VEGA-LITE
A Grammar of Interactive Graphics

Kanit "Ham" Wongsuphasawat @kanitw
Dominik Moritz @domoritz
Arvind Satyanarayan @arvindsatya1
Jeffrey Heer @jeffrey_heer

Interactive Data Lab @uwdata
University of Washington
Data-Driven Documents

Ggplot2 – part of the tidyverse

Vega – A Visualization Grammar

Tableau
Building Blocks
for composing
an expresssive range of graphics

https://gunsmokeandknitting.files.wordpress.com/2014/12/boston-tea-party-16.jpg
Grammar of Graphics
Grammar of Graphics

Data  
Input data source to visualize.
Grammar of Graphics

Data
Input data source to visualize.

Transform
Filter, aggregation, binning, etc.
Grammar of Graphics

Data  Input data source to visualize.
Transform  Filter, aggregation, binning, etc.
Mark  Data-representative graphics.
Grammar of Graphics

Data
Input data source to visualize.

Transform
Filter, aggregation, binning, etc.

Mark
Data-representative graphics.

Encoding
Mapping between data and mark properties.
Grammar of Graphics

Data
Input data source to visualize.

Transform
Filter, aggregation, binning, etc.

Mark
Data-representative graphics.

Encoding
Mapping between data and mark properties.

Scale
Functions that map data values to visual values.
Grammar of Graphics

Data  Input data source to visualize.
Transform  Filter, aggregation, binning, etc.
Mark  Data-representative graphics.
Encoding  Mapping between data and mark properties.
Scale  Functions that map data values to visual values.
Guides  Axes & legends that visualize scales.
Grammar of Graphics for Customized Designs
Grammar of Graphics for Customized Designs

Offer fine-grained control for composing interactive graphics.
Grammar of Graphics for Customized Designs

Offer **fine-grained control** for composing interactive graphics.

But require **verbose** specifications and technical expertise.
Grammar of Graphics for Exploration
Grammar of Graphics for Exploration

Facilitate rapid exploration with concise specifications by omitting low-level details.
Grammar of Graphics for Exploration

Facilitate rapid exploration with concise specifications by omitting low-level details.

Infer sensible defaults and allow customization by overriding defaults.
Grammar of Graphics for Exploration

Facilitate **rapid exploration** with **concise** specifications by omitting low-level details.

Infer **sensible defaults** and allow customization by overriding defaults.

But **limited** support for **interactions**.
Vega-Lite's Mission

Facilitate exploratory data analysis with an expressive yet concise language to specify interactive multi-view graphics
Vega-Lite: a Grammar of Graphics

**Histogram**

**Multi-series Line Chart**

**Stripplot**

**Slope Graph**

**Binned Scatterplot**

**Area Chart**
Vega-Lite: a Grammar of Multi-View Graphics

Scatterplot Matrix

Concatenated & Layered View

Faceted View
Vega-Lite: a Grammar of Interactive Multi-View Graphics

Indexed Chart

Focus+Context

Cross-filtering
Vega-Lite: a Grammar of Interactive Multi-View Graphics

Indexed Chart

Focus+Context

Cross-filtering
Vega-Lite: a Grammar of Interactive Graphics

The Design of Vega-Lite

**Single View** Specification

**Layered** and **Multi-view** Composition

**Interactions** with Selections

Using Vega-Lite

**Programming** with Vega-Lite

Higher-level **Tools** and **Recommendations**
Vega-Lite: a Grammar of Interactive Graphics

The Design of Vega-Lite

**Single View** Specification

- **Layered** and **Multi-view** Composition
- **Interactions** with Selections

Using Vega-Lite

**Programming** with Vega-Lite

- Higher-level **Tools** and **Recommendations**
Specifying Single Views
Specifying Single Views

Abstract Data
Specifying Single Views

Abstract Data → Visual Representation
### Specifying Single Views

#### Abstract Data

<table>
<thead>
<tr>
<th>date</th>
<th>temperature</th>
<th>precipitation</th>
<th>weather</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/1</td>
<td>10.6</td>
<td>10.9</td>
<td>&quot;rain&quot;</td>
</tr>
<tr>
<td>1/2</td>
<td>11.7</td>
<td>0.8</td>
<td>&quot;drizzle&quot;</td>
</tr>
<tr>
<td>1/3</td>
<td>12.2</td>
<td>10.2</td>
<td>&quot;rain&quot;</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

