AXLE INFORMATICS

User Guide
LS Vision Chart Library

AXLE INFORMATICS

User Guide

Document Version History

Version	Description of Change	Approved By	Effective Date
0.9	Draft	N/A	09/16/2020
1.0	Version 1.0 Final		

AXLE INFORMATICS

Contents

Introduction

LS Vision is a high-level grammar of interactive charts and provides a concise, declarative JSON syntax to create a range of visualizations. As an Angular library, it was built on top of the Vega-Lite grammar / syntax, which itself is a visualization specification language on top of D3.

LS Vision allows the user to configure charts in intuitive way for easy use out of the box. As it is a superset of the Vega-Lite Grammar, it provides all of the power and flexibility of the well documented library. It was designed for use in the Angular framework and can be embedded inside components.

LS Vision uses an Angular directive to interface with Vega-Lite and create charts. There are two ways in which it can be configured: by using a pure Vega-Lite configuration or using the LS Vision configuration. Figure 1 shows the input parameters of the directive. In order to use a pure Vega-Lite configuration the user can provide the json template for the chart (example shown in Figure 3) to the *config* input parameter. This configuration object must include the schema and syntax noted here and shown on the second line in Figure 3. In order to use the LS Vision configuration, the json template (example shown in Figure 4) must be provided to the *lsConfig* object.

Figure 1

```
// Vega-Lite configuration of chart.
// Either this or lsConfig need to be used
@Input() config: Coordinate;

// Configuration with LS-Chart syntax.
// Either this or @config need to be used
@Input() lsConfig: LsConfig;

// Vega defined theme
@Input() theme;

// Object to be used as data
@Input() data;

// Predefined chart type
@Input() chartType: string;
```

Example usage of the directive

Figure 2

```
<div visionChart [lsConfig]="bar" [data]="barData" [chartType]="'simpleBar'"></div>
```

To render your chart, you define your chart configuration / declarative json template, define your chart type, and define your data. Default Vega-Lite chart types work out of the box with minimal configuration and can be specified with the *chartType* property. The possible chart types:

stackedBar	simpleBar	horizontalBar
groupedBar	Areachart	Donut
Pie	pieLabels	lineChart
multiseriesLine	Scatterplot	scatterplotColored
Histogram	Heatmap	Stream
box		

Examples of each of these charts appear later in the document.

Below is an example of a Vega Lite configuration for a grouped bar chart located here. (Other chart examples are located in the example gallery on the Vega-Lite website.)

Figure 3

```
export const oldGroupedBar = {
  $schema: 'https://vega.github.io/schema/vega-lite/v4.json',
  transform: [
    {
      filter: 'datum.year == 2000'
    },
    {
      calculate: 'datum.sex == 2 ? \'Female\' : \'Male\'',
      as: 'gender'
    }
  ],
  width: {
    step: 12
  },
  mark: 'bar',
  encoding: {
    column: {
      field: 'age',
      type: 'ordinal',
      spacing: 10
    },
    y: {
      aggregate: 'sum',
      field: 'people',
      title: 'population',
      axis: {
        grid: false
      }
```

```
    },
    x: {
      field: 'gender',
      axis: {
        title: ''
      }
    },
    color: {
      field: 'gender',
      scale: {
        range: [
          '#675193',
          '#ca8861'
        ]
```

Below is the configuration of an LS-Vision chart configuration

Figure 4

```
export const groupedbar: LsConfig = {
    column: {
        field: 'age',
    },
    y: {
        field: 'people',
        title: 'population',
    },
    x: {
        field: 'sex',
        title: '',
    },
    color: {
        field: 'sex',
        range: ['#675193', '#ca8861'],
    },
};
```

Both configurations produce the same chart visualization. In the LS-Vision configuration the x and y axes map to LsAxis objects that have properties to configure features and behavior relating to the axes. Above, the “field” property refers to the attribute for objects in the json data array to pull from and map onto the chart. The “title” property refers to the text written for the title of each axis. Note: these axes have been moved from the Vega-Lite encoding object to the root configuration object to simplify object nesting.

