Vega-Lite
A Grammar of Interactive Graphics

Arvind Satyanarayan @arvindsatya1
Stanford University

Dominik Moritz @domoritz
Kanit "Ham" Wongsuphasawat @kanitw
Jeffrey Heer @jeffrey_heer
University of Washington
Vega

Declarative Interactive Data Visualization
Vega

Declarative Interactive Data Visualization

describes *what* the visualization should look like vs. *how* it should be computed.
**Vega**: Declarative Interactive Data Visualization describes *what* the visualization should look like vs. *how* it should be computed.

- **Data**
- **Event Streams**: `[mousedown, mouseup] > mousemove`

- **Transforms**
- **Signals**: `minX := min(downX, event.x)`

- **Scales**
- **Scale Inversions**: `minVal := xScale.invert(minX)`

- **Guides**
- **Predicates**: `p(t) := t.value ∈ [minVal, maxVal]`

- **Marks**
- **Production Rules**: `fill := p(t) → colorScale(t.category)`
  - `∅ → gray`


Vega: Maximize Expressivity and Performance

Interactive performance
~2x faster than D3.
It is both ridiculous and amazing that this can be expressed in 593 lines (!) of purely declarative spec: vega.github.io/vega-editor/in ...
It is both ridiculous and amazing that this can be expressed in 593 lines (!) of purely declarative spec: vega.github.io/vega-editor/in ...

Lynn Cherry @oceankidbilly sure... but no way do i want to hard code that for a specific case. it's tiny code in dc.js.
Expressivity & performance are important. But so is **concise specification**.

Rapid authoring critical for *exploratory* visualization.

Small language vocabulary promotes alternative designs and systematic enumeration for higher-level applications.
Vega-Lite
A Grammar of Interactive Graphics

Arvind Satyanarayan @arvindsatya1
Stanford University

Dominik Moritz @domoritz
Kanit "Ham" Wongsuphasawat @kanitw
Jeffrey Heer @jeffrey_heer
University of Washington
Vega-Lite

Grammar of Graphics + Grammar of Interaction

Concise, high-level Vega-Lite specifications
Complete, low-level Vega specifications

compiled to

Arvind Satyanarayan @arvindsatya1
Stanford University

Dominik Moritz @domoritz
Kanit "Ham" Wongsuphasawat @kanitw
Jeffrey Heer @jeffrey_heer
University of Washington
Vega-Lite
Grammar of Graphics + Grammar of Interaction

Concise, high-level Vega-Lite specifications
Complete, low-level Vega specifications

Arvind Satyanarayan @arvindsatya1
Stanford University

Dominik Moritz @domoritz
Kanit "Ham" Wongsuphasawat @kanitw
Jeffrey Heer @jeffrey_heer
University of Washington
Single View: Unit Specification

```
{
  "data": {"url": "data/weather.csv"},
  "mark": "circle",
  "encoding": {
    "x": {
      "field": "wind",
      "type": "quantitative"
    },
    "y": {
      "field": "temp_max",
      "type": "quantitative"
    }
  }
}
```
Single View: Unit Specification

```json
{
  "data": {
    "url": "data/weather.csv"
  },
  "mark": "circle",
  "encoding": {
    "x": {
      "field": "wind",
      "type": "quantitative"
    },
    "y": {
      "field": "temp_max",
      "type": "quantitative"
    }
  }
}
```
Single View: Unit Specification

```
{
  "data": {
    "url": "data/weather.csv"
  },
  "mark": "circle",
  "encoding": {
    "x": {
      "field": "wind",
      "type": "quantitative"
    },
    "y": {
      "field": "temp_max",
      "type": "quantitative"
    }
  }
}
```
Single View: Unit Specification

```json
{
    "data": {"url": "data/weather.csv"},
    "mark": "circle",
    "encoding": {
        "x": {
            "field": "wind",
            "type": "quantitative"
        },
        "y": {
            "field": "temp_max",
            "type": "quantitative"
        }
    }
}
```
Single View: Unit Specification

```json
{
  "data": {
    "url": "data/weather.csv"
  },
  "transform": {
    "filter": "datum.location === 'Seattle'"
  },
  "mark": "circle",
  "encoding": {
    "x": {
      "field": "wind",
      "type": "quantitative"
    },
    "y": {
      "field": "temp_max",
      "type": "quantitative"
    }
  }
}
```
Single View: Unit Specification