#### Visual Representation

**Weather Data for Seattle**
Specifying Single Views

Weather Data for Seattle

<table>
<thead>
<tr>
<th>date</th>
<th>temperature</th>
<th>precipitation</th>
<th>weather</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/1</td>
<td>10.6</td>
<td>10.9</td>
<td>&quot;rain&quot;</td>
</tr>
<tr>
<td>1/2</td>
<td>11.7</td>
<td>0.8</td>
<td>&quot;drizzle&quot;</td>
</tr>
<tr>
<td>1/3</td>
<td>12.2</td>
<td>10.2</td>
<td>&quot;rain&quot;</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

Abstract Data

Visual Representation

Strip Plot of Temperature
Strip Plot = (Tick with x=field)

Tick Mark

Temperature as x-position
Strip Plot = (Tick with x=field)

Tick Mark

Temperature as x-position
(Quantitative)
Strip Plot = (Tick with x=field)

```
{ data: {url: "weather-seattle.json"},
```

Tick Mark

Temperature as x-position (Quantitative)
Strip Plot = (Tick with x=field)

Tick Mark

Temperature as x-position (Quantitative)

```javascript
{
  data: {url: "weather-seattle.json"},
  mark: "tick",
}
```
**Strip Plot** = (Tick with x=field)

**Tick Mark**

Temperature as x-position (Quantitative)

```
{
  data: {url: "weather-seattle.json"},
  mark: "tick",
  encoding: {
    x: {
      field: "temperature",
      type: "quantitative"
    }
  }
}
```
Strip Plot = (Tick with x=field)

Tick Mark

Temperature as x-position (Quantitative)

Vega-Lite is portable JSON specification
Strip Plot = (Tick with \texttt{x=field})

```json
{
  data: {url: "weather-seattle.json"},
  mark: "tick",
  encoding: {
    x: {
      field: "temperature",
      type: "quantitative"
    }
  }
}
```
Strip Plot: Default Scales and Axes

```
{
  data: {url: "weather-seattle.json"},
  mark: "tick",
  encoding: {
    x: {
      field: "temperature",
      type: "quantitative",
      scale: {type: "linear", domain: [0, 8], ...}
    }
  }
}
```
Strip Plot

{  
  data: {url: "weather-seattle.json"},
  mark: "tick",
  encoding: {
    x: {
      field: "temperature",
      type: "quantitative"
    }
  }
}
Strip Plot

```json
{
  data: {url: "weather-seattle.json"},
  mark: "tick",
  encoding: {
    x: {
      field: "temperature",
      type: "quantitative"
    }
  }
}
```

How many days?
Histogram

Goal
Histogram = (Bar with $x=binned$ field, $y=\text{count}$)
Histogram = (Bar with $x=binned\ field$, $y=count$)

```json
{
  data: {url: "weather-seattle.json"},
  mark: "tick",
  encoding: {
    x: {
      field: "temperature",
      type: "quantitative"
    }
  }
}
```
Histogram = (Bar with $x=binned$ field, $y=count$)

```json
{
  data: {url: "weather-seattle.json"},
  mark: "tick",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    }
  }
}
```
Histogram = (Bar with $x=\text{binned field}$, $y=\text{count}$)

```json
{
    data: {url: "weather-seattle.json"},
    mark: "tick",
    encoding: {
        x: {
            bin: true,
            field: "temperature",
            type: "quantitative"
        }
    }
}
```
Histogram = (Bar with $x=binned$ field, $y=count$)

```json
{
  data: {url: "weather-seattle.json"},
  mark: "tick",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
```
Histogram = (Bar with x=binned field, y=count)

```json
{
    data: {url: "weather-seattle.json"},
    mark: "tick",
    encoding: {
        x: {
            bin: true,
            field: "temperature",
            type: "quantitative"
        },
        y: {
            aggregate: "count",
            type: "quantitative"
        }
    }
}
```
Histogram = (Bar with \( x = \text{binned field}, \ y = \text{count} \))

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
```
Histogram = (Bar with \( x = \) binned field, \( y = \text{count} \))