The color object specifies which field to use for coloring. It treats each unique value in this field as a group and creates a color for each. In the data, the “sex” field shown has two values which are mapped to colors. The “range” attribute of the color object allows us to specify the two colors that the values will be mapped to. (To map a color to a specific value you should also use the “domain” attribute discussed later). As a note this object has also been simplified and moved from the encoding object in the root configuration object for simplification.

This visualization is a “grouped” bar chart. The “column” field specifies which attribute each group represent. Each group will define a different value from that attribute.

As a note, the transform object in LS-Vision is not allowed as all data transformations must be performed on the data outside the directive.

Schema

Chart Configuration for LS-Charts

```
export interface LsConfig {

    // Y Axis
    x?: LsAxis;

    // X Axis
    y?: LsAxis;

    // Grouping by color
    color?: LsColor;

    // Bar chart color
    fill?: string;

    // For line chart points
    point?: Point;

    // For scatterplot point shapes
    shape?: Field;

    // For grouped columns
    column?: Column;

    // Bubble chart
    size?: Field;

    // For circular plots
    circular?: CircularPlots;

    height?: number;
    width?: number;
    title?: string;
    description?: string;

    // Multiplier for text size
    textSizeMult?: number;
}
```

LS-Axis for x / y axes

```
export interface LsAxis  {

    // Name of the property on the json object data to pull values and plot
    field: string;

    // Number of bins, if not provided, skip binning
    bins?: number | boolean;

    // If false hide the grid marks on the chart
    grid?: boolean;

    // For time plots, this will plot by year, yearmonth, etc
    timeUnit?: string;

    // Title of the axis
    title?: string;

    // Formats the title (useful for time),`datum.value` or %Y
    titleFormat?: string;

    // If you want to change the type
    type?: string;

}
```

```
export interface Column {
    field: string;

    // Space between groups. TODO: implement this
    spacing?: number;
}

export interface LsColor {
    field: string;

    // Ordered list of colors to map
    range?: string[];

    // Ordered list of values to map to colors
    domain?: string[];

    // Title for the legend, if value is null, don't show legend
    legend?: string;
}

export interface Field {
    field: string;
}

export interface CircularPlots {
    innerRadius?: number;
    outerRadius?: number;
    textRadius?: number;
    // Field for specifying text labels
    text?: string;
    theta: string;
}

export interface Point {
    filled?: boolean;
    fill?: string;
}
```

Examples

Bar Chart

Directive Usage

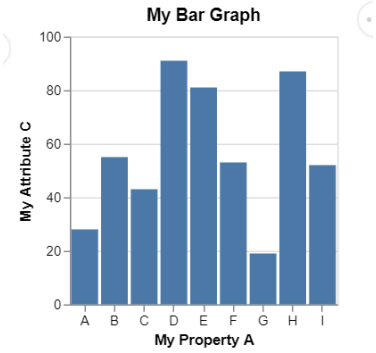
```
<div visionChart [lsConfig]="bar" [data]="barData" [chartType]="'simpleBar'"></div>
```

Configuration object “bar”

```
{
    "height": "200",
    "width": "200",
    "title": "My Bar Graph",
    "description": "This is a bar chart",
    "x": {"field": "a", "title": "My Property A"},
    "y": {"field": "c", "title": "My Attribute C"}
  }
```

Data object barData

```
 {
    "values": [
      {"a": "A", "c": 28}, {"a": "B", "c": 55}, {"a": "C", "c": 43},
      {"a": "D", "c": 91}, {"a": "E", "c": 81}, {"a": "F", "c": 53},
      {"a": "G", "c": 19}, {"a": "H", "c": 87}, {"a": "I", "c": 52}
    ]
  }
```



Above is a simple bar graph that defines a height, width, title, and x / y axes for the chart. The default of the x axis is grouped by values in property “a”. By default this axis is “ordinal” in that even if numbers are provided here, they will be treated like a category rather than numerical values.