{  "data": {"url": "data/weather.csv"},
  "transform": {
    "filter": "datum.location === 'Seattle'"
  },
  "mark": "circle",
  "encoding": {
    "x": {
      "field": "wind",
      "type": "quantitative",
      "bin": true
    },
    "y": {
      "field": "temp_max",
      "type": "quantitative",
      "bin": true
    },
    "size": {
      "field": "*",
      "aggregate": "count",
      "type": "quantitative"
    }
  }
}
Single View: Unit Specification

```json
{
  "data": {"url": "data/weather.csv"},
  "transform": {
    "filter": "datum.location === 'Seattle'"
  },
  "mark": "circle",
  "encoding": {
    "x": {
      "field": "wind",
      "type": "quantitative",
      "bin": true
    },
    "y": {
      "field": "temp_max",
      "type": "quantitative",
      "bin": true
    },
    "size": {
      "field": "*",
      "aggregate": "count",
      "type": "quantitative"
    }
  }
}
```
Single View: Unit Specification

{...

Data

"encoding": {
  "x": {
    "field": "wind",
    "type": "quantitative",
    "bin": true
  },
  "y": {
    "field": "temp_max",
    "type": "quantitative",
    "bin": true
  },
  "size": {
    "field": "*",
    "aggregate": "count",
    "type": "quantitative",
    "scale": {"zero": true},
    "legend": {"title": "# of days"}
  }
}

Mark

Encoding

Transforms

(Scales + Guides)

Data + Mark + Encoding + Transforms

Transforms (Scales + Guides)

Single View: Unit Specification
https://vega.github.io/vega-lite/docs/
View Composition
View Composition

layer: [ ] = [ ]
View Composition

layer: [ ] = [ ]

vconcat: [ ] , [ ] = [ ]
View Composition

layer: [ , ] =  
vconcat: [ , ] =

repeat row: [A,B]  =
View Composition

layer: [ ] =  
vconcat: [ ] = 

repeat row: [A,B]  =  

facet row: C  = 

View Composition

Concat

Layer

Single View

Single View

Single View
Layering

```json
{
  "data": {
    "url": "data/weather-sea.csv",
    "mark": "bar",
    "encoding": {
      "x": {
        "timeUnit": "month",
        "field": "date",
        "type": "T"
      },
      "y": {
        "aggregate": "mean",
        "field": "precipitation",
        "type": "Q"
      },
      "color": {"value": "#77b2c7"}
    }
  }
}
```
Layering

```json
{
    "layer": [
        {
            "data": {
                "url": "data/weather-sea.csv"
            },
            "mark": "line",
            "encoding": {
                "x": {
                    "timeUnit": "month",
                    "field": "date",
                    "type": "T"
                },
                "y": {
                    "aggregate": "mean",
                    "field": "temp_max",
                    "type": "Q"
                }
            }
        }
    ]
}
```
Layering

```json
{
    "layer": [
        {
            "layer": []
        },
        {
            "layer": []
        }
    ],
    "resolve": {
        "y": {
            "scale": "independent"
        }
    }
}
```
Vega-Lite

Grammar of Graphics + Grammar of Interaction

Concise, high-level Vega-Lite specifications
Complete, low-level Vega specifications

compiled to

Arvind Satyanarayan @arvindsatya1
Stanford University

Dominik Moritz @domoritz
Kanit "Ham" Wongsuphasawat @kanitw
Jeffrey Heer @jeffrey_heer
University of Washington
Vega-Lite