```javascript
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
```
Sensible Defaults for Binning

Channel determines guide and bin parameters
Sensible Defaults for Binning

Channel determines guide and bin parameters

<table>
<thead>
<tr>
<th>Guide</th>
<th># of Bins</th>
<th>Position</th>
</tr>
</thead>
</table>

![Graph showing BIN(temperature) distribution](image-url)
Sensible Defaults for Binning

Channel determines guide and bin parameters

<table>
<thead>
<tr>
<th>Guide</th>
<th># of Bins</th>
<th>Position</th>
<th>Quantitative axis</th>
</tr>
</thead>
</table>

- BIN(temperature)

![Bar chart showing the distribution of BIN(temperature) values]
# Sensible Defaults for Binning

Channel determines guide and bin parameters

<table>
<thead>
<tr>
<th>Guide</th>
<th># of Bins</th>
</tr>
</thead>
</table>

**Position**

- Quantitative axis
- More bins

![Graph showing binning with temperature on the x-axis and counts on the y-axis.](image-url)
## Sensible Defaults for Binning

Channel determines guide and bin parameters

<table>
<thead>
<tr>
<th>Guide</th>
<th>Color/Opacity/Shape</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guide</td>
<td>Legend with range labels</td>
<td>Quantitative axis</td>
</tr>
<tr>
<td># of Bins</td>
<td>Fewer bins</td>
<td>More bins</td>
</tr>
</tbody>
</table>

### Hottest Temperature
- -10–40
- 0–10
- 10–20
- 20–30
- 30–40

![Graph showing binning for temperature](image)
Histogram

```json
{
    data: {url: "weather-seattle.json"},
    mark: "bar",
    encoding: {
        x: {
            bin: true,
            field: "temperature",
            type: "quantitative"
        },
        y: {
            aggregate: "count",
            type: "quantitative"
        }
    }
}
```
Histogram + Color

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    },
    color: {
      field: "weather",
      type: "nominal"
    }
  }
}
```
Histogram + Color

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    },
    color: {
      field: "weather",
      type: "nominal",
      "scale": {
        "domain": ["sun", "fog", "drizzle", "rain", "snow"],
        "range": ["#e7ba52", "#c7c7c7", "#aec7e8", "#1f77b4", "#9467bd"]
      }
    }
  }
}
```
Histogram + **Color** = **Stacked Histogram**

```javascript
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    },
    color: {
      field: "weather",
      type: "nominal",
      ...
    }
  }
}
```
Stacked Histogram: **Sensible Defaults**

- **no stack**
- **stack (default)**

![Histogram](image-url)
Stacked Histogram: **Sensible Defaults**

Channel (color) + Mark (bar) automatically enables stacking: a layout transform.

**no stack**

**stack (default)**
Stacked Histogram: Sensible Defaults

Channel (color) + Mark (bar) automatically enables stacking: a layout transform.

no stack $\rightarrow$ overlap

stack (default)
Histogram + Color = Stacked Histogram

```
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    },
    color: {
      field: "weather",
      type: "nominal"
    }
  }
}
```
Histogram + Color = Stacked Histogram

```json
{
  data: {'url': 'weather-seattle.json'},
  mark: 'bar',
  encoding: {
    x: {
      bin: true,
      field: 'temperature',
      type: 'quantitative'
    },
    y: {
      aggregate: 'count',
      type: 'quantitative'
    },
    color: {
      field: 'weather',
      type: 'nominal'
    }
  }
}
```

hard to compare without common baseline
Histogram + Color = Stacked Histogram

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {bin: true, field: "temperature", type: "quantitative"},
    y: {aggregate: "count", type: "quantitative"},
    color: { field: "weather", type: "nominal" }
  }
}
```
Histogram + Column = Trellis Histogram

```
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {bin: true, field: "temperature", type: "quantitative"},
    y: {aggregate: "count", type: "quantitative"},
    column: { field: "weather", type: "nominal" }
  }
}
```
Vega-Lite: a Grammar of Interactive Graphics

The Design of Vega-Lite

Single View Specification

Layered and Multi-view Composition

Interactions with Selections

Using Vega-Lite

Programming with Vega-Lite

Higher-level Tools and Recommendations
View Composition Operators
View Composition Operators

facet row: C

\[ C = \begin{array}{c}
  \text{c1} \\
  \text{c2}
\end{array} \]
View Composition Operators

facet row: C

= 

layer: [ ]
View Composition Operators

facet row: $C$

$vconcat: [\begin{array}{c}
\text{\includegraphics[width=0.1\textwidth]{image1}} \\
\text{\includegraphics[width=0.1\textwidth]{image2}}
\end{array}] = [\begin{array}{c}
\text{\includegraphics[width=0.1\textwidth]{image3}} \\
\text{\includegraphics[width=0.1\textwidth]{image4}}
\end{array}]$