Line Chart

Directive Usage (Without X Grid lines)

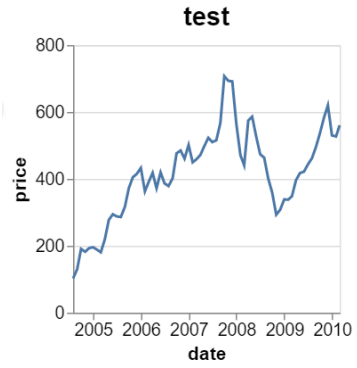
```
<div visionChart [lsConfig]="line" [data]="lineData" [chartType]="'lineChart'"></div>
```

Configuration object “line”

```
{
  "title": "test",
  "description": "Google's stock price over time.",
  "x": {"field": "date", "grid": false},
  "y": {"field": "price"},
  "textSizeMult": 1.3
}
```

Sample Data “lineData”

```
[
   { "symbol": "AAPL", "date": "Jan 1 2000", "price": 39.81 },
   { "symbol": "AAPL", "date": "Feb 1 2000", "price": 36.35 },
   ...
   { "symbol": "AAPL", "date": "Jan 1 2010", "price": 192.06 },
   { "symbol": "AAPL", "date": "Feb 1 2010", "price": 204.62 },
   { "symbol": "AAPL", "date": "Mar 1 2010", "price": 223.02 }
]
```



Above is a simple line chart. The x axis defaults to type “temporal”. Here the “date” field is read in, processed as a date, and graphed by year. If you want a numerical value or “quantitative” type you can override this value by providing the “type” attribute on the axis. Once the type is “quantitative” the values graphed will be processed as numbers. In addition “textSizeMult” is a multiplier that sets text size.

Pie Chart

Directive Usage (Text Size Multiplier)

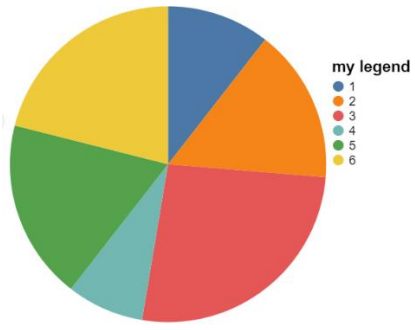
```
<div visionChart [lsConfig]="pie" [data]="pieData" [chartType]="'pie'"></div>
```

Configuration object “pie”

```
{
  "width": 275,
  "description": "A simple pie chart with embedded data.",
  "circular": {
    "outerRadius": 150,
    "theta": "value"
  },
  "color": {"field": "category", "legend": "my legend"},
  "textSizeMult": 1
  }
```

Data object “pieData”

```
[
   {"category": 1, "value": 4},
   {"category": 2, "value": 6},
   {"category": 3, "value": 10},
   {"category": 4, "value": 3},
   {"category": 5, "value": 7},
   {"category": 6, "value": 8}
]
```



The above pie chart defines a circular chart by using the “circular” object. The property “outterRadius” defines the radius of the pie chart, “theta” represents the filed defining the pie wedges, and on the “color” object, the “legend” property defines the title for the lengnd for the pie wedges.

AXLE INFORMATICS

Scatterplot

Directive Usage

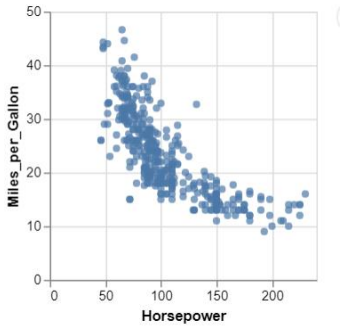
```
<div visionChart [lsConfig]="scatter" [data]="scatterData" [chartType]="'scatterplot'"></div>
```

Configuration object “scatter”

```
{
  "x": {"field": "Horsepower"},
  "y": {"field": "Miles_per_Gallon"}
}
```

Data object “scatterData”

```
[
   {
      "Name":"chevrolet chevelle malibu",
      "Miles_per_Gallon":18,
      "Cylinders":8,
      "Displacement":307,
      "Horsepower":130,
      "Weight_in_lbs":3504,
      "Acceleration":12,
      "Year":"1970-01-01",
      "Origin":"USA"
   },
    …
]
```



Above is a simple scatterplot with x / y axes defined. These axes are defaulted to the “quantitative” type and is processed as numbers.

