outline

radius: point's radius

stroke-dasharray: dash array for point's outline

fill: point's fill

event attributes

click-e: mouse click event

mouse-over-e: mouse over event

mouse-out-e: mouse out event

`<error-bars>`

Error bars are a visual element which can provide a visual representation of uncertainty around measures. In IVML, these are described by a center location and values describing the uncertainty in the positive and negative x and y directions.

ivml attributes

xcenter-function: accessor for data's function for the center x point

data: the javascript object for this plot

ycenter-function: accessor function for the center y point

left-function: accessor fir data's uncertainty in the positive x direction

up-function: accessor for data's uncertainty in the positive y direction

down-function: accessor fir data's uncertainty in the negative y direction

right-function: accessor fir data's uncertainty in the negative x direction

svg attributes

stroke: line color

stroke-width: line opacity

stroke-opacity: line width

event attributes

click-e: mouse click event

mouse-over-e: mouse over event

mouse-out-e: mouse out event

`<line-segments>`

Line segments are visual elements defined with a starting and ending point.

ivml attributes

x1-function: accessor for data's x start point

y1-function: accessor for data's y start point

data: the javascript data object to be plotted

x2-function: accessor for data's x end point

y2-function: accessor for data's y end point

svg attributes

stroke: color of the line

stroke-width: width of the line

stroke-dasharray: dashing of the line

stroke-opacity: opacity of the line

event attributes

click-e: mouse click event

mouse-over-e: mouse over event

mouse-out-e: mouse out event

`<line-group>`

Plots a group of `<paths>` elements cumulatively as a stacked area chart.

`<bar-group>`

Group of `<bars>` elements, intended for bar charts. This directive requires the data to be index by a nominal value on the axis.

ivml attributes

padding: pixel spacing between bars

type: specifies a `grouped` or `stacked` chart.

arrangement: specifies a `vertical` or `horizontal` chart.