Grammar of Graphics + Grammar of Interaction

Concise, high-level Vega-Lite specifications
Complete, low-level Vega specifications

compiled to

Arvind Satyanarayan @arvindsatya1
Stanford University

Dominik Moritz @domoritz
Kanit "Ham" Wongsuphasawat @kanitw
Jeffrey Heer @jeffrey_heer
University of Washington
Selection
Selection

an abstraction that defines points of interest, event processing, and a predicate function for inclusion testing.
Vega-Lite Selections

```json
{
  "data": {
    "url": "data/cars.json"
  },
  "mark": "circle",
  "encoding": {
    "x": {
      "field": "Horsepower",
      "type": "Q"
    },
    "y": {
      "field": "Miles_per_Gallon",
      "type": "Q"
    },
    "color": {
      "field": "Origin",
      "type": "N"
    }
  }
}
```

Selections define backing data points, event processing, and a predicate function.
Vega-Lite Selections: A Single Point

```json
{
    "data": {"url": "data/cars.json"},
    "mark": "circle",
    "select": {
        "picked": {"type": "point"}
    },
    "encoding": {
        "x": {"field": "Horsepower", "type": "Q"},
        "y": {"field": "Miles_per_Gallon", "type": "Q"},
        "color": {"field": "Origin", "type": "N"}
    }
}
```

**Selections** define backing data points, event processing, and a predicate function.

Selection **types** provide defaults values for these three components.
Vega-Lite Selections: A Single Point

```json
{
  "data": {
    "url": "data/cars.json"
  },
  "mark": "circle",
  "select": {
    "picked": {"type": "point"}
  },
  "encoding": {
    "x": {
      "field": "Horsepower", "type": "Q"
    },
    "y": {
      "field": "Miles_per_Gallon", "type": "Q"
    },
    "color": [
      {"if": "picked", "field": "Origin", "type": "N"},
      {"value": "grey"}
    ]
  }
}
```

**Selections** define backing data points, event processing, and a predicate function.

Selection **types** provide defaults values for these three components.
Vega-Lite Selections: Multiple Points

```
{
    "data": {"url": "data/cars.json"},
    "mark": "circle",
    "select": {
        "picked": {"type": "list"}
    },
    "encoding": {
        "x": {"field": "Horsepower", "type": "Q"},
        "y": {"field": "Miles_per_Gallon", "type": "Q"},
        "color": [
            {"if": "picked", "field": "Origin", "type": "N"
             "value": "grey"}
        ]
    }
}
```

Selections define backing data points, event processing, and a predicate function.
Selection types provide defaults values for these three components.
Vega-Lite Selections: Multiple Points

Selections define backing data points, event processing, and a predicate function.
Selection types provide defaults values for these three components.
Vega-Lite Selections: Continuous Region

```
{
  "data": {
    "url": "data/cars.json"
  },
  "mark": "circle",
  "select": {
    "picked": {"type": "interval"}
  },
  "encoding": {
    "x": {
      "field": "Horsepower", "type": "Q"
    },
    "y": {
      "field": "Miles_per_Gallon", "type": "Q"
    },
    "color": [
      {"if": "picked", "field": "Origin", "type": "N"
       "value": "grey"}
    ]
  }
}
```

Selections define backing data points, event processing, and a predicate function. Selection types provide defaults values for these three components.
Selection Transforms

manipulate a selection's semantics (points of interest, event processing, or predicate function).
Selection Transforms

manipulate a selection's semantics (points of interest, event processing, or predicate function).

toggle – when events occur, toggles a point in/out of selection;
  automatically instantiated for list selections.

translate – when events occur, offsets spatial properties/data values;
  automatically instantiated for interval selections.

zoom – when events occur, applies a scale factor to selected points.

nearest – accelerates selection via voronoi tessellations.

project – modifies predicate to determine inclusion based on fields or channels.
Vega-Lite Selections: **Multiple Points**

```json
{
    "data": {
        "url": "data/cars.json"
    },
    "mark": "circle",
    "select": {
        "picked": {
            "type": "list",
            "on": "mouseover"
        }
    },
    "encoding": {
        "x": {
            "field": "Horsepower",
            "type": "Q"
        },
        "y": {
            "field": "Miles_per_Gallon",
            "type": "Q"
        },
        "color": [
            {
                "if": "picked",
                "field": "Origin",
                "type": "N"
            },
            {
                "value": "grey"
            }
        ]
    }
}
```