Histogram

Directive Usage

```
<div visionChart [lsConfig]="histogram" [data]="histogramData" [chartType]="'histogram'"></div
```

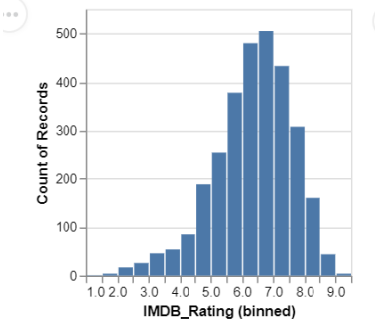
Configuration object “histogram”

```
{
  "x": {
    "bins": "20",
    "field": "IMDB_Rating"
  }
}
```

Data object “histogramData”

```
[{"Title":"The Land Girls","US_Gross":146083,"Worldwide_Gross":146083,"US_DVD_Sales":null,"Production_Budget":8000000,"Release_Date":"Jun 12 1998","MPAA_Rating":"R","Running_Time_min":null,"Distributor":"Gramercy","Source":null,"Major_Genre":null,"Creative_Type":null,"Director":null,"Rotten_Tomatoes_Rating":null,
"IMDB_Rating":6.1,"IMDB_Votes":1071}

    …
]
```



For histograms, the user need only define a single axis for binning. The number of bins is provided by the axis’ “bins” property and the

Horizontal Bar

with Red Fill (you can do it through the more advanced type property)

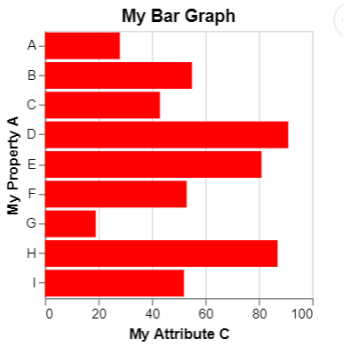
```
<div visionChart [lsConfig]="horizontal" [data]="barData" [chartType]="'horizontalBar'"></div>
```

Configuration object “horizontal”

```
{
    "height": "200",
    "width": "200",
    "title": "My Bar Graph",
    "fill": "red",
    "y": {"field": "a", "title": "My Property A"},
    "x": {"field": "c", "title": "My Attribute C"}
  }
```

Data object barData

```
 {
    "values": [
      {"a": "A", "c": 28}, {"a": "B", "c": 55}, {"a": "C", "c": 43},
      {"a": "D", "c": 91}, {"a": "E", "c": 81}, {"a": "F", "c": 53},
      {"a": "G", "c": 19}, {"a": "H", "c": 87}, {"a": "I", "c": 52}
    ]
  }
```



This horizontal bar chart defaults the x axis as “quantitative” and to be processed as numbers. The y axis is expected to be a group and is defaulted to “ordinal”. Also, the “fill” attribute is depicted here which changes the color of all of the bars to red.

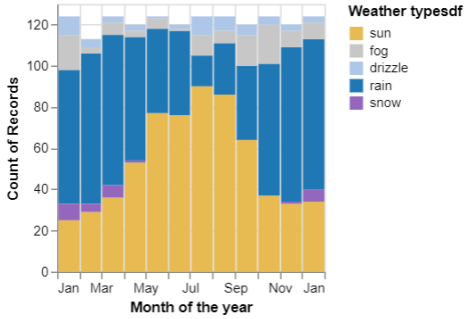
Stacked Bar Chart

(Aggregates, might want to construct a different datasource that doesn't aggregate)

```
<div visionChart [lsConfig]="stacked" [data]="stackedData" [chartType]="'stackedBar'"></div>
```

```
{
  "x": {
    "timeUnit": "month",
    "field": "date",
    "title": "Month of the year"
  },
  "color": {
    "field": "weather",
    "domain": ["sun", "fog", "drizzle", "rain", "snow"],
    "range": ["#e7ba52", "#c7c7c7", "#aec7e8", "#1f77b4", "#9467bd"],
    "legend": "Weather typesdf"
  }
}
```

```
[
    {
      "date": "2012-01-01",
      "precipitation": 0,
      "temp_max": 12.8,
      "temp_min": 5,
      "wind": 4.7,
      "weather": "drizzle"
    },
       ...
}
```