layer: $[\begin{array}{c}
\text{\includegraphics[width=0.1\textwidth]{image5}} \\
\text{\includegraphics[width=0.1\textwidth]{image6}}
\end{array}], [\begin{array}{c}
\text{\includegraphics[width=0.1\textwidth]{image7}} \\
\text{\includegraphics[width=0.1\textwidth]{image8}}
\end{array}] = [\begin{array}{c}
\text{\includegraphics[width=0.1\textwidth]{image9}} \\
\text{\includegraphics[width=0.1\textwidth]{image10}}
\end{array}]$
View Composition Operators

facet row: C

layer: [ ] =

vconcat: [ ] =

repeat row: [A,B] =
Monthly Precipitation

```
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      timeUnit: "month",
      field: "date",
      type: "quantitative"
    },
    y: {
      aggregate: "mean",
      field: "precipitation",
      type: "quantitative"
    }
  }
}
```
Monthly Precipitation

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      timeUnit: "month",
      field: "date",
      type: "quantitative"
    },
    y: {
      aggregate: "mean",
      field: "precipitation",
      type: "quantitative"
    }
  }
}
```
Monthly Precipitation

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      timeUnit: "month",
      field: "date",
      type: "quantitative"
    },
    y: {
      aggregate: "mean",
      field: "precipitation",
      type: "quantitative"
    }
  }
}
```
Layering

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      timeUnit: "month",
      field: "date",
      type: "quantitative"
    },
    y: {
      aggregate: "mean",
      field: "precipitation",
      type: "quantitative"
    }
  }
}
```
Layering

```json
{
  data: {url: "weather-seattle.json"},
  layer: [{
    mark: "bar",
    encoding: {
      x: {
        timeUnit: "month",
        field: "date",
        type: "quantitative"
      },
      y: {
        aggregate: "mean",
        field: "precipitation",
        type: "quantitative"
      }
    }
  }]
}```
Layering

```
{
  data: {url: "weather-seattle.json"},
  layer: [
    {
      mark: "bar",
      encoding: {
        x: {
          timeUnit: "month",
          field: "date",
          type: "quantitative"
        },
        y: {
          aggregate: "mean",
          field: "precipitation",
          type: "quantitative"
        }
      }
    },
    {
      mark: "rule",
      encoding: {
        y: {
          aggregate: "mean",
          field: "precipitation",
          type: "quantitative"
        }
      }
    }
  ]
}
```
Concat

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      timeUnit: "month",
      field: "date",
      type: "quantitative"
    },
    y: {
      aggregate: "mean",
      field: "precipitation",
      type: "quantitative"
    }
  }
}
```
Concat

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      timeUnit: "month",
      field: "date",
      type: "quantitative"
    },
    y: {
      aggregate: "mean",
      field: "precipitation",
      type: "quantitative"
    }
  }
}
```
Concat

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      timeUnit: "month",
      field: "date",
      type: "quantitative"
    },
    y: {
      aggregate: "mean",
      field: "precipitation",
      type: "quantitative"
    }
  }
}

{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      timeUnit: "month",
      field: "date",
      type: "quantitative"
    },
    y: {
      aggregate: "mean",
      field: "temperature",
      type: "quantitative"
    }
  }
}
```
Concat

```json
vconcat: [{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      timeUnit: "month",
      field: "date",
      type: "quantitative"
    },
    y: {
      aggregate: "mean",
      field: "precipitation",
      type: "quantitative"
    }
  }
},
  {data: {url: "weather-seattle.json"},
   mark: "bar",
   encoding: {
     x: {
       timeUnit: "month",
       field: "date",
       type: "quantitative"
     },
     y: {
       aggregate: "mean",
       field: "temperature",
       type: "quantitative"
     }
   }
}]
```
Concat