**Selection Transforms** – manipulate a selection's semantics (backing data points, event processing, or predicate function).
Vega-Lite Selections: **Multiple Points**

```
{
  "data": {
    "url": "data/cars.json"
  },
  "mark": "circle",
  "select": {
    "picked": {
      "type": "list",
      "on": "mouseover",
      "nearest": true
    }
  },
  "encoding": {
    "x": {
      "field": "Horsepower",
      "type": "Q"
    },
    "y": {
      "field": "Miles_per_Gallon",
      "type": "Q"
    },
    "color": [
      {
        "if": "picked",
        "field": "Origin",
        "type": "N",
        "value": "grey"
      }
    ]
  }
}
```

**Selection Transforms** – manipulate a selection’s semantics (event processing).
Vega-Lite Selections: **Multiple Origins**

```
{
  "data": {
    "url": "data/cars.json"
  },
  "mark": "circle",
  "select": {
    "id": {
      "type": "list",
      "on": "mouseover"
    }
  },
  "encoding": {
    "x": {
      "field": "Horsepower",
      "type": "Q"
    },
    "y": {
      "field": "Miles_per_Gallon",
      "type": "Q"
    },
    "color": [
      {
        "if": "id",
        "field": "Origin",
        "type": "N"
      },
      {"value": "grey"}
    ]
  }
}
```

**Selection Transforms** – manipulate a selection’s semantics (predicate function).
Vega-Lite Selections: Continuous Region

```json
{
  "data": {
    "url": "data/cars.json"
  },
  "mark": "circle",
  "select": {
    "picked": {"type": "interval"}
  },
  "encoding": {
    "x": {
      "field": "Horsepower", "type": "Q"
    },
    "y": {
      "field": "Miles_per_Gallon", "type": "Q"
    },
    "color": [
      {"if": "picked", "field": "Origin", "type": "N" "value": "grey"}
    ]
  }
}
```
Vega-Lite Selections: Continuous Region (X)

```
{
  "data": {
    "url": "data/cars.json"},
  "mark": "circle",
  "select": {
    "picked": {
      "type": "interval",
      "project": {
        "channels": ["x"]
      }
    }
  },
  "encoding": {
    "x": {
      "field": "Horsepower", "type": "Q"
    },
    "y": {
      "field": "Miles_per_Gallon", "type": "Q"
    },
    "color": [
      {
        "if": "picked",
        "field": "Origin",
        "type": "N"
      },
      {"value": "grey"}
    ]
  }
}
```
Vega-Lite Selections: Continuous Region (Y)

```json
{
  "data": {
    "url": "data/cars.json"
  },
  "mark": "circle",
  "select": {
    "picked": {
      "type": "interval",
      "project": {
        "channels": ["y"]
      }
    }
  },
  "encoding": {
    "x": {
      "field": "Horsepower", "type": "Q"
    },
    "y": {
      "field": "Miles_per_Gallon", "type": "Q"
    },
    "color": [
      {
        "if": "picked",
        "field": "Origin", "type": "N"
      },
      {
        "value": "grey"
      }
    ]
  }
}
```
Vega-Lite Selections

```json
{
  "data": {"url": "data/cars.json"},
  "mark": "circle",
  "encoding": {
    "x": {"field": "Horsepower", "type": "Q"},
    "y": {"field": "Miles_per_Gallon", "type": "Q"},
    "color": {"field": "Origin", "type": "N"}
  }
}
```
Vega-Lite Selections

```
{
  "data": {"url": "data/cars.json"},
  "mark": "circle",
  "select": {
    "grid": {
      "type": "interval"
    }
  },
  "encoding": {
    "x": {"field": "Horsepower", "type": "Q"},
    "y": {"field": "Miles_per_Gallon", "type": "Q"},
    "color": {"field": "Origin", "type": "N"}
  }
}
```
Vega-Lite Selections can be initialized

```json
{
  "data": {
    "url": "data/cars.json",
    "mark": "circle",
    "select": {
      "grid": {
        "type": "interval",
        "init": {
          "scales": true
        }
      }
    },
    "encoding": {
      "x": {
        "field": "Horsepower", "type": "Q"
      },
      "y": {
        "field": "Miles_per_Gallon", "type": "Q"
      },
      "color": {
        "field": "Origin", "type": "N"
      }
    }
  }
}
```

(1) Selection is **populated** with **scale domains**.