The above chart depicts a stacked bar graph. This graph has a number of important defaults. Note that there is no explicit y axis. This is because it is defaulted to an aggregate of 'count' in Vega Lite terms.

This means the y axis is counted by aggregating values. In addition the timeUnit is set to “month”. This means that it separates the time line by month. Then it aggregates all of the different types on “weather”, which defined in the color object, counts the records for the individual months, then colors the stacks based on the weather type. It is important to note here that the use of domain in the color object here denotes the values of weather to be mapped to colors. These items match up in order with the “range” attribute which lists the colors that the values map to.

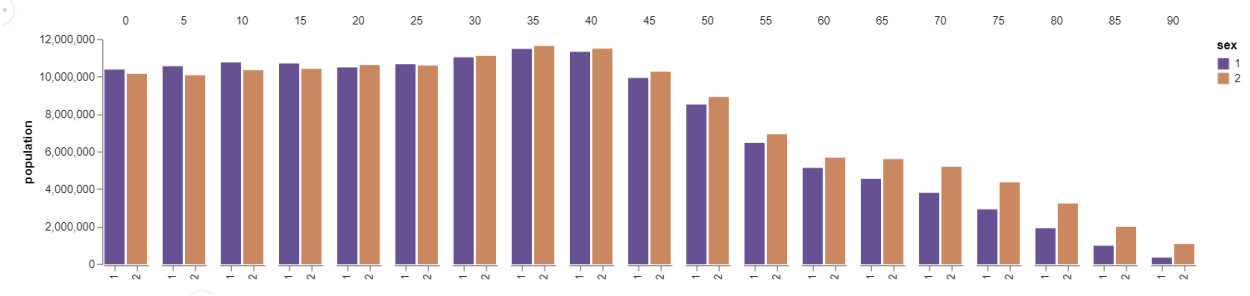
AXLE INFORMATICS

Grouped Bar

```
<div visionChart [lsConfig]="groupedBar" [data]="groupedData" [chartType]="'groupedBar'"></div>
```

```
{
      "column": {
        "field": "age"
      },
      "y": {
         "field": "people",
        "title": "population"
      },
      "x": {
        "field": "sex",
        "title": ""
      },
      "color": {
        "field": "sex",
        "range": ["#675193", "#ca8861"]
      }
  }
```

```
[
{"year":1850,"age":0,"sex":1,"people":1483789},
{"year":1850,"age":0,"sex":2,"people":1450376},
{"year":1850,"age":5,"sex":1,"people":1411067},
{"year":1850,"age":5,"sex":2,"people":1359668}
...
]
```



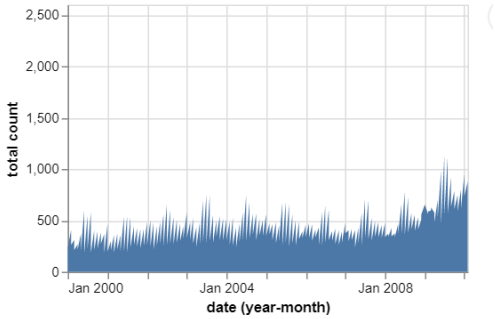
The above grouped bar chart shows

Area Chart Time

```
<div visionChart [lsConfig]="area" [data]="areaData" [chartType]="'areachart'"></div>
```

```
{
  "x": {
    "field": "date", "timeUnit": "yearmonth"
  },
  "y": {
    "field": "count",
    "title": "total count"
  }
}
```

```
[
    {
        "series": "Government",
        "year": 2000,
        "month": 1,
        "count": 430,
        "rate": 2.1,
        "date": "2000-01-01T08:00:00.000Z"
    },
      ...
]
```