```json
{  
  vconcat: [{
    data: {url: "weather-seattle.json"},
    mark: "bar",
    encoding: {
      x: {
        timeUnit: "month",
        field: "date",
        type: "quantitative"
      },
      y: {
        aggregate: "mean",
        field: "precipitation",
        type: "quantitative"
      }
    }
  },
  {  
    data: {url: "weather-seattle.json"},
    mark: "bar",
    encoding: {
      x: {
        timeUnit: "month",
        field: "date",
        type: "quantitative"
      },
      y: {
        aggregate: "mean",
        field: "temperature",
        type: "quantitative"
      }
    }
  }
]}
```
Repeat

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      timeUnit: "month",
      field: "date",
      type: "quantitative"
    },
    y: {
      aggregate: "mean",
      field: "precipitation",
      type: "quantitative"
    }
  }
}
```
Repeat

```json
{
  repeat: {
    column: ["precipitation", "temperature", "wind"]
  },
  spec: {
    data: {url: "weather-seattle.json"},
    mark: "bar",
    encoding: {
      x: {
        timeUnit: "month",
        field: "date",
        type: "quantitative"
      },
      y: {
        aggregate: "mean",
        field: {repeat: "column"},
        type: "quantitative"
      }
    }
  }
}
```
Repeat

```javascript
{  
  repeat: {  
    column: ["precipitation", "temperature", "wind"]  
  },  
  spec: {  
    data: {url: "weather-seattle.json"},  
    mark: "bar",  
    encoding: {  
      x: {  
        timeUnit: "month",  
        field: "date",  
        type: "quantitative"  
      },  
      y: {  
        aggregate: "mean",  
        field: {repeat: "column"},  
        type: "quantitative"  
      }  
    }  
  }}
```
Repeat

```
{
  repeat: {
    column: ["precipitation", "temperature", "wind"]
  },
  spec: {
    data: {url: "weather-seattle.json"},
    mark: "bar",
    encoding: {
      x: {
        timeUnit: "month",
        field: "date",
        type: "quantitative"
      },
      y: {
        aggregate: "mean",
        field: {repeat: "column"},
        type: "quantitative"
      }
    }
  }
}
```
Concat

{  
  repeat: {  
    column: ["precipitation", "temperature", "wind"]  
  },  
  spec: {  
    data: {url: "weather-seattle.json"},  
    mark: "bar",  
    encoding: {  
      x: {  
        timeUnit: "month",  
        field: "date",  
        type: "quantitative"  
      },  
      y: {  
        aggregate: "mean",  
        field: {repeat: "column"},  
        type: "quantitative"  
      }  
    }  
  }  
}
Repeat

```json
{  
  repeat: {  
    column: [  
      "temperature",
      "precipitation",
      "wind"],
    row: [  
      "wind",
      "precipitation",
      "temperature",
    ]  
  },  
  spec: {}  
}
```
Repeat: SPLOM

```json
{
  repeat: {
    column: [
      "temperature",
      "precipitation",
      "wind"],
    row: [
      "wind",
      "precipitation",
      "temperature"],
  },
  spec: {
  }
}
```
Dashboards

Many views, some of them composed

Manual data, and view management
Dashboards

Many views, some of them composed

Manual data, and view management
Dashboards

Many views, some of them composed

Manual data, and view management

Composition
Composition
Hierarchical View Composition

Facet

Weather

Repeat
Prec, Temp, Wind

Layer

Repeat
Temp, Prec, Wind

Layer
Hierarchical View Composition

Facet

Repeat

Layer

HConcat

Facet

Repeat

Layer

HConcat

Repeat

Prec, Temp, Wind

Facet

Weather

Repeat

Temp, Prec, Wind

Layer
Hierarchical View Composition

- **VConcat**
- **HConcat**
- **Repeat** Prec, Temp, Wind
- **Facet** Weather
- **Repeat** Temp, Prec, Wind
- **Layer**
Layer + Repeat

{ 
  Repeat
  Prec, Temp, Wind 'precipitation', "temperature", "wind"
  s,
  spec: {
    Layer
  }
}
Layer + Repeat

{  
  repeat: {  
    column: ["precipitation", "temperature", "wind"]  
  },  
  spec: {  
    layer: [  
      
    ]  
  }  
}
Vega-Lite: a Grammar of Interactive Graphics

The Design of Vega-Lite

Single View Specification
Layered and Multi-view Composition
Interactions with Selections -> *We’ll discuss this next lecture!*

Using Vega-Lite

Programming with Vega-Lite
Higher-level Tools and Recommendations
Vega-Lite: a Grammar of Interactive Graphics

The Design of Vega-Lite

- Single View Specification
- Layered and Multi-view Composition
- Interactions with Selections

Using Vega-Lite

- Programming with Vega-Lite
- Higher-level Tools and Recommendations
Vega-Lite: a Grammar of Interactive Graphics

The Design of Vega-Lite

Single View Specification
Layered and Multi-view Composition
Interactions with Selections

Using Vega-Lite

Programming with Vega-Lite
Higher-level Tools and Recommendations
Using Vega-Lite
Using Vega-Lite