(2) Selection now drives scale domains.
Vega-Lite Selections can be initialized

```
{
    "data": {"url": "data/cars.json"},
    "mark": "circle",
    "select": {
        "grid": {
            "type": "interval", "zoom": true,
            "init": {"scales": true}
        }
    },
    "encoding": {
        "x": {
            "field": "Horsepower", "type": "Q",
            "scale": {"domain": {"selection": "grid"}}
        },
        "y": {
            "field": "Miles_per_Gallon", "type": "Q",
            "scale": {"domain": {"selection": "grid"}}
        },
        "color": {"field": "Origin", "type": "N"}
    }
}
```

1. Selection is **populated** with scale domains.  
2. Selection now drives scale domains.
Vega-Lite Layered CrossFilter

```json
{
  "data": {"url": "data/flights.json"},
  "mark": "bar",
  "encoding": {
    "x": {"field": "hour", "type": "Q", "bin": true},
    "y": {"aggregate": "count", "field": ","},
    "type": "Q"
  }
}
```
Vega-Lite Layered CrossFilter

```json
{
  "repeat": {
    "column": ["hour", "delay", "distance"]
  },
  "spec": {
    "data": {
      "url": "data/flights.json"
    },
    "mark": "bar",
    "encoding": {
      "x": {
        "field": {
          "repeat": "column"
        },
        "type": "Q",
        "bin": true
      },
      "y": {
        "aggregate": "count",
        "field": "*",
        "type": "Q"
      }
    }
  }
}
```
Vega-Lite Layered CrossFilter

```json
{
  "repeat": {"column": ["hour", "delay", "distance"]},
  "spec": {
    "layers": [{
      "data": {"url": "data/flights.json"},
      "mark": "bar",
      "encoding": {
        "x": {"field": "repeat": "column"}, "type": "Q", "bin": true,
        "y": {"aggregate": "count", "field": "*", "type": "Q"}
      }
    }, {
      ...
      "color": {"value": "goldenrod"}
    }
  }
}
```
Vega-Lite Layered CrossFilter

```json
{
    "repeat": {
        "column": ["hour", "delay", "distance"]
    },
    "spec": {
        "layers": [
            {
                ...
            }
        ]
    }
}
```

Vega-Lite Layered CrossFilter can be initialized

![Graphs showing distribution of data in different bins.](image)
Vega-Lite Layered CrossFilter

```json
{
  "repeat": {
    "column": ["hour", "delay", "distance"]
  },
  "spec": {
    "layers": [
      {
        ...
      },
      {
        ...
      },
      ...
    ]
  }
}
```
Vega-Lite Layered CrossFilter

```
{
  "repeat": {
    "column": ["hour", "delay", "distance"]
  },
  "spec": {
    "layers": [
      {...
        "select": {
          "region": {
            "type": "interval",
            "project": {
              "channels": ["x"]
            }
          }
        }
      }, ...
      {
        "transform": {
          "filterWith": "region"
        }
      }
    ]
  }
}
```

35 Lines of JSON!
Vega-Lite: Systematically Enumerate Designs


An Ecosystem of Tools

- Wikipedia Graph
- Lyra
- Vega-Lite
- Vega
- D3
- JavaScript
- SVG
- Canvas

- Voyager
- PoleStar
- Altair
An Ecosystem of Tools

Led by Brian Granger and Jake VanderPlas.

Python API 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/

Interest from other Python visualization vendors (Matplotlib, Bokeh, Plotly) to make their own Vega-Lite renderers.

“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.
An Ecosystem of Tools

<table>
<thead>
<tr>
<th>Wikipedia Graph</th>
<th>Lyra</th>
<th>Vega-Lite</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vega</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JavaScript</td>
<td>SVG</td>
<td>Canvas</td>
<td></td>
</tr>
</tbody>
</table>

How can we **automatically partition** interactive workloads between client and server?


What **inference** can we perform over interactive visualization to accelerate analysis and design?
Prototype in supplementary material, a release by the end of the year!