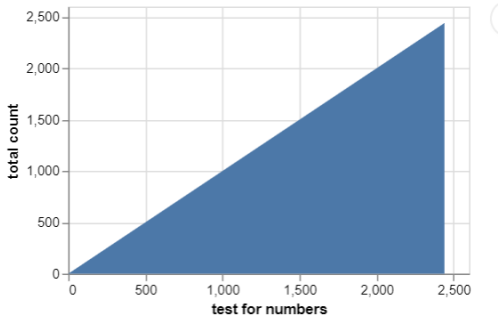
Above depicts an area chart. The x axis is defaulted to “temporal” in Vega-Lite terms. The field “date” is processed as a date, and the “timeUnit” is set to “yearmonth” so the date labels along the x axis show month and year.

Area Number

```
<div visionChart [lsConfig]="areaNumber" [data]="areaData" [chartType]="'areachart'"></div>
```

```
{
  "x": {
    "field": "count", "type": "quantitative", "title": "test for numbers"
  },
  "y": {
    "field": "count",
    "title": "total count"
  }
}
```

```
[
    {
        "series": "Government",
        "year": 2000,
        "month": 1,
        "count": 430,
        "rate": 2.1,
        "date": "2000-01-01T08:00:00.000Z"
    },
      ...
]
```



Above is an area chart that has both axes as numerical, or “quantitative” values. The y axis defaults to quantitative where, as specified by the previous chart, the x axis is defaulted to temporal. In order to change the x axis to “quantitative” the type must be specified.

AXLE INFORMATICS

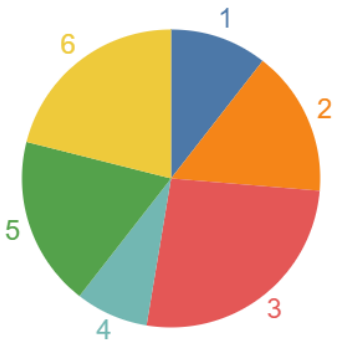
Pie Chart with labels

(and text radius)

```
<div visionChart [lsConfig]="pieLabels" [data]="donutData" [chartType]="'pieLabels'"></div>
```

```
{
  "description": "A simple pie chart with labels.",
  "circular": {
    "outerRadius": 80,
    "textRadius": 90,
    "text": "category",
    "theta": "value"
  },
  "color": {"field": "category", "type": "nominal"},
  "textSizeMult": 1.2
}
```

```
[
      {"category": 1, "value": 4},
      {"category": 2, "value": 6},
      {"category": 3, "value": 10},
      {"category": 4, "value": 3},
      {"category": 5, "value": 7},
      {"category": 6, "value": 8}
    ]
```



The above is a pie chart with labels for each pie wedge. The pure Vega-Lite configuration that produces this chart is rather complicated. Here, you need only provide an “outerRadius” for the pie chart and a “textRadius” to define how far the text will be from the center. Then define the json attribute for the label text and the property that defines the angle for each wedge, “theta”.

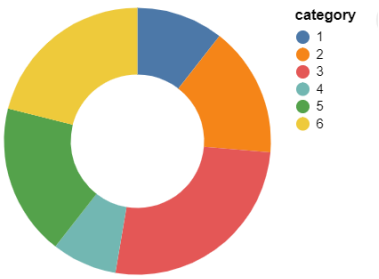
Donut

(and Inner Radius)

```
<div visionChart [lsConfig]="donut" [data]="donutData" [chartType]="'donut'"></div>
```

```
{
  "description": "A simple donut chart with embedded data.",
  "circular": {
    “outerRadius": 150,
    "innerRadius": 50,
    "theta": "value"
  },
  "color": {"field": "category"}
}
```

```
[
      {"category": 1, "value": 4},
      {"category": 2, "value": 6},
      {"category": 3, "value": 10},
      {"category": 4, "value": 3},
      {"category": 5, "value": 7},
      {"category": 6, "value": 8}
    ]
```