Compile to Vega and use Vega's runtime
Using Vega-Lite

Compile to Vega and use Vega's runtime

Retarget to different renderers
(Web-based/Server, Canvas/SVG)
Using Vega-Lite

Compile to Vega and use Vega's runtime

Retarget to different renderers (Web-based/Server, Canvas/SVG)

Support streaming data
Using Vega-Lite

Compile to Vega and use Vega's runtime

Retarget to different renderers (Web-based/Server, Canvas/SVG)
Support streaming data

Declarative JSON Syntax
Using Vega-Lite

Compile to Vega and use Vega's runtime

Declarative JSON Syntax

Retarget to different renderers (Web-based/Server, Canvas/SVG)

Support streaming data

Serve as file format
Using Vega-Lite

Compile to Vega and use Vega's runtime

Declarative JSON Syntax

Retarget to different renderers (Web-based/Server, Canvas/SVG)

Support streaming data

Serve as file format

Bindings for different languages
Altair is a declarative statistical visualization library for Python, based on [Vega-Lite](https://vega.github.io/vega-lite/).

With Altair, you can spend more time understanding your data and its meaning. Altair’s API is simple, friendly and consistent and built on top of the powerful [Vega-Lite](https://vega.github.io/vega-lite/) visualization grammar. This elegant simplicity produces beautiful and effective visualizations with a minimal amount of code.
Histogram in Altair

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "Temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
```
Histogram in Altair

```
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "Temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
```
Histogram in Altair

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "Temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
from altair import Chart, expr
```
Histogram in Altair

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "Temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
```

from altair import Chart, expr

altair-viz.github.io
Histogram in Altair

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "Temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
```

```python
from altair import Chart, expr
data_weather = expr.DataFrame('data/weather-seattle.json')
Chart(data_weather)
```
from altair import Chart, expr

data_weather = expr.DataFrame('data/weather-seattle.json')

Chart(data_weather)
from altair import Chart, expr
data_weather = expr.DataFrame('data/weather-seattle.json')
Chart(data_weather).mark_bar()
Histogram in Altair

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "Temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
```

```python
from altair import Chart, expr
data_weather = expr.DataFrame('data/weather-seattle.json')
Chart(data_weather).mark_bar()
```
Histogram in Altair

```json
{
  "data": {"url": "weather-seattle.json"},
  "mark": "bar",
  "encoding": {
    "x": {
      "bin": true,
      "field": "Temperature",
      "type": "quantitative"
    },
    "y": {
      "aggregate": "count",
      "type": "quantitative"
    }
  }
}
```

```python
from altair import Chart, expr
data_weather = expr.DataFrame('data/weather-seattle.json')
Chart(data_weather)
    .mark_bar()
    .encode(
        x=X(bin=True, field='Temperature'),
        y=Y(aggregate='count')
    )
```
Histogram in Altair

```json
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "Temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
```

```python
from altair import Chart, expr
data_weather = expr.DataFrame('data/weather-seattle.json')
Chart(data_weather)
  .mark_bar()
  .encode(
    x=expr.X(bin=True, field='Temperature'),
    y=expr.Y(aggregate='count'))
```

Altair's API is automatically generated from the Vega-Lite JSON schema.
“It is this type of 1:1:1 mapping between thinking, code, and visualization that is my favorite thing about [Altair]”

– Dan Saber.

https://dansaber.wordpress.com/2016/10/02/a-dramatic-tour-through-pythons-data-visualization-landscape-including-ggplot-and-altair/
“We see this portion of the effort as much bigger than Altair itself: the Vega and Vega-Lite specifications are perhaps the best existing candidates for a principled lingua franca of data visualization” – Altair Team.
Vega-Lite: a Grammar of Interactive Graphics

The Design of Vega-Lite

- Single View Specification
- Layered and Multi-view Composition
- Interactions with Selections

Using Vega-Lite

- Programming with Vega-Lite
- Higher-level Tools and Recommendations
Voyager

Augment manual specification with recommendation to **promote breadth & reduce tedium** in exploration.

Use Vega-Lite to **recommend** data and visual encodings.

[https://github.com/vega/voyager](https://github.com/vega/voyager)
No specified visualization yet. Start exploring by dragging a field to encoding pane on the left or examining univariate summaries below.

Univariate Summaries

Cylinders

Origin

Year

Univariate

Specification

Related Views

Filter

Invalid numbers
No specified visualization yet. Start exploring by dragging a field to encoding pane on the left or examining univariate summaries below.

Related Views

Univariate Summaries

- Cylinders
  - amc ambassador brougham
  - amc ambassador dp
  - amc ambassador std
  - amc concord
  - amc concord cl
  - amc concord dl
  - amc concord d
  - amc concord dl 6
  - amc gremlin
  - amc hornet
  - amc hornet sportabout (s)
  - amc matador
  - amc matador (sw)
  - amc pacer
  - amc pacer dl
  - amc rebel std (sw)
  - amc spirit d
  - audi 100 ls
  - audi 100s
  - audi 400d
  - audi 5000
  - audi 5000s (diesel)
  - audi 80
  - bmw 2002
  - bmw 300i
  - buick century

- Year (Year)
  - BIN (Acceleration)

- Origin

- BIN (Acceleration)
1. Browsing Univariate Summaries
1. Browsing Univariate Summaries
1. Browsing Univariate Summaries
1. Browsing Univariate Summaries

No specified visualization yet. Start exploring by dragging a field to encoding pane on the left or examining univariate summaries below.