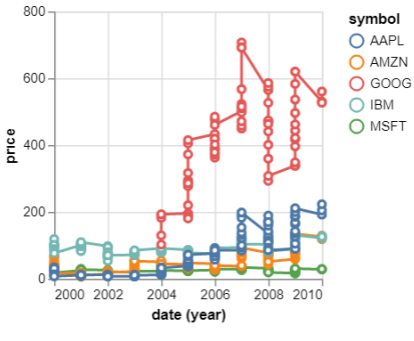
Here a donut chart is drawn. To create the donut hole, one must provide the “innerRadius” for the hole along with the outerRadius for the edge of the circle.

Line with points

```
<div visionChart [lsConfig]="linePoints" [data]="multiseriesData" [chartType]="'multiseriesLine'"></div>
```

```
{
  "description": "Stock prices of 5 Tech Companies over Time.",
  "x": {
    "timeUnit": "year",
    "field": "date"
  },
  "point": {
    "fill": "white",
    "filled": false
  },
  "y": { "field": "price"},
  "color": {"field": "symbol"}
  }
```

```
[
    {
      "symbol": "MSFT",
      "date": "Jan 1 2000",
      "price": 39.81
    },
    {
      "symbol": "MSFT",
      "date": "Feb 1 2000",
      "price": 36.35
    },
...
]
```



The line chart with points defines those points via the “point” object. The user must provide the “fill” color and if it is “filled”.

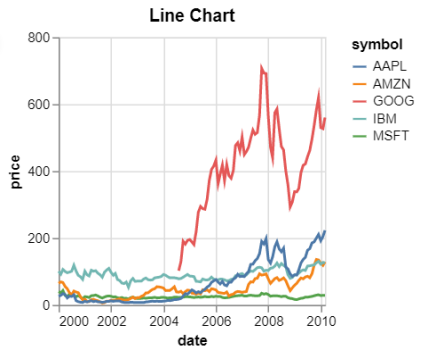
AXLE INFORMATICS

Multiseries

```
<div visionChart [lsConfig]="multiseries" [data]="multiseriesData" [chartType]="'multiseriesLine'"></div>
```

```
{
  "description": "Stock prices of 5 Tech Companies over Time.",
  "title": "Line Chart",
  "x": {"field": "date"},
  "y": {"field": "price"},
  "color": {"field": "symbol"}
}
```

```
[
    {
      "symbol": "MSFT",
      "date": "Jan 1 2000",
      "price": 39.81
    },
    {
      "symbol": "MSFT",
      "date": "Feb 1 2000",
      "price": 36.35
    },
...
]
```

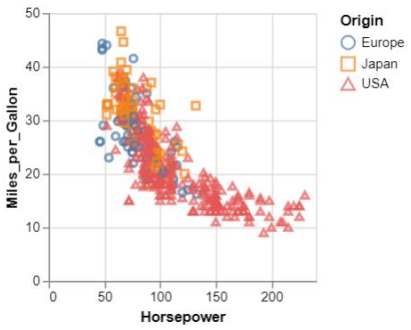


Scatter plot (Shapes and Colors)

```
<div visionChart [lsConfig]="scatterColor" [data]="scatterData" [chartType]="'scatterplotColored'"></div>
```

```
{
  "description": "A scatterplot showing horsepower and miles per gallons.",
  "x": {"field": "Horsepower"},
  "y": {"field": "Miles_per_Gallon"},
  "color": {"field": "Origin"},
  "shape": {"field": "Origin"}
}
```

```
[
   {
      "Name":"chevrolet chevelle malibu",
      "Miles_per_Gallon":18,
      "Cylinders":8,
      "Displacement":307,
      "Horsepower":130,
      "Weight_in_lbs":3504,
      "Acceleration":12,
      "Year":"1970-01-01",
      "Origin":"USA"
   },
...
]
```