```json

```
mark: "tick",
encoding: {
  x: {
    field: "Horsepower",
type: "quantitative"
  }
}
```

```json
{
  mark: "tick",
  encoding: {
    x: {
      field: "Horsepower",
      type: "quantitative"
    }
  }
}
```
3. Related Views
3. Related Views
3. Related Views
3. Related Views
3. Related Views
3. Related Views
3. Related Views

```json
mark: "tick",
encoding: {
  x: {
    field: "Horsepower",
    type: "quantitative"
  }
}
```
mark: "tick",
encoding: {
x: {
    field: "Horsepower",
    type: "quantitative"
  }
}
3. Related Views

```json
{
  mark: "tick",
  encoding: {
    x: {
      field: "Horsepower",
      type: "quantitative"
    }
  }
}
```
3. Related Views

- Enumerate Transform
  - mark: "tick",
    encoding: {
      x: {
        fn: bin/mean?,
        field: "Horsepower",
        type: "quantitative"
      }
    }

3. Related Views

Enumerate
Transform

mark: "tick",
encoding: {
x: {
  fn: bin/mean?,
  field: "Horsepower",
  type: "quantitative"
}
}
3. Related Views

```json
{
    mark: "tick",
    encoding: {
        x: {
            fn: "bin",
            field: "Horsepower",
            type: "quantitative"
        }
    }
}
```
3. Related Views

```json
{
    "mark": "tick",
    "encoding": {
        "x": {
            "fn": "bin",
            "field": "Horsepower",
            "type": "quantitative"
        }
    }
}
```
3. Related Views
3. Related Views

```json
{
  "mark": "tick",
  "encoding": {
    "x": {
      "fn": "bin",
      "field": "Horsepower",
      "type": "quantitative"
    }
  }
}
```
3. Related Views

```javascript
{
  mark: "tick",
  encoding: {
    x: {
      fn: "bin",
      field: "Horsepower",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
```
mark: "tick",
encoding: {
  x: {
    fn: "bin",
    field: "Horsepower",
    type: "quantitative"
  },
  y: {
    aggregate: "count",
    type: "quantitative"
  }
}
3. Related Views

{ mark: `?`,
  encoding: {
    x: {
      fn: "bin",
      field: "Horsepower",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
3. Related Views

Enumerate & Rank Encodings

```json
{
  mark: 'mark',
  encoding: {
    x: {
      fn: 'bin',
      field: 'Horsepower',
      type: 'quantitative'
    },
    y: {
      aggregate: 'count',
      type: 'quantitative'
    }
  }
}
```
3. Related Views

Enumerate & Rank

Encodings

```
{ mark: '?',
  encoding: {
    x: {
      fn: "bin",
      field: "Horsepower",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
```
3. Related Views

Enumerate & Rank

Encodings

mark: ?,
encoding: {
  x: {
    fn: "bin",
    field: "Horsepower",
    type: "quantitative"
  }
  y: {
    aggregate: "count",
    type: "quantitative"
  }
}
3. Related Views

Enumerate & Rank

Encodings

```
mark: ?,
encoding: {
  x: {
    fn: "bin",
    field: "Horsepower",
    type: "quantitative"
  },
  y: {
    aggregate: "count",
    type: "quantitative"
  }
}
```
3. Related Views
3. Related Views
3. Related Views
3. Related Views
3. Related Views
4. Wildcard Specification
4. Wildcard Specification
4. Wildcard Specification

Enumerate Variable

```javascript
mark: "tick",
encoding: {
  x: {
    field: "Horsepower",
    type: "quantitative"
  },
  y: {
    field: ?,
    type: "quantitative"
  }
}
```
4. Wildcard Specification

```json
{  
  mark: "tick",  
  encoding: {  
    x: {  
      field: "Horsepower",  
      type: "quantitative"  
    },  
    y: {  
      field: "Weight in lbs",  
      type: "quantitative"  
    }  
  }  
}
```
4. Wildcard Specification
4. Wildcard Specification
Voyager

Augment manual specification with recommendation to **promote breadth & reduce tedium** in exploration.

Use Vega-Lite to **recommend** data and visual encodings.

https://github.com/vega/voyager
Vega-Lite: a Grammar of Interactive Graphics

The Design of Vega-Lite

- Single View Specification
- Layered and Multi-view Composition
- Interactions with Selections

Using Vega-Lite

- Programming with Vega-Lite
- Higher-level Tools and Recommendations
Exploring Data

In this tutorial, you’ll learn a few more techniques for creating visualizations in Vega-Lite. If you are not familiar with Vega-Lite, please read the getting started tutorial first.

For this tutorial, we will create visualizations to explore weather data for Seattle, taken from NOAA. The data file contains columns for the temperature (in Celsius), precipitation (in centimeter), wind (in meter/second), and sun hours. We have one row for each day from January 1st, 2012 to December 31st, 2015.

To load the CSV file with Vega-Lite, we need to provide a URL and set the format type in the data section of the specification.

```json
"data": 
  "url": "data/seattle-weather.csv"
```

Let's start by looking at the precipitation. Precipitation is a quantitative variable. Let's use a tick mark to visualize the distribution of precipitation.

It looks as though precipitation is skewed towards lower values: that is, when it rains, it usually doesn’t rain much. If we want to better see this, we can create a histogram of the precipitation data. For this, we have to add an encoding that uses a discrete field that is categorical. While it is difficult to see patterns across continuous data, we can therefore transform the data on a categorical scale.

Vega-Lite is a high-level visualization grammar. It provides a concise JSON syntax for supporting rapid generation of visualizations to support analysis. Vega-Lite can serve as a declarative format for describing and creating data visualizations. Vega-Lite specifications can be compiled to a lower-level, more detailed Vega specifications and rendered using Vega’s compiler.

This documentation describes the JSON specification language and how to use Vega-Lite visualizations in a web application.
```json
"data": {
  "url": "data/seattle-weather.csv",
  "format": {
    "type": "csv"
  }
},
"mark": "bar",
"encoding": {
  "x": {
    "field": "date",
    "type": "temporal",
    "timeUnit": "month",
    "axis": {
      "title": "Month of the year"
    }
  },
  "y": {
    "field": "*",
    "type": "quantitative",
    "aggregate": "count"
  }
},
"color": {
  "field": "weather",
  "type": "nominal",
  "scale": {
    "domain": ["sun", "fog", "drizzle", "rain", "snow"]
  },
  "range": ["#e7ba52", "#c7c7c7", "#aec7e8", "#1f77b4", "#9467bd"]
},
"legend": {
  "title": "Weather type"
}
```
Vega-Lite – A High-Level Visualization Grammar

Vega-Lite is a high-level visualization grammar. It provides a concise JSON syntax for supporting rapid generation of visualizations to support analysis. Vega-Lite specifications can be compiled to Vega specifications.

Vega-Lite specifications describe visualizations as mappings from data to properties of graphical marks (e.g., points or bars). It automatically produces visualization components including axes, legends, and scales. It then determines properties of these components based on a set of carefully designed rules. This approach allows Vega-Lite specifications to be succinct and expressive, but also provide user control. As Vega-Lite is designed for analysis, it supports data transformations such as aggregation, binning, filtering, sorting, and visual transformations including stacking and faceting.

Read our introduction article on Medium, check out the documentation and take a look at our example gallery.

Example

This is a Vega-Lite specification to create a bar chart that shows the average temperature in Seattle for each month.

```
{
  "schema": "https://vega.github.io/schema/vega-lite/v2.json",
  "data": {
    "name": "data/Seattle-temp.csv"
  },
  "mark": "bar",
  "encoding": {
    "x": {
      "field": "month",
      "type": "temporal"
    },
    "y": {
      "field": "data",
      "type": "quantitative"
    }
  }
}
```

vega.github.io/vega-lite