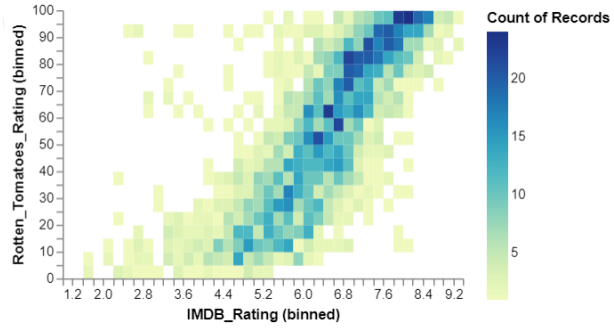
The scatterplot above makes use of the “shape” object. As like the color object, the shape object separates all of the values in the specified attribute into distinct groups. Above, the “Origin” attribute on all of the data has only three values. Those are each given a shape and plotted in the chart and shown in the legend.

Heatmap

```
<div visionChart [lsConfig]="heatmap" [data]="histogramData" [chartType]="'heatmap'"></div>
```

```
{
  "width": 300,
  "height": 200,
    "x": {
      "bins":60,
      "field": "IMDB_Rating"
    },
    "y": {
      "bins": 40,
      "field": "Rotten_Tomatoes_Rating"
    }
}
```


```
[{"Title":"The Land Girls","US_Gross":146083,"Worldwide_Gross":146083,"US_DVD_Sales":null,"Production_Budget":8000000,"Release_Date":"Jun 12 1998","MPAA_Rating":"R","Running_Time_min":null,"Distributor":"Gramercy","Source":null,"Major_Genre":null,"Creative_Type":null,"Director":null,"Rotten_Tomatoes_Rating":null,"IMDB_Rating":6.1,"IMDB_Votes":1071}
```

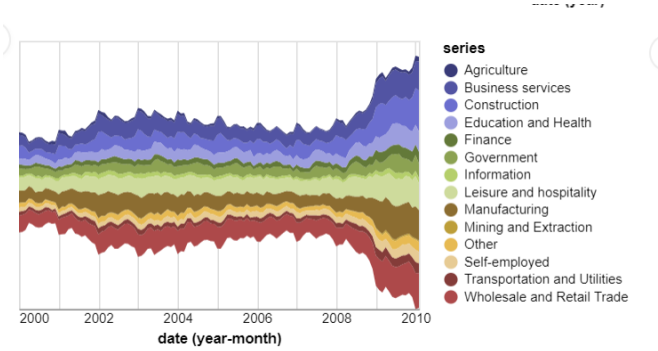
Above is a heatmap or a two dimensional histogram. As stated before, the bins attribute determines how many bins will be created along the axis. In a heatmap both axes have binning and the intensity of the color denotes how many fall in that bin.

Stream graph

```
<div visionChart [lsConfig]="stream" [data]="areaData" [chartType]="'stream'"></div>
```

```
{
  "x": {
    "field": "date", "timeUnit": "yearmonth"
  },
  "y": {
    "field": "count",
    "title": "total count"
  }
}
```

```
[
    {
        "series": "Government",
        "year": 2000,
        "month": 1,
        "count": 430,
        "rate": 2.1,
        "date": "2000-01-01T08:00:00.000Z"
    },
      ...
]
```



The graph drawn above is a stream graph. The x axis is defaulted to “temporal” and the y axis is defaulted to aggregate type “sum” in Vega Lite terms. It is also has the property “stack” set to “center” by default to center the different streams along the y axis. In addition it has a specific color scheme set by default.

AXLE INFORMATICS

Box Plot

```
<div visionChart [lsConfig]="box" [data]="groupedData" [chartType]="'box'"></div>
```

```
{
    "description": "A vertical 2D box plot showing median, min, and max in the US
 population distribution of age groups in 2000.",
      "x": {"field": "age"},
      "y": {
        "field": "people",
        "title": "population"
      }
  }
```

```
[
    {
        "series": "Government",
        "year": 2000,
        "month": 1,
        "count": 430,
        "rate": 2.1,
        "date": "2000-01-01T08:00:00.000Z"
    },
      ...
